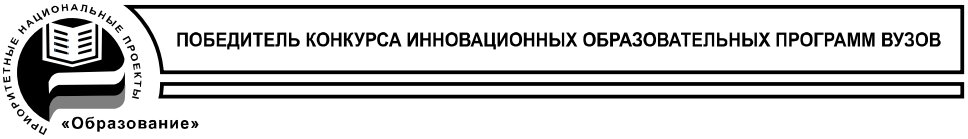
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ



НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК

САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

март–апрель 2008

№ 54

ТЕХНОЛОГИИ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ И КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ

ISSN 1819-222X

Адрес: 197101, Санкт-Петербург, Кронверкский пр., 49, СПбГУ ИТМО

Телефон: (812) 233 12 70

Факс: (812) 233 12 70 (с пометкой: для редакции
Научно-технического вестника)

http: //books.ifmo.ru/ntv

E-mail:karmanov@mail.ifmo.ru

Подписано к печати 25.04.2008 Тираж 350 экз. Заказ № 2(54)

Отпечатано в учреждении «Университетские телекоммуникации»
Адрес: 197101, Санкт-Петербург, Кронверкский пр., 49

---

1 РЕЗУЛЬТАТЫ РЕАЛИЗАЦИИ НАУЧНО-ИССЛЕДОВАТЕЛЬСКИХ ПРОЕКТОВ ФЦП «ИССЛЕДОВАНИЯ И РАЗРАБОТКИ ПО ПРИОРИТЕТНЫМ НАПРАВЛЕНИЯМ РАЗВИТИЯ НАУЧНО-ТЕХНИЧЕСКОГО КОМПЛЕКСА НА 2007–2012 ГОДЫ»

УДК 004.021

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЙ ПРОГРАММНЫЙ КОМПЛЕКС МОДЕЛИРОВАНИЯ НАНОРАЗМЕРНЫХ АТОМНО-МОЛЕКУЛЯРНЫХ СИСТЕМ

В.Н. Васильев, А.В. Бухановский, С.А. Козлов, В.Г. Маслов, Н.Н. Розанов

Предложен подход к созданию высокопроизводительного программного комплекса нового поколения, осуществляющего моделирование наноразмерных структур и комплексов в рамках концепции iPSE. Обосновывается высокоуровневая архитектура комплекса, определяются состав и назначение основных предметно-ориентированных сервисов, рассматриваются вопросы организации человеко-компьютерного взаимодействия, управления данными и знаниями, а также визуализации результатов расчетов.

Ключевые слова: моделирование наноструктур, высокопроизводительные вычисления, архитектура научного программного обеспечения.

Введение

Развитие технологической базы формирования наносистем (атомно-молекулярных или твердотельных структур с размерами менее 100 нм) и материалов на их основе с заданными характеристиками требует создания соответствующих вычислительных методов, алгоритмов и программных комплексов компьютерного моделирования. Одиночная наноструктура интерпретируется как атомно-молекулярная система с большим количеством частиц, взаимодействующих на основе первопринципных закономерностей (ab initio). Для ее описания применимы методы и модели квантовой химии как раздела теоретической и вычислительной физики [1, 2]. Ресурсоемкость вычислительных методов квантовой химии нелинейно возрастает с увеличением числа частиц; как следствие, достижение приемлемого качества расчетов требует использования технологий высокопроизводительных вычислений.

Мировая практика создания высокопроизводительного программного обеспечения для квантовохимических расчетов в настоящее время в основном следует экстенсивному подходу, использующему формальные методы анализа и распараллеливания кода уже существующих программных решений для традиционных (последовательных) вычислительных архитектур. Характерными примерами являются такие универсальные, многоцелевые программные комплексы, как Gaussian [3], GAMESS [4], QChem [5], Jaguar [6], Dalton [7], Molpro [8], Molcas [9] и ORCA [10]. Альтернативой им является линейка программ, изначально реализующих параллельные алгоритмы и специально ориентированных на использование многопроцессорных вычислительных архитектур. Она включает в себя, в частности, системы NWChem [11], MPQC [12] и PQS [13]. При этом развитие и модификация существующих методов квантовохимического моделирования приводят к усложнению структуры программных комплексов, превращая их в интегрированные многофункциональные среды с возможностью организации сложных вычислительных сценариев. К таким средам относится, в частности, Molecular Science Software Suite (MS3), которая включает в себя вышеупомянутую программную систему

NWChem, оболочку PSE Ecce (Problem Solving Environment Extensible Computational Chemistry Environment) и набор библиотек ParSoft, обеспечивающий функционирование NWChem на различных параллельных архитектурах [14].

Несмотря на широкие возможности современных проблемно-ориентированных интегрированных сред (PSE [15]), их специфической особенностью является ориентация, в первую очередь, на распределенные вычисления в слабо связанной вычислительной среде (гиперсети или Грид) [16]. Основной упор делается на распределение отдельных вычислительных подзадач по ресурсам, что отодвигает проблему эффективности их *совокупной* утилизации на второй план. Такой подход является допустимым для достаточно простых вычислительных цепочек. Однако для сложных схем, допускающих ветвления и асинхронное взаимодействие параллельно выполняемых подзадач, требуется использование специфического механизма управления параллельными вычислительными процессами с учетом иерархии «система–узел–процессор–ядро». В данной статье обсуждаются концептуальные основы создания высокопроизводительного программного комплекса нового поколения, реализующего указанный механизм на основе концепции iPSE (Intelligent Problem Solving Environment).

Высокоуровневая архитектура программного комплекса

Концепция iPSE определяет принципы построения интеллектуальной среды управления параллельными вычислительными процессами в распределенной иерархической среде [17], включающей в себя одну или несколько суперЭВМ различной архитектуры. Преимущества такого подхода перед традиционными представлениями о программных комплексах (как совокупности расчетных модулей, библиотек и соответствующей программной документации) состоят в следующем:

– в рамках iPSE формализуются (в форме программных кодов) не только методы и вычислительные алгоритмы, но и *экспертные знания* об организации процесса изучения явления средствами компьютерного моделирования. Другими словами, iPSE реализует функции интеллектуальной системы поддержки принятия решений исследователя, что принципиально важно для практического внедрения такого комплекса;
– iPSE предоставляет *единый интерфейс* взаимодействия для предметно-ориентированных программных модулей и компонентов, которые могут разрабатываться различными коллективами, могут быть написаны на разных языках программирования и иметь различные условия распространения и использования;
– iPSE изначально ориентирована на поддержку высокопроизводительных вычислений, причем не только для суперкомпьютерных систем с традиционной (кластерной) архитектурой, но и для *неоднородных систем*, например, гиперкластеров (суперкомпьютеров, объединенных высокоскоростным каналом). При этом управление эффективностью выполнения сценария является прерогативой iPSE, что позволяет избегать конфликтных ситуаций при разделении ресурсов между различными вычислительными модулями и разными пользователями.

Принципиальным аспектом разработки комплекса в рамках концепции iPSE является выбор парадигмы интеграции его компонентов, которая тесно связана с общемировой тенденцией снижения сложности процессов разработки, тестирования и поддержки программных продуктов на фоне увеличения общей сложности решаемых задач. В частности, это привело к переходу от структурного, объектно-ориентированного и компонентно-ориентированного подходов в инженерии программного обеспечения к специфическим концепциям, таким как CCA (Common Component Architecture [18]), AOA (Aspect Oriented Architecture [19]) и (наиболее общая) – SOA (Service Oriented Architecture). Эти концепции ориентированы на достижение высокого уровня функциональной изоляции компонентов системы, что (в идеальном случае) позволяет интерпре-

тировать отдельные программные элементы как сервисы (SaaS, Software as a Service [20]). Парадигма SaaS позволяет условно разделить программный комплекс на четыре базовые подсистемы (репозиторий вычислительных сервисов, управляющую оболочку, подсистему управления знаниями и данными, подсистему человеко-компьютерного взаимодействия), компоненты которых могут перекрестно взаимодействовать друг с другом на основе общих интерфейсов (рис. 1).

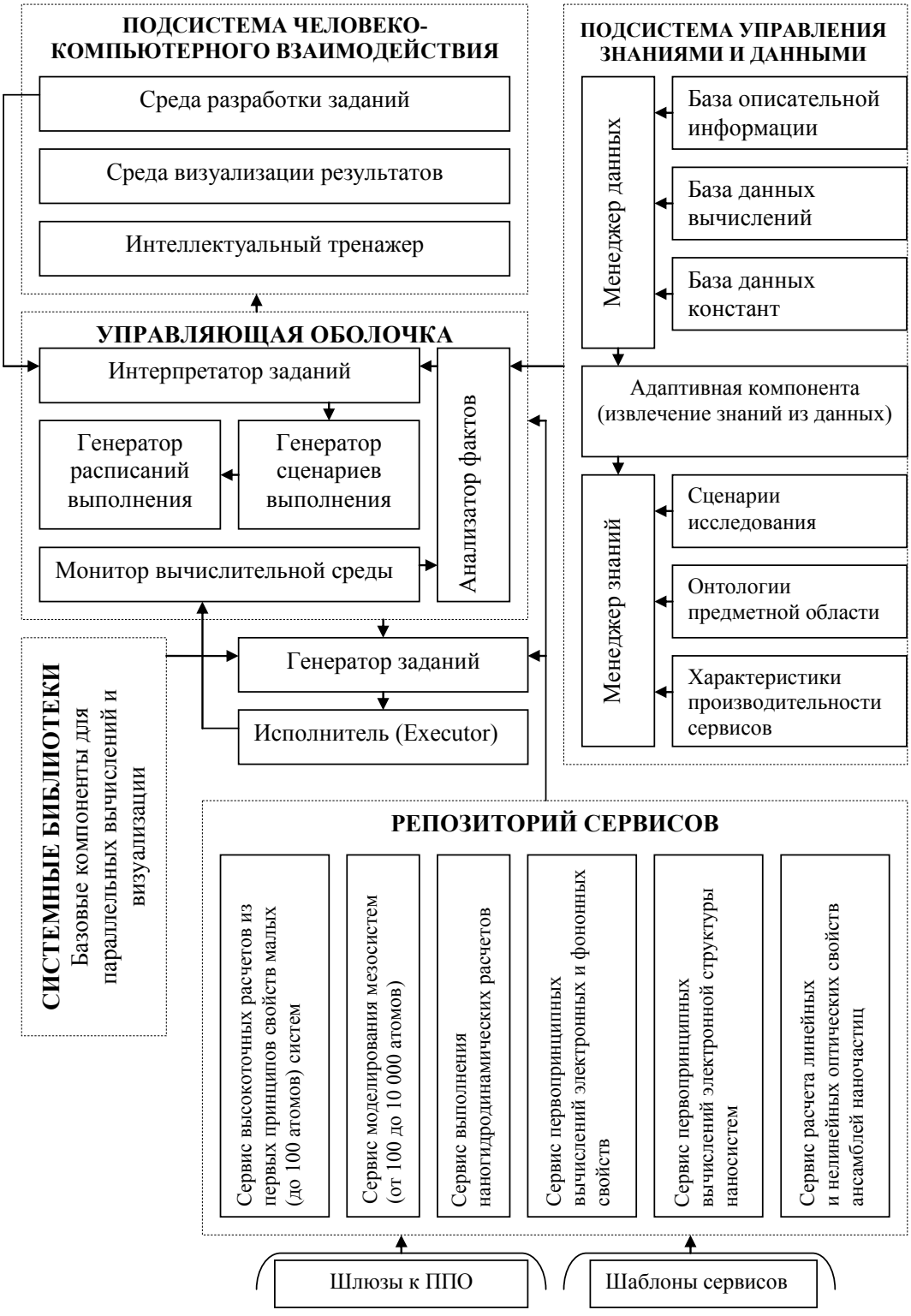


Рис. 1. Концептуальная схема программного комплекса моделирования наноразмерных атомно-молекулярных систем и комплексов на основе концепции iPSE

Репозиторий вычислительных сервисов является основным содержательным элементом системы. Под вычислительным сервисом в данном случае понимается прикладная программа или компонент, *обернутая* (wrap) сервисной оболочкой, которая обеспечивает единый интерфейс передачи данных и управления в среде. Каждый сервис, помимо процедурной части (самого программного модуля), имеет декларативную составляющую, содержащую подробную информацию о заложенных в нем математических моделях, методах и алгоритмах [21]. И процедурная, и декларативная компоненты сервиса могут храниться на разных узлах суперЭВМ и даже на разных компьютерах; информация об их физическом расположении представлена в репозитории. Помимо базовых сервисов, репозиторий содержит и сервисы-шлюзы для подключения независимого программного обеспечения (в том числе коммерческих систем), и «пустые» оболочки для написания или встраивания новых пользовательских сервисов.

Подсистема управления знаниями и данными в основном реализует декларативную составляющую в рамках заданной предметной области. База метазнаний включает в себя информацию о структуре применяемых в сервисах методов и моделей (сложности, степени адекватности конкретной архитектуре и пр.), знания об управлении и решающие знания, которые позволяют строить функционалы качества для оценки программной архитектуры работы пользовательских приложений в среде. База знаний в области нанотехнологий содержит онтологию предметных понятий [22], позволяющую однозначно классифицировать вычислительные сервисы. База технологических знаний о вычислительных сервисах содержит собственно факты (в форме конкретных выражений для моделей производительности, таблиц и аппроксимаций) и описатели (как область применимости конкретной модели производительности). Учитывая специфическую структуру и разветвленность знаний, описывающих сразу две предметных области (компьютерные технологии и нанотехнологии), предполагается использовать комбинированную (фреймово-продукционную) модель представления знаний [23]. Подсистема управления данными, в свою очередь, обеспечивает доступ к данным по четырем основным категориям: данные предметной области (константы), данные текущих расчетов, системные данные, декларативные данные (описания, отчеты и пр.). В подсистему также входит сервис извлечения знаний из данных, который, в первую очередь, необходим для пополнения информации о функционировании самой вычислительной среды.

Управляющая оболочка, по сути, является интеллектуальной системой поддержки принятия решений исследователя, которая, исходя из пользовательского запроса, формирует архитектуру расчетного приложения таким образом, чтобы оно оптимально выполнялось на заданной архитектуре с точки зрения оптимизации системы функционалов (производительности, точности, надежности и пр.) [17]. Система интерпретирует задание (метаописание которого представляется пользователем через систему человеко-компьютерного взаимодействия) и, формируя набор активных фактов предметной области, определяет набор подходящих вычислительных сервисов и сценарий их взаимодействия, после чего строит последовательность конкурирующих оптимальных расписаний их параллельного выполнения, основываясь на мониторинге текущего состояния системы. В результате система определяет оптимальную архитектуру, генерирует описание задания, производит автоматическую генерацию и компиляцию кода с подключением соответствующих библиотек и отправляет его на выполнение через специальный сервис-executor.

Подсистема человеко-компьютерного взаимодействия реализует функции ввода и вывода информации в программном комплексе. Она содержит три базовых компоненты, слабо связанных логически: компоненту пользовательского интерфейса системы (которая позволяет пользователю разрабатывать и запускать задания), компоненту представления информации (научная визуализация) и компоненту системы обучения работы с интегрированной средой (интеллектуальный тренажер) [23].

Предметно-ориентированные вычислительные сервисы

Основной содержательной частью программного комплекса являются предметно-ориентированные вычислительные сервисы, информация о которых хранится в репозитории. Каждый сервис включает в себя одну или несколько вычислительных программ с заданными интерфейсами и условиями эксплуатации. Такой способ организации позволяет инкорпорировать в программный комплекс не только авторские программные коды, но уже существующие свободно распространяемые (в том числе в рамках лицензии GPL) решения, не нарушая авторских и смежных прав их разработчиков.

Сервис высокоточных расчетов из первых принципов свойств малых (до 100 атомов) атомно-молекулярных систем. Под высокоточными расчетами понимаются расчеты, выполненные при таком максимально возможном уровне физического приближения, при котором они могут быть проведены для систем данной степени сложности на данных аппаратных средствах за разумное время. На современном уровне развития суперЭВМ становится очевидным, что такой потенциально точный квантовохимический метод, как метод полного конфигурационного взаимодействия [24], не может быть применен, учитывая экспоненциальную зависимость его трудоемкости от размеров системы. Потому в состав сервиса включены программные реализации методов квантового Монте-Карло (КМК, вариационного и диффузионного) [25], связанных кластеров [26] и многоконфигурационного самосогласованного поля [27]. Эти методы различаются по соотношению «точность / ресурсоемкость вычислений» при заданном размере системы, что и определяет их конкурентность в рамках вычислительной схемы.

Сервис моделирования мезосистем (от 100 до 10000 атомов). Для расчета мезосистем в силу ресурсоемкости не могут быть напрямую использованы первопринципные методы. Как следствие, это требует использования макромоделей определенного уровня, что, в силу полуэмпирического характера их построения, приводит к неоднозначности расчетных подходов. Потому в состав сервиса включены два альтернативных метода. Первый метод основан на методике КМК при ненулевой температуре и позволяет вычислять термодинамические характеристики объекта исследования [28]. Второй метод основан на теории функционала плотности (DFT) [29] и является традиционным для моделирования систем, содержащих сотни и тысячи атомов. Предусматриваются различные реализации метода функционала плотности в базисе как плоских волн, так и атомных орбиталей с применением линейно масштабируемых алгоритмов, с использованием наиболее употребительных функционалов, включая функционалы с исправленным асимптотическим поведением. Для вычисления энергий и интенсивностей электронных переходов предусматривается возможность использования метода Time Dependent DFT (TDDFT) [30].

Сервис термодинамических расчетов и вычисления фононных спектров. В результате вычислений на основе метода DFT, помимо величин, характеризующих электронную структуру объекта (орбитальные волновые функции, матрица плотности и величины, получаемые из нее), могут быть также найдены фононные спектры (зависимости плотности фононных состояний от энергии фонона). Используя фононный спектр, рассчитываются такие термодинамические характеристики, как теплоемкость и абсолютная энтропия [31].

Сервис первопринципных вычислений электронной структуры наносистем, учитывающих сильное межэлектронное взаимодействие в этих системах. Методы на основе DFT не учитывают ряд физически важных эффектов, в частности, взаимодействие между параллельно расположенными плоскими системами, обусловленное динамическими корреляциями. Потому для вычислений электронной структуры используется метод Хартри–Фока с последующим применением теории возмущений Мёллера–

Плессета 2-го порядка (MP2) [32]. В случае систем больших размеров (сотни и тысячи атомов) допустимо использовать только локальные варианты теории возмущений Мёллера–Плессета (Local MP2, LMP2), обеспечивающие приемлемое масштабирование относительно размеров системы [33]. Кроме того, в состав сервиса включен комплекс программных компонентов для расчетов квазиодномерных атомно-молекулярных систем, включая свойства основного и возбужденных состояний мезосистем (до 10^5 атомов). Расчеты выполняются методом функций Грина на основе модельных гамильтонианов для систем с коллективизированными электронами. Для определения параметров гамильтонианов будут проводиться расчеты из первых принципов для ряда идеальных систем.

Сервис расчета линейных и нелинейных оптических свойств ансамблей наночастиц, погруженных в матрицу или нанесенных на поверхность. Сервис включает в себя программные компоненты, реализующие расчет. Они ориентированы на случаи (1) металлических наночастиц на прозрачных диэлектрических подложках, (2) молекулярных *J*-агрегатов, (3) квантоворазмерных объектов (квантовых точек) в полупроводниковых слоях и лазерных схемах, включая лазеры на квантовых ямах и точках [34], (4) материалов для твердотельных лазеров с высокой концентрацией редкоземельных нанокластеров и металлических частиц и (5) компьютерно-генерированных дифракционных элементов нанофотоники.

Для металлических наночастиц учитывается локальное усиление оптических полей вблизи них и на поверхности.

Для *J*-агрегатов в модель включается межмолекулярное взаимодействие, в том числе посредством излучения, и неоднородное уширение; модель будет описывать основные свойства найденных недавно участниками проекта наносолитонов – устойчивых локализованных молекулярных структур с размерами около 1 нм [35].

Для полупроводниковых наноматериалов будут учитываться процессы переноса между одиночными наноструктурами и матрицей; программы должны позволить прогнозировать свойства многослойных полупроводниковых структур, микрорезонаторов и микролазеров с активными (усиливающими) и пассивными (поглощающими) секциями, содержащими квантовые ямы и точки.

Для лазерных материалов, активированных редкоземельными нанокластерами и металлическими наночастицами, основное внимание будет уделяться моделированию процессов трансформации возбуждений [36]. Для дифракционных элементов нанофотоники интерес представляют прямая и обратная задачи дифракции. Эффективность предлагаемого подхода обосновывается сочетанием аналитических и разнообразных численных алгоритмов, включая спектральные и сеточные методы и метод Монте-Карло. Система сервисов позволит прогнозировать оптические свойства наноматериалов, исходя из результатов первопринципных расчетов электронных свойств мезосистем. Будет также предусмотрена возможность решения обратной задачи – подбора характеристик наночастиц для обеспечения наперед заданных свойств наноматериала.

Сервис моделирования процессов наногидродинамики и формирования наночастиц, нанопластин и нанотрубок. Сервис содержит программные компоненты, выполняющие расчеты формирования зародыша новой фазы в растворе или расплаве (методы молекулярной динамики). Выполняется оптимизация характеристик расплава для максимальной степени упорядоченности структуры, моделируется рост наночастиц и взаимодействия между ними методом самосогласованного поля, динамика формирования ансамбля нанопластин и нанотрубок из окислов переходных металлов, их транспортные и механические свойства, а также процесс скручивания нанотрубки с учетом эффектов наногидродинамики [37].

Отображение вычислительных алгоритмов на иерархическую параллельную архитектуру суперЭВМ

Анализ состава предметно-ориентированных сервисов продемонстрировал разнообразие вычислительных методов, соответствующих им параллельных алгоритмов и способов их программной реализации. Потому отображение схемы вычислений (включающей в себя несколько взаимодействующих сервисов) является нетривиальной задачей, требующей использования специальных управляющих механизмов. При создании управляющей оболочки (см. рис. 1) реализуется подход к организации сервисов в неоднородной распределенной среде, позволяющий осуществлять *интеллектуальное управление* их взаимодействием на основе динамической топологии. В отличие от традиционного подхода к организации сервисного взаимодействия посредством шины, ориентированного, прежде всего, на сравнительно однородные системы и сетевые коммуникации, в данном проекте предполагается использовать централизованную систему управления. Это связано как с многообразием типов сервисов, так и с неоднородностью сетевых коммуникаций (характерной для суперЭВМ с большим количеством узлов) с учетом относительной непредсказуемости состояния многопользовательской среды. Возможность организовать такое взаимодействие обеспечивается способом построения сервисов, когда уже на этапе создания сервисной оболочки разработчик предоставляет информацию не только об интерфейсах взаимодействия, но и о характеристиках производительности данного сервиса (времени его работы в зависимости от характеристик данных и вычислительной архитектуры). Фактически эта информация представляет собой экспертное знание, заданное в форме уравнения или табличной функции (профиля приложения). Эффективное взаимодействие сервисов в этом случае организуется самой оболочкой управления, которая выполняет операцию логического вывода (строит оптимальное расписание) на основе знаний о производительности, заложенных в функциональных сервисах, и данных о функционировании информационно-телекоммуникационной системы в целом, получаемых посредством ее мониторинга в реальном времени. Это позволяет выбрать оптимальную схему организации взаимодействия между вычислительными сервисами за счет управления последовательностью их выполнения на ресурсах, способами распараллеливания, балансировкой нагрузки и маршрутами передачи данных (формируя, таким образом, *динамическую топологию взаимодействия* под конкретное композитное приложение). В том случае, если функциональный сервис по какой-либо причине не содержит знаний о характеристиках своей производительности, они могут быть получены для частного случая путем оценочного тестирования (построения профиля приложения) или имитационного моделирования при вызове для этого соответствующих системных сервисов. Основным достоинством такого подхода является возможность гибкого управления производительностью распределенной вычислительной системы с целью повышения степени утилизации ее ресурсов даже в случае, когда архитектура вычислительной системы динамически изменяется (например, перегружаются отдельные каналы или выходят из строя вычислительные узлы). Предлагаемый подход успешно отработан на примере вычислительных SOA-приложений для суперкомпьютеров терафлопной производительности [17].

Визуализация результатов расчетов наноструктур

Для обеспечения человеко-компьютерного взаимодействия в состав программного комплекса включается молекулярный редактор, облегчающий процесс подготовки исходных данных при проведении квантовохимических расчетов, и визуализатор, позволяющий представить результаты расчетов в наглядном виде. Необходимость в таких программах связана с исключительной трудоемкостью процесса подготовки исходных

данных и интерпретации результатов при большом числе атомов. В принципе возможны совмещение функций молекулярного редактора и визуализатора с точки зрения обеспечения визуализации геометрии молекулы, анимационная визуализация (в том числе – нормальных колебаний), построение карт волновых функций, электронной плотности (орбитальной и полной) и электростатического потенциала. Типы карт, построенных по результатам расчета молекулы антрацена, приведены на рис. 2.

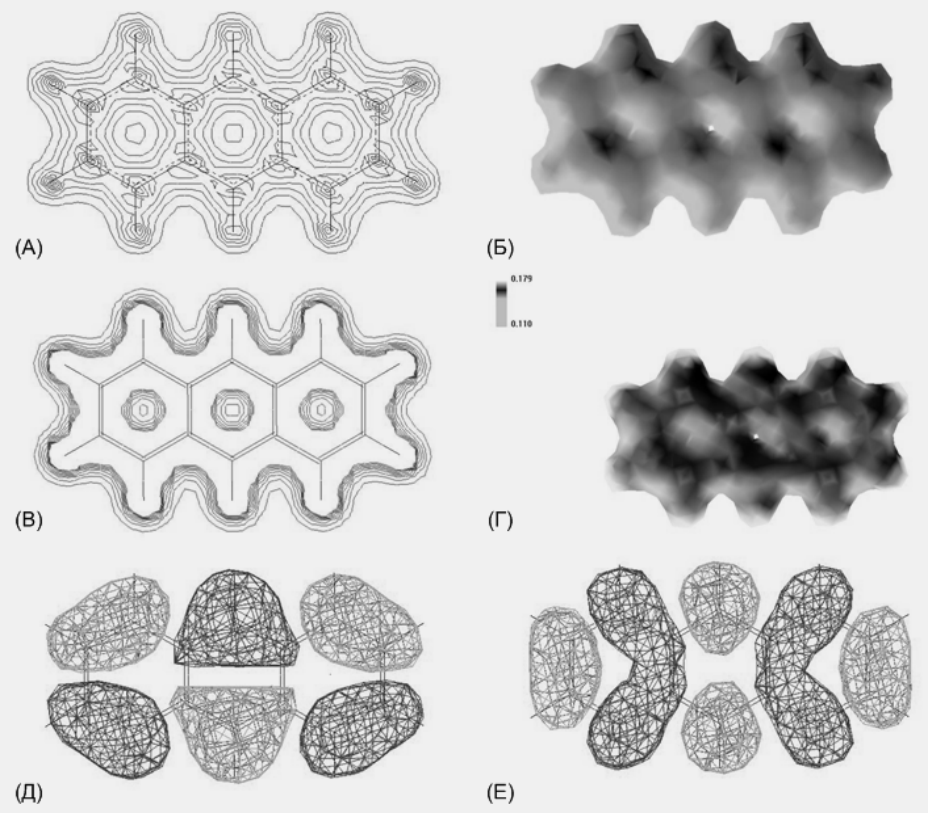


Рис. 2. Формы визуализации результатов расчетов: (А) – 2D-карта суммарной электронной плотности; (Б) – 3D-карта суммарной электронной плотности; (В) – 2D-карта электростатического потенциала; (Г) – 3D-карта электростатического потенциала; (Д) – 3D-карта высшей заполненной (HOMO) орбитали; (Е) – 3D-карта низшей вакантной (LUMO) орбитали

Заключение

В работе предложен подход к созданию программного комплекса нового поколения, осуществляющего компьютерное моделирование атомно-молекулярных наноразмерных структур и комплексов. Обоснована высокоуровневая архитектура комплекса, реализующая основные положения концепции iPSE на основе сервисно-ориентированной архитектуры. Разработан состав и архитектура основных классов сервисов в рамках заданной онтологии; представлен механизм динамического управления производительностью в условиях иерархической архитектуры вычислительной среды. Обоснован выбор расчетных методов и вычислительных технологий моделирования малых систем (до 100 атомов), мезосистем (от 100 до 10 000 атомов), расчета термодинамических характеристик таких систем, расчета линейных и нелинейных оптических свойств ансамблей наночастиц и моделирования процессов наногидродинамики и формирования наночастиц, нанопластин и нанотрубок.

Работа выполнена в рамках ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007–2012 годы», проект 2008-04-2.4-15-003.

Литература

1. Levine N. Quantum Chemistry. – 4th ed. – Prentice Hall, Englewood Cliffs, NJ, 1991. W.J.
2. Pauling L., Wilson E.B. Introduction to Quantum Mechanics with Applications to Chemistry. – Dover Publications, 1989.
3. The Official Gaussian Website. – Режим доступа: http://www.gaussian.com, свободный.
4. The General Atomic and Molecular Electronic Structure System. – Режим доступа: http://www.msg.ameslab.gov/GAMESS, свободный.
5. Q–Chem. – Режим доступа: http://www.q–chem.com, свободный.
6. Schrödinger. – Режим доступа: http://www.schrodinger.com, свободный.
7. Dalton quantum chemistry program. – Режим доступа: http://www.kjemi.uio.no/software/dalton/dalton.html, свободный.
8. Molpro quantum chemistry. – Режим доступа: package http://www.molpro.net, свободный.
9. Molcas 7. – Режим доступа: http://www.teokem.lu.se/molcas, свободный.
10. ORCA – Home. – Режим доступа: http://www.thch.uni–bonn.de/tc/orca, свободный.
11. NWChem Home Page. – Режим доступа: http://www.emsl.pnl.gov/docs/nwchem/nwchem.html, свободный.
12. The Massively Parallel Quantum Chemistry Program. – Режим доступа: http://www.mpqc.org, свободный.
13. Parallel Quantum Solutions. – Режим доступа: http://www.pqs–chem.com, свободный.
14. Environmental Molecular Sciences Laboratory. – Режим доступа: www.emsl.pnl.gov:2080/mscf/about/descr_ms3.htm, свободный.
15. Gallopoulos S., Houstis E., Rice J. Computer as Thinker/Doer: Problem–Solving Environments for Computational Science // IEEE Computational Science and Engineering. – Summer 1994.
16. Grid Computing Environments 2001 Special Issue of Concurrency and Computation: Practice and Experience. – Режим доступа: http://aspen.ucs.indiana.edu/gce, свободный.
17. Ковальчук С.В. и др. Особенности проектирования высокопроизводительных программных комплексов для моделирования сложных систем // Информационно–управляющие системы. – 2008. – Т. 34. – №3. – С. 10–18.
18. Bernholdt D.E., Elwasif W.R., Kohl J.A., Epperly T.G.W. A Component Architecture for High–Performance Computing // Proceedings of the Workshop on Performance Optimization via High–Level Languages and Libraries, New York, USA, 22 June 2002.
19. Navasa A., Pérez M.A., Murillo J.M. Aspect Modelling at Architecture Design // Lecture Notes in Computer Science. – 2005. – Vol. 3527. – P. 41–58.
20. Cohen S. Ontology and Taxonomy of Services in a Service–Oriented Architecture // The Architecture Journal. – Microsoft. – 2007. – №11.– P. 30–35.
21. Jeffery K. GRIDs offer a clear view of the future // The future of the GRID – eStrategies Projects (Special Issue). – 2008. – P. 8–9.
22. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. – Санкт–Петербург: Питер, 2000.
23. Нечаев Ю.И. Искусственный интеллект: концепции и приложения. – СПб, 2002.
24. Szabo A., Ostlund N.S. Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory. – New York: McGraw–Hill, 1989.
25. Kent P.R.C. Techniques and Applications of Quantum Monte Carlo: PhD Dissertation / Robinson College. – Cambridge, August 1999. – Режим доступа: http://www.physics.uc.edu/~pkent/thesis/Thesis.html, свободный.

26. Bartlett R.J. Coupled–Cluster Theory: An Overview of Recent Developments // Modern Electronic Structure Theory, Part II.
27. Schaefer H.F. Ph.D. thesis / Stanford University. – Stanford, CA, 1969.
28. Замалин В.М., Норман Г.Э., Филинов В.С. Метод Монте–Карло в статистической термодинамике. – М.: Наука, 1977. – 228 с.
29. Dreizler R.M., Gross E.K.U. Density Functional Theory. – Berlin: Springer, 1990.
30. Dreuw, Head–Gordon M. Single–Reference ab Initio Methods for the Calculation of Excited States of Large Molecules // Chem. Rev. – 2005. – V. 105. – P.4009–4037.
31. Ismer L., Ireta J., Boeck S., Neugebauer J. Phonon spectra and thermodynamic properties of the infinite polyalanine a helix: Adensity–functional–theory–based harmonic vibrational analysis // Phys. Rev. E 71. – 2005. – № 031911. – 031911–5.
32. Pople J.A., Binkley J.S., Seeger R. Theoretical Models Incorporating Electron Correlation // Int. J. Quant. Chem. Symp. – 1976. – V. 10. – № 1.
33. Head–Gordon M., Lee M.S. and Maslen P.E. Simulation and Theory of Electrostatic Interactions in Solution; Ed. by L.R. Pratt and G. Hummer. – AIP Conference Proceedings, 492. – AIP, New York, 1999. – P. 301.
34. Fedorov S.V. Laser solitons and excitons in semiconductor media // Proc. SPIE. – 2007. – Vol. 6724. – P. 672521.
35. Киселев Ал.С., Киселев Ан.С., Розанов Н.Н. Наноразмерные диссипативные дискретные солитоны в резонансно возбуждаемых J–агрегатах // Письма в ЖЭТФ. – 2008.– Т. 87. – № 11–12. – С. 763–766.
36. Nikonorov N.V., Przhevuskii A. Monte–Carlo simulation of up-conversion processes in Erbium-doped materials // Optical Materials. – 2003. – Vol. 21. – P. 729–741.
37. Chivilikhin S.A., Gusarov V.V., Popov I.Yu., Svitenkov A.I. Model of fluid flow in nano–channel // Russian Journal of Mathematical Physics. – 2008. – Vol. 15. – № 3. – P. 410–412.

Васильев Владимир Николаевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, ректор, д.т.н., профессор, vasilev@mail.ifmo.ru

Бухановский Александр Валерьевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, д.т.н., профессор, boukhanovsky@mail.ifmo.ru

Козлов Сергей Аркадьевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, декан, д.ф.-м.н., профессор, kozlov@mail.ifmo.ru

Маслов Владимир Григорьевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, вед. научн. сотр., д.ф.-м.н., maslov@sp.ru

Розанов Николай Николаевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, д.ф.-м.н., профессор, rosanov@nr3748.spb.edu

УДК 004.021, 004.896

КОНЦЕПЦИЯ И МЕТОДОЛОГИЧЕСКИЕ ОСНОВЫ СОЗДАНИЯ ИНТЕЛЛЕКТУАЛЬНОГО БАЗИСА ГРИД-СИСТЕМ

Ю.И. Нечаев, А.В. Бухановский, В.Н. Васильев

Обсуждается проблема создания интеллектуальных Грид-систем. Основное внимание уделяется формулировке концептуального базиса и принципов обработки информации в сложных динамических средах. Приведено формальное описание логического пространства для интеграции моделей знаний, включающих использование Neuro-Fuzzy систем и мультиагентных технологий. Рассмотрены особенности вычислительного интеллекта, сформулированы модель выбора, определения и аксиомы интеллектуальных Грид-систем. Дается интерпретация Грид-системы как параллельной вычислительной архитектуры.
Ключевые слова: Грид, интеллектуальные системы, нечеткая логика.

Введение

Системный анализ и синтез моделей обработки информации в интеллектуальных Грид-системах определяют методы решения сложных научно-технических проблем, особенно в условиях непрерывного функционирования исследуемых динамических объектов. При этом актуальными представляются проблемы создания эволюционно-самоорганизующихся баз данных и знаний, а также систем адаптивного синтеза информационно-вычислительных конфигураций. В совокупности эти проблемы определяют общую проблему разработки теоретических основ построения самоорганизующихся программно-аппаратных комплексов интеллектуальных Грид-систем [1, 2].

Задачи, решаемые в рамках Грид-систем, по своему характеру оказываются близкими к традиционным задачам искусственного интеллекта. Сравнительный анализ показывает, что можно выделить ряд общих свойств этих задач:

- высокая сложность алгоритмов и большое количество обрабатываемых данных с существенно различной структурой;
- жесткие требования к ресурсам вычислительной системы и ее производительности, необходимость вычислений в режиме реального времени;
- существенная нерегулярность вычислений, операции с множеством разнообразных по своим свойствам объектов и отношений между ними.

При реализации этих задач адаптивный синтез и распараллеливание компьютерных программ осуществляется путем систематического выполнения трансформаций программ, представленных в виде схемных правил некоторого специального вида. Однако многие распараллеливающие преобразования удобнее задавать в процедурном виде. Представленные в виде правил трансформации знания (схемные правила) о методах распараллеливания программ можно накапливать и использовать с помощью специально разработанных интеллектуальных Грид-систем. Схемное правило может быть представлено в виде входного и выходного образов, которые описывают программные конструкции, дополненные условием применимости [3]. Специфика логических правил предусматривает локальные и глобальные (многокомпонентные) преобразования, когда фрагменты программы, связанные семантическими или логическими отношениями, разнесены в тексте. Условия применимости логических правил формулируются на основе базовых предикатов и функций, большая часть которых задается на графах представления программы [4, 5]. В процессе создания интеллектуальной Грид-системы можно разработать различные наборы правил распараллеливания, а также предикатов и функций, характеризующих параметры различных архитектур используемой системы.

Применение многопроцессорных систем при решении сложных задач анализа и интерпретации информации становится эффективным только при наличии такой программной поддержки, которая обеспечила бы максимальную загрузку всех компонентов системы. Это условие относится к любой вычислительной системе параллельной обра-

ботки данных, в том числе и к Грид-системам. Потенциал применения современных быстродействующих компьютерных систем (кластеров, метакластеров и Грид) существенно зависит от организации вычислений и качества используемого программного обеспечения. Таким образом, актуальной становится задача формулировки концепции интеллектуальной Грид-системы и поддерживающего инструментария для автоматизации процесса ее функционирования.

Концептуальный базис реализации Грид-систем как интеллектуальной среды нового поколения

Интеллектуальную Грид-систему можно рассматривать как вычислительную среду нового поколения. Главной особенностью такой интеллектуальной среды является реализация в ней базовых принципов и процедур, обеспечивающих эффективное решение сложных задач анализа и интерпретации информации в режиме реального времени. Усложнение алгоритмов обработки информации в мультипроцессорной вычислительной среде приводит к необходимости широкого использования методов и средств искусственного интеллекта при поиске новых эффективных вычислительных процедур и их параллельной реализации.

Концепция интеллектуальных Грид-систем сформулирована как обобщение и развитие традиционных моделей обработки информации, использующих методы искусственного интеллекта [6, 7]. Разработанная концепция основана на следующем определении интеллектуальной Грид-системы: *интеллектуальной Грид-системой будем называть динамическую интеллектуальную систему, основанную на распределенной базе знаний и формальном представлении взаимосвязанных модельных уровней (структурного, логического), ориентированную на решение типичных задач Грид-технологий.* Структурная схема, определяющая концептуальную модель интеллектуальной Грид-системы, представлена на рис. 1. Эта модель предусматривает использование интеллектуальной Грид-системы не только для решения традиционных больших задач, но и для обеспечения функционирования виртуальных вычислительных комплексов (виртуальный полигон, виртуальная лаборатория и др.).

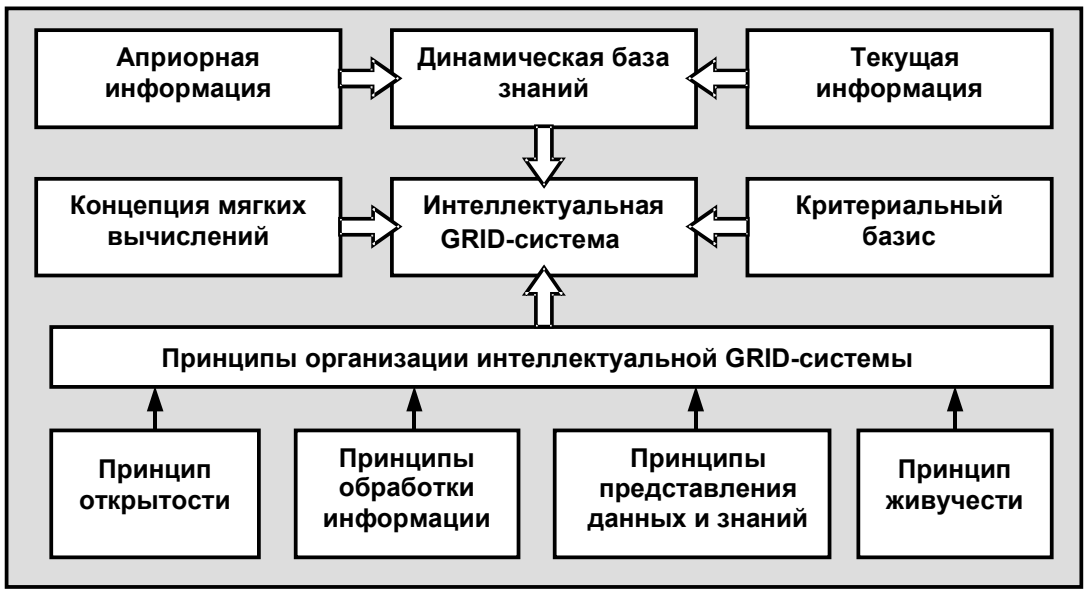


Рис. 1. Концептуальная модель интеллектуальной Грид-системы

При разработке концептуальной модели сформулированы принципы построения и особенности применения интеллектуальных Грид-систем при решении сложных научно-технических задач. Среди этих принципов следует выделить адаптивность, мно-

гопроцессорность, распределенность и глобальность обработки информации, унифицированность и эволюционность представления данных и знаний, активное формирование требований по изменению конфигурации системы, вычислительную мощность и максимальное быстродействие, открытость, непрерывность функционирования и живучесть. Структура интеллектуальных Грид-систем должна обладать возможностью эволюционного наращивания, а при необходимости – и кардинального изменения в условиях непрерывности функционирования [8].

Повышение достоверности оценки и прогноза ситуаций в сложных задачах на основе Грид-систем достигается с использованием нового подхода к обработке информации, основанного на развитии концепции «мягких вычислений» [9]. Этот подход предусматривает использование теоретических принципов (рис. 2), позволяющих рационально организовать вычислительную технологию обработки данных, а также формализовать поток информации при реализации нечеткого логического вывода в мультипроцессорной вычислительной среде [10]. Реализация этих принципов дает возможность повысить эффективность функционирования Грид-системы при непрерывном изменении динамики объекта и внешней среды. Проверка корректности алгоритмов управления и принятия решений осуществляется на основе общих требований к алгоритмическому обеспечению системы. Применительно к параллельным алгоритмам логического управления понятие корректности связано со специфическими свойствами таких алгоритмов – непротиворечивостью, устойчивостью и самосогласованностью. Таким образом, интеллектуальная Грид-система представляет собой сложный многопроцессорный вычислительный комплекс, который можно рассматривать как самоорганизующееся динамическое информационное пространство унифицированного представления данных и знаний.

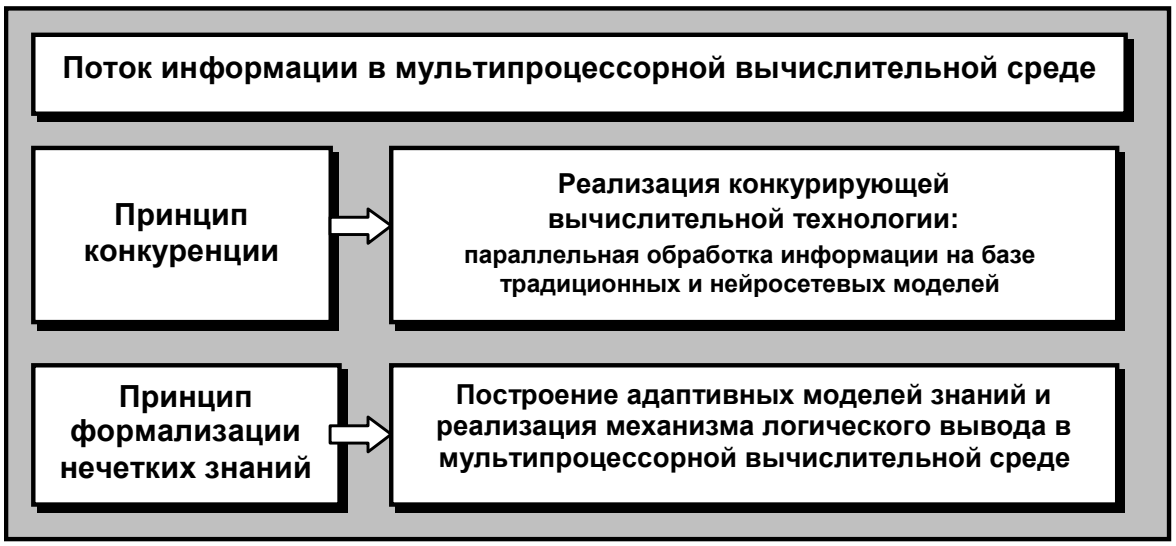


Рис. 2. Принципы обработки информации
в мультипроцессорной вычислительной среде

При формализации знаний интеллектуальной Грид-системы рассматривается понятие о *логическом пространстве* [11]. Такая модель позволяет определить операции над векторным логическим пространством, элементы которого представляются в виде логических векторов, состоящих из логических скаляров. Используя понятия дизъюнкции и конъюнкции логических векторов, можно сформулировать аксиомы логического пространства и представить предикатную интерпретацию логических сетей и линейных логических преобразований как основного средства представления объектов произвольной природы при обработке информации. При формальном описании логического пространства в интеллектуальной Грид-системе разрабатываются методы

формирования альтернативно адаптивных алгоритмов, с помощью которых реализуются принципы обработки информации в мультипроцессорной среде. Алгоритмы, реализующие различные подходы к решению поставленной задачи на основе методов классической математики (стандартные алгоритмы), нечетких и нейросетевых моделей, образуют некоторое множество функционально эквивалентных (функционально близких) алгоритмов. Повышение функциональной эффективности алгоритмов с одинаковым функциональным назначением достигается путем их адаптации к исходным данным. Выбор предпочтительной вычислительной технологии осуществляется путем анализа альтернатив. Для алгоритмов одинаковой функциональной направленности $A_i|X, Y$ с областью определения X и областью значения Y при заданной функции качества $\rho(y)\in R+$ можно представить следующий адаптивный алгоритм [12]:

$$A = A_0\left|X_1,...,X_N \bullet A_1\right|X_1,Y_1 \bullet ... \bullet A_N\left|X_N,Y_N = A_0\right|X_1,...,X_N \bullet \prod_{i=1}^{N} A_i\left|X_iY_i\right.\right., \quad (1)$$

где $(\cdot\bullet)$ – операция композиции, определяющая последовательное выполнение алгоритмов, который разбивает область определения X на подмножества $X_1,\ldots X_N$,

$$\bigcup_{i=1}^{N} X_i = X,\ \forall i \neq j \left|X_i \cap X_j = \varnothing,\ \forall y_i \in Y_i, y_j \in Y_j \right| \rho(y_i) < \rho(y_j), \quad (2)$$

со степенью превосходства i-го алгоритма над j-м на ограниченном множестве X

$$. S\left(A_i, A_j\right)\left|X = \frac{1}{N_{x_i \in X}} \sum_{x_i \in X} \frac{\rho\left(A_j\left|x_i\right.\right) - \rho\left(A_i\left|x_j\right.\right)}{\max\left[\rho\left(A_j\left|x_i\right.\right), \rho\left(A_i\left|x_j\right.\right)\right]}\right.. \quad (3)$$

При создании интеллектуальных Грид-систем возникает проблема разработки обучаемых, адаптивных механизмов логического вывода. Опыт разработки базы знаний сложных систем позволяет рассматривать в качестве одного из эффективных решений использование принципа адаптивного резонанса, широко применяемого в теории нейронных сетей [13] и в бортовых интеллектуальных системах [14]. Формируемые на основе этого принципа свойства базы знаний наиболее четко проявляются в задачах интерпретации ситуаций в условиях неопределенности и неполноты исходной информации (рис. 3).

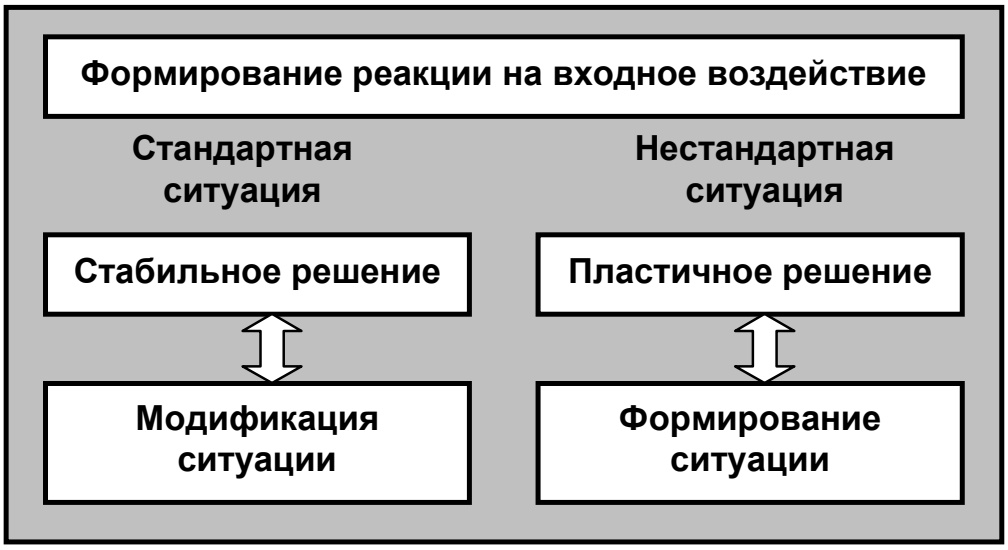


Рис. 3. Адаптивная компонента интеллектуальной Грид-системы

Задачей адаптивной компоненты интеллектуальной Грид-системы является формирование правильной реакции в случае «пластичного» решения о появлении нового образа и «стабильного» решения о совпадении со старым образом. Другой важной особенностью динамической базы знаний, использующей принцип адаптивного резо-

нанса, является *самоадаптация алгоритма поиска* нестандартных ситуаций. Адаптивная компонента использует управляемый алгоритм поиска, основанный на динамической самоорганизации классов-прототипов, соответствующих классам векторов в условной части логического правила, путем наращивания структуры, определяющей «действие» в выходной части логического правила. Среди других формализмов представления знаний в интеллектуальных Грид-системах следует выделить достаточно сложную конструкцию модели знаний, получившую название *гиперправила с мультиактивизаторами* [15]. Такая модель представляет собой логическую конструкцию с детерминированной операционной семантикой. Благодаря естественному совмещению процедурной и непроцедурной компонент гиперправила представляют собой достаточно удобное средство описания локальных функций обработки в едином процессе на основе специального механизма «доски объявлений» и наличия информации в базе данных.

При создании интеллектуальных Грид-систем актуальной является разработка эффективных внутренних структур взаимодействия программных компонентов, реализующих обучение, целенаправленное функционирование, оценку результата и принятие решений. Парадигма, определяющая модель взаимодействия в такой системе, предполагает интеграцию разнообразных компонентов, объединенных функциональными связями. В настоящее время разработано большое количество специализированных программ и инструментальных средств, реализующих Грид-системы, которые можно использовать в качестве базовой платформы при интеграции имеющихся программных решений в единый интеллектуальный комплекс. Это значительно упрощает процесс создания интеллектуальных Грид-систем, которые включают в себя принципиально различные уровни обработки и абстракции информации и позволяют организовать параллельную обработку всех внутренних информационных потоков на основе эффективной, гибкой и легко масштабируемой архитектуры. В такой архитектуре интеграция интеллектуальных модулей может быть осуществлена на основе традиционных методов искусственного интеллекта, включая мультиагентные системы.

Создание интегрированных знаний в интеллектуальных Грид-системах базируется на современных методах инженерии знаний. Разрабатываемые логические модели задают конкретную формальную систему. В рамках такой системы функционирует исследуемый динамический объект (сложная задача), находясь в одном из состояний. Импликативные аксиомы формальной системы определяют логические зависимости между понятиями-соотношениями, расширяя множество фактов и образуя дедуктивный уровень динамической базы знаний. Структура логических формул позволяет описывать зависимости между понятиями и отношениями исследуемой предметной области на основе многоуровневых типов переменных.

Реализация отмеченных принципов формализации знаний дает возможность обеспечить эффективность функционирования интеллектуальных Грид-систем при различной сложности решаемых задач, в том числе и при непрерывном изменении динамики объекта и внешней среды в виртуальных моделирующих комплексах (виртуальный полигон, виртуальная лаборатория). Практическая значимость обработки информационных потоков в реальном времени обусловлена стремлением повысить скорость машинных вычислений путем распараллеливания вычислительных алгоритмов и реализации их на суперкомпьютерных платформах.

При проектировании интеллектуальных Грид-систем существует проблема выбора и оптимизации параметров, которые зависят от средств реализации. На начальных этапах разработки необходимо определить и обосновать структурную схему системы, определить режим ее работы и условия эксплуатации. Для формализации задачи выбора и обоснования средств реализации интеллектуальной Грид-системы вводится обобщенная характеристика, учитывающая различные варианты ее построения, вари-

анты программного обеспечения и используемые вычислительные устройства, реализующие заданные функции системы. В качестве обобщенной характеристики можно рассматривать сложность интеллектуальной Грид-системы, определяющую многопараметрические и разнородные условия, при которых необходимо спроектировать систему. *Функция выбора* β_{DS} модели интеллектуальной Грид-системы связывает параметры алгоритмов обработки информации P_{Ai}, исходные параметры $P_{\text{O}j}$ и средства реализации алгоритмов R_{Ak} [16]:

$$\beta_{DS} = F(P_{A1}, \ldots, P_{Am}, P_{\text{O}1}, \ldots, P_{\text{O}n}, R_{A1}, \ldots, R_{\text{A}q}), \tag{4}$$

где i, j, k – общее число параметров для заданной структуры функции выбора.

Частные параметры в (4) объединяются в динамический диапазон параметров, представленных в различных физических величинах. Переход к безразмерным параметрам осуществляется путем их нормирования, причем множества частных параметров связываются между собой, образуя некоторую обобщенную характеристику:

$$\alpha_{DS} = \sum_{i=1}^{m} \gamma_i P_{A_i}^{*} + \sum_{j=1}^{m} \gamma_j P_{A_j}^{*} + \sum_{i=1}^{m} \gamma_k P_{A_k}^{*}, \tag{5}$$

где γ_i, γ_j, γ_k – весовые коэффициенты, которые устанавливаются исходя из влияния каждого из рассматриваемых параметров на обобщенную характеристику α_{DS} (символ * указывает на безразмерность параметра).

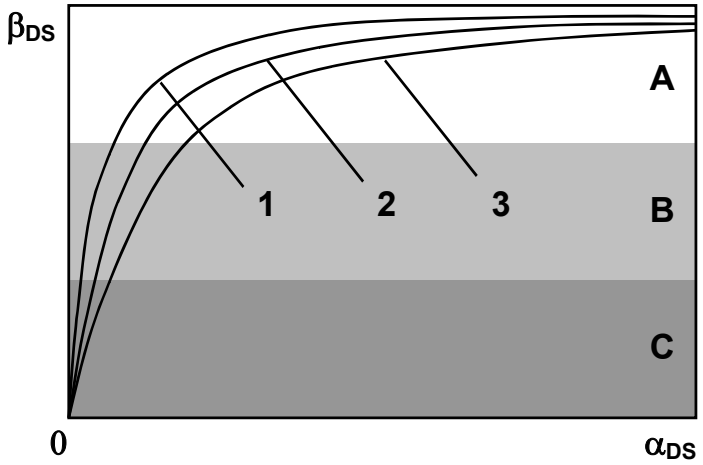


Рис. 4. Функция выбора для различных алгоритмов принятия решений:
1, 2, 3 – сравниваемые алгоритмы; A, B, C – средства их реализации (A – программная логика; B – микропроцессорные средства; C – программные средства)

Выражения (4) и (5) позволяют сформулировать функцию выбора для различных моделей и алгоритмов. В частности, для управляющих и вычислительных алгоритмов, используемых в интеллектуальных Грид-системах, выражение для функции выбора может быть представлено с помощью экспоненциальной функции

$$\beta_{DS} = 1 - \exp\left[-\alpha_{DS} K_A K_T\right] \tag{6}$$

(где K_A – коэффициент, учитывающий особенности алгоритма решения задачи; K_T – коэффициент, учитывающий время выполнения алгоритма), а также в виде зависимости (рис. 4)

$$\beta_{DS} = f(\alpha_{DS}, I_R), \tag{7}$$

позволяющей выделить зоны допустимых значений функции выбора исходя из особенностей рассматриваемого алгоритма обработки варианта решения и интервалов I_R их реализации.

Систематизация и унификация знаний об оптимизации программных комплексов реализуются на основе *модели онтологии S* и онтологической системы $M(S)$, формальное описание которых имеет вид [17]

$$S = <C, R, F>, M(S) = <C_{\text{META}}, \{C_{D\&T}\}, M_{LI}>, \quad (8)$$

где C – конечное множество концептов предметной области, R – конечное множество отношений между концептами; F – конечное множество функций интерпретации (аксиоматизация), заданных на концептах и/или отношениях онтологии; C_{META} – онтология верхнего уровня (метаонтология); $\{C_{D\&T}\}$ – множество предметных онтологий и задач предметной области (предметная онтология); M_{LI} – модель механизма логического вывода (онтология задач), ассоциированного с онтологической системой $M(S)$.

Метаонтология оперирует общими понятиями и отношениями конкретной предметной области. *Предметная онтология* содержит понятия, описывающие конкретную предметную область, семантически значимые отношения и множество интерпретаций понятий и отношений. *Онтология задач* в качестве понятий содержит типы решаемых задач, которые с помощью отношений осуществляют декомпозицию задач на подзадачи.

При формализации предметной области интеллектуальных Грид-систем рассматривают онтологии предметных областей (domain ontology), онтологии классов решаемых задач (task ontology), онтологии приложений (application ontology), онтологии верхнего уровня (top-level ontology). Отдельным типом считаются онтологии методов решения проблем. Возможные пути модификации онтологий в интеллектуальных Грид-системах представлены на рис. 5.

Рис. 5. Пути модификации онтологий в интеллектуальной Грид-системе

Таким образом, модель онтологий в интеллектуальных Грид-системах следует рассматривать как универсальный инструмент междисциплинарных научных исследований, интегрирующий наиболее важные обобщенные функции систем, основанных на знаниях. Эти функции могут быть концептуально реализованы и описаны в терминах проблем искусственного интеллекта. Базы знаний таких систем формируются как концепция онтологий (онтологии в процессе создания интеллектуальных Грид-систем) общей теории баз знаний.

Определения и аксиомы интеллектуальных Грид-систем

Создание интеллектуальной Грид-системы основано на знаниях о предметных областях каждого класса задач (линейная алгебра, нелинейные уравнения и системы, обыкновенные дифференциальные уравнения и системы и др.). Знания представляются в виде функциональных программных модулей, которые описывают логически законченные части машинных алгоритмов исследования и решения задач, а также их семантики. Технологическая схема построения базы знаний каждого программного средства включает анализ предметной области и ее формализацию, построение модели пользователя и модели взаимодействия, компьютерное представление знаний о моделях предметной области (совокупность задач по каждому классу и алгоритмов их исследования с оценками достоверности результатов) в виде фактов и правил. Реализация интеллектуальной Грид-системы предусматривает создание программной технологии решения сложной задачи в рамках информационно-коммуникационной среды, включающей механизмы моделирования, формирования предметной области и ее отображения в компьютерную программу. Разработка технологии реализации такой среды должна поддерживать выполнение сложной задачи территориально распределенными группами исполнителей и автоматически отображать результаты решения в компьютерную систему. Для формирования целостной картины рассматриваемой проблемы функционирования интеллектуальной Грид-системы необходимо сформулировать базовые понятия и построить соответствующие модели.

Инфраструктура, в рамках которой осуществляется решение сложной задачи с помощью интеллектуальной Грид-системы, определяется как *глобальная информационно-коммуникационная среда*, элементами которой являются Интернет, корпоративные сети и др. Решаемая задача P (Problem) представляет собой совокупность составляющих, связанных с постановкой задачи S, требованиями к решению R (Req), результатом решения D (Decision) и руководством пользователя по выполнению результата G (Guide) [18],

$$P = <S, R, D, G>. \quad (9)$$

Свойство структурированности требует декомпозиции решаемой задачи P на подзадачи $P_1, \ldots, P_n$. Знания, необходимые для решения задачи P, представляют собой структурированную и формализованную информацию. Для решения общей задачи вводится такое понятие, как *паттерн знаний* – знания, необходимые и достаточные для решения подзадач $P_1, \ldots, P_n$ (постановка подзадачи, теоретическое решение, практическое приложение и руководство пользователя). В качестве посредника используется мобильная программа, обеспечивающая перемещение паттерна между центром и исполнителями в рамках глобальной информационно-коммуникационной среды. Предметная область выполнения задачи рассматривается как совокупность паттернов, необходимых и достаточных для реализации поставленной задачи. Для формализации знаний интеллектуальной Грид-системы вводятся аксиомы, отражающие эволюцию решения сложных задач в рамках глобальной информационно-коммуникационной среды. Аксиоматическое представление знаний основано на общих понятиях предметной области и специфично для прикладной онтологии [19] (рис. 6).

Для решения логических и вычислительных задач при функционировании интеллектуальной Грид-системы используются различные подходы, методы и модели. Для модели предметной области, представленной в виде онтологии, можно выделить метод предельных обобщений [20]. Этот метод позволяет представить модель предметной области в виде кортежа $<ONT, ASK>$, где ONT – модель онтологии; ASK – модель адекватной системы знаний. Адекватность модели предметной области свидетельствует о том, что модель действительности A ($<ONT, ASK>$) совпадает с множеством моделей

всех ситуаций, входящих в модель предметной области. Так, если рассматривать α как модель любой ситуации действительности, то $\alpha \in A(<ONT, ASK>)$, в противном случае имеем $\alpha^* \notin A(<ONT, ASK>)$. Если ASK_1 эквивалентно ASK_2 по модели действительности ($ASK_1 \sim ASK_2$), то $A(<ONT, ASK_1>) = A(<ONT, ASK_2>)$. Эти утверждения позволяют представить модель знаний в следующем виде:

$$ASK = \{f \mid \mu: ASK_1 \rightarrow ASK_2\}\ P_k, \tag{10}$$

где f – функциональные отображения, реализующие математические модели; μ – различные механизмы реализации отображений; ASK_1 – входные данные задачи (описание информационной среды и задания); ASK_2 – выходные данные задачи; P_k – правила композиции схем задач (правила, описывающие способы объединения локальных задач).

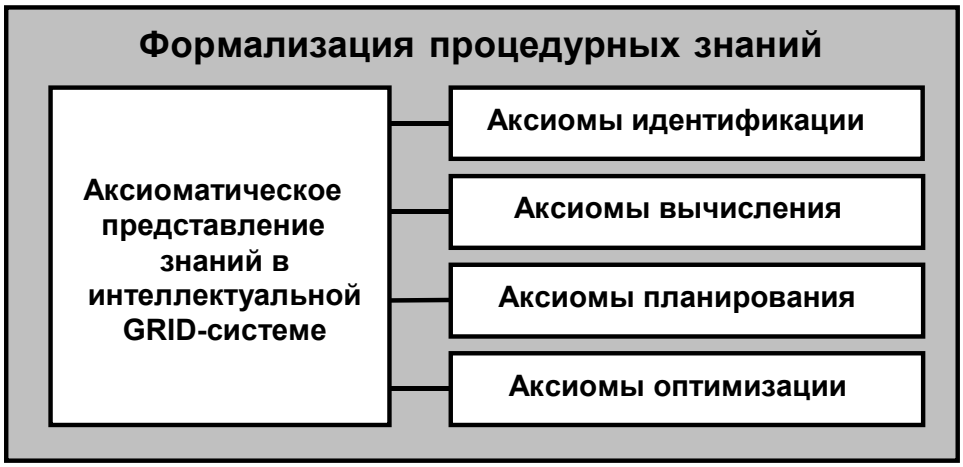


Рис. 6. Формализация и аксиоматическое представление процедурных знаний в интеллектуальной Грид-системе

Выделяя рассматриваемые классы моделей $F_1, \ldots, F_N$, описывающих интерпретацию состояний интеллектуальной Грид-системы, можно представить общую модель знаний в следующем виде:

$$F_1 \cup F_2 \cup \ldots \cup F_N \subseteq ASK. \tag{11}$$

Используя правила композиции P_k, можно построить замыкание множества функциональных отображений $F^+|P_k$ в процессе решения конкретной задачи.

При реализации рассмотренной информационной структуры в интеллектуальной Грид-системе следует учитывать особенности автоматизации программирования и выполнения параллельных программ в мультипроцессорной вычислительной среде. Среди этих особенностей наиболее важной представляется проблема «семантического разрыва» [21], которая характерна для Грид-систем и охватывает многие аспекты решения задач на различных вычислительных средствах. Одним из путей решения проблемы «семантического разрыва» является использование сетевых языков программирования, предназначенных для параллельного программирования в распределенных системах и основанных на идеологии мультипроцессоров с динамической архитектурой [22].

Открытый многоуровневый характер базы знаний интеллектуальных Грид-систем связан с необходимостью использования для формального описания концептуальной модели предметной области семиотического подхода [7]. Такой подход позволяет формально описать систему взаимосвязанных модельных уровней (структурного, логического и т.д.) Грид-систем. Каждому уровню соответствует открытая база знаний о множестве возможных решений рассматриваемых задач. Семиотическая модель открытой формальной системы при реализации интеллектуальной Грид-системы имеет вид

$$F = <B, R, A, C, Q_1, \dots, Q_4>, \quad (12)$$

где B – множество базовых элементов системы; R – множество правил построения синтаксически правильных формул; A – множество аксиом F (подмножество формул из C, которым априорно присваивается статус истинности); C – множество правил вывода (семантические правила, позволяющие получать из A новые синтаксически правильные формулы, которым присваивается статус истинности); Q_1, …, Q_4 – правила изменения для множеств B, R, A, C.

Семиотическая модель является разрешимой, если существует конструктивная процедура, дающая однозначный ответ на вопрос, является ли данный синтаксически корректный элемент семантически верным. Эта процедура задает алгоритм управления в семиотической модели пространства и времени. Основной данного алгоритма является потоковый алгоритм интерпретации недоопределенных динамических вычислительных моделей [23]. Таким образом, главным требованием к моделям функций является способность обеспечить указанный в (12) алгоритм интерпретации.

Проблемы выбора моделей и осуществимости решения задач на основе интеллектуальных Грид-систем

В процессе создания интеллектуальной Грид-системы рассматриваются и сравниваются различные технические решения о структуре системы, механизме ее функционирования, выборе параметров и других элементов. Сравнение решений и выбор оптимального (рационального) решения достигается с помощью моделей, основанных на знаниях (фреймы и продукционные модели). Важной проблемой при создании интеллектуальной Грид-системы является учет показателей осуществимости решения сложных задач в распределенных вычислительных средах.

Знание-ориентированный подход к адаптации алгоритмов в интеллектуальной Грид-системе позволяет реализовать методы формирования альтернативно адаптивных алгоритмов на основе показателей степени превосходства. Алгоритмы, реализующие различные методы решения одной задачи в такой системе, образуют некоторое множество функционально эквивалентных (функционально близких) алгоритмов. В целях повышения функциональной эффективности алгоритмов, реализуемых в интеллектуальной Грид-системе, можно применять к решению сложных задач адаптивные алгоритмы (в терминах [24] – алгоритмы альтернативной адаптации). В качестве базовой интеллектуальной компоненты такой системы рассматривается ее адаптивное структурно-инвариантное ядро, особенностями которого являются знание-ориентированный подход, наличие информационно-расширяемой (адаптивной) метамодельной компоненты, унифицированное представление метаданных о классах и отношениях рассматриваемой предметной области. Другая компонента реализует гибко настраиваемую адаптивную технологию репликации данных между удаленными вычислительными средствами.

При анализе эффективности вычислительной технологии решения сложных задач на основе знание-ориентированного подхода используются показатели осуществимости принятых решений [25]. В практических приложениях удобно использовать графическую интерпретацию решения многокритериальной задачи (метод эталонов) [26], позволяющую избежать использования дополнительной информации и получать решения, близкие к оптимальным.

Достаточно представительным классом моделей при реализации парадигмы интеллектуальных Грид-систем являются Neuro-Fuzzy модели, разработанные в рамках нечеткого и нейросетевого логического базиса. Такие модели могут найти применение не только при формализации знаний и задачах анализа и интерпретации информации, но и в качестве конкурирующих вычислительных технологий при разработке вирту-

альных вычислительных комплексов (виртуальный полигон, виртуальная лаборатория и др.). В больших задачах интерпретации информации можно использовать преимущества мультиагентных моделей [27], определяющих создание инфраструктуры для обеспечения глобальной интеграции информационных и вычислительных ресурсов.

Построение среды многопроцессорного моделирования, сочетающей в себе различные комбинации открытых и трансформируемых сред, осуществляется в виде ситуационной модели игры с динамически меняющимся классом стратегий и управляемым сценарием [28]. Для решения этой задачи предварительно формулируется сценарий, описываемый конечным графом

$$G = (S_C, W_S) \tag{13}$$

(где S_C – стратегии, а P_S – переходы между ними), позволяющий построить алгоритм решения задачи в виде следующих процедур:

– представить S_C в виде

$$S_C = \bigcup_{t_j} S_C^{t_j}, \tag{14}$$

т.е. как объединение стратегий (альтернатив) $S_C^{t_j}$ и моментов управления t_j (j=1, …, N);

– представить P_S в виде $P_S \subseteq S_C \times S_C$, описывающем переходы между стратегиями с помощью отображения множества эффективных стратегий I_1, в виде двух множеств, первое из которых соответствует множеству дуг $\delta_A t_j : A^{i(t_j)} \to P_S$, второе – множеству полезностей этих стратегий, отображаемому в множество дуг, $\delta_U t_j : U^{i(t_j)} \to P_S$.

При анализе процесса принятия решений на сетевых моделях, характерных для интеллектуальных Грид-систем, альтернатива Alt_{W1} определяется некоторыми подмножествами инцидентных позиций $\{p_{j1}\}$ [29, 30] и разрешенных иерархических переходов $\{t_{i1}\}$, моделирующих процессы принятия решений $DR_i(L)$ из множества $\{DR_i{}^{(S)}\}$ для достижения поставленной цели:

$$\begin{aligned} &Alt_{W1} \in \{Alt_W\},\ w \in W; \\ &\{p_{j1}\} \subseteq (p_j),\ j_1 \in J_1,\ J_i \subseteq J; \\ &\{t_{i1}\} \subseteq (t_i),\ i_1 \in I_1,\ I_1 \subseteq I, \\ &DR_i(L)^{(S)} \subseteq DR_i{}^{(S)}. \end{aligned} \tag{15}$$

Проблема выбора решения при интерпретации данных в интеллектуальных Грид-системах состоит в реализации последовательности операций анализа иерархической структуры. Выделенные в этой структуре элементарные подзадачи предварительно анализируются методом иерархий [31]. Реализация метода иерархий связана с выделением приоритетов (весов) признаков в целях выбора наилучших из них с использованием алгебраической теории матриц и экспертных процедур:

$$W\bar{\pi} = \lambda_{\max}\bar{\pi};$$

$$\Pi = \begin{bmatrix} \pi_{11} & \dots & \pi_{1m} \\ \dots & \pi_{ij} & \dots \\ \pi_{n1} & \dots & \pi_{nm} \end{bmatrix} \times \begin{bmatrix} g_1 \\ \dots \\ g_1 \end{bmatrix}, \tag{16}$$

где W – обратно-симметричная матрица значений парных сравнений признаков относительно данного атрибута; $\bar{\pi}$ – нормированный вектор весов признаков; $\lambda_{\max}$ – наибольшее собственное значение матрицы W; Π – результат определения глобальных приоритетов признаков Π_1, …, Π_N; N – число признаков; π_{ij} (i=1, …, n, j=1, …, m) – относительный вес i-го признака по j-му атрибуту; g_j – относительный вес j-го атрибута.

Таким образом, метод [31] связан с проектированием многоуровневой иерархии критериев (факторов) и заданием численных значений мер предпочтения между сопоставляющими сущностями. Такая информация позволяет вычислить вектор приоритетов сущностей на основе собственного вектора обратно-симметричной матрицы, а также глобальные приоритеты объектов на основе приоритетов, определенных по локальным критериям.

Безопасность и отказоустойчивость интеллектуальных Грид-систем

В настоящее время существует достаточное количество решений по организации Грид-систем. Поскольку большинство транзакций в таких системах происходит через открытые коммутационные среды, возникает ряд проблем, связанных с безопасностью Грид [32]. Традиционные меры предусматривают изоляцию систем и защиту ресурсов за счет поддержки правил, ограничивающих действия пользователей, что противоречит главной идее – совместному использованию ресурсов вне зависимости от географических рубежей и границ организаций [33]. Современные подходы к организации механизмов защиты информации в компьютерных системах предусматривают централизацию средств защиты (сертификационные агентства, службы регистрации и авторизации, единые для всей системы), что накладывает значительные ограничения при динамическом масштабировании Грид-систем [34].

Большинство из требований безопасности, которые могут быть использованы при разработке интеллектуальной Грид-системы, входят в стандарт Security Architecture for Open Grid Services, разработанный Open Grid Forum, и на сегодняшний день Globus Toolkit (GT) – широко распространения версия этого стандарта.

Анализ функционирования Грид-систем [35] показал, что наиболее важной проблемой при обеспечении их безопасности является установление показателей доверия к узлам системы. Уровень доверия можно определить непосредственно по опыту работы Грид-системы, а также по данным от узлов, ранее интегрированных в такую систему. В качестве решения этой проблемы в работе [36] предлагается ввести понятия *сервера-хранилища метрик доверия и вектора репутации узлов*, с помощью которого формируются показателя доверия:

а) усредненный показатель доверия к *i*-му узлу

$$T_{Si}^{*} = T_{Si} + \sum_{j=1}^{n} \frac{T_{ji} * T_{Sj}}{n+1}, \tag{17}$$

б) общий показатель доверия

$$T_{j} = \sum_{j=1}^{n} \frac{1 - W_{i}}{n_{j}}, \tag{18}$$

где T_{ji} – показатель доверия *j*-го узла к *i*-му; n_j – количество транзакций с *j*-м узлом; W_i – *i*-е значение вектора репутации узла.

Показатель доверия находится в интервале (0,1), причем 0 соответствует полному отсутствию доверия, а 1 – полному доверию узлу со стороны Грид-системы.

Интерфейс пользователя в интеллектуальных Грид-системах

Организация совместной работы пользователей в интеллектуальной Грид-системе существенно зависит от того, насколько эффективно решены проблемы выбора решения и предусмотрены различные сценарии развития событий при выполнении высокопроизводительных вычислений. Взаимная адаптация пользователей и компьютерных средств обработки информации требует разработки специального инструментария подготовки решений (перечень решаемых задач, наборы вопросов, заданий, под-

сказок, сценариев и др.) как интеллектуального помощника и советчика для лица, принимающего решения. Функциональная схема, характеризующая поддержку оператора в интеллектуальной Грид-системе, представлена на рис. 7. Схема формализует поток информации, определяющий компьютерное решение комплекса взаимосвязанных задач интеллектуальной поддержки оператора при анализе и интерпретации информации в сложных динамических средах.

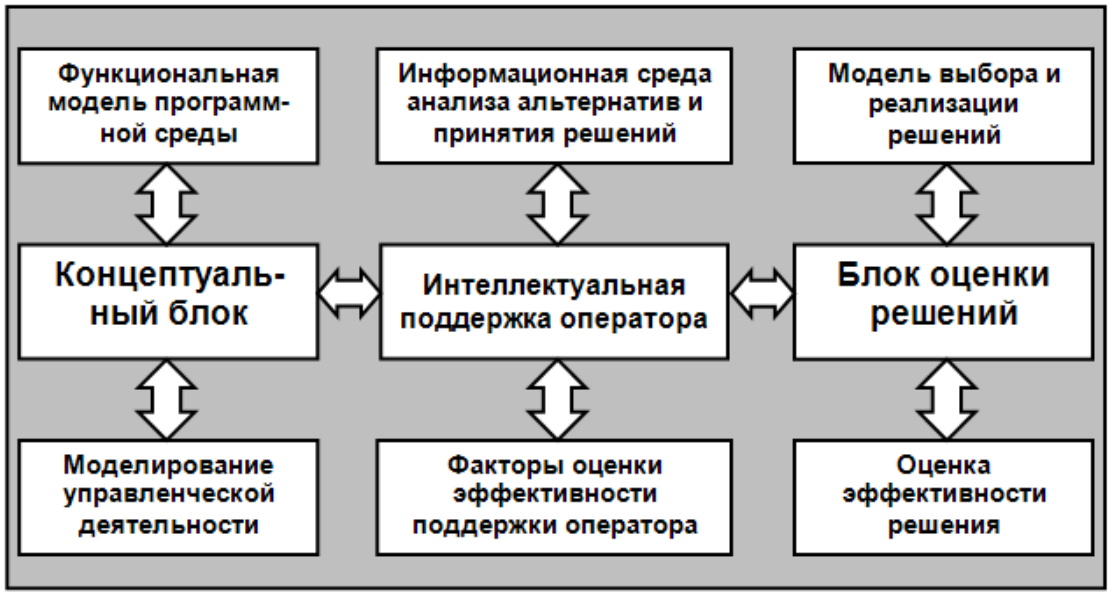


Рис. 7. Поток информации при поддержке оператора в интеллектуальной Грид-системе

Используемые характеристики интерфейсов позволяют оценить эффективность реализации программно-аппаратной структуры интеллектуальной Грид-системы с учетом стоимости, производительности, надежности, а также провести проектирование информационной структуры, обеспечивающей выполнение требований по сбору и передаче потоков данных и результатов их обработки. Большая вычислительная размерность задач, решаемых на основе интеллектуальных Грид-систем, огромные информационные массивы и скорости их обработки, высокая пропускная способность каналов связи требуют решения проблемы *интеграции компонент* системы и человеко-машинного интерфейса.

Надежность работы пользователя (оператора) интеллектуальной Грид-системы можно приближенно оценить, используя формулу

$$P = \prod_{i=1}^{m} P^{ki} = \exp\left[-\sum_{i=1}^{m} (1 - P_i)^{ki} \right] = \exp\left[\sum_{i=1}^{m} Q_i T_i k_i \right], \qquad (19)$$

где P_i – вероятность безошибочного выполнения операции i-го типа; Q_i – интенсивность ошибок i-го типа; k_i – число выполненных операций i-го типа; m – общее число различных типов операций.

Показатели P_i и Q_i могут быть определены по статистическим оценкам, полученным в процессе тестирования Грид-системы:

$$P_i^* = \frac{1}{N_i}(N_i - n_i), \quad Q_i^* = \frac{n_i}{N_i T_i}, \qquad (20)$$

где N_i – общее число выполняемых операций i-го типа; n_i – число ошибок, допущенных оператором при выполнении операции i-го типа; T_i – среднее время выполнения операции i-го типа.

Заключение

Проведенный анализ приложений методов искусственного интеллекта в многопроцессорной вычислительной среде позволил сформулировать концепцию разработки интеллектуальной Грид-системы, функционирующей на основе логико-лингвистических моделей обработки информации и структурированной базы знаний. В рамках этой концепции:

- предложены системные принципы построения моделей обработки информации пространственно-временных полей интеллектуальных Грид-систем;
- сформулированы определения и аксиомы обработки информации в интеллектуальных Грид-системах, позволяющие формализовать сложные процессы обработки знаний при решении больших задач анализа и интерпретации информации;
- определен подход к выбору многоуровневых моделей знаний и осуществимости решения задач на основе интеллектуальных Грид-систем**;**
- приведены алгоритмы контроля режима функционирования, оценки безопасности и риска принимаемых решений в интеллектуальных Грид-системах;
- обоснована реализация интерфейса пользователей при интеграции структурных компонентов в интеллектуальных Грид-системах.

Таким образом, новая парадигма функционирования Грид-систем, реализуемая на базе достижений искусственного интеллекта, позволяет расширить традиционные подходы к обработке информации, дополнить их новыми методами, моделями и алгоритмами поддержки принятия решений с учетом неопределенности и неполноты исходной информации. Развитие внутреннего потенциала теории управления и принятия решений на основе принципа конкуренции позволяет на базе анализа альтернатив при решении сложных задач в информационно-коммуникационной среде выбирать предпочтительную вычислительную технологию с использованием методов классической математики, нечетких и нейросетевых моделей. Расширяя функциональные возможности традиционных Грид-систем и повышая эффективность решения поставленных задач, разработанный подход позволяет обеспечить новое качество – способность предсказания и предвидения критических и аварийных ситуаций, что особенно важно при реализации в сложных вычислительных комплексах обработки информации (виртуальный полигон, лаборатория и др.).

Литература

1. Ковальчук С.В., Иванов С.В., Колыхматов И.И., Бухановский А.В. Особенности проектирования высокопроизводительных комплексов для моделирования сложных систем // Информационно–управляющие системы. – 2008. – № 3. – С. 10–18.
2. Воеводин В.В. Решение больших задач в распределенных вычислительных средах // Автоматика и телемеханика. – 2007. – № 5. – С. 32–45.
3. Partch H. Steinbruggen R. Program transformation systems // ACM Computer Survey. – 1993. – Vol. 15. – № 3. – P. 199–236.
4. Жегуло О.А. Представление знаний о методах распараллеливания в экспертной систем поддержки распараллеливания программ // Искусственный интеллект. – 2001. – № 3. – С. 323–330.
5. Букатов А.А. Разработка средств непроцедурной реализации распараллеливающих преобразований программ // Труды Всероссийской научной конференции «Фундаментальные и прикладные аспекты разработки больших распределенных программных комплексов». – Абрау–Дюрсо, 1998. – С. 109–115.
6. Нечаев Ю.И. Искусственный интеллект: концепции и приложения. – СПб: ГМТУ, 2002.

7. Поспелов Д.А. Ситуационное управление: теория и практика. – М.: Наука, 1986.– 288 с.
8. Варламов О.О. Эволюционные базы данных и знаний для адаптивного синтеза интеллектуальных систем. – М.: Радио и связь, 2002. – 288 с.
9. Zadeh L. Fuzzy logic, neural networks and soft computing // Commutation on the ASM. – 1994. – Vol. 37. – № 3. – P. 77–84.
10. Нечаев Ю.И. Математическое моделирование в бортовых интеллектуальных системах реального времени // Труды 5–й Всероссийской научно–технической конференции «Нейроинформатика–2003». Лекции по нейроинформатике. Часть 2.– С. 119–179.
11. Четвериков Г.Г., Вечирская И.Д. Формальное описание логического пространства // Искусственный интеллект, Донецк. – 2003. – № 3. – С. 781–789.
12. Ширяев В.И. Управление динамическими системами в условиях неопределенности // Искусственный интеллект, Донецк. – 2003. – № 3. – С. 224–231.
13. Уоссермен Ф. Нейрокомпьютерная техника. Теория и практика. – М.: Мир, 1992. – 240 с.
14. Нечаев Ю.И., Петров О.Н. Система поддержки принятия решений на основе нечетких знаний о динамике судна в экстремальных ситуациях // Сборник докладов Международной конференции по мягким вычислениям и измерениям SCM–2005. Санкт–Петербург, 2005. – Т. 2. – С. 66–69.
15. Варламов О.О. Разработка адаптивного механизма логического вывода на эволюционной интерактивной сети гиперправил с мультиактивизаторами, управляемой потоком данных // Искусственный интеллект, Донецк. – 2002. – № 3.– С. 363–370.
16. Литвинская О.С. Функция выбора решения задачи обоснования средства реализации алгоритмов в информационных системах // Сборник докладов 4–й Всероссийской научно-технической конференции «Управление и информационные технологии УИТ–2006». – Санкт–Петербург, 2006. – С. 162–164.
17. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. – М.: Питер, 2000. – 384 с.
18. Виссия Х., Краснопрошин В.В.. Вальвачев А.Н. Технология выполнения IT-проектов коллективами распределенных исполнителей // Искусственный интеллект, Донецк. – 2008. – № 3.– С. 63–69.
19. Девятков В.В. Онтологии в проектировании систем // Сборник докладов Международной конференции по мягким вычислениям и измерениям SCM–99. – Санкт–Петербург, 1999. – Т. 2. – С. 137–140.
20. Прокопчук Ю.А, Метод предельных обобщений и эффективный принцип работы вычислительного интеллекта // Искусственный интеллект, Донецк. – 2008. – № 3. – С. 727–736.
21. Майерс Г. Архитектура современных ЭВМ. – М.: Мир, 1985. – 364 с.
22. Царев И.В. ЯРД – язык сетевого программирования в распределенных вычислительных системах с динамической архитектурой // Искусственный интеллект, Донецк. – 2008. – № 3. – С. 761–770.
23. Нариньяни А.С. Недоопределенность в системах представления и обработки знаний // Изв. АН ССР. Техническая кибернетика. – 1986. – №5. – С. 3–28.
24. Растригин Л.А. Адаптация сложных систем. – Рига: Зинатие, 1981. – 375 с.
25. Хорошевский В.Г. Архитектура вычислительных систем. – М.: МГТУ им. Н.Э. Баумана, 2005. – 512 с.
26. Groppen V.O. Smart computing principles. Models and algorithms // Proceedings of the 5-th conference on evolutionary methods of design, optimization and control with applications to industrial and social problems. – Spain, Barselona, 2003. – P. 133–134.

27. Городецкий В.И. Многоагентные системы: современное состояние исследований и перспективы применения // Новости искусственного интеллекта. – 1996. – № 1. – С. 44–59.
28. Вовк С.П. Разработка технологии нечеткого моделирования ситуаций принятия решений в частично формализуемых средах // Программные продукты и системы. – 2004. – № 3. – С. 16–22.
29. Шаталов А.С. Отображение процессов управления в пространство состояний. – М.: Энергоатомиздат, 1986. – 256 с.
30. Миркин Б.Г. Проблема группового выбора. – М.: Наука, 1974. – 256 с.
31. Saaty T.L. Exploring the interface between hierarchies, multiple objectives and fuzzy sets // Fuzzy sets and systems. – 1978. – P. 67–68.
32. The security architecture for open GRID services / Nagaratnam N., Janson P., Dayka J. and all. – IBM Corporation, 2003. – 74 p.
33. Foster I., Kesselman C., Nick J., Tuecke S. The physiology of the GRID: open GRID services architecture for distributed systems integration. – Springer Verlag, 2002. – 31 p.
34. Charabarty A. GRID computing security // International workshop Advanced computing and commutations ADCOM. – Ahmedabad (INDIA), 2004. – 12 p.
35. Security for GRID services / Welch V., Siebenlist F., Foster I and al. // 12–th International sy on high performance distributed computing (HPDC–12). – IEEE Press, 2003.
36. Мухин В.Е. Средства защиты GRID–систем на основе дифференцирования уровня доверия к узлам системы // Искусственный интеллект, Донецк. – 2003. – № 3.– С. 187–196.

Нечаев Юрий Иванович – Санкт-Петербургский морской технический университет, д.т.н., профессор, petr_oleg@mail.ru

Бухановский Александр Валерьевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, д.т.н., профессор, boukhanovsky@mail.ifmo.ru

Васильев Владимир Николаевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, ректор, д.т.н., профессор, vasilev@mail.ifmo.ru

УДК 004.4'22, 681.324

ИНСТРУМЕНТАЛЬНАЯ ОБОЛОЧКА ПРОЕКТИРОВАНИЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ПРИЛОЖЕНИЙ В СРЕДЕ ГРИД. ЧАСТЬ I: БАЗОВЫЕ ПОЛОЖЕНИЯ

А.В. Ларченко, А.В. Дунаев, А.В. Бухановский

Рассматриваются особенности разработки высокопроизводительных приложений на основе прикладных Грид-сервисов. Предложен способ формализации процесса проектирования в виде графовой структуры потоков заданий (workflow) с последовательно уточняемыми характеристиками. Обоснована целесообразность поддержки принятия решений разработчика в среде Грид посредством использования интеллектуальной системы и предложена ее принципиальная архитектура.

Ключевые слова: Грид, проектирование, интеллектуальная система, высокопроизводительные вычисления.

Введение

На современном этапе развития технологий распределенных вычислений и систем под Грид подразумевается географически распределенная, согласованная, открытая и стандартизованная среда разделения вычислительных и информационных ресурсов [1]. Это понятие обобщает частные определения, независимо сформулированные I. Foster и C. Kesselman [2] и M. Livny [3], которые, по мере внедрения технологий Грид в научно-исследовательскую и производственную практику, расширяются и модифицируются. В настоящее время наиболее привычной является инфраструктурная интерпретация Грид как среды, предоставляющей совокупность высокопроизводительных вычислительных ресурсов для выполнения независимых задач различных пользователей [4]. Однако в рамках концепции [1] допустим альтернативный подход, который рассматривает Грид как специфическую («мягкую», soft) среду параллельных вычислений, наряду с более традиционными (например, кластерными, SMP, гибридными, P2P) параллельными архитектурами.

Проблема эффективной организации параллельных вычислений в Грид связана с необходимостью объединения и синхронизации большого количества вычислительных систем для решения одной задачи. Решение этой проблемы связано с учетом ряда специфических факторов, в общем случае отрицательно влияющих на параллельную производительность. К ним относятся неоднородность вычислительных ресурсов и стохастическая изменчивость параметров коммуникационных сетей и вычислительных систем, характеризуемая нестационарным поведением во времени [5]. Потому принципы проектирования эффективных параллельных вычислительных приложений в Грид существенно отличаются от традиционных подходов, характерных, например, для кластерных систем [6]. Как следствие, «ручной» процесс разработки таких приложений является весьма трудоемким и требует высокой квалификации разработчиков. Это ограничение может быть отчасти устранено за счет развития специализированного программного инструментария (технологических сред и оболочек проектирования), в частности, инструментальных оболочек семейства PEG (Parallel Execution on the Grid). В данной статье представлен способ формализации процесса проектирования композитных приложений в Грид и обоснована концептуальная схема оболочки PEG как интеллектуальной системы поддержки принятия решений.

Процесс проектирования вычислительных приложений в среде Грид

Целью проектирования является формализация и обоснование внутренней структуры приложения (элементов и связей между ними), обеспечивающей оптимальные значения характеристик выполнения с точки зрения пользователя при заданных огра-

ничениях. Применительно к задачам высокопроизводительных вычислений такими характеристиками являются общее время выполнения, а также связанные с ним параллельное ускорение и эффективность. Эти величины совокупно зависят от особенностей применяемых параллельных алгоритмов, характеристик исходных данных и специфики параллельной вычислительной архитектуры [7].

Современный подход к задаче проектирования технических объектов и сооружений традиционно формулируется в терминах задачи многокритериальной оптимизации. Она заключается в определении оптимальных характеристик системы с точки зрения удовлетворения различным, зачастую противоречивым или взаимозависимым требованиям. В области проектирования программного обеспечения в настоящее время такая постановка не является традиционной; основным инструментом проектирования остается использование полуинтуитивных подходов, основанных на использовании шаблонов [14], проверочных списков [15] или инструментария специальных парадигм программирования [16]. Это связано как с многообразием форм требований к программному обеспечению (пользовательские, функциональные, к надежности, к безопасности и пр.), так и с достаточно слабым уровнем формализации многих из них. Однако для такой области, как высокопроизводительные вычисления, ведущей характеристикой является собственно производительность (выражаемая через время выполнения задачи, параллельное ускорение, реактивность или эффективность приложения), которая находится в однозначном соответствии с характеристиками данных, спецификой алгоритма и особенностями вычислительной среды.

Отличительной особенностью процесса проектирования приложений в Грид является изначальная ориентация на использование предметно-ориентированных вычислительных компонентов – прикладных Грид-сервисов (ПГС). Под ПГС подразумевается прикладная программа, зарегистрированная в Грид посредством ее оборачивания (wrapping) соответствующей сервисной оболочкой и размещения на вычислительном узле, которая позволяет использовать ПГС другим системным и прикладным сервисам.

Организация процесса проектирования в значительной степени зависит от способа формального описания композитного приложения. Для описания приложений в Грид допустимо использовать нотацию потоков задач, или *workflow* (WF) [8]. WF представляет собой набор блоков, соответствующих основным операциям, реализуемым ПГС, и взаимосвязей между ними, определяющих обмен данными. В отличие от блок-схем алгоритмов, в WF очередность операций задана неявно и подчиняется принципу передачи управления по наличию исходных данных для блоков. Это качество является принципиальным для описания выполнения вычислительных задач в Грид в силу неконтролируемой изменчивости производительности вычислительных узлов и пропускной способности коммуникационной среды.

На рис. 1 изображен процесс проектирования приложений в среде Грид в виде последовательного уточнения заданного пользователем WF. На начальном этапе пользователь описывает постановку задачи в терминах своей предметной области, создавая семантическое описание композитного приложения, или *мета-workflow* (MWF), без указания условий ее выполнения в Грид. Второй этап проектирования состоит в переходе к описанию в терминах *абстрактного workflow* (AWF). AWF отличается от MWF тем, что неопределенное описание сервисов в терминах предметной области заменено указаниями на *метасервисы*. Метасервисы являются обобщением ПГС, объединяя в себе свойства прикладных программ нескольких ПГС без учета особенностей реализации на конкретных вычислительных узлах. Метасервисы подбираются посредством семантического поиска в Грид на основе пользовательских критериев (требований к функциональным характеристикам, ресурсоемкости, точности и пр.) Третьим этапом проектирования является конкретизация WF до уровня *конкретного workflow* (CWF) с целью формирования оптимальной по производительности архитектуры ком-

позитного приложения. При этом базовыми механизмами формирования оптимальной параллельной программной архитектуры такого приложения являются выбор конкретных сервисов, выбор эффективных способов распараллеливания по каждому из сервисов (внутренний параллелизм), выбор эффективных способов распараллеливания на уровне композитного приложения, статическая и динамическая балансировки, миграция задач между узлами в Грид и пр. Описание приложения в форме CWF эквивалентно алгоритмической записи, что позволяет автоматически генерировать сценарий исполнения, который может быть использован непосредственно для запуска в Грид.

Практическая реализация процесса проектирования (рис. 1) ограничена лишь достаточно простыми приложениями, состоящими из небольшого количества ПГС, объединенных в WF простой (преимущественно линейной) структуры. Увеличение количества ПГС в условиях неопределенности, характерной для параметров среды Грид, приводит к неполиномиальному усложнению переходов MWF–AWF и AWF–CWF. Стохастичность параметров Грид требует применения специфических методов проектирования, предназначенных для работы в условиях неопределенности входных данных и слабой формализации постановки задачи.

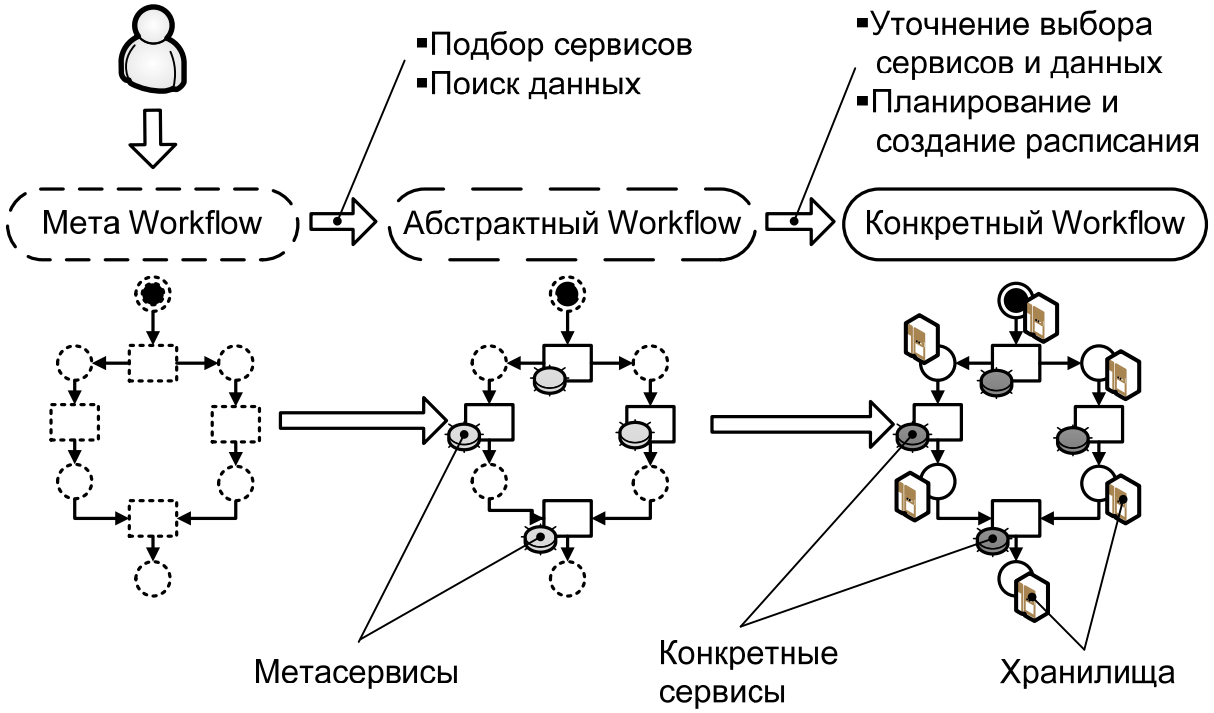


Рис. 1. Процесс проектирования вычислительных приложений в среде Грид

Проектирование приложений в Грид как задача искусственного интеллекта

В общем случае преимущества выбора механизмов (или их комбинации) формирования оптимальной архитектуры неочевидны в силу относительно слабой формализации требований и ограничений, изменчивости внешней среды (в данном случае – инфраструктуры Грид) и неопределенности характеристик задачи. Как следствие, решение такой задачи целесообразно осуществлять не формальными методами, а средствами искусственного интеллекта [9]. В соответствии с критериями из [9], это обусловлено тем, что:

– не представляется возможным разработать количественную модель производительности приложений, применимую к различным классам параллельных вычислительных архитектур и видам приложений. Предлагаемые решения носят частный характер; условия их применимости для других архитектур дискуссионны [10];

– создание моделей производительности ПГС является прерогативой их разработчиков, которые в данном случае выступают как эксперты;
– решение задачи проектирования, по сути, сводится к проблеме прогнозирования, или предсказания характеристик разрабатываемого приложения на этапе описания и обоснования его структуры;
– данные, характеризующие условия эксплуатации композитного приложения, зашумлены за счет стохастической изменчивости характеристик среды Грид;
– задачу проектирования в рамках процесса, изображенного на рис. 1, допустимо решать методом формальных рассуждений – путем выбора наиболее адекватного образца из ограниченного набора вариантов, удовлетворяющих формальным требованиям (пользовательским или системным) как на уровне AWF, так и на уровне CWF.

Как следствие, программный инструментарий реализации процесса проектирования в заданной предметной области должен быть реализован в форме интеллектуальной системы поддержки принятия решений (ИСППР). Понятие ИС связано со способностью компьютерной программы решать сложные, трудно формализуемые задачи, способностью к обучению, обобщению и аналогиям с возможностью взаимодействия с *внешней средой*. Применительно к задаче проектирования композитных приложений в качестве внешней среды выступает сама среда Грид. В табл. 1 отмечены основные черты Грид, которые, в соответствии с критериями [11], обосновывают целесообразность применения ИСППР реального времени.

Таблица 1. Особенности среды Грид, обосновывающие применение ИСППР

Взаимодействие управляющих систем с внешним миром	Взаимодействие ИСППР с Грид посредством мониторинга и управления выполнением
Совершенствование собственного поведения в ходе работы	Оценка опыта предыдущих запусков для адаптации к изменениям Грид
Механизмы прогноза изменений собственного поведения и поведения внешнего мира	Прогнозирование производительности с учетом стохастического характера Грид
Наличие многоуровневой иерархической структуры	Наличие иерархического параллелизма (multicore, GPGPU, MPI)
Сохраняемость функционирования при отказе некоторых подсистем	Адаптивный ответ на сбои выполнения (миграция, перепланирование)

Отличительной чертой ИСППР является наличие *базы знаний* (БЗ), на интерпретации которой основан принцип действия системы. БЗ, определяющая процесс проектирования приложений в Грид, является распределенной; ее фрагменты (локальные БЗ, или ЛБЗ) ассоциированы с ПГС. БЗ содержит две категории знаний: предметно-ориентированные (декларативные) описания сервисов, зарегистрированных в Грид, и знания об их параллельной производительности. Последние включают в себя метазнания (информацию о структуре закономерностей, условия применимости, решающие знания), а также собственно предметные знания, или факты, в форме конкретных моделей производительности, таблиц и аппроксимаций, связывающих время выполнения задачи с характеристиками входных и выходных данных и параметрами вычислительной среды. Приобретение знаний (как процесс их передачи от эксперта или из других источников в БЗ) выполняется как непосредственно при разработке проблемно-ориентированных модулей, так и в процессе самообучения системы по результатам работы индуктивных программ имитационного моделирования вычислительных процессов в Грид [12].

Помимо этого, система имеет обратную связь с пользователем и поддерживает при необходимости диалог с ним на языке его предметной области на всех этапах создания описания, его интерпретации и выполнения. Это необходимо в следующих случаях:

- при недостатке знаний для построения процесса вычислений по данным, ограничениям и критериям, которые указал пользователь. В этом случае система должна определить, каких именно знаний не хватает, и предложить варианты их восполнения, если таковые возможны;
- в случае необходимости участия пользователя для осуществления явного выбора среди предложенных системой альтернатив. Это может объясняться желанием пользователя в некоторых случаях иметь более полный контроль над работой интеллектуальной системы, корректируя процесс вывода в тех местах, в которых он хочет самостоятельно производить выбор. Примером может служить предложение системой ранжированного по некоторым параметрам (время прогнозируемого исполнения, точность и пр.) списка альтернативных расписаний для «ручного» выбора;
- в случае возникновения критических ситуаций, когда требуется непосредственное участие пользователя – при сбоях в Грид, при значительном отставании от составленного прогноза и пр.
- в других особых случаях, когда избежать устранения пользователя от участия невозможно.

Принцип действия интеллектуальной системы поддержки принятия решений разработчика высокопроизводительных приложений в среде Грид

На рис. 2 приведена концептуальная схема, иллюстрирующая принцип действия ИСППР разработчика высокопроизводительных приложений в Грид. Человеко-компьютерное взаимодействие в ИСППР осуществляется через оболочку визуального проектирования, в которой пользователь описывает композитное приложение в форме MWF, используя специально разработанную графическую нотацию. На первом этапе работы ИСППР выполняется интерпретация MWF, на основании которой осуществляется семантический поиск сервисов, зарегистрированных в Грид, в рамках заданной предметной области.

Таким образом, создается набор активных ПГС, установленных на конкретных вычислительных системах, входящих в Грид, и готовых к использованию. Результатом является набор AWF, построенных с использованием метасервисов, формально подходящих под пользовательские критерии. Второй этап работы ИСППР связан с осуществлением логического вывода, который представляет собой создание расписания, оптимального с точки зрения критериев пользователя. Для этого используются экспертные знания о производительности ПГС, предоставляемые их разработчиками при регистрации сервиса в Грид, например, в форме параметрической модели времени выполнения в зависимости от параметров задачи и характеристик вычислительной системы. Используя данные мониторинга текущего состояния Грид и известные характеристики пользовательской задачи, на основе этих знаний оценивается время выполнения конкретных ПГС, т.е. формируется набор активных фактов, описывающих различные альтернативы. Эти факты используются для построения системы конкурирующих расписаний; при этом конкуренция обусловлена как различными эвристиками, применяемыми для ускорения процесса построения расписания, так и разнообразием AWF, порождаемых одним MWF в силу применения различных способов распределения данных, балансировки и других механизмов управления параллельной производительностью. Как следствие, выбор оптимального расписания требует выполнения процедуры сопоставления альтернатив путем их ранжирования в условиях неопределенности входных данных,

стохастической изменчивости параметров Грид и экспертного характера знаний о производительности ПГС. Это ранжирование выполняется на основании сопоставления распределений времени работы для каждого из расписаний с учетом ошибки мискласcификации. Процесс сопоставления проиллюстрирован в [13].

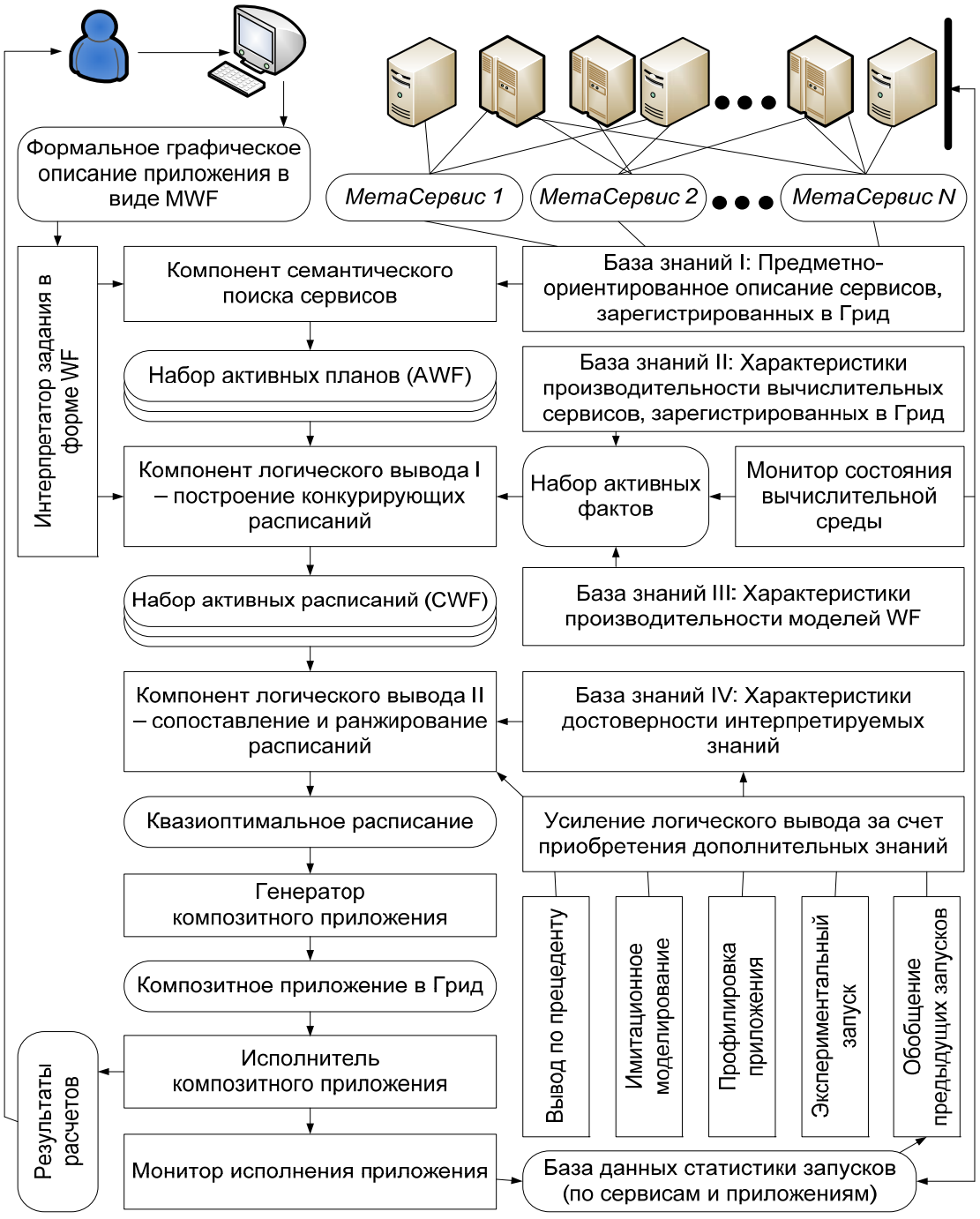


Рис. 2. Концептуальная схема ИСППР разработчика высокопроизводительных приложений в среде Грид

В результате пользователю предлагается один или несколько (в том случае, если достоверно нельзя отдать приоритет ни одному) CWF. Каждый CWF является полным описанием параллельного выполнения задачи в Грид. Таким образом, пользователь может непосредственно трансформировать CWF в соответствующие сценарии и отправить на выполнение в Грид. Результатом выполнения сценария является набор данных, который отправляется в указанное хранилище Грид или на машину пользователя.

Существенным отличием данной ИСППР от традиционных экспертных систем является наличие адаптивной компоненты, которая позволяет усиливать процедуру логического вывода системы за счет приобретения дополнительных знаний в ходе ее функционирования. Дополнительные знания могут приобретаться косвенным путем за счет использования следующих возможностей:

- *поиск прецедентов* – система осуществляет поиск знаний о предыдущих запусках и на их основании делает вывод;
- *имитационное моделирование* – система может произвести моделирование процесса вычислений с помощью симулятора Грид и основывать свой вывод на его результатах;
- *тестирование и профилирование* – производятся отдельные запуски программных модулей сервисов с участием эксперта, который по оценкам этих запусков самостоятельно пополнит базу знаний.

При этом динамически формируется и обновляется база данных статистики предыдущих запусков; ее анализ позволяет не только уточнить интерпретируемую информацию (например, коэффициенты моделей производительности), но и судить о качестве знаний, предоставляемых экспертами, регулируя соответствующие коэффициенты достоверности. Таким образом, предлагаемая ИСППР позволяет автоматизировать процесс разработки параллельных приложений в Грид даже в условиях существенной неопределенности, когда разработчик не сообщает достоверной информации о производительности своих ПГС.

Заключение

Проектирование высокопроизводительных приложений в Грид является сложным и ресурсоемким процессом, требующим высокой квалификации разработчика и специализированного программного инструментария. Неполнота исходной информации о характеристиках ПГС и стохастическая изменчивость характеристик среды Грид оправдывают решение задачи проектирования средствами искусственного интеллекта. Предложенная концепция ИСПРР реального времени позволяет описать задачу проектирования в рамках последовательности MWF–AWF–CWF, автоматизируя процесс разработки композитных приложений на основе ПГС и гибко управляя их параллельной производительностью, исходя из конкретных условий эксплуатации инфраструктуры Грид. Особенности программной реализации и примеры функционирования ИСППР приведены в работах [12, 13].

Работа выполнена в рамках НИР 2007-4-1.4-20-01-025 «Разработка инструментальной оболочки проектирования высокопроизводительных приложений для Грид-архитектур в целях создания прикладных сервисов компьютерного моделирования и обработки данных».

Литература

1. Stockinger H. Defining the Grid – a snapshot of the current view // Journal of Supercomputing. – Springer Science+Business Media. – 2007.
2. Foster I., Kesselman C. The Grid: Blueprint for a New Computing Infrastructure. – Morgan–Kaufman, 1999.
3. Thain D., Tannenbaum T., and Livny M. Condor and the Grid // Berman F. (Editor), Fox G. (Editor), Hey J.G.A. (Editor). Grid Computing: Making The Global Infrastructure a Reality. – John Wiley, 2003.

4. Дмитриев В.Ю. Сети высокопроизводительных вычислений – новое направление ИТ // JetInfo: Информационный бюллетень. Выпуск 175. – М.: Компания «Инфосистемы Джет», 2007. – 28 с.
5. Ларченко А.В., Дунаев А.В., Бухановский А.В. Инструментальная оболочка проектирования высокопроизводительных прикладных Грид-сервисов компьютерного моделирования и обработки данных // Труды Всероссийской научной конференции «Научный сервис в сети Интернет». – М.: Издательство МГУ, 2008. – С. 163.
6. Гергель В.П. Теория и практика параллельных вычислений: Учебное пособие. – М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2007. – 423 с.
7. Ковальчук С.В., Иванов С.В., Колыхматов И.И., Бухановский А.В. Особенности проектирования высокопроизводительных программных комплексов для моделирования сложных систем // Информационно-управляющие системы. – 2008. – № 3. – С. 10–18.
8. Blythe J., Deelman E. et al. The Role of Planning in Grid Computing. – Режим доступа: http://www.isi.edu/~gil/papers/icaps03–submission.pdf, свободный.
9. Нечаев Ю.И. Искусственный интеллект: концепции и приложения. – СПб: Изд. СПбГМТУ, 2000. – 294 с.
10. Foster I. Designing and Building Parallel Programs: Concepts and Tools for Parallel. – Addison Wesley, 1995. – 430 p.
11. Аверкин А.Н., Гаазе–Рапопорт М.Г., Поспелов Д.А. Толковый словарь по искусственному интеллекту. – М.: Радио и связь, 1992.
12. Дунаев А.В., Ларченко А.В., Бухановский А.В. Инструментальная оболочка проектирования высокопроизводительных приложений в Грид. Часть III: Приобретение и формализация знаний // Научно-технический вестник СПбГУ ИТМО. – 2008. – Вып. 54. – С. 46–55.
13. Ларченко А.В., Дунаев А.В, Бухановский А.В. Инструментальная оболочка проектирования высокопроизводительных приложений в Грид. Часть II: Архитектура, реализация и применение // Научно-технический вестник СПбГУ ИТМО. – 2008. – Вып. 54. – С. 37–45.
14. Соммервилл И. Инженерия программного обеспечения. 6–е изд. / Пер. с англ.– М.: Издательский дом «Вильямс», 2002. – 624 с.
15. Foster I. What is the Grid. A three point checklist // GridToday. – July 22, 2002 – Vol. 1. – № 6. – Режим доступа: http://www.gridtoday.com/02/0722/100136.html, свободный.
16. Шалыто А.А. Логическое управление. Методы аппаратной и программной реализации алгоритмов. – СПб: Наука, 2000. – 780 с.

Ларченко Алексей Викторович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, м.н.с., aleksey.larchenko@gmail.com

Дунаев Антон Валентинович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, м.н.с., anton.dunaev@gmail.com

Бухановский Александр Валерьевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, д.т.н., профессор, boukhanovsky@mail.ifmo.ru

УДК 004.4’22, 681.324

ИНСТРУМЕНТАЛЬНАЯ ОБОЛОЧКА ПРОЕКТИРОВАНИЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ПРИЛОЖЕНИЙ В СРЕДЕ ГРИД. ЧАСТЬ II: АРХИТЕКТУРА, РЕАЛИЗАЦИЯ И ПРИМЕНЕНИЕ

А.В. Ларченко, А.В. Дунаев, А.В. Бухановский

Описывается архитектура инструментальных оболочек семейства PEG, обеспечивающих поддержку принятия решений разработчика композитных приложений в Грид. Обсуждаются принципы их функционирования, особенности программной реализации и использования. Работоспособность предложенных решений иллюстрируется на примере композитного приложения расчета климатических спектров морского волнения.

Ключевые слова: Грид, проектирование, интеллектуальная система, высокопроизводительные вычисления.

Введение

Математическое и программное обеспечение Грид-вычислений отличается существенным разнообразием, обусловленным независимым развитием технологий распределенных вычислений в рамках различных направлений [1]. С точки зрения процесса разработки приложений оно может быть разделено на следующие иерархические группы:

- промежуточное программное обеспечение Грид, или middleware, обеспечивающее объединение разрозненных ресурсов в единое вычислительное и информационное пространство на основе стандартизированных технологий взаимодействия. К этому классу относятся широко известные пакеты Globus [2], UNICORE [3], gLite [4] и др.;
- инструментальные среды разработки приложений в Грид, предоставляющие разработчику набор типовых компонентов, использующих функциональности middleware. К ним относятся такие разработки, как Intel GPE [5] и SAGA [6];
- системы визуального прототипирования приложений в Грид, обычно использующие нотацию потоков заданий, или workflow (WF). Этот класс включает в себя решения Taverna [7], Triana [8] и пр. Такие системы упрощают процесс создания программного кода за счет интерпретации визуального описания, которое формируется и обосновывается разработчиком на основе собственного опыта и навыков.

Отдельную группу программного обеспечения Грид-вычислений представляют интеллектуальные системы поддержки принятия решений (ИСППР) разработчиков, к которым относятся инструментальные оболочки семейства PEG (Parallel Execution on the Grid) [9, 10], K-WF Grid [13], Pegasus [13]. В отличие от систем визуального прототипирования, ИСППР обеспечивают автоматизацию самого процесса проектирования, формируя оптимальную по производительности структуру композитного приложения на основе пользовательского описания в терминах предметной области. Пользователю предоставляется возможность создать структуру параллельного композитного приложения (выполнение операций декомпозиции, связывания, агломерации) посредством визуального проектирования с использованием графического языка. Системой выполняется мониторинг среды выполнения с целью выяснения различных ее характеристик, таких как пропускная способность сети, вычислительные мощности отдельных вычислительных узлов и их доступность, права на исполнение приложений и пр. На основании этих данных производится моделирование процесса исполнения приложения с целью прогнозирования его производительности на этапе проектирования и построения оптимального расписания его выполнения. По расписанию выполняется автоматическая генерация композитного приложения, а также сопутствующих файлов, необходимых для выполнения заданного приложения в Грид.

В данной статье описывается архитектура инструментальных оболочек семейства PEG, назначение и принцип действия основных функциональных компонентов, а также иллюстративный пример использования оболочки для разработки композитного приложения расчета климатических спектров морского волнения.

Эволюция архитектуры инструментальных оболочек семейства PEG

Семейство инструментальных оболочек PEG поддержки принятия решений разработчика высокопроизводительных приложений в Грид включает в себя следующие образцы:

- PEG1 – инструментальная оболочка параллельного исполнения приложений в Грид. PEG1 позволяет создавать параллельные приложения из готовых или разрабатываемых прикладных Грид-сервисов (ПГС) и оперативно отслеживать процесс их исполнения;
- PEG2 – инструментальная оболочка информационной и технологической поддержки процесса проектирования приложений; поддерживает визуальную среду прототипирования композитного приложения в терминах абстрактного описания (абстрактный workflow, или AWF [9]) и предоставляет набор специфичных для Грид инструментов проектирования (монитор Грид, контроллер ресурсов, монитор исполнения и пр.);
- iPEG (Intelligent PEG) – интеллектуальная среда, реализующая всю последовательность операций в рамках концепции [9], от задания пользователем описания задачи в терминах предметной области (мета-workflow, или MWF) до выполнения композитного приложения, созданного на основе квазиоптимального CWF.

Как видно из рис. 1, в оболочке PEG1 реализована основная функциональность работы с инфраструктурой Грид. Для этого используется PEG Grid API, который реализует основные приемы работы с Грид на основе Intel GPE. Для этого он использует библиотеки GPE API версии 1.5, которые, в свою очередь, предоставляют уровень абстракции над Globus. Такая схема позволяет отделить реализацию собственно программного комплекса от реализации подлежащей Грид-архитектуры. PEG Grid API применительно к оболочке PEG2 выполнен таким образом, что позволяет динамически «подменять» промежуточное программное обеспечение Грид, что означает, что на схеме вместо GPE API может появиться, например, UNICORE [3] API. Это достигается абстрактной реализацией основных элементов Грид, таких как задача, сервер, целевая система и пр., конкретные реализации которых могут взаимодействовать с различными Грид-архитектурами. Таким образом, элемент PEG Grid API представляет собой «окно» в Грид, через которое производятся все низкоуровневые операции работы с ним.

Выбор среды GPE сводится к следующим аргументам.

- Среда GPE ориентирована на использование Globus Toolkit: выбор Globus позволяет значительно расширить круг систем, которые можно привлечь к вычислениям. К тому же Globus в своей реализации имеет полную или потенциальную поддержку многих технологий и стандартов, присущих современным Грид-системам.
- Среда GPE поддерживает общепринятые в настоящее время технологии для распределенных информационных систем, которые напрямую не реализованы в Globus: JSDL, BPEL, RandomByteIO. Globus, являясь достаточно низкоуровневым пакетом, предоставляет широкие, но ограниченные возможности, что компенсируется такими объектно-ориентированными продуктами, как GPE (или аналогичное решение – SAGA).
- Среда GPE позволяет осуществлять бесшовное взаимодействие приложений, работающих под разными операционными системами и на различных вычислительных платформах. Интероперабельность и кроссплатформенность обусловлена тем, что

среда GPE разработана с использованием языка Java, реализация виртуальной машины которого существует для большинства современных архитектур.

– Открытость кода: проект GPE реализуется в рамках лицензии GPL, позволяя получать доступ к исходным кодам, тем самым повышая адекватность разрабатываемых моделей, так как учитывается внутренняя реализации приложений и сервисов.
– Клиентские библиотеки: GPE в своем составе имеет хорошо разработанный API для работы со средой из любой программы, что позволяет создавать собственные приложения, непосредственно получающие доступ к Грид-среде.
– Поддержка протокола GridFTP: для более быстрого, надежного и защищенного обмена данными есть возможность использовать протокол GridFTP, что существенно ускоряет работу с данными.

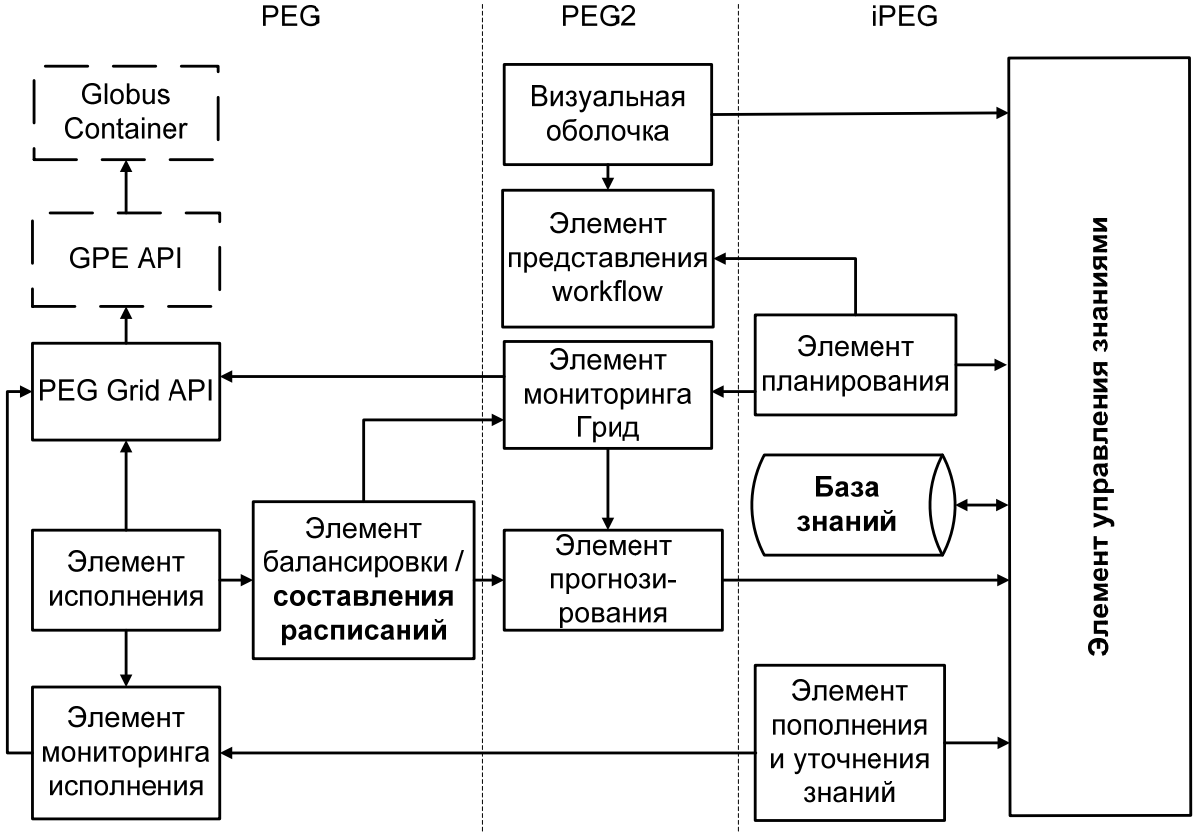


Рис. 1. Эволюция архитектуры оболочек семейства PEG

На рис. 1 представлен процесс эволюции архитектуры оболочек семейства PEG. Элемент исполнения в оболочках PEG отвечает за выполнение пользовательского задания на удаленных целевых системах: он осуществляет вызовы интерпретатора WF, балансировочных алгоритмов, а также производит мониторинг, т.е. является организующим ядром системы.

Элемент балансировки / составления расписаний в различных видах присутствует во всех прототипах семейства PEG, однако предоставляет различную функциональность. В PEG1 он способен лишь произвести декомпозицию данных, используя равномерное разбиение по доступным целевым системам, тогда как в PEG2 он позволяет использовать четыре различных типа расписаний, которые учитывают работу всего WF. В iPEG данный элемент реализуется в виде полнофункционального инструмента составления квазиоптимальных расписаний выполнения композитного приложения с использованием различных эвристик и текущих данных о Грид-среде.

В инструментальной оболочке PEG2, по сравнению с PEG1, вводятся дополнительные функциональные элементы, в частности: визуальная оболочка проектирования, элемент представления WF, элемент мониторинга Грид, элемент прогнозирования времени выполнения приложений в Грид.

Введение визуальной оболочки позволило реализовать одну из ключевых функциональностей программного комплекса – возможность визуальной компоновки пользовательских заданий в виде AWF. В соответствии с этим был введен также элемент представления WF, который отвечает за внутреннее представление и интерпретацию заданного пользователем WF, который в дальнейшем может использоваться остальными элементами программного комплекса. Элементы мониторинга Грид и прогнозирования позволили реализовать полнофункциональную оптимизацию эффективности процесса исполнения, основываясь на данных о текущем состоянии Грид и информации о производительности отдельных сервисов. Оба они используются элементом балансировки в своей работе. В PEG2, как было указано, функциональность элемента была расширена, что достигается за счет включения в его работу вызовов функций новых блоков.

В интеллектуальной среде iPEG вводится ряд принципиально новых элементов, главным из которых является элемент управления знаниями. Он предоставляет основной функционал для работы с базами знаний, в том числе инструменты их использования, активизации, извлечения, пополнения и обновления, а также проверки целостности и достоверности.

Для обеспечения работы iPEG как интеллектуальной системы необходимо также введение таких элементов, как элемент пополнения и уточнения знаний и элемент планирования. Элемент планирования необходим в силу того, что в предлагаемой концепции выдвигается требование, заключающееся в возможности задания пользователем нечеткого описания процесса выполнения в виде MWF. При таком задании необходимо введение этапа трансляции пользовательского описания в набор AWF на основании данных, указанных пользователям в блоках действий. Для этого элементу планирования требуется обращаться к элементу управления знаниями за информации о доступных сервисах и их онтологических описаниях. Это обеспечивает соответствие общей архитектуры оболочек семейства PEG основным функциональным характеристикам ИСППР [9].

Основные программные компоненты инструментальных оболочек семейства PEG

На рис. 2 приводятся обобщенные функциональные схемы инструментальных оболочек PEG1 и PEG2. На них отражены основные этапы работы системы (средний ряд), компоненты, ответственные за выполнение этих этапов (нижний ряд) и получающиеся по окончании этапов результаты (верхний ряд).

Этап проектирования композитного приложения в PEG1 состоит в создании пользователем специального файла описания параллельного задания в формате TSDL. Этот файл передается консольному клиенту, который производит интерпретацию задания и создает его внутреннее представление в виде класса Task. Далее происходит разбиение данных, указанных пользователем в файле описания, на части, предназначенные для отправки на доступные целевые системы в Грид. Эта операция выполняется компонентом декомпозиции данных, одновременно осуществляющим простейшую балансировку на основе равномерной декомпозиции. В результате получается набор задач в виде массива объектов класса Job, каждый из которых содержит информацию о данных и целевой системе, на которой планируется производить запуск. На следующем этапе компонент Executor производит запуск получившихся задач в параллельном режиме, после чего система ожидает завершения работы задач, регулярно производя мониторинг их

состояния на целевых системах. По завершении задач результаты передаются с целевых систем на компьютер пользователя (с которого производится управление) и объединяются компонентом Composer. Результат слияния помещается в указанную пользователем директорию.

Процесс работы PEG2 при внешнем сходстве с PEG1 является более сложным за счет использования визуальной оболочки для описания структуры приложения в форме WF и использования набора конкурирующих методов построения расписаний (балансировки).

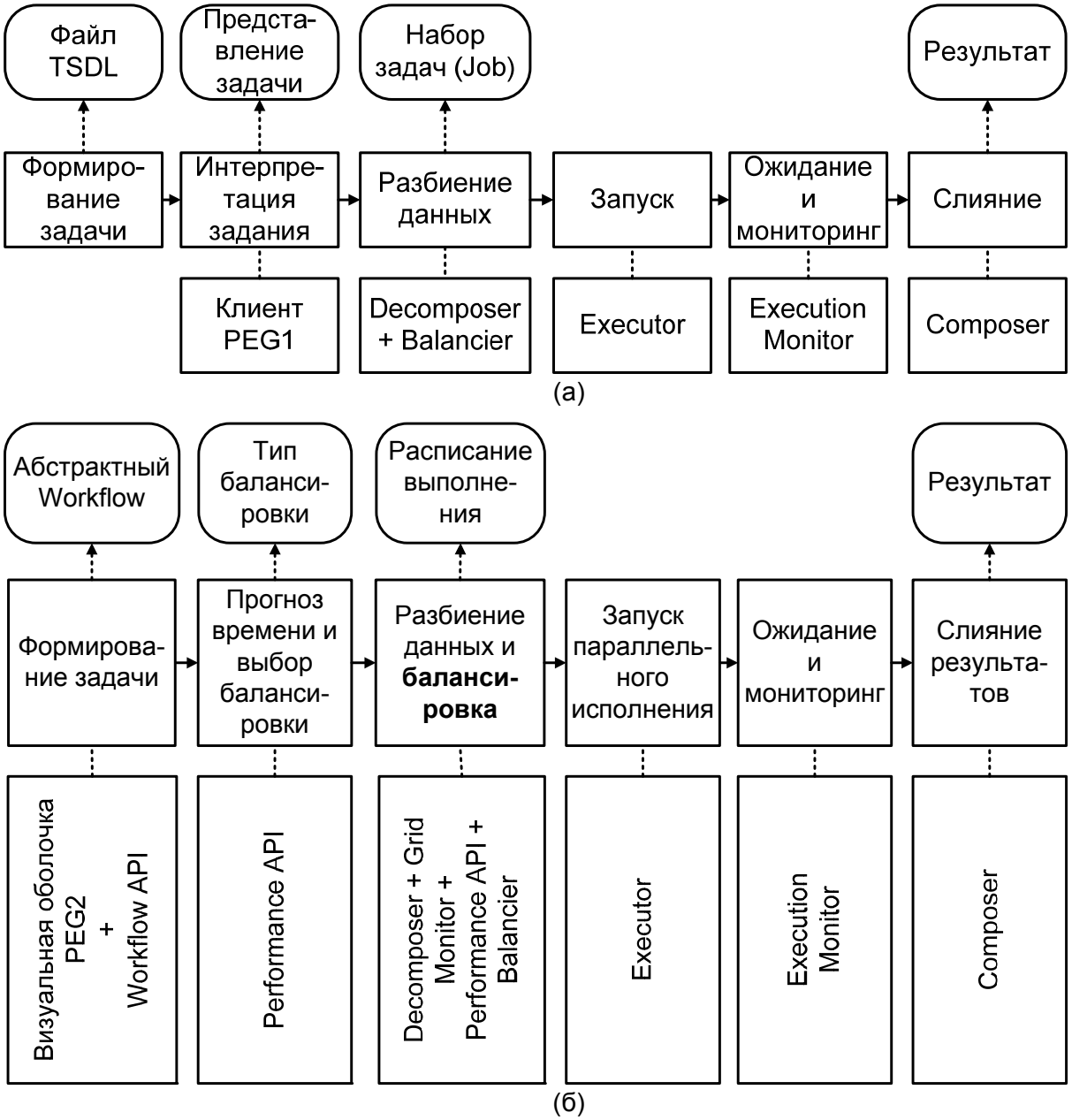


Рис. 2. Функциональные схемы инструментальных оболочек PEG1 и PEG2 (нотация элементов приведена по тексту)

На этапе проектирования в PEG2 пользователь, используя визуальную оболочку, компонует структуру процесса вычислений в виде AWF. Для полученного описания система прогнозирования выполняет оценку времени выполнения для набора конкурирующих расписаний, реализуемых различными типами статических балансировок по данным. На основе этой информации пользователь принимает решение и выбирает вариант, наиболее

удовлетворяющий его критериям. На основании пользовательского выбора компонент балансировщик выполняет разбиение данных, используя данные о производительности ПГС, включенных в WF, и текущем состоянии Грид-среды. В результате формируется расписание выполнения и набор файлов, представляющих собой фрагменты исходных данных для отправки на целевые системы.

Компонент Executor производит запуск композитного приложения в параллельном режиме и ожидает завершения всех его задач, регулярно производя мониторинг их состояния на целевых системах. Результаты мониторинга отображаются с помощью визуальной оболочки в реальном времени. По завершении исполнения происходит сбор результатов с целевых систем и их объединение. Пользователь при этом имеет возможность визуализировать полученные данные.

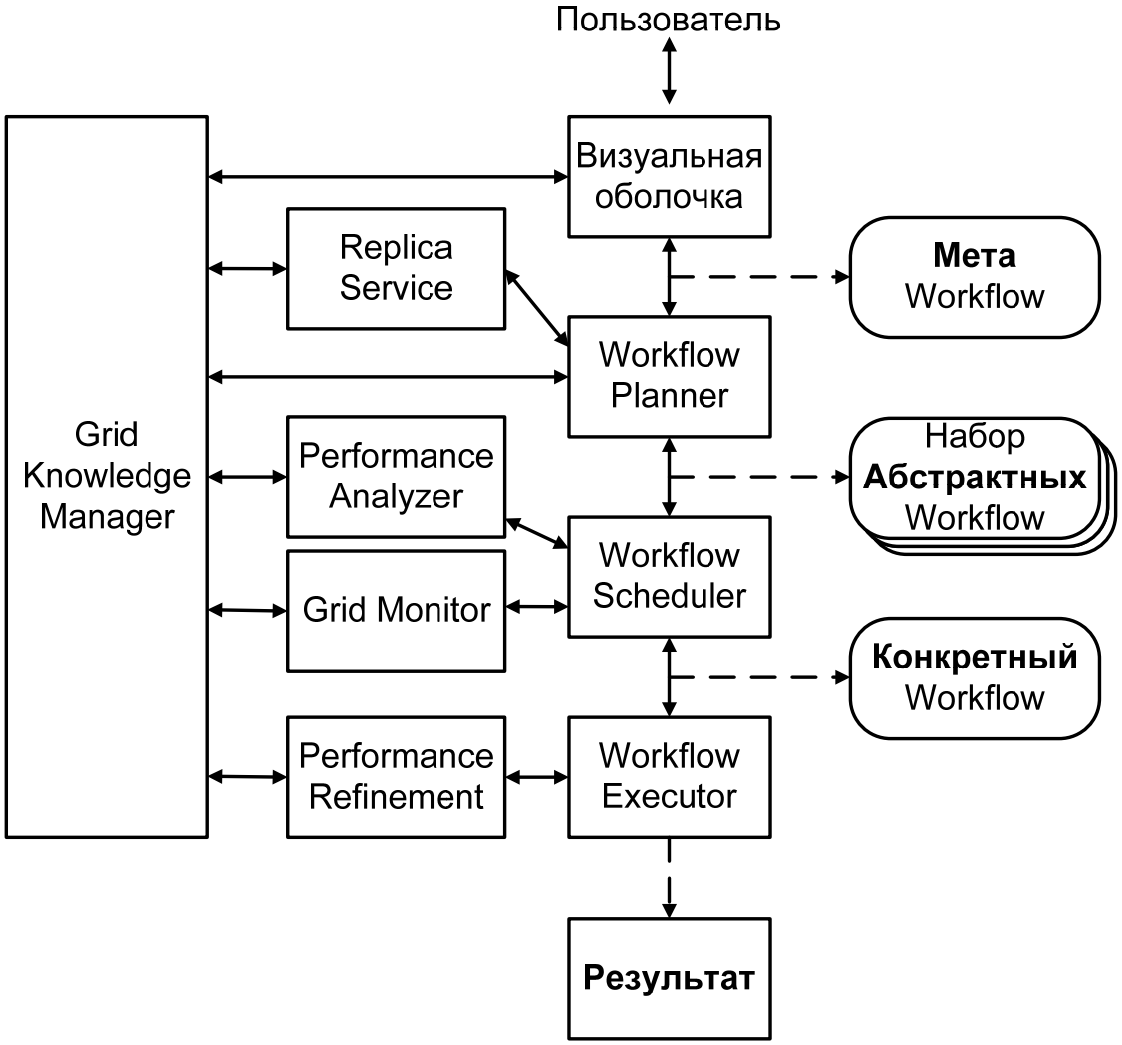


Рис. 3. Функциональная схема инструментальной оболочки iPEG
(нотация элементов приведена по тексту)

На рис. 3 представлена схема функционирования инструментальной оболочки iPEG. Пользователь посредством визуальной оболочки создает описание композитного приложения в форме MWF, используя знания о соответствующем наборе доступных ПГС в Грид, предоставляемом системой управления знаниями (Grid Knowledge Manager). Это описание передается планировщику (Workflow Planner), который на основе знаний о сервисах и данных создает набор альтернативных AWF. Каждый AWF представляет собой непротиворечивый процесс вычислений с фиксированными ПГС и данными. Для указания местоположения данных используется компонент Replica Service,

который позволяет осуществлять доступ к распределенным хранилищам данных, включая результаты предыдущего использования ПГС.

По завершении выполнения задач на целевых системах Workflow Executor осуществляет сбор результатов работы; пользователь получает окончательный результат (например, в виде файла данных). Таким образом, интеллектуальная оболочка iPEG позволяет осуществлять полномасштабную информационную поддержку принятия решений разработчика высокопроизводительных приложений в Грид.

Применение инструментальной оболочки iPEG для создания высокопроизводительных композитных приложений в Грид

Проиллюстрируем основные принципы работы системы iPEG на примере создания композитного приложения, осуществляющего расчет климатических спектров морского волнения [11].

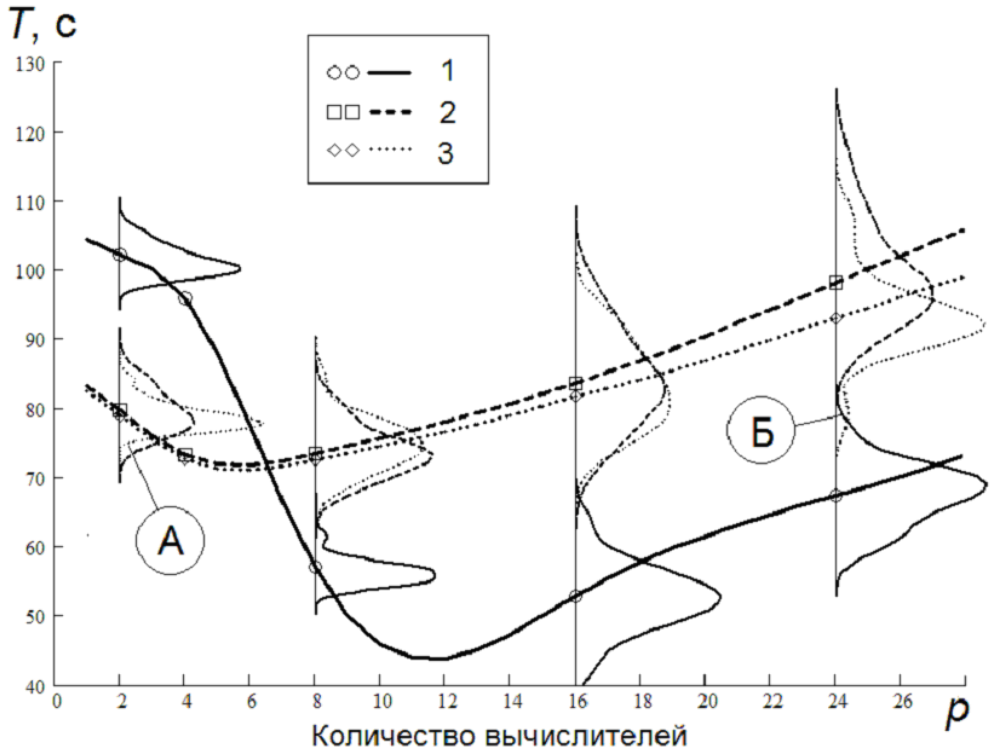


Рис. 4. Время выполнения иллюстративного композитного приложения по обработке 8 000 спектров морского волнения в Грид:
1 – равномерная схема; 2 – каскадная схема; 3 – обратная каскадная схема.
А, Б – см. по тексту

В процессе вычислений производится статистическое обобщение массива таблично заданных функций (24×25 значений) распределения энергии волн по частотам и направлениям в фиксированной точке пространства за длительный временной интервал. Каждой из таблиц сопоставляется нелинейная функция двух аргументов, параметры которой определяются методом адаптивного случайного поиска. Обработка таблиц ведется независимо, что позволяет выполнять распараллеливание вычислительного процесса по данным.

На рис. 4 приведены характеристики времени обработки массива из 8 000 спектров, полученные на экспериментальном стенде – корпоративной Грид-системе, состоящей из 24 целевых систем на базе процессоров Intel Pentium D и Intel Core 2 Quad.

По ходу работы в ИСППР, реализуемой инструментальной оболочкой iPEG, рассматривались три конкурирующих расписания: равномерное распределение данных,

прямая и обратная каскадные схемы, реализующие синхронные способы взаимодействия с вычислителями. Видно, что в среднем для небольшого количества вычислителей (p=2–4) каскадные схемы являются более предпочтительными; с увеличением p накладные расходы на осуществление синхронного взаимодействия возрастают, и, как следствие, предпочтительным становится использование равномерной схемы распределения данных по вычислителям.

На рис. 4 также приведены ядерные оценки плотности распределения времени работы для каждой из трех схем распределения данных, полученные путем обобщения априорных измерений. Видно, что с увеличением количества вычислителей разброс времени выполнения возрастает, что связано с ростом влияния стохастических эффектов коммуникаций в Грид. При этом взаимное расположение плотностей распределения определяет правила ранжирования конкурирующих расписаний. Например, в случае А (рис. 4) для p=2 прямая и обратная каскадные схемы приводят в среднем к практически одинаковым оценкам времени работы. Различие между ними существенно меньше соответствующего диапазона изменчивости, что не позволяет автоматически выбрать наилучший способ. Однако разброс времени работы для прямой каскадной схемы в 1,5 раза больше, чем для обратной каскадной схемы. Как следствие, окончательный выбор схемы распараллеливания остается за пользователем, исходя из предпочитаемой им стратегии (меньший риск – меньшая производительность, больший риск – большая производительность).

С другой стороны, существуют ситуации (например, Б при p=24), когда плотности распределения времени работы для разных схем перекрываются, однако различие между средними значениями сопоставимо с диапазоном их изменчивости. Тогда ранжирование конкурирующих расписаний должно вестись с учетом заданного уровня значимости ошибки; в частности, в ситуации Б в 5% случаев обратная каскадная схема будет давать лучший по производительности результат, чем равномерная, несмотря на то, что в среднем равномерная схема работает в 1,5 раза быстрее.

Из рис. 4 также видно, что кривые производительности имеют ярко выраженные минимумы (для каскадных схем – при p=6, для равномерной – при p=12), которые и определяют оптимальный режим выполнения задачи. Для рассмотренного примера максимальное ускорение в среднем составляет около 2,5 раз, а выигрыш по сравнению с конкурирующими каскадными расписаниями – в 2 раза. Столь невысокие показатели ускорения характерны для вычислений в Грид в силу существенного вклада коммуникационной составляющей; однако они являются только одним из определяющих факторов (наравне с объемом доступной памяти, дискового пространства и пр.), которые заставляют пользователей обратиться к технологиям распределенных вычислений и систем.

Таким образом, в ходе работы интеллектуальной системы iPEG рассмотренный выше анализ проводится автоматически, что позволяет обосновать выбор оптимальной схемы распараллеливания композитного приложения применительно к доступным в Грид вычислительным ресурсам и ПГС.

Заключение

Предложенная в [9] концепция ИСППР, осуществляющая поддержку принятия решений разработчика приложений в Грид, реализуется в рамках семейства инструментальных оболочек PEG. В отличие от традиционных инструментальных средств разработки (например, систем визуального прототипирования WF), ИСППР обеспечивает автоматизацию самого процесса проектирования, формируя оптимальную по производительности архитектуру композитного приложения на основе пользовательского описания в терминах предметной области и знаний о производительности используемых

ПГС. Методы и технологии приобретения соответствующих знаний изложены в работе [12].

Работа выполнена в рамках НИР 2007-4-1.4-20-01-025 «Разработка инструментальной оболочки проектирования высокопроизводительных приложений для Грид-архитектур в целях создания прикладных сервисов компьютерного моделирования и обработки данных».

Литература

1. Stockinger H. Defining the Grid – a snapshot of the current view // Journal of Supercomputing. – Springer Science+Business Media. – 2007.
2. Globus Toolkit Homepage. – Режим доступа: http://globus.org/toolkit/, свободный.
3. UNICORE – Distributed computing and data resources. – Режим доступа: http://www.unicore.eu/, свободный.
4. gLite. – Режим доступа: http://glite.web.cern.ch/glite/, свободный.
5. GPE. – Режим доступа: http://gpe4gtk.sourceforge.net/, свободный.
6. SAGA. – Режим доступа: http://forge.ogf.org/sf/projects/saga–rg/, свободный.
7. Taverna Project Website. – Режим доступа: http://taverna.sourceforge.net/, свободный.
8. Triana Collaboration Projects. – Режим доступа: http://www.trianacode.org/collaborations/index.html, свободный.
9. Ларченко А.В., Дунаев А.В., Бухановский А.В. Иинструментальная оболочка проектирования высокопроизводительных приложений в Грид. Часть I: Базовые положения // Научно–технический вестник СПбГУ ИТМО. – 2008. – Вып. 54.– С. 29–36.
10. Дунаев А.В., Ларченко А.В., Бухановский А.В. Инструментальная оболочка поддержки принятия решений разработчика высокопроизводительных приложений в Грид // Научно–технические ведомости СПбГПУ. – 2008. – № 5. – С. 98–104.
11. Boukhanovsky A.V., Lopatoukhin L.J., Guedes Soares C. Spectral wave climate of the North Sea // Applied Ocean Research. – July 2007. – Vol. 29. – Issue 3. – P. 146–154.
12. Дунаев А.В., Ларченко А.В., Бухановский А.В. Инструментальная оболочка проектирования высокопроизводительных приложений в Грид. Часть III: Приобретение и формализация знаний // Научно–технический вестник СПбГУ ИТМО. – 2008. – Вып. 47. – С. 46–55.
13. K–Wf Grid – Home. – Режим доступа: http://www.kwfgrid.net/, свободный.
14. Pegasus Homepage. – Режим доступа: http://pegasus.isi.edu/, свободный.

Ларченко Алексей Викторович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, м.н.с., aleksey.larchenko@gmail.com

Дунаев Антон Валентинович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, м.н.с., anton.dunaev@gmail.com

Бухановский Александр Валерьевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, д.т.н., профессор, boukhanovsky@mail.ifmo.ru

УДК 004.4’22, 681.324

ИНСТРУМЕНТАЛЬНАЯ ОБОЛОЧКА ПРОЕКТИРОВАНИЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ПРИЛОЖЕНИЙ В СРЕДЕ ГРИД. ЧАСТЬ III: ПРИОБРЕТЕНИЕ И ФОРМАЛИЗАЦИЯ ЗНАНИЙ

А.В. Дунаев, А.В. Ларченко, А.В. Бухановский

Обсуждается проблема приобретения, представления и формализации знаний о производительности вычислительных сервисов в среде Грид. Рассмотрена иерархия аналитических моделей, описывающих время работы в форме детерминированной функции случайных аргументов – параметров среды Грид. Предложен способ косвенного приобретения знаний с использованием индуктивной программы имитационного моделирования параллельных вычислительных процессов в Грид. Представлен способ определения вероятностных характеристик параллельного ускорения на основе знаний о вероятностном распределении времени работы приложения.

Ключевые слова: Грид, проектирование, интеллектуальная система, высокопроизводительные вычисления.

Введение

Интеллектуальные технологии проектирования высокопроизводительных композитных приложений в среде Грид [1–3] основываются на знаниях о параллельной производительности прикладных Грид-сервисов (ПГС) как предметно-ориентированных вычислительных компонентов, зарегистрированных в Грид и доступных другим приложениям и сервисам. Эти знания включают в себя метазнания (информацию о структуре закономерностей, условия применимости, решающие знания), а также собственно предметные знания, или факты, в форме конкретных моделей производительности, таблиц и аппроксимаций, связывающих время выполнения задачи с характеристиками входных и выходных данных и параметрами вычислительной среды.

Приобретение таких знаний (как процесс их передачи от эксперта или извлечения из других источников в базу знаний интеллектуальной системы [4]) в Грид имеет существенное отличие по сравнению с монопольными вычислительными средами. Оно состоит в том, что отдельные ПГС разрабатываются и поддерживаются специалистами в различных предметных областях, принадлежащими к различным научным школам. Как следствие, разработчик ПГС является единственным (и узкоспециализированным) экспертом в области функционирования этого ПГС; при этом уровень представления экспертных знаний существенно варьируется в зависимости от квалификации разработчика в области высокопроизводительных вычислений. Таким образом, задача обоснования архитектуры композитного приложения, требующая использования нескольких ПГС, приводит к необходимости объединения знаний, полученных в неоднородной группе экспертов, что делает затруднительным применение традиционных методов инженерии знаний [5, 6].

Процесс приобретения знаний о производительности приложений в Грид дополнительно осложняется коммунальным характером вычислительной архитектуры, что значит, что физическая инфраструктура может использоваться другими программными средствами (пользователями) одновременно с таковыми в Грид. Как следствие, такие характеристики, как количество и производительность вычислительных узлов, структура и пропускная способность коммуникаций случайным образом меняются во времени [7]. Потому экспертные знания, отражающие специфику вычислительного алгоритма ПГС, *искажаются* при его исполнении в Грид. Для приобретения таких знаний в условиях неопределенности необходимо использовать косвенные методы, основанные на обработке данных экспериментальных измерений производительности ПГС. Получать такие данные можно путем накопления результатов измерений по предыдущим запускам ПГС, а для редких (ненаблюдаемых) ситуаций – посредством индуктивных

программ имитационного моделирования вычислительных процессов в Грид. Данная статья посвящена вопросам формализации и приобретения знаний о производительности приложений в Грид в условиях неопределенности, обусловленной стохастическим характером вычислительной среды.

Аналитические модели параллельной производительности в Грид

Аналитические модели в форме зависимости $T = f(\Xi, \Pi)$ времени выполнения T_Π от характеристик задачи Ξ и параметров вычислительной системы Π используются для исследования характеристик параллельных алгоритмов для традиционных (например, кластерных) архитектур [8, 9]. Учет стохастичности набора параметров Π позволяет интерпретировать T_Π в форме детерминированной функции случайных аргументов и использовать для ее описания и анализа соответствующие методы теории вероятностей [10].

Одной из наиболее простых моделей производительности вычислений в Грид является сетевой закон Амдала, где коэффициент деградации вычислений τ – случайная величина:

$$T_p = \frac{T_0}{p}\left(1 + \tau \cdot p\right). \qquad (1)$$

Здесь T_p – время вычислений на p вычислителях (целевых системах), T_0 – время вычислений на ПГС на пользовательской системе (вне Грид). Модель (1) требует однородности целевых систем и коммуникаций между ними. Ее характерной особенностью является гомоскедастичность, выраженная в том, что в зависимости от p изменяется только математическое ожидание $M[T_p]$, а дисперсия времени выполнения остается постоянной.

Построение более сложных моделей параллельной производительности требует конкретизации структуры параллельного алгоритма и входных данных. Это позволяет выразить время выполнения задачи через удельные времена выполнения коммуникационной операции t_w и вычислительной операции t_c. Например, для параллельного алгоритма, распределяющего массив из N данных между p вычислителями с целью расчета M величин на каждом и последующего их сбора на одном узле, модель времени выполнения принимает вид

$$T_p = \left(N + pM\right)t_w + \frac{N}{p}t_c, \qquad (2)$$

где t_w – случайная величина. Модель (2) описывает однородную вычислительную систему, в которой t_w характеризует усредненную пропускную способность коммуникационной среды, которая случайным образом изменяется от запуска к запуску приложения одинаково для всех коммуникационных каналов. В отличие от (1), эта модель является гетероскедастической: с увеличением p дисперсия времени вычислений возрастает.

Выражение (2) является оценкой снизу для времени выполнения приложения, поскольку не учитывает коммуникационного дисбаланса, обусловленного тем, что пропускные способности коммуникационных каналов в Грид могут меняться асинхронно, т.е. $t_w^{(i)} = t_w + \eta_i$, где η_i – гауссов случайный шум с заданными математическим ожиданием m_η и среднеквадратичным отклонением σ_η. С учетом этого факта время работы приложения определяется как $T_p = \max\limits_{i=1,p}\left[T_i\right]$, где T_i – время работы с каждым вычислителем.

Оно также является гауссовой случайной величиной со средним значением m_T и среднеквадратичным отклонением σ_T:

$$m_T = \left(t_w + m_\eta\right)\left(\frac{N}{p} + M\right) + t_c \frac{N}{p},\ \sigma_T = \left(\frac{N}{p} + M\right)\sigma_\eta, \tag{4}$$

соответственно. Тогда закон распределения величины T_p (как максимума случайной выборки) асимптотически (при значительных p) становится предельным распределением I типа (Гумбеля, или Фишера-Типпета) [11]

$$F_{T_c}(x) = \exp\left[-\exp\left[-a_p(x - b_p)\right]\right], \tag{5}$$

где коэффициенты

$$a_p = \frac{\sqrt{2\ln(p)}}{\sigma_T},\ b_p = \left[\sqrt{2\ln(p)} - \frac{\ln(\ln(p)) + \ln(4\pi)}{2\sqrt{2\ln(p)}}\right]\sigma_T + m_T \tag{6}$$

зависят от количества рабочих узлов p.

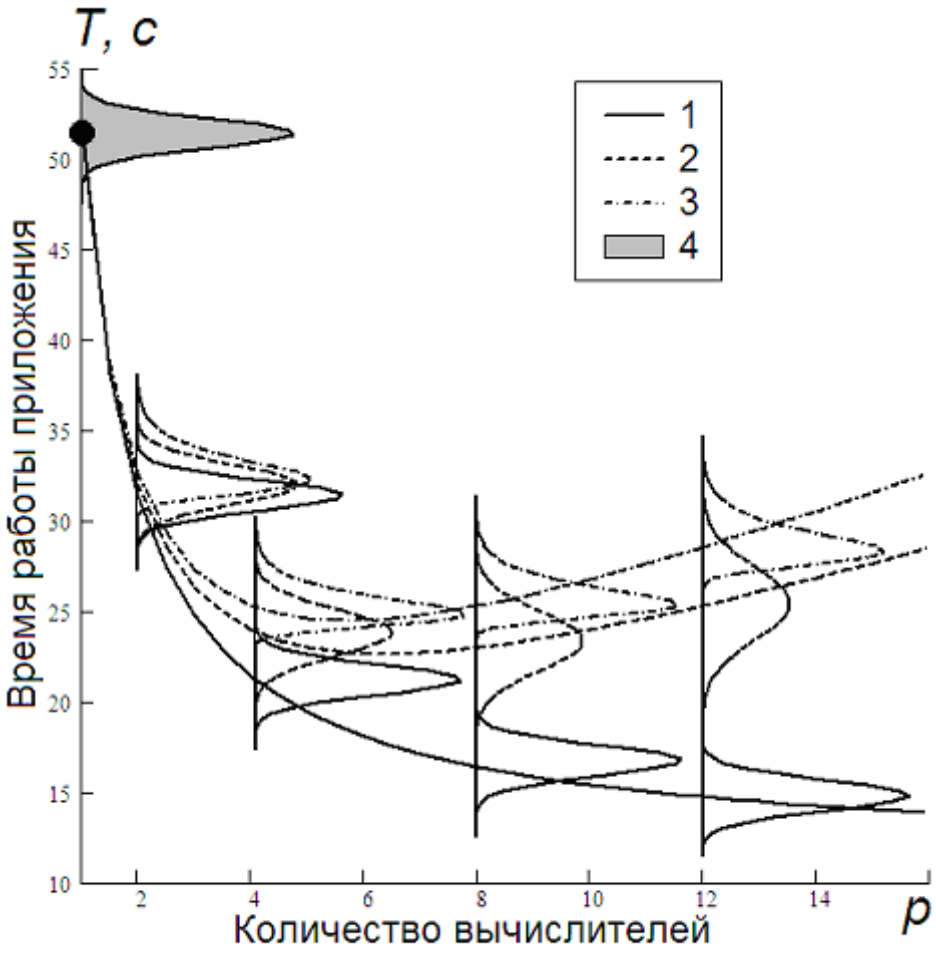


Рис. 1. Сопоставление аналитических моделей параллельной производительности в Грид: 1 – сетевой закон Амдала (1); 2 – параметрическая модель (2); 3 – модель на основе теории экстремальных значений (5); 4 – экспериментальное распределение для настройки параметров моделей

На рис. 1 приведены результаты сопоставления аналитических моделей производительности (1), (2) и (5), построенных по результатам измерений производительности

тестового приложения на экспериментальном стенде на основе системы PEG1 [2]. Для корректности сопоставления при идентификации параметров всех трех моделей использовались только данные измерений T_0 и T_1.

Из рис. 1 следует, что мера различия моделей возрастает с увеличением количества вычислителей. Учет специфических факторов Грид в моделях приводит к увеличению оценки среднего времени работы приложения. При переходе от модели (2) к модели (5) изменяется тип распределения – оно становится асимметричным. При этом размах распределения (5) в целом меньше, чем распределения (2), поскольку учет коммуникационного дисбаланса позволяет уточнить оценку времени работы за счет выделения систематической составляющей – разности математических ожиданий оценок (2) и (5).

Модель (5) с коэффициентами (6) применима для описания вычислительных архитектур со стохастической изменчивостью производительности вычислителей и постоянной пропускной способностью коммуникаций [7]. Для более сложных случаев, когда стохастическое поведение свойственно одновременно и производительности узлов, и пропускной способности коммуникационной сети, класс экстремального распределения (5), по-видимому, сохраняется. Однако его коэффициенты уже не могут быть заданы в аналитическом виде. Потому для исследования таких систем необходимо применять методы имитационного моделирования.

Имитационное моделирование выполнения приложений в Грид

Для исследования производительности параллельных приложений в Грид-системах со сложной топологией, напрямую не описываемой аналитическими моделями, разработана система имитационного моделирования вычислительных процессов в Грид. Разработанная модель комбинирует элементы процессного и транзактного блочного имитационного моделирования и реализует средний уровень детализации процессов в Грид-среде [12], учитывая влияние большого числа факторов, ни один из которых не имеет лидирующего значения. Система имитационного моделирования позволяет задавать произвольные сетевые топологии, производительность и загруженность для вычислительных узлов, пропускную способность и загруженность для сетевых каналов, законы распределения для шума загруженностей вычислительных узлов и сетевых каналов. При этом на ее основе могут быть реализованы различные методы параллельной декомпозиции и связывания компонентов исследуемого приложения.

Для описания вычислительной задачи применяется специальная нотация, которая позволяет описывать любые программы со статической схемой выполнения. В описании используются 4 базовых элемента: *send*, *calc*, *thread* и *threadGroup*. Первые два задают элементарные операции передачи и обработки данных, а последние позволяют организовывать последовательные и параллельные потоки.

На рис. 2 приведена общая функциональная схема разработанной системы имитационного моделирования. Систему можно разделить на 2 слабо связанных блока – *Балансировщик* и *Ядро*. На вход *Балансировщика* подаются файл конфигурации среды Грид, объем входных и выходных данных, а также первоначальное описание вычислительной задачи, которое не указывает, на каких элементах Грид и какие именно действия должны быть выполнены. Помимо этого, пользователь задает метод балансировки и декомпозиции данных, который должен быть применен при уточнении описания вычислительной задачи и проецировании ее на элементы Грид. Далее уточненная схема вычислительной задачи вместе с конфигурацией среды Грид передается в исполнительный блок ядра системы, который является его организующим элементом. Он интерпретирует схему задачи и, используя блоки *Сетевой канал* и *Вычислительный узел,* производит имитацию работы приложения в среде Грид. Блоки *Сетевой канал* и *Вычислительный узел* производят учет работы элементов среды Грид, позволяя рассчиты-

вать время работы операций send и calc. Использование блока *Линия времени* позволяет при моделировании учитывать процессы разделения элементов инфраструктуры (сетевых каналов, вычислительных узлов, серверов Грид) между несколькими независимыми потоками действий и корректировать расчетное общее время работы схемы задачи.

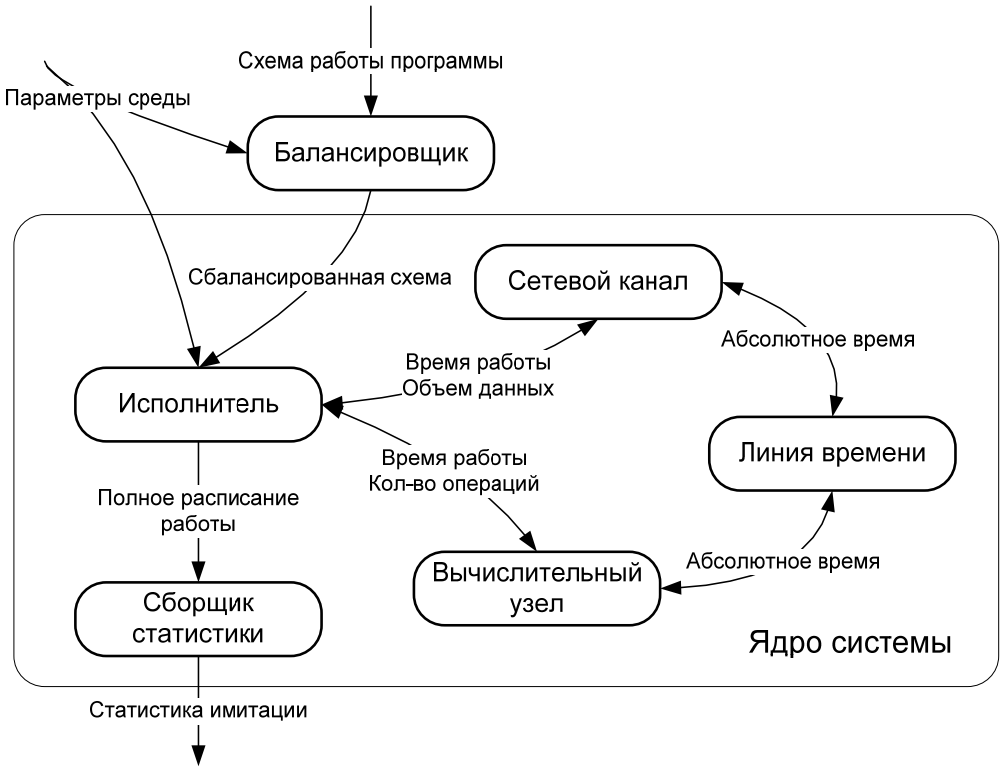


Рис. 2. Функциональная схема системы имитационного моделирования

Результатом моделирования становится полное расписание выполнения схемы вычислительной задачи, которое передается блоку статистической обработки *Сборщик статистики*. Он производит расчет общего времени работы схемы задачи и дисбаланса времени выполнения параллельных потоков. Этот блок легко поддается расширению функциональности для получения дополнительных сведений о процессе работы задачи.

На рис. 3 представлена принципиальная схема имитационного процесса. Ее можно разделить на две тесно связанные составляющие: блочную схему выполнения процесса моделирования и линии учета времени выполнения.

Первая составляющая описывает элементарные действующие компоненты, такие, как блоки передачи данных, блоки выполнения операций, а также описывает отношения между элементарными компонентами – блоком последовательного объединения и блоком параллельного объединения. В блоке «Схема выполнения параллельного приложения» описывается сам параллельный алгоритм. Например, объединяющий блок Thread #1 устанавливает отношение последовательного выполнения между блоками Send #1, ThreadGroup и Send #4. Блок ThreadGroup устанавливает отношение параллельного выполнения между двумя блоками Thread #2 и Thread #3.

Помимо описания отношений, группирующие блоки Thread и ThreadGroup описывают порядок выполнения. Для приведенной схемы он будет выглядеть так: Send #1, (Send #2, Calc #2, Send #3 | Calc #1, Send #5) выполняются в параллельном режиме, Send #4. Необходимо обратить внимание на обязательный синхронизационный барьер, который всегда присутствует после блока ThreadGroup. Операции, следующие за ThreadGroup, не начнут выполняться, пока не выполнятся все потоки операций, входящие в состав ThreadGroup.

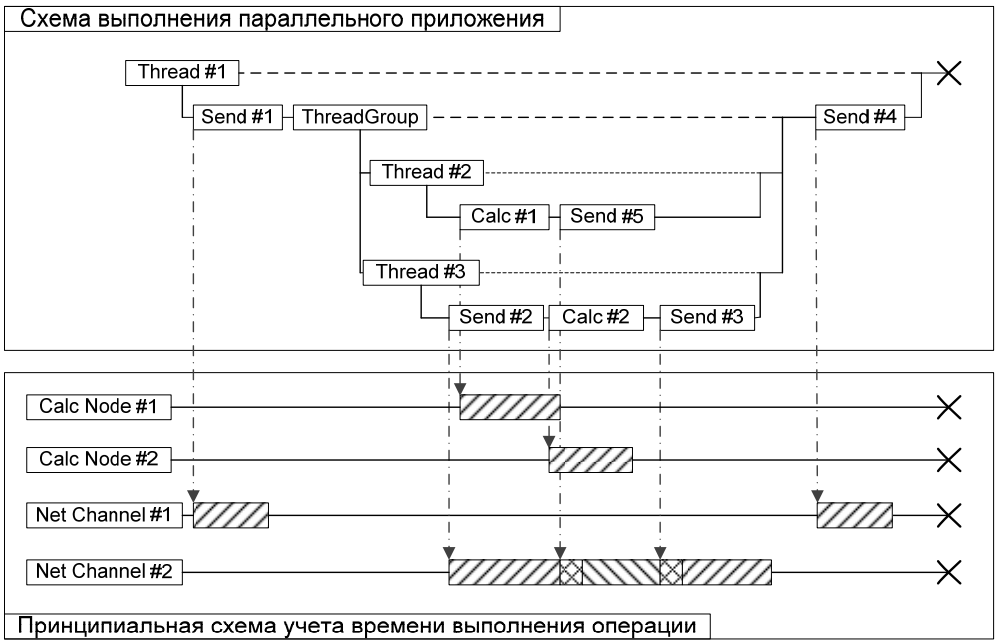


Рис. 3. Принципиальная схема работы программного симулятора

Схема выполнения, помимо задания отношений следования между элементарными блоками, задает характеристики элементарных блоков. К таким характеристикам можно отнести объем вычислений и идентификатор вычислительного узла, на котором будут производиться вычисления – для вычислительной операции calc; объем передаваемых данных и идентификатор сетевого канала, по которому будет производиться передача данных – для операции передачи данных send.

Вторая составляющая процесса имитационного моделирования (на рис. 3 отображена в виде блока «Принципиальная схема учета времени выполнения операций») позволяет производить подсчет времени работы операций, описанных схемой выполнения. Линии учета времени – это структуры, которые закрепляются за всеми элементами топологии, участвующими в процессе симуляции, и хранят данные о своей занятости тем или иным блоком схемы выполнения. На рисунке приведены четыре линии – две, относящиеся к вычислительным узлам, и две, относящиеся к сетевым каналам. На них схематично изображены прямоугольники, отражающие время выполнения того или иного элементарного блока с участием элемента топологии, за которым закреплена данная линия. Такой подход был применен для того, чтобы корректно учитывать взаимовлияние независимых, параллельно выполняющихся элементарных операций друг на друга посредством разделения элемента топологии. Подобные случаи часто встречаются при разделении сетевых каналов несколькими процессами передачи данных. Поэтому для учета влияния требуется во время моделирования производить синхронизацию процессов, выполняющихся параллельно, и вносить корректировки в конечное время выполнения того или иного блока схемы выполнения.

На рис. 4 приведены результаты сопоставления плотностей времени работы вычислительной задачи, рассчитанных на основе имитационной модели и стендовых экспериментов. На стенде Грид-среды вычислялась задача расчета климатических спектров морского волнения [14]. Стенд состоял из 6 целевых систем (вычислительных узлов) различной производительности и программного оснащения (ОС, различные сборки ПГС), 1 выделенного сервера Грид и 1 терминала пользователя, осуществляющего декомпозицию исходных данных, их посылку, получение и композицию результатов вычислений. Для объединения компьютеров была применена активно используемая локальная сеть. Так как большое число различных независимых источников стохастичности не позволяет использовать описанные аналитические модели, то единственным

способом моделирования процессов, происходящих при вычислении описанной задачи на стенде, является применение имитационной модели. Для составления файла описания Грид-среды были проведены специализированные тесты по измерению характеристик инфраструктуры стенда. Задача расчета климатических спектров морского волнения была описана в нотации системы имитационного моделирования.

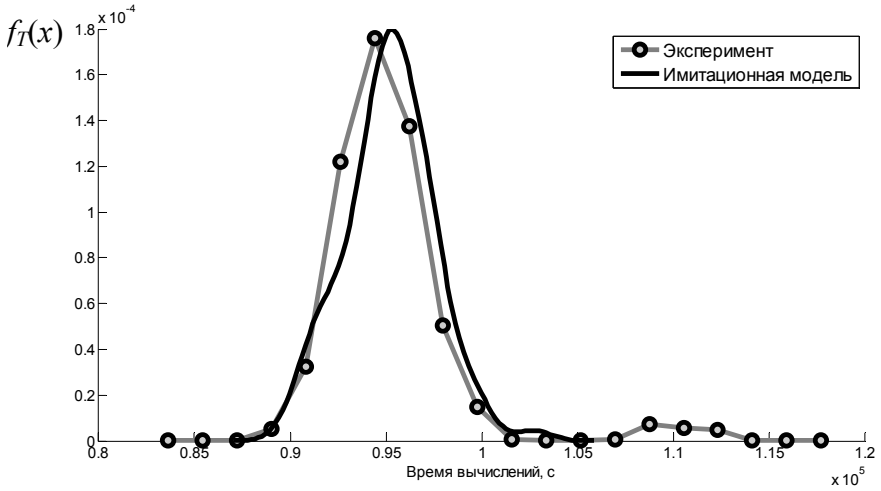


Рис. 4. Сопоставление результатов моделирования с экспериментом

Из рис. 4 видно, что, хотя имитационная модель достаточно точно описывает процесс вычислений на стенде, она имеет заметные отклонения от экспериментальных результатов. Эти отклонения объясняются тем, что имитационная модель имеет средний уровень детализации и не отслеживает влияния таких низкоуровневых эффектов, как:

- работа библиотек шифрации и дешифрации передаваемых между узлами Грид данных;
- задержки при постановке задачи на выполнение;
- задержки при передаче сообщений по протоколу SOAP для обмена данными (в том числе управляющими).

Определение параллельного ускорения в Грид

Аналитические или имитационные модели производительности приложений в Грид позволяют оценить плотность распределения времени работы $f_T(x)$. Помимо времени работы, важной характеристикой масштабируемости приложения является параллельное ускорение S_p. Классическое определение параллельного ускорения в виде

$$S_p = T_0 / T_p \tag{7}$$

не может быть использовано для непосредственного определения производительности в Грид по экспериментальным данным, поскольку значение T_p может изменяться от запуска к запуску. Однако, интерпретируя (7) как детерминированную функцию случайной величины T_p, можно выразить плотность распределения параллельного ускорения через $f_T(x)$:

$$f_S(x) = \frac{T_1}{x^2} f_T\left[\frac{T_1}{x}\right]. \tag{8}$$

В качестве примера на рис. 5 приведены вероятностные характеристики параллельного ускорения, рассчитанного по методике (7)–(8) для композитного приложения, осуществляющего расчет климатических спектров морского волнения. Подробное опи-

сание приложения и оценки времени его работы на экспериментальном стенде – корпоративной Грид-системе, состоящей из 24 целевых систем на базе процессоров Intel Pentium D и Intel Core 2 Quad под управлением оболочки iPEG – приведены в работе [2]. Для различного количества вычислителей p рассчитывались распределения $f_T(x)$ посредством ядерной оценки с гауссовым ядром, по которым на основе (8) вычислялись математическое ожидание и характерные квантили (99%, 10% и 1% обеспеченностей).

Видно, что кривые характеристик параллельного ускорения имеют ярко выраженные максимумы (при $p \cong 10$–14), которые и определяют оптимальный режим выполнения задачи. При этом стохастическая изменчивость ускорения (выражаемая, например, в размахе распределения) возрастает по мере увеличения количества вычислителей p, достигая максимума в районе максимума математического ожидания ускорения (соответствующего оптимальному *в среднем* количеству вычислителей). Это связано с тем, что выбор оптимального количества вычислителей p^* обусловлен балансом между выигрышем по времени за счет распараллеливания вычислительных операций и влиянием коммуникационных расходов. Как следствие, стохастическая изменчивость характеристик Грид в данном случае приводит к нарушению этого баланса как в одну, так и в другую сторону, что и влияет на степень разброса результатов измерений. По мере дальнейшего увеличения количества вычислителей, $p >> p^*$, размах распределений ускорения уменьшается.

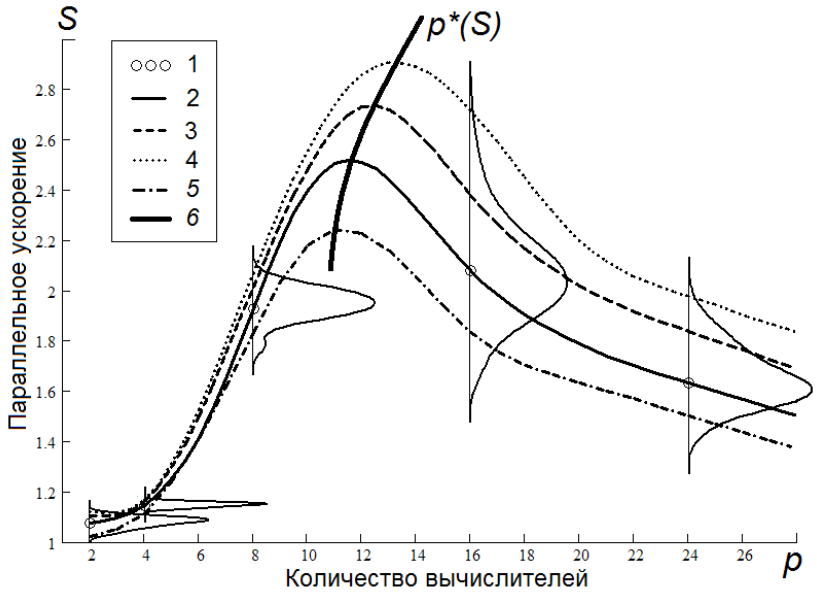


Рис. 5. Параллельное ускорение выполнения иллюстративного композитного приложения по обработке 8 000 спектров морского волнения в Грид:
1 – средние значения (эксперимент); 2 – математическое ожидание;
3 – квантиль 10%; 4 – квантиль 1%; 5 – квантиль 90%;
6 – оптимальное количество вычислителей для заданного уровня ускорения

Из рис. 5 также следует, что максимумы квантильных кривых ускорения смещаются в сторону большего значения p^* по мере уменьшения их обеспеченности. Как следствие, для вычислений в Грид понятие оптимального количества процессоров p^*, по-видимому, лишено смысла, поскольку является случайной величиной, условное распределение которого $f_P(x \mid S)$ в общем случае зависит от ускорения S, на достижение которого ориентируется разработчик. Это заставляет перейти от безусловной ха-

рактеристики p^* к вероятностной характеристике – *усредненному* оптимальному количеству вычислителей для заданного уровня ускорения:

$$p^*(S) = \frac{\sum\limits_{k:\,p_k \geq S} p_k \int\limits_S^{p_k} f_{S(p_k)}(x)dx}{\sum\limits_{k:\,p_k \geq S} \int\limits_S^{p_k} f_{S(p_k)}(x)dx}. \qquad (9)$$

Величина (9) носит характер условного математического ожидания. По мере повышения уровня S величина $p^*(S)$ также увеличивается; это связано с тем, что потенциально для получения большего ускорения требуется больше вычислителей p.

С величиной (9) связана вероятность α, определяющая шанс получить ускорение не ниже заданного уровня S при выборе оптимального количества вычислителей:

$$\alpha = \int\limits_S^{p^*(S)} f_{S(p_k)}(x)dx. \qquad (10)$$

Таким образом, производительность параллельного приложения в Грид, в отличие, например, от традиционных кластерных систем, характеризуется тройкой взаимосвязанных величин (S, p^*, α) – заданного ускорения, оптимального количества вычислителей для его достижения и соответствующей вероятности α.

Заключение

Экспертные знания разработчиков ПГС, связывающие параллельную производительность со спецификой алгоритма, *искажаются* при исполнении сервиса в Грид в силу стохастического характера вычислительной среды. Для приобретения таких знаний в условиях неопределенности адаптировано семейство аналитических моделей, позволяющих обобщать данные экспериментальных измерений производительности ПГС, и разработана система имитационного моделирования, выступающая в роли индуктивной программы для интеллектуальных систем семейства PEG [1, 2]. Для интерпретации приобретаемых знаний в терминах параллельного ускорения предложен метод его определения на основе формализма детерминированной функции случайного аргумента.

Работа выполнена в рамках НИР 2007-4-1.4-20-01-025 «Разработка инструментальной оболочки проектирования высокопроизводительных приложений для Грид-архитектур в целях создания прикладных сервисов компьютерного моделирования и обработки данных».

Литература

1. Ларченко А.В., Дунаев А.В., Бухановский А.В. Инструментальная оболочка проектирования высокопроизводительных приложений в Грид. Часть I: Базовые положения // Научно–технический вестник СПбГУ ИТМО: Технологии высокопроизводительных вычислений и компьютерного моделирования. – 2008. – Вып. 54. – С. 29–36.
2. Ларченко А.В., Дунаев А.В., Бухановский А.В. Инструментальная оболочка проектирования высокопроизводительных приложений в Грид. Часть II: Архитектура, реализация и применение // Научно–технический вестник СПбГУ ИТМО: Технологии высокопроизводительных вычислений и компьютерного моделирования. – 2008. – Вып. 54. – С. 37–45.
3. Инструментальная оболочка поддержки принятия решений разработчика высокопроизводительных приложений в Грид / А.В. Дунаев, А.В. Ларченко, А.В. Бухановский // Научно–технические ведомости СПбГПУ. – 2008. – № 5. – С. 98–104.

4. Нечаев Ю.И. Искусственный интеллект: концепции и приложения. – СПб: Изд. СПбГМТУ, 2000. – 294 с.
5. Частиков А.П., Гаврилова Т.А., Белов Д.Л. Разработка экспертных систем. Среда CLIPS. — СПб: БХВ-Петербург, 2003. – 608 с.
6. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. – СПб: Питер, 2000. – 384 с.
7. Дунаев А.В., Ларченко А.В., Бухановский А.В. Моделирование параллельных вычислительных процессов в среде Грид на примере Intel Grid Programming Environment акселераторов // Параллельные вычислительные технологии (ПаВТ 2008): Труды международной научной конференции, Санкт–Петербург, 28 января – 1 февраля, 2008. – Челябинск: Изд. ЮУрГУ, 2008. – С. 383–389.
8. Foster I., Kesselman C. The Grid: Blueprint for a New Computing Infrastructure. – Morgan–Kaufman, 1999.
9. Гергель В.П. Теория и практика параллельных вычислений: Учебное пособие. – М.: Интернет–Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2007.– 423 с.
10. Лившиц Н.А., Пугачев В.Н. Вероятностный анализ систем автоматического управления. – М.: Советское радио, 1963. – Т. 1. – 896 с.
11. Лидбеттер М., Линдгрен Г., Ротсен Х. Экстремумы случайных последовательностей и рядов. – М., Мир, 1989, 392 с.
12. Филиппович А.Ю. Интеграция систем ситуационного, имитационного и экспертного моделирования.– М.: Изд–во «ООО Эликс+», 2003. – 300 с.
13. Кельтон В.Д., Лоу А.М. Имитационное моделирование. Классика CS. – 3-е изд. – СПб: Питер, 2004. – 847 с.
14. Boukhanovsky A.V., Lopatoukhin L.J., Guedes Soares C. Spectral wave climate of the North Sea // Applied Ocean Research. – July 2007. – Vol. 29. – Issue 3. – P. 146–154.

Дунаев Антон Валентинович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, м.н.с., anton.dunaev@gmail.com

Ларченко Алексей Викторович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, м.н.с., aleksey.larchenko@gmail.com

Бухановский Александр Валерьевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, д.т.н., профессор, boukhanovsky@mail.ifmo.ru

УДК 004.021, 681.324

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЙ ПРОГРАММНЫЙ КОМПЛЕКС МОДЕЛИРОВАНИЯ ЭКСТРЕМАЛЬНЫХ ГИДРОМЕТЕОРОЛОГИЧЕСКИХ ЯВЛЕНИЙ. ЧАСТЬ I: ПОСТАНОВКА ЗАДАЧИ, МОДЕЛИ, МЕТОДЫ И ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ

А.В. Бухановский, О.И. Зильберштейн, С.В. Иванов, С.В. Ковальчук, Л.И. Лопатухин, С.К. Попов, М.М. Чумаков

Обсуждаются вопросы практической реализации новой концепции получения информации об экстремальных гидрометеорологических явлениях на основе компьютерного моделирования. Формулируются основные этапы вычислительной технологии, рассматриваются особенности современных подходов, численных методов и параллельных алгоритмов для гидродинамического и статистического моделирования полей ветра, волнения, течений и уровня моря. Приводится оценка ресурсоемкости основных вычислительных операций и обосновывается выбор суперкомпьютерной архитектуры.

Ключевые слова: экстремальные явления, гидрометеорологические моделирование, параллельные алгоритмы.

Введение

Освоение пространства Мирового океана (прежде всего – шельфовой зоны) требует информации о внешних нагрузках на морские объекты и сооружения. Внешние нагрузки обусловлены совместным действием ветра, волнения и морских течений (в некоторых случаях – льда) при определенном уровне моря. Их характеристики подразделяются на оперативные (определяющие режимы эксплуатации объектов и сооружений) и экстремальные (характеризующие режим выживания) [1]. В отличие от оперативных характеристик, традиционно получаемых посредством статистического анализа исторических массивов гидрометеорологической информации, экстремальные характеристики обычно не обеспечены данными наблюдений и, как следствие, должны определяться с использованием инструментария математического моделирования [2, 3].

В практике проектирования морских сооружений в качестве экстремальных обычно рассматриваются расчетные характеристики гидрометеорологических явлений, возможных 1 раз в T лет, где T соответствует классу сооружения. Для судов и объектов океанотехники традиционно $T = 1$–100 лет, а для некоторых гидротехнических сооружений (как, например, защитные сооружения Санкт-Петербурга от наводнений) могут рассматриваться периоды повторяемости до $T = 10\,000$ лет. Поэтому моделирование экстремальных гидрометеорологических явлений является существенно более ресурсоемкой задачей, чем традиционные гидрометеорологические задачи прогноза погоды и моделирования климата (например, [4–6]).

Обычно при прогнозе погоды оперируют данными на объемной пространственной сетке, однако все расчеты выполняются в синоптическом диапазоне (в среднем от 2 до 5 суток). Задача моделирования климата охватывает временной интервал в десятки и сотни лет, однако она может оперировать декадными или даже среднемесячными данными. В то же время современная концепция получения информации об экстремальных гидрометеорологических явлениях [7] основана на полностью синтетическом подходе [8].

В данной работе рассматриваются вопросы практической реализации концепции [7, 8], исходя из особенностей применяемых математических моделей и численных методов, специфики параллельных алгоритмов и потенциальных возможностей современных суперкомпьютерных систем.

Эволюция подходов, методов и технологий математического моделирования экстремальных гидрометеорологических явлений

В настоящее время существует несколько кардинально различных подходов к расчету гидрометеорологических характеристик, возможных 1 раз в T лет. Наиболее известными их реализациями являются методы IDM (Initial Distribution Method), AMS (Annual Maxima Series) и POT (Peak Over Threshold), которые основаны на интерпретации экстремального значения как детерминированной квантили распределения, крайнего члена выборки и выброса случайного процесса за уровень [9]. Несмотря на различие вероятностных моделей, современные методы расчета экстремальных гидрометеорологических явлений являются комбинированными, что позволяет им гибко сочетать особенности математического аппарата и специфику описываемого им явления. В частности, авторский метод BOLIVAR [9, 10], как и POT, интерпретирует экстремальные явления в терминах выходов за уровень, однако использует для описания их совместной изменчивости формализм квантильной функции годовых максимумов, что позволяет сохранить теоретическую обоснованность методов группы AMS. В табл. 1 сопоставляются основные характеристики указанных методов.

Таблица 1. Сопоставление методов расчета гидрометеорологических экстремумов

Критерии	Методы			
	IDM	AMS	POT	BOLIVAR
Моделирование «хвоста» распределения	Эвристически (логнормальное или Вейбулл)	Класс предельных распределений I, II, III типа или обобщенное экстремальное	Обобщенное распределение Парето	Класс предельных распределений I, II, III или обобщенное экстремальное
Объем выборки для оценки параметров	$365Tn$, где n – число синоптических сроков в сутки	T	1–3T, зависит от уровня	40–70T (в зависимости от акватории)
Определение вероятности для оценки периода повторяемости	Используя среднее число независимых наблюдений	Точно (как годовой максимум)	В среднем (с учетом среднего числа штормов в год)	Точно (как в методе AMS)
Учет сезонной и межгодовой изменчивости	Отдельно не рассматривается и косвенно учитывается в режимном распределении	Не учитывается	В среднем (в зависимости от уровня)	Учитывается для каждого диапазона отдельно
Требования к информационной базе	Разрозненные наблюдения по районам за длительный промежуток времени	Значения годовых максимумов (обычно по измерениям особо опасных явлений)	Значения характеристик в отобранных штормах	Непрерывные временные ряды в синоптические сроки за длительный промежуток времени
Методы вычислений	Оценивание параметров распределения и экстраполяция по формуле	Оценивание параметров распределения и экстраполяция по формуле	Оценивание параметров распределения и экстраполяция по формуле	Метод Монте-Карло

Из табл. 1 следует, что различия в подходах обусловлены не только спецификой применяемого математического аппарата, но, в первую очередь, особенностями информационной базы, на основании которой выполняются расчеты. Так, метод IDM весьма неприхотлив к качеству данных и может использоваться на достаточно больших разрозненных выборках (например, попутные судовые наблюдения за ветром и волне-

нием). Метод AMS требует выполнения регулярных наблюдений в фиксированной точке в течение ряда лет, однако он использует только характеристики самого сильного шторма в каждом году (что обычно доступно при наличии регулярных измерений на гидрометеорологических станциях). Метод POT использует только информацию о характеристиках процессов в выбранных (на уровне построения вероятностной модели) штормах, что существенно увеличивает объемы выборок и облегчает процедуру расчетов, однако повышает степень неопределенности в оценках экстремумов. Наконец, метод BOLIVAR использует информацию о штормах как *импульсах* заданной интенсивности, длительности и времени появления. Потому для его применения необходимы *непрерывные* временные ряды измерений в синоптические сроки за длительный промежуток времени.

Современная концепция расчета экстремальных гидрометеорологических характеристик на основе данных гидродинамического моделирования

Практическое применение метода BOLIVAR связано с необходимостью иметь соответствующую информационную базу, необходимую для идентификации вероятностных моделей. Однако получение таких данных об океанографических характеристиках (волнении, течениях и уровне моря) применительно к произвольному району Мирового океана ограничено тем, что существующие наблюдения, как правило, нерегулярны или могут вообще отсутствовать. В частности, спутниковые данные имеют непрерывную продолжительность 20–25 лет, причем над одной и той же акваторией (а тем более, точкой) спутник может пролетать через несколько суток (следовательно, эквидистантность измерений неочевидна), данные автоматических буев имеют продолжительность 5–15 лет, однако они, в основном, находятся в прибрежных районах. Наблюдения на береговых гидрометеостанциях могут иметь длительность 100 и более лет, однако они не отражают специфики гидрометеорологических процессов в открытом море. Поэтому применительно к задаче расчета экстремальных гидрометеорологических явлений современная концепция [7] постулирует следующие допущения.

- Основным источником данных об океанографических процессах являются результаты расчетов по гидродинамическим моделям динамики океана. Такой подход позволяет, используя данные реанализа метеорологических полей как входные данные, получать информационные массивы океанографических характеристик непрерывной продолжительностью несколько десятков лет.
- Для статистического оценивания экстремальных характеристик, возможных 1 раз в T лет, используется система стохастических моделей, описывающих совместную многомасштабную (синоптическую, сезонную, межгодовую) изменчивость пространственно-временных полей океанографических характеристик. Это позволяет методом Монте-Карло воспроизвести ансамбль их реализаций, таким образом, экстраполируя значения экстремумов на заданный временной интервал.
- Экстремальность гидрометеорологического явления по отношению к конкретному объекту определяется интегральной совокупностью всех факторов путем рассмотрения функций риска, специфичных для определенных классов морских объектов и сооружений.

В настоящее время доступность глобальных (NCEP/NCAR, ERA-40) и региональных (SMHI, JRA25, NMI и др.) массивов реанализа метеорологических полей, эволюция гидродинамических моделей и развитие высокопроизводительных вычислительных технологий сделали возможной практическую реализацию данной концепции в полном объеме. Практическая реализация рассматриваемой концепции моделирования предусматривает последовательное решение трех задач ((а) подготовки и усвоения метеорологических данных, (б) гидродинамического моделирования, (в) статистического ана-

лиза, моделирования и расчета экстремумов), объединенных в вычислительную цепочку. Каждая задача дополнительно распадается на систему взаимосвязанных модулей (см. рис. 1).

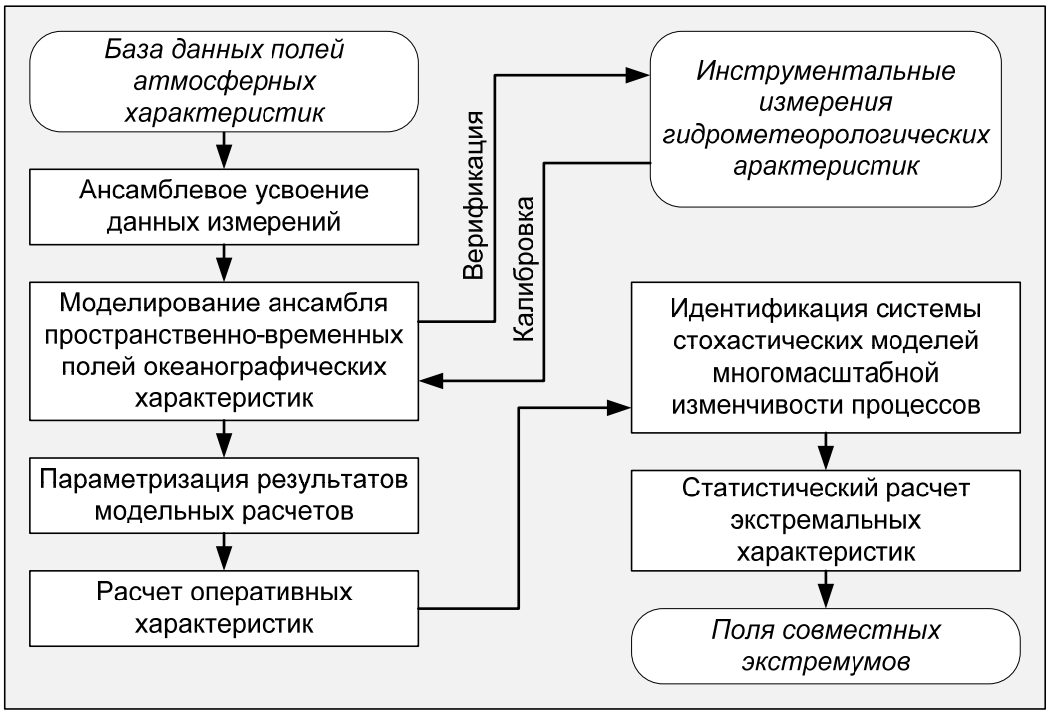


Рис. 1. Концепция расчета экстремальных гидрометеорологических характеристик с использованием компьютерного моделирования

Этапы расчета экстремальных гидрометеорологических характеристик: методы, модели и параллельные алгоритмы

В соответствии с рис. 1, первым этапом является подготовка массива метеорологической информации. В качестве входных данных для расчета океанографических характеристик (ветрового волнения, течений и уровня моря) могут быть использованы поля атмосферного давления и скорости приводного ветра на основе данных реанализа, в которые усваиваются данные высокоточных инструментальных измерений. Процедура усвоения [11] может проводиться как для каждой метеорологической величины независимо, так и путем пересчета одних величин через другие. Так, приводный ветер может быть определен из геострофического соотношения (по полям приземного атмосферного давления с усвоением) с учетом кривизны изобар, агеострофики и приводного трения. Наиболее ресурсоемким элементом таких расчетов является собственно определение геострофического ветра и радиусов кривизны изобар. Эта процедура может быть эффективно распараллелена по времени (распределяя T лет данных между разными вычислителями) в рамках модели как общей, так и распределенной памяти. В последнем случае распределение данных между узлами выполняется с перекрытием, обусловленным конечной разностью второго порядка при расчете временной производной поля давления.

Второй этап включает в себя формирование массива полей океанографических характеристик, используя гидродинамические модели. Используя массив метеорологической информации как входные данные, выполняется совместное гидродинамическое моделирование волнения, течений и уровня моря, что обусловлено необходимостью учета фактической глубины акватории по разгону волн. В том случае, когда штормовые колебания уровня моря сопровождаются развитым ветровым волнением, этот эффект

может привести к появлению существенно больших волн, чем предельно возможные при среднем многолетнем положении уровня моря.

Гидродинамическая модель морского волнения в спектральной форме представляется в виде уравнения баланса волновой энергии:

$$\frac{\partial N}{\partial t}+\frac{\partial N}{\partial \varphi}\dot{\varphi}+\frac{\partial N}{\partial \theta}\dot{\theta}+\frac{\partial N}{\partial k}\dot{k}+\frac{\partial N}{\partial \beta}\dot{\beta}+\frac{\partial N}{\partial \omega}\dot{\omega}=G\,. \qquad (1)$$

Здесь N – спектральная плотность волнового действия; она является функцией от широты φ, долготы θ, волнового числа k и угла β между направлением волнового вектора и параллелью, а также от частоты ω и времени t. Это уравнение связывает между собой явления притока энергии от ветра, диссипации и ее перераспределения и нелинейного взаимодействия между частотными составляющими процесса волнения. Чаще всего функция источника G записывается в виде суммы трех компонент, $G=G_{in}+G_{nl}+G_{ds}$ (поступления энергии от ветра к волнам, слабонелинейного взаимодействия в спектре ветрового волнения и диссипации волновой энергии соответственно).

Для расчетов уровня моря и течений в общем случае используется трехмерная гидродинамическая бароклинная модель со свободной поверхностью. Исходная система уравнений в декартовой системе координат в приближении гидростатики и плоскости записывается в следующем виде:

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0\,, \qquad (2)$$

$$\frac{\partial u}{\partial t}+\frac{\partial}{\partial x}(uu)+\frac{\partial}{\partial \mathrm{y}}(vu)+\frac{\partial}{\partial \mathrm{z}}(wu)-fv=-\frac{1}{\rho}\frac{\partial p}{\partial x}+N_h\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+\frac{\partial}{\partial \mathrm{z}}\left(N_z\frac{\partial u}{\partial z}\right), \qquad (3)$$

$$\frac{\partial \mathrm{v}}{\partial t}+\frac{\partial}{\partial x}(uv)+\frac{\partial}{\partial y}(vv)+\frac{\partial}{\partial z}(wv)+fu=-\frac{1}{\rho}\frac{\partial \mathrm{p}}{\partial \mathrm{y}}+N_h\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)+\frac{\partial}{\partial \mathrm{z}}\left(N_z\frac{\partial u}{\partial z}\right), \qquad (4)$$

$$\frac{\partial p}{\partial z}=\rho g\,, \qquad (5)$$

$$\frac{\partial T}{\partial t}+\frac{\partial}{\partial x}(uT)+\frac{\partial}{\partial y}(vT)+\frac{\partial}{\partial z}(wT)=\frac{\partial}{\partial \mathrm{z}}\left(K_z\frac{\partial T}{\partial z}\right), \qquad (6)$$

$$\frac{\partial S}{\partial t}+\frac{\partial}{\partial x}(uS)+\frac{\partial}{\partial y}(vS)+\frac{\partial}{\partial z}(wS)=\frac{\partial}{\partial \mathrm{z}}\left(K_z\frac{\partial S}{\partial z}\right), \qquad (7)$$

$$\rho=\Psi(T,S,p)\,. \qquad (8)$$

Здесь f – параметр Кориолиса; p – давление; ρ – плотность; T – температура, S – соленость, u,v,w – составляющие поля скорости течений по осям (x,y,z), N_z, N_h – коэффициенты вертикальной и горизонтальной вязкости; K_z – коэффициент вертикальной диффузии. Для приливных морей в правой части (3)–(4), (6)–(7) дополнительно учитывается приливная составляющая. Уравнения (1)–(8) интегрируются совместно. Распараллеливание вычислительного алгоритма возможно несколькими путями. Традиционный для вычислительной гидродинамики подход (в рамках модели общей памяти) реализует конвейерный параллельный подход к организации рабочего цикла [12]. Это объясняется учетом зависимостей, возникающих в поле расчетных характеристик между точками пространства, что делает невозможным способ простой доменной декомпозиции циклов. Как следствие, это требует предварительного выявления направлений зависимостей. Альтернативой такому подходу к распараллеливанию (1)–(8) является естественное, или физическое распараллеливание на основе декомпозиции по времени. Временная связность исследуемых процессов требует выполнения дополни-

тельных расчетов, осуществляющих, по сути, перекрытия между последовательными интервалами времени, на которых выполняется моделирование волнения, течений и уровня моря в параллельном режиме. Длительность перекрытия зависит от физических особенностей моделируемого процесса и специфики акватории и составляет 3–7 суток для модели волнения и 1–3 месяца для модели течений и уровня моря.

В третий этап включаются процедуры статистической обработки расчетных данных, идентификации стохастических моделей и расчета экстремумов. Статистическая обработка информационной базы гидрометеорологических характеристик выполняется средствами многомерного ста тистического анализа (МСА) пространственно-временных полей. Для расчета экстремальных гидрометеорологических явлений, возможных 1 раз в T лет, и оценки соответствующих рисков от их воздействий на морские объекты и сооружения используется авторский метод BOLIVAR (см. табл. 1). В рамках этого метода последовательность значений гидрометеорологического процесса (модуля скорости ветра и течений, высоты волн и уровня моря) в фиксированной точке (x, y) рассматривается как набор выборок, состоящих из максимумов h_{ij}^{+} в n самых сильных штормах в i-м году в течение T лет ($i = 1,...,T; j = 1,...,n$). Наибольшее значение $h_{\max}$, возможное 1 раз в T лет, является крайним членом выборки. Порядковые статистики h_{ij}^{+} являются оценками квантилей y_p распределения максимумов процесса в штормах, их вероятностные свойства описываются совместной функцией распределения

$$G(y_1,\ldots,y_n) = P\left\{h_{i1}^{+} < y_1, \ldots, h_{in}^{+} < y_n\right\}, \qquad (9)$$

называемой квантильной функцией. Квантильная функция (9) несет, в принципе, всю информацию, необходимую для точечного и интервального оценивания экстремальных характеристик, возможных в заданное число лет. Однако ее определение в аналитическом виде в общем случае невозможно, что требует применения метода Монте-Карло на основе системы стохастических моделей, идентифицированных по данным гидродинамического моделирования. Учет многомасштабной (мелкомасштабной, синоптической, сезонной, межгодовой) изменчивости требует использования стохастических моделей разных классов, подробно описанных в [12]. Для их распараллеливания применяются подходы, основанные на декомпозиции по статистическому ансамблю, применении принципа перемешивания, а также декомпозиции по индексирующей переменной (например, времени t в задаче декомпозиции массива спектров волнения) [13]. В общем случае ресурсоемкость процедуры стохастического моделирования зависит от количества испытаний Монте-Карло (до 10^8–10^9), числа расчетных точек акватории, а также размерности задачи (количества характеристик, определяющих экстремальное гидрометеорологическое явление).

Оценка вычислительной сложности технологии

Из предыдущего раздела следует, что в настоящее время разработаны основные модели, методы и вычислительные алгоритмы, успешно реализующие каждый из элементов концепции, отраженной на рис. 1. Это позволяет рассмотреть возможность их совместной реализации в рамках единой технологии получения информации о совместных характеристиках экстремальных гидрометеорологических явлений на основе компьютерного моделирования. При этом существенной проблемой является вычислительная сложность, или ресурсоемкость выполняемых расчетов. Например, в [14] показано, что ресурсоемкость расчетов характеристик экстремальных явлений с периодом повторяемости не более 100 лет для акватории Каспийского моря составляет около 120 ПФлоп. Иными словами, для выполнения всех расчетов за одни сутки необходима реальная производительность вычислительных мощностей 1,4 ТФлопс. В табл. 2 при-

ведены характеристики ресурсоемкости основных вычислительных операций, применительно к выполнению расчетов для Каспийского моря [12]. Как следствие, практическая реализация разработанной технологии требует эффективного использования вычислительных систем терафлопной производительности, имеющих несколько сотен или даже тысяч процессоров.

Таблица 2. Характеристики ресурсоемкости вычислительных операций

№	Операция	Объем генерируемых данных	Относительное время выполнения
1	Усвоение метеорологической информации	$1 \cdot 10^{10}$	6%
2	Расчет полей характеристик морского волнения	$5 \cdot 10^{13}$	28%
	Расчет полей морских течений и уровня		40%
3	Параметризация частотно-направленных спектров	$3 \cdot 10^{12}$	13%
	Стохастическое моделирование экстремумов	$3 \cdot 10^{15}$	10%
4	Расчет функций риска	$5 \cdot 10^{7}$	3%

Заключение

Таким образом, появление информационных массивов реанализа метеорологических полей, эволюция вычислительных гидродинамических моделей динамики океана и развитие методов, средств и технологий высокопроизводительных вычислений сделали возможным воплощение новой концепции расчета экстремальных гидрометеорологических характеристик на основе данных компьютерного моделирования. В настоящее время разработаны *все* вычислительные модели, методы и параллельные алгоритмы, реализующие базовые элементы данной концепции. Однако принципиальной задачей, требующей применения вычислительных систем терафлопного диапазона производительности, является создание интегрированной вычислительной технологии и разработки высокопроизводительного программного комплекса на ее основе. Решению этих задач посвящены следующие статьи данного цикла [14, 15].

Литература

1. Справочные данные по режиму ветра и волнения Баренцева, Охотского и Каспийского морей; Под ред. Лопатухина Л.И. и др. – Российский Морской регистр судоходства. – СПб, 2003. – 214 с.
2. Бухановский А.В., Лопатухин Л.И., Чернышева Е.С. Подходы, опыт и некоторые результаты исследований волнового климата океанов и морей. II. Расчет волнения по гидродинамическим моделям, режимные распределения и климатические спектры волн // Вестник СПбГУ. – 2005. – Сер. 7. – Вып. 4. – С. 56–69.
3. Справочные данные по режиму ветра и волнения Балтийского, Северного, Черного, Азовского и Средиземного морей / Лопатухин Л.И. и др. Российский Морской регистр судоходства. – СПб, 2006. – 450 с.
4. Володин Е.М., Толстых М.А. Параллельные вычисления в задачах моделирования климата и прогноза погоды // Вычислительные методы и программирование. – 2007. – Т. 8. – С. 113–122.
5. Joppich W., Quaas J. Coupling general circulation models on a meta-computer // Lecture Notes in Computer Science. – 2003. – Vol. 2657. – P. 161–170.
6. Stankova E., Zatevakhin M. Numerical simulation of cloud dynamics and microphysics // Lecture Notes in Computer Science. – 2003. – Vol. 2657. – P. 171–178.

7. Мирзоев Д.А. и др. Концепция обеспечения специализированной гидрометеорологической информацией проектирования сооружений на шельфе арктических морей // Труды Четвертой Международной конференции «Освоение шельфа арктических морей» (RAO 99). – СПб, 1999.
8. Бухановский А.В., Лопатухин Л.И., Рожков В.А. Подходы, опыт и некоторые результаты исследований волнового климата океанов и морей. III. Экстремальные и необычные волны // Вестник СПбГУ. – 2006. – Сер. 7. – Вып. 4. – С. 58–69.
9. Lopatoukhin L.J. et al. Estimation of extreme wind wave heights // World Meteorological Organization. WMO/TD. – 2000. – № 1041. – 76 p.
10. Рожков В.А. и др. Моделирование штормового волнения // Физика атмосферы и океана. – 2000. – Т. 36. – № 5. – С. 689–699.
11. Бухановский А.В., Лопатухин Л.И., Иванов С.В. Подходы, опыт и некоторые результаты исследований волнового климата океанов и морей. I. Постановка задачи и входные данные // Вестник СПбГУ. – Сер. 7. – 2005. – Вып. 3. – С. 62–74.
12. Бухановский А.В. и др. Моделирование экстремальных явлений в атмосфере и океане как задача высокопроизводительных вычислений // Вычислительные методы и программирование. – 2008. – Том 9. – С. 141–153.
13. Bogdanov A.V., Boukhanovsky A.V. High performance parallel algorithms for data processing // Lecture Notes in Computer Science. – 2004. – Vol. 3036. – P. 239–246.
14. Ковальчук С.В. и др. Высокопроизводительный программный комплекс моделирования экстремальных гидрометеорологических явлений. Часть II: Разработка и оценка программной архитектуры // Научно-технический вестник СПбГУ ИТМО. – 2008. – Вып. 54. – С. 64–71.
15. Иванов С.В. и др. Высокопроизводительный программный комплекс моделирования экстремальных гидрометеорологических явлений. Часть III: Интерпретация и использование результатов расчетов // Научно-технический вестник СПбГУ ИТМО. – 2008. – Вып. 54. – С. 72–79.

Бухановский Александр Валерьевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, д.т.н., профессор, boukhanovsky@mail.ifmo.ru

Иванов Сергей Владимирович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, м.н.с., svivanov@mail.ifmo.ru

Ковальчук Сергей Валерьевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, м.н.с., kovalchuk@mail.ifmo.ru

Зильберштейн Олег Иделевич – ГУ «Гидрометцентр России», зав. лаб., к.ф.-м.н., lmpi@yandex.ru

Попов Сергей Константинович – ГУ «Гидрометцентр России», ст.н.с., к.ф.-м.н., lmpi@yandex.ru

Лопатухин Леонид Иосифович – Санкт-Петербургский государственный университет, д.г.н., профессор, leonid-lop@yandex.ru

Чумаков Михаил Михайлович – ГУ «Гидрометцентр России», ст.н.с., к.ф.-м.н., lmpi@yandex.ru

УДК 004.021, 681.324

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЙ ПРОГРАММНЫЙ КОМПЛЕКС МОДЕЛИРОВАНИЯ ЭКСТРЕМАЛЬНЫХ ГИДРОМЕТЕОРОЛОГИЧЕСКИХ ЯВЛЕНИЙ. ЧАСТЬ II: РАЗРАБОТКА И ОЦЕНКА ПРОГРАММНОЙ АРХИТЕКТУРЫ

С.В. Ковальчук, С.В. Иванов, А.В. Бухановский

Рассматриваются особенности проектирования программного комплекса моделирования экстремальных гидрометеорологических явлений в рамках парадигмы сервисно-ориентированной архитектуры. На основе прототипирования и анализа параметрических моделей производительности показано, что выбор способа параллельной декомпозиции для каждого из вычислительных модулей в рамках комплекса определяется особенностями постановки задачи и спецификой вычислительной архитектуры. Предложен подход к динамической организации вычислительно эффективной архитектуры путем введения интеллектуального управляющего сервиса. Представлены результаты оценки архитектуры на основе вычислительных экспериментов.

Ключевые слова: высокопроизводительные вычисления, архитектура параллельных приложений, управление параллельной производительностью.

Введение

Высокоуровневое проектирование (или разработка архитектуры программного обеспечения, ПО) ставит своей целью формализацию и обоснование внутренней структуры ПО, которая наилучшим образом удовлетворяет проектным требованиям при заданном наборе ограничений в области работоспособности, безопасности, безотказности, защищенности [1]. Эволюция способов формализации структуры ПО определяется, в первую очередь, потребностью снижения сложности процессов разработки, тестирования и поддержки программных продуктов на фоне увеличения общей сложности решаемых задач. Это последовательно порождает структурный [2], объектно-ориентированный [3] и компонентно-ориентированный [4] подходы в инженерии ПО. Дальнейшая тенденция функциональной изоляции компонентов отражается в архитектурах на базе SOA (Service Oriented Architecture) [5], REST (Representational State Transfer) [6], AOA (Aspect Oriented Architecture) [7] и др. Такие подходы достаточно успешно используются при разработке коммерческих приложений и часто могут определять весь технологический процесс создания ПО [8].

При разработке наукоемкого ПО (scientific computing software engineering) необходимо учитывать, что приоритеты проектных требований, их структура, процесс проектирования, создания, проверки, внедрения, поддержки и использования обладают рядом индивидуальных особенностей [9]. Во многих случаях важным критерием работоспособности ПО является производительность (productivity), характеризующая получение результата вычислений с приемлемой точностью за заданное время. Она обеспечивается за счет применения технологий высокопроизводительных вычислений в многопроцессорных и/или распределенных средах.

Использование традиционных компонентно-ориентированных подходов к проектированию распределенных программных систем (на основе DCOM [10], CORBA [11], Enterprise Java Bean [12] и пр.) допустимо не во всех случаях, поскольку они не ориентированы непосредственно на достижение наибольшей производительности. Несмотря на то, что для наукоемкого ПО предлагаются специфические проблемно-ориентированные подходы, в частности CCA (Common Component Architecture) [13], в общем случае архитектура таких систем определяется особенностями решаемой задачи, спецификой предметной области и характеристиками параллельной вычислительной системы. В данной статье обсуждаются особенности проектирования программного

комплекса моделирования экстремальных гидрометеорологических явлений на основе концепции, изложенной в [14].

Сервисно-ориентированная архитектура программного комплекса

Практическая реализация концепции расчета характеристик экстремальных гидрометеорологических явлений требует сочетания в одном программном комплексе набора взаимосвязанных функциональных компонент, отвечающих за основные этапы гидродинамического и статистического моделирования [14]. Поскольку компоненты основаны на различных математических моделях и используют различные программные технологии, то принципиальной проблемой является организация взаимодействия (как между компонентами, так и внутри них) таким образом, чтобы обеспечить наиболее эффективное использование ресурсов вычислительной системы.

Для решения этой задачи взаимодействие вычислительных модулей программного комплекса осуществляется на основе сервисно-ориентированной архитектуры (SOA). Это позволяет добиться максимального уровня изоляции частей программного комплекса, тем самым делая более строгой его структурированность и облегчая процесс разработки и тестирования. На рис. 1 представлена общая структура взаимодействия сервисов программного комплекса. Для нее характерна следующая типизация сервисов:

– управляющие сервисы (композиция, декомпозиция, управления задачей, управление вычислительными узлами);
– сервисы доступа к данным (входная гидрометеорологическая информация, результаты расчетов);
– вычислительные сервисы (модели подготовки метеорологической информации, гидродинамические и стохастические модели, статистическая обработка и расчет экстремумов).

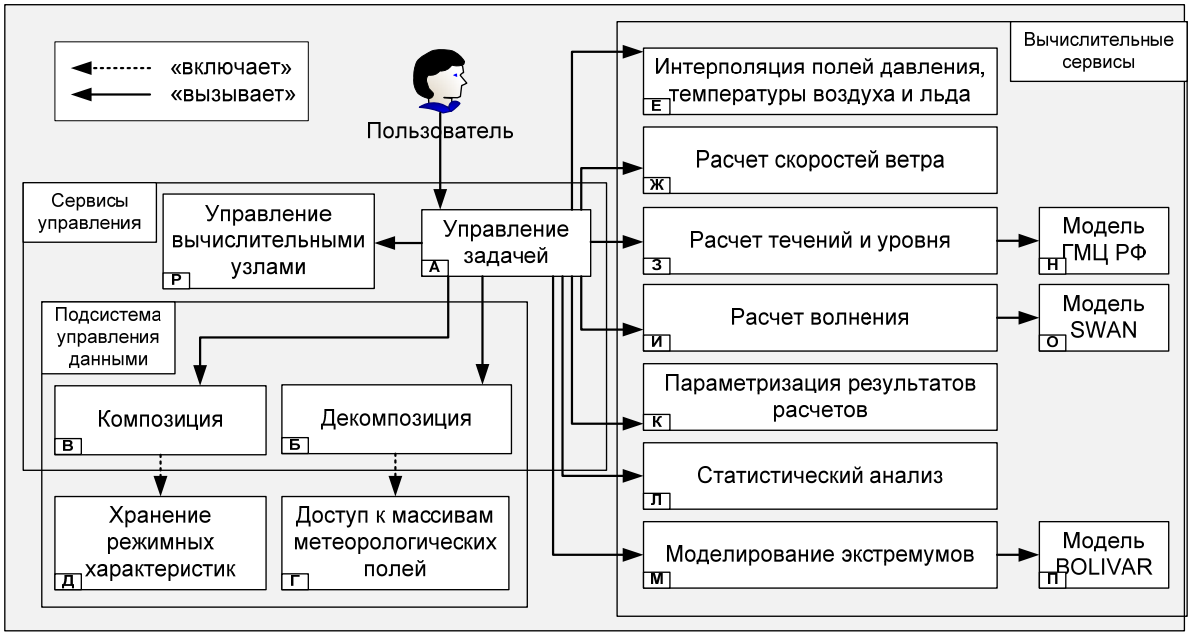


Рис. 1. Архитектура программного комплекса

В общем случае функциональные компоненты, указанные на рис. 1, могут быть представлены в виде совокупности сервисов, исполняющихся на отдельных узлах вычислительной системы и обеспечивающих наиболее полное использование вычислительных ресурсов.

Для оценки предложенной архитектуры на этапе проектирования выполнено прототипирование и моделирование производительности разрабатываемого программного обеспечения для прогнозирования особенностей его использования и формализации требований. Так, можно приближенно описать время выполнения произвольного сервиса на p вычислителях с общей памятью (внутри одного узла) в форме [15]

$$T_{calc}(p,\alpha,\gamma)_{omp} = T_0\left[\left[\alpha(p-1)+\frac{1}{p}\right]\cdot\gamma+(1-\gamma)\right], \qquad (12)$$

где α – параметр, характеризующий рост накладных расходов с увеличением количества вычислителей, а γ – доля параллельных вычислений. Общее время коммуникаций между узлами определено как сумма времени рассылки и сбора данных, а также времени на передачу перекрытий между независимыми блоками данных:

$$T_{comm} = t_w\left[\left(3L_0\tau_0+\Pi\mu L\right)+\Pi L\cdot\vartheta+\left(\eta\xi L+\nu(\xi L)^2\right)(\Pi-1)\right], \qquad (13)$$

где τ_0, L_0 и τ, L – число узлов расчетной сетки по времени и пространству для метеорологических (входных) и океанографических (выходных) данных соответственно, Π – количество узлов, η, ν, ξ, μ – коэффициенты, характеризующие кратность использования данных в моделях, а ϑ – доля репликации данных между узлами. Величина t_w соответствует времени передачи единицы информации по коммуникационной сети. Предполагается, что время работы T_0 каждой задачи (расчета ветра, волн, течений, параметризации спектров и расчета статистики) определяется только производительностью вычислителя (например, ядром CPU) и объемом обрабатываемых данных; выражения для времени работы конкретных вычислительных модулей приведены в [15].

На рис. 2 показано влияние архитектуры программной системы на общее ускорение вычислений. При небольшом количестве узлов оптимальным является распараллеливание по времени [14] с использованием отдельных процессоров узлов с общей памятью как независимых вычислителей. Однако с увеличением числа процессоров тенденция изменяется, и многопроцессорные узлы становятся более предпочтительными. С увеличением размера входного массива (число моделируемых лет, рис. 2, а–б) в результате увеличения гранулярности общее ускорение также возрастает. Рис. 2, в–г, иллюстрирует влияние перекрытия независимых блоков данных (что является обязательным при распараллеливании по времени) на производительность алгоритма. При уменьшении величины перекрытия с трех до одного месяца ускорение в среднем вырастает примерно в полтора раза, а эффективность от использования многопроцессорных узлов сдвигается в зону большего числа узлов и меньшего числа процессоров на узле. Эта характеристика может меняться в зависимости от специфики акватории (океанские процессы обладают большим «временем жизни», чем процессы в закрытых морях), а также от особенностей моделируемого процесса (синоптический интервал атмосферных процессов и волнения составляет несколько суток, а эволюция вихрей водных масс может длиться несколько месяцев). Из рис. 2 следует, что в рамках данной задачи невозможно предложить статический способ распараллеливания вычислительных модулей комплекса, который был бы одинаково эффективен для всех вариантов входных данных при заданных характеристиках вычислительной системы. Эта проблема решается путем внедрения в состав программного комплекса (рис. 1) управляющего сервиса специального вида, который, основываясь на характеристиках задачи, используя данные текущего мониторинга состояния вычислительной системы и знания о параллельной производительности расчетных модулей, организует процедуры параллельной декомпозиции и композиции данных таким образом, чтобы совокупная производительность программного комплекса была наилучшей.

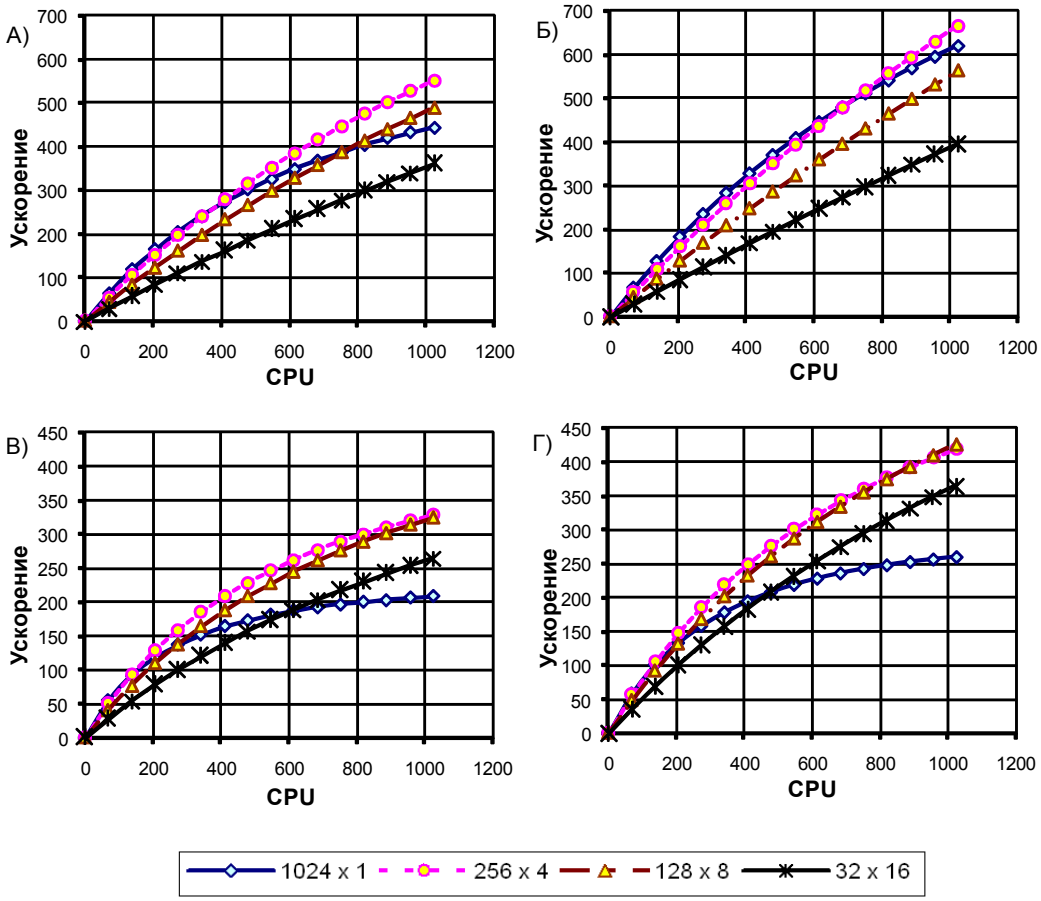


Рис. 2. Параллельное ускорение в зависимости от конфигурации вычислительных узлов, длины входного массива и размера перекрытия независимых блоков данных: а) 15 лет, перекрытие 1 месяц; б) 30 лет, перекрытие 1 месяц; в) 5 лет, перекрытие 3 месяца; г) 5 лет, перекрытие 1 месяц. В легенде – схема распараллеливания: число процессов × число потоков

Интеллектуальный механизм управления параллельной производительностью

Проблема формирования *динамической архитектуры* высокопроизводительного программного комплекса в условиях неопределенности, обусловленной отсутствием надежных теоретических соотношений, описывающих производительность его компонентов, решается средствами искусственного интеллекта. Для этого в архитектуру на рис. 1 вводится дополнительный управляющий сервис, реализующий интеллектуальную подсистему управления параллельными вычислениями. Такая модификация обусловлена необходимостью автоматизации процесса принятия решения на базе знаний, определяющих возможности используемых вычислительных сервисов, а также данных, характеризующих как решаемую в данный момент задачу, так и используемую программно-аппаратную вычислительную среду. Оптимальность данного решения оценивается временем, затраченным на вычисления, проводимые по выбранной схеме. Сложность принятия решений в данном случае определяется разнообразием способов распараллеливания для каждого из используемых вычислительных модулей, отсутствием надежных моделей, описывающих параллельную производительность, а также динамикой вычислительной среды (изменение приоритетов работающих процессов, производительности узлов, пропускной способности каналов и пр.). На рис. 3 представлен общий принцип работы интеллектуальной подсистемы и ее основные компоненты.

В процессе работы механизма вывода подсистема, оперируя знаниями экспертов (включенных в описание конкретных вычислительных сервисов), на основании имеющихся данных производит оценку весов ребер и вершин графа параллельных реализаций, варьируя вариантами распараллеливания отдельных вычислительных моделей (рис. 4). Варианты распараллеливания каждой из моделей строятся с использованием технологий параллельного программирования различных уровней (управление потоками и процессами). Каждому из вариантов распараллеливания соответствует своя специфика поведения кривой ускорения на используемой архитектуре. Для перехода к классическому представлению взвешенного графа достаточно заменить вершины ребрами с соответствующим весом, что позволит применять к построенному графу без изменения классические алгоритмы. Очевидно, что кратчайший путь в графе от узла «Начало» до узла «Конец» определит оптимальную (согласно модельным оценкам) схему распараллеливания вычислительных модулей и последовательности их запуска. В силу неопределенности во входных данных в результате анализа выявляется несколько конурирующих, или «допустимых» путей, которые потом ранжируются.

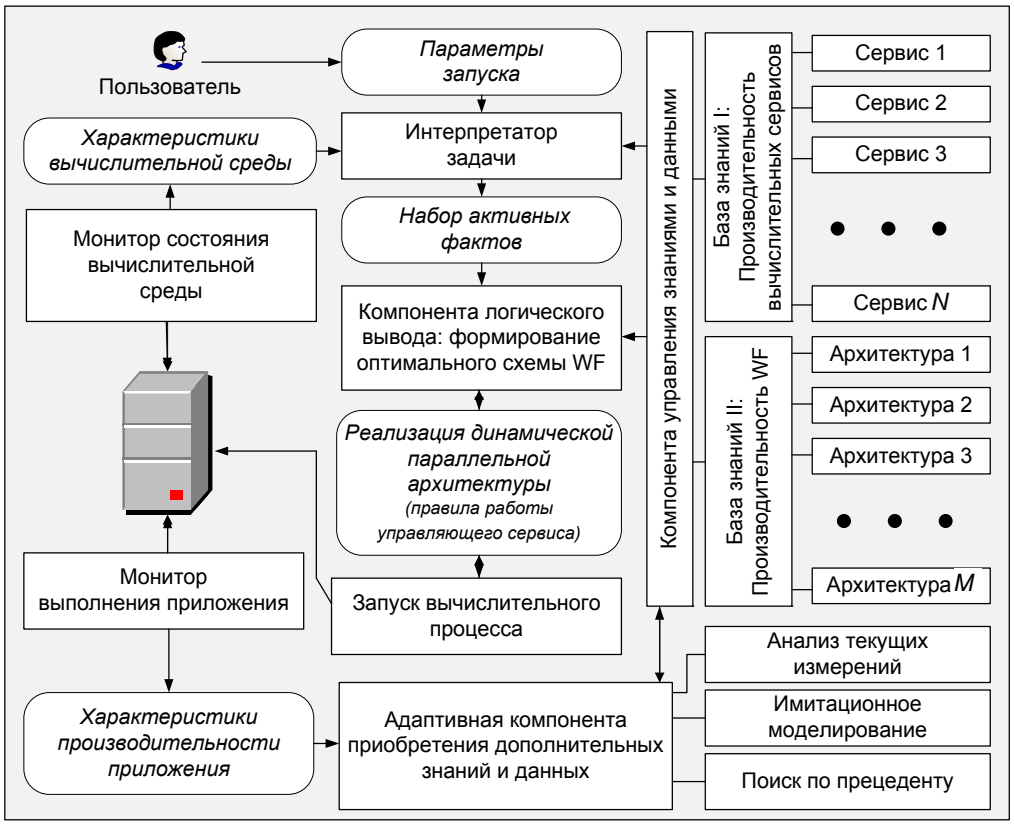


Рис. 3. Структура интеллектуальной подсистемы

В табл. 1 приведен пример анализа оптимальной схемы распараллеливания, построенной посредством обхода графа на рис. 4, на основании результатов численных экспериментов, выполненных на кластере со 128 вычислителями (рассчитывались характеристики для Каспийского моря при T=5). Из таблицы видно, что для различных вычислительных модулей в качестве оптимальной (в рамках работы комплекса в целом) выбраны различные схемы распараллеливания (количество узлов × количество логических процессоров на узле × количество потоков на каждом из процессоров). Примечательным является тот факт, что использование комбинации для различных методов распараллеливания позволяет существенно повысить масштабируемость не только комплекса в целом, но и отдельных расчетных модулей. Например, для международной модели морского волнения SWAN в расчетном примере параллельное ускоре-

ние составляет около 84 раз на 128 вычислителях, в то время как авторами этой модели указывается, что даже для систем с общей памятью эта модель эффективно не масштабируется более, чем на 64 вычислителя [16]. С увеличением временного интервала T степень масштабируемости вычислений возрастает.

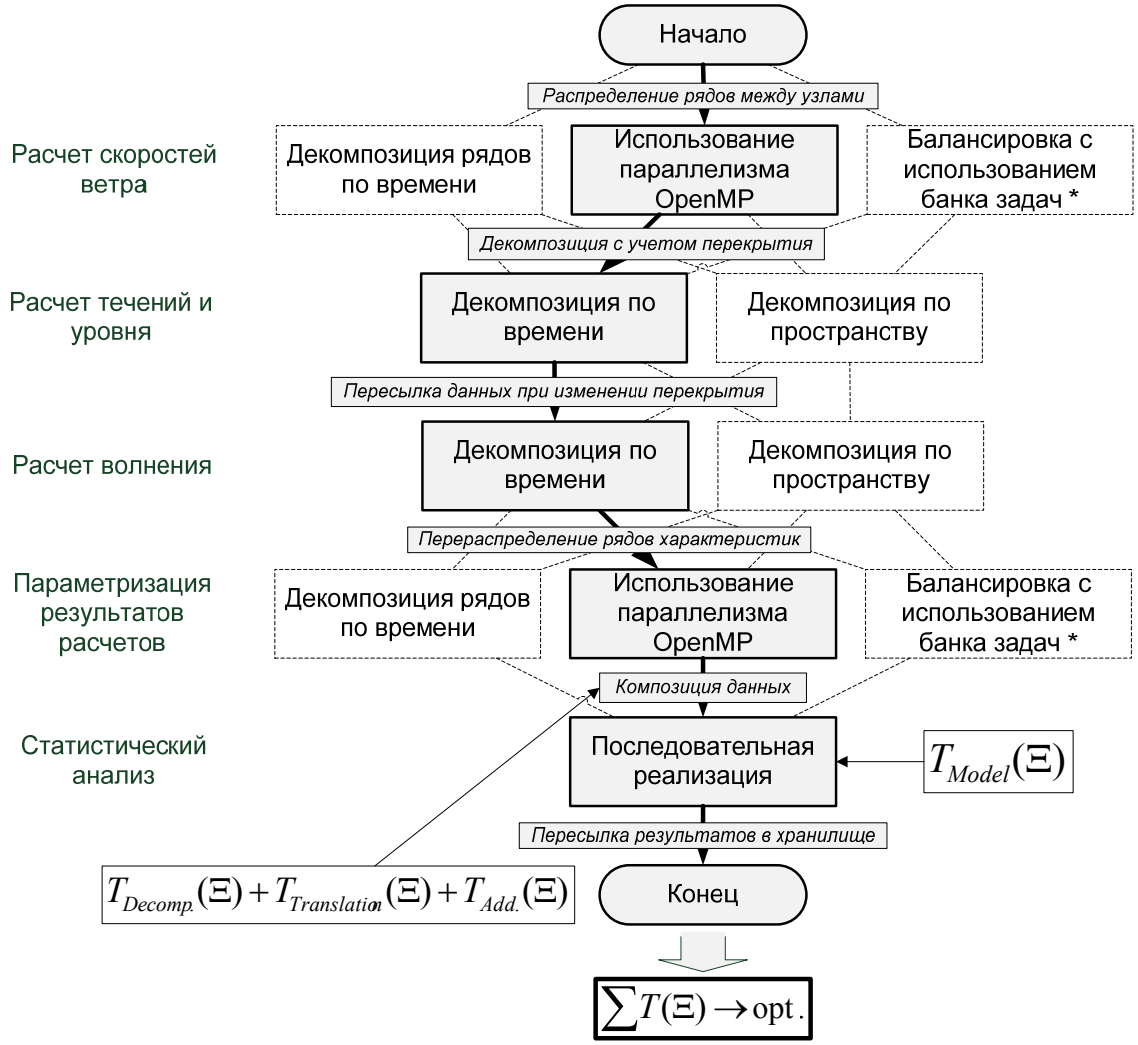


Рис. 5. Анализ графа параллельных реализаций в интеллектуальной подсистеме управления параллельной производительностью

Таблица 1. Анализ оптимальной схемы распараллеливания по данным вычислительных экспериментов (общее ускорение на 128 вычислителях – 83 раза)

Функциональный компонент	Схема распараллеливания	Модельное время работы, с	Измеренное время работы, с	Относительная ошибка, %	Ускорение
Расчет ветра	16×8×1	106	102	4	71
Модель SWAN	16×1×8	4749	4699	1	84
Модель течений и уровня	16×8×1	1999	2175	8	70
Статистическое обобщение	16×2×4	726	719	1	127

Организация доступа к данным

Поскольку функционирование комплекса связано с оперированием большими объемами информации, в рамках данной задачи применяется технология REST, ориен-

тированная на обеспечение эффективного представления ресурсов [17]. Доступ к ресурсам обладает фиксированным интерфейсом, что обеспечивает унификацию ресурсов и упрощает взаимодействие с пользователем. Совместное использование технологий SOA и REST позволяет добиться организации унифицированного доступа к различным данным, используемым программным комплексом. На рис. 1 ряд рассмотренных сервисов (подсистема управления данными) отвечает за организацию доступа к большим объемам данных, необходимых для работы программного комплекса. Несмотря на то, что для коммерческого программного обеспечения традиционным способом хранения данных являются реляционные базы данных, для представления расчетной гидрометеорологической информации эффективнее использовать файловые форматы GRIB и NetCDF. Однако каждому из расчетных файлов сопоставляются метаданные (характеристики сетки, условия расчета и пр.), которые допустимо хранить в реляционной базе данных.

В процессе организации взаимодействия системы с разделяемой памятью задача повышения эффективности коммуникаций решается посредством применения механизмов сериализации–десериализации. Они позволяют осуществить преобразование объектов, доступных программному коду, к форме, удобной для передачи посредством механизма потоков. Это дает возможность оптимизировать процесс передачи данных по сети за счет сжатия, синхронизации и дополнительного контроля [18].

Заключение

Таким образом, в работе обоснована онтология сервисов, необходимых для формализации структуры программного комплекса моделирования экстремальных гидрометерологических явлений на базе SOA. Предложен принцип организации интеллектуальной подсистемы отображения параллельных вычислительных алгоритмов на заданную аппаратную архитектуру с целью достижения наибольшей производительности; разработаны механизмы оптимизации времени выполнения задачи на основе обхода графа параллельной реализации расчетной схемы. Продемонстрирована эффективность предложенных методов организации динамической параллельной архитектуры программного комплекса на примере расчетов экстремальных характеристик ветра, волнения, течений и уровня Каспийского моря. В работе [19] обсуждаются вопросы интерпретации и использования полученных результатов.

Литература

1. Соммервилл И. Инженерия программного обеспечения. – 6–е изд. / Пер. с англ. – М.: Издательский дом «Вильямс», 2002.– 624 с.
2. Вирт Н. Алгоритмы + структуры данных = программы. – М.: Мир, 1985. – 406 с.
3. Буч Г. Объектно–ориентированный анализ и проектирование с примерами приложений на C++, 2–е изд. / Пер. с англ. – М.: «Бином». – СПб: «Невский диалект», 1999. – 560 с.
4. Szyperski C. Component Software: Beyond Object–Oriented Programming. – Addison–Wesley Professional, 1998. – 411 p.
5. Lublinsky B. Defining SOA as an architectural style. 9 January 2007. – Режим доступа: http://www.ibm.com/developerworks/architecture/library/ar–soastyle/, свободный.
6. Fielding R.T. Architectural Styles and the Design of Network–based Software Architectures: Doctoral thesis / University of California. – Irvine, 2000. – 162 p.
7. Navasa A., Pérez M.A., Murillo J.M. Aspect Modelling at Architecture Design // Lecture Notes in Computer Science. – 2005. – Vol. 3527. – P. 41–58.

8. Якобсон А., Буч Г., Рамбо Дж. Унифицированный процесс разработки программного обеспечения. – СПб: Питер, 2002.– 496 с.
9. Carver J.C., Kendall R.P., Squires S.E., et al. Software Development Environments for Scientific and Engineering Software: A Series of Case Studies // Proceedings of the 29th International Conference on Software Engineering. – Washington, DC, USA. – IEEE Computer Society Press, 2007. – P. 550–559.
10. Sessions R. COM and DCOM: Microsoft's Vision for Distributed Objects. – New York, USA: John Wiley & Sons, 1998.– 492 p.
11. Object Management Group. OMG's CORBA website. – Режим доступа: http://www.corba.org, свободный.
12. Matena V., Stearns B. Applying Enterprise JavaBeans: Component–Based Development for the J2EE Platform. – Prentice Hall, 2000. – 464 p.
13. Bernholdt D.E., Elwasif W.R., Kohl J.A., Epperly T.G.W. A Component Architecture for High–Performance Computing // Proceedings of the Workshop on Performance Optimization via High–Level Languages and Libraries, New York, USA, 22 June 2002. – Режим доступа: http://www.ece.lsu.edu/jxr/pohll–02/papers/bernholdt.pdf , свободный.
14. Бухановский А.В. и др. Высокопроизводительный программный комплекс моделирования экстремальных гидрометеорологических явлений. Часть I: Постановка задачи, модели, методы и параллельные алгоритмы // Научно-технический вестник СПбГУ ИТМО. – 2008. – Вып. 54. – С. 56–63.
15. Ковальчук С.В. Особенности проектирования высокопроизводительных программных комплексов для моделирования сложных систем // Информационно-управляющие системы. – 2008. – № 3. – С. 10–18.
16. John E. Cazes, Tim Campbell, Erick Rogers. OpenMP Parallel Implementation of SWAN // Navo MSRC Navigator. – 2001. – Vol. 13.
17. Vinoski S. REST Eye for the SOA Guy // IEEE Internet Computing. – 2007. – Vol. 11. – № 1. – P. 82–84.
18. Obasanjo D. XML Serialization in the .NET Framework // Microsoft MSDN Library. 23 January 2003. – Режим доступа http://msdn2.microsoft.com/en–us/library/ms950721.aspx, свободный.
19. Иванов С.В. и др. Высокопроизводительный программный комплекс моделирования экстремальных гидрометеорологических явлений. Часть III: Интерпретация и использование результатов расчетов // Научно-технический вестник СПбГУ ИТМО. – 2008. – Вып. 54. – С. 72–79.

Ковальчук Сергей Валерьевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, м.н.с., kovalchuk@mail.ifmo.ru

Иванов Сергей Владимирович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, м.н.с., svivanov@mail.ifmo.ru

Бухановский Александр Валерьевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, д.т.н., профессор, boukhanovsky@mail.ifmo.ru

УДК 004.021, 681.324

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЙ ПРОГРАММНЫЙ КОМПЛЕКС МОДЕЛИРОВАНИЯ ЭКСТРЕМАЛЬНЫХ ГИДРОМЕТЕОРОЛОГИЧЕСКИХ ЯВЛЕНИЙ. ЧАСТЬ III: ИНТЕРПРЕТАЦИЯ И ИСПОЛЬЗОВАНИЕ РЕЗУЛЬТАТОВ РАСЧЕТОВ

С.В. Иванов, С.В. Ковальчук, Л.И. Лопатухин, Е.С. Чернышева, А.В. Бухановский

Обсуждаются способы и формы представления характеристик экстремальных гидрометеорологических явлений на основе компьютерного моделирования. Рассматриваются основные виды экстремумов: в точке, в поле, по месяцам и направлениям, многомерные характеристики. Приводятся оценки экстремальных высот волн и скоростей ветра для Баренцева и Каспийского моря.
Ключевые слова: гидрометеорологические экстремумы, атлас экстремальных характеристик.

Введение

В работе [1] излагаются основные положения современной концепции изучения экстремальных гидрометеорологических явлений посредством компьютерного моделирования, реализуемые в рамках высокопроизводительного программного комплекса [2]. В данной статье обсуждаются вопросы интерпретации и дальнейшего использования результатов расчетов как альтернативы традиционным гидрометеорологическим атласам и справочникам на основе данных наблюдений. Для иллюстрации совместно рассматриваются скорость ветра и волнение моря как две наиболее значимые для морской практики гидрометеорологические характеристики.

Первые справочники по режиму ветра и волнения были основаны на визуальных наблюдениях. Они появились после Второй мировой войны, сыграв большую роль в понимании волнового климата, и не потеряли свою актуальность до настоящего времени [3]. Последний справочник, базирующийся на данных визуальных наблюдений, был издан в Великобритании в 1986 г. не только в печатном виде, но и в форме компьютерной информационной системы. В опубликованных пособиях по данным визуальных наблюдений в виде таблиц и графиков представлены сведения о повторяемости волнения по градациям для отдельных районов, месяцев или сезонов, приведены другие элементарные статистические данные (средние значения, дисперсии, параметры распределений и т.п.). С середины 70-х гг. XX века для составления справочников стали активно использоваться инструментальные измерения с автоматических буев и буровых установок. Сейчас эти данные, несмотря на их многочисленность, относятся в основном к прибрежным районам и не отражают режим волнения в открытых районах океанов и морей. Началом спутниковых измерений волнения можно считать 1975 г. Накопленная информация позволила издать в 1996 г. первые атласы по режиму ветра и волнения по спутниковым данным. Подробный обзор истории создания справочников приведен в работах [4, 5].

Не останавливаясь на многочисленных методических вопросах, возникающих при создании подобных справочников, отметим, что их данные отражают пространственно-временную изменчивость режима волнения больших акваторий, поскольку основаны на принципах районирования. В то же время освоение шельфа океанов и морей выдвинуло многочисленные дополнительные требования к составу, полноте и достоверности сведений о ветро-волновом климате, которые часто необходимы для конкретных точек моря или отдельных, небольших по площади месторождений, где отсутствуют наблюдения или их недостаточно. Таким образом, единственным способом получения такой информации на настоящий день остается компьютерное моделирование. Ниже излагаются примеры использования высокопроизводительного программного комплекса для

моделирования экстремальных гидрометеорологических явлений в Баренцевом и Каспийском морях.

Гидрометеорологические экстремумы в фиксированной точке пространства

Значения скалярной гидрометеорологической величины (высоты волн, модуля скорости ветра, уровня моря и пр.), возможные 1 раз в T лет в фиксированной точке, являются наиболее распространенной характеристикой экстремальных гидрометеорологических явлений. В качестве примера рассмотрим процесс их получения для акватории Баренцева моря с использованием высокопроизводительного программного комплекса [2]. Исходными данными для расчета служили поля приводного ветра из массива реанализа NCEP/NCAR с 1967 по 2006 гг. Для усвоения использованы данные измерений ветра на пяти автоматических буйковых станциях в Баренцевом море (с 1989 по 1995 гг.). Учитывая, что волнение в Баренцевом море развивается в условиях, формируемых штормовой активностью в Северной Атлантике, для калибровки полей ветра дополнительно использовались данные наблюдений в Норвежском море, а также измерения судов погоды M, L в Атлантическом океане. Расчет полей волнения проводился на системе из трех вложенных пространственно-временных сеток посредством интегрирования уравнения баланса волновой энергии в форме спектральной модели IV поколения Wave Watch III [6]. Общее время расчета с учетом распараллеливания составило около 140 часов. На рис. 1 приведены результаты верификации модельных расчетов высот волн по данным инструментальных измерений, выполненных в 2006 г. в двух точках восточной части Баренцева моря (рис. 2).

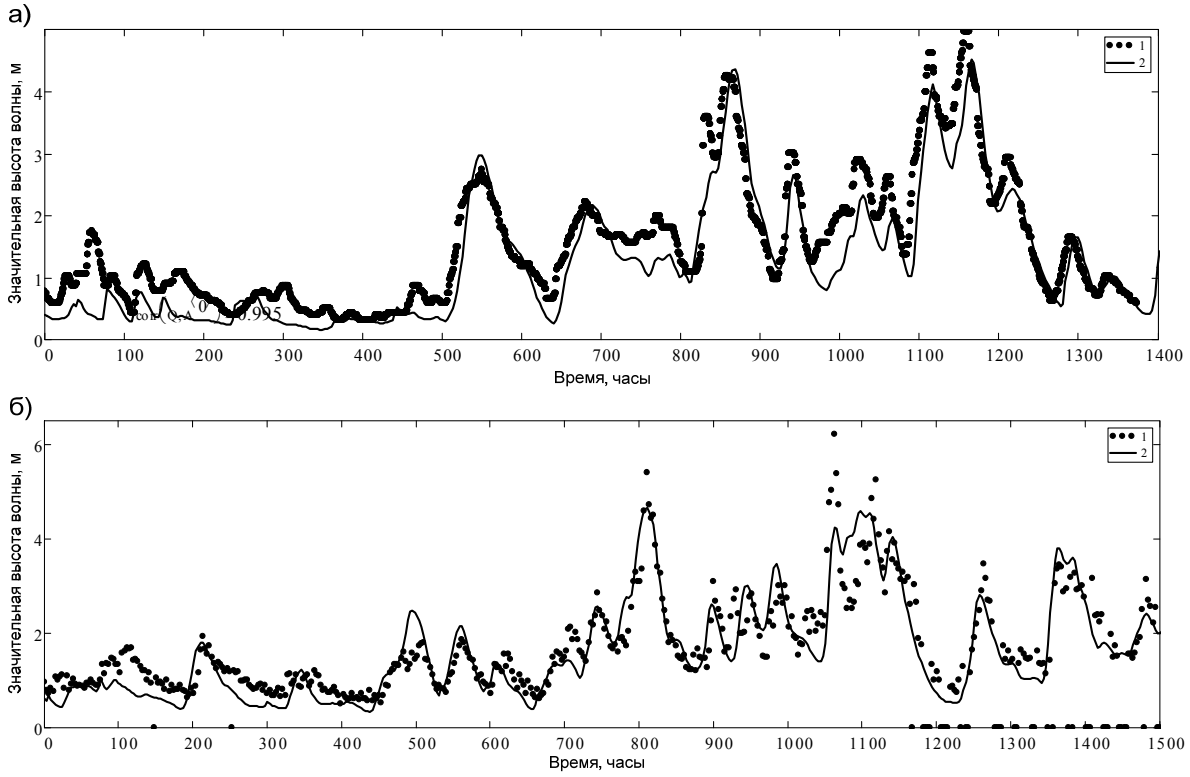


Рис. 1. Результаты верификации модельных расчетов высот волн по данным инструментальных измерений; (а) и (б) соответствуют точкам на рис. 2:
1 – данные измерений; 2 – расчеты по модели

В обоих случаях видно, что результаты моделирования достаточно хорошо согласуются с данными измерений, несмотря на то, что за период измерений усвоения данных по ветру не проводилось. Это дает основание для дальнейшего использования по-

лученных расчетных данных с целью расчета экстремальных характеристик методом BOLIVAR [1, 7]. На рис. 2 приведен пример расчетов наибольших (0,1%) высот волн в Баренцевом море, возможных 1 раз в 100 лет. Видно, что в западной части моря высота экстремальных волн может превышать 30 м; по мере продвижения штормов к востоку (юго-востоку) эта величина несколько снижается, достигая в Печорском море только 8–16 м.

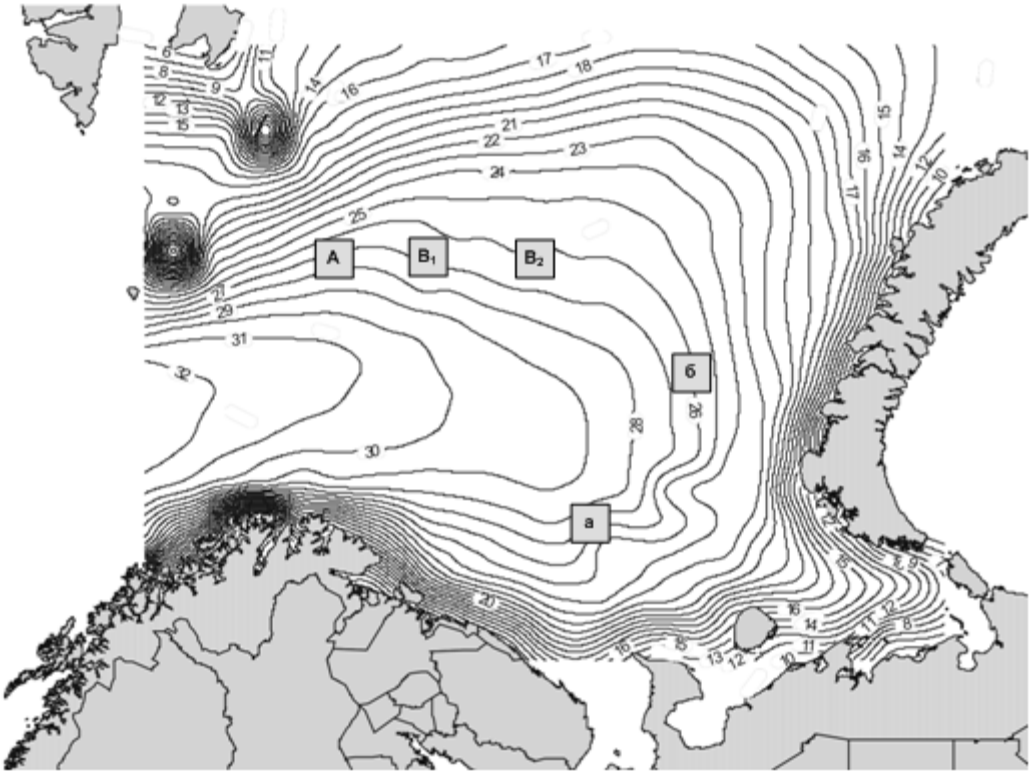


Рис. 2. Наибольшая высота волны, возможная 1 раз в 100 лет, в Баренцевом море. Значками (а) и (б) показаны точки измерений с рис. 1, (А) и (В) – с рис. 3

Пространственно-временная изменчивость гидрометеорологических экстремумов

Изображенный на рис. 2 пространственный образ не соответствует реальному шторму, пусть даже самому сильному, поскольку во всех точках акватории экстремальные волны не реализуются одновременно. Так, на рис. 3 приведены точечные диаграммы годовых максимумов наибольших высот волн h^A в точке А (рис. 2) и соответствующих им условных (ассоциированных) значений высот волн $h^{B|A}$ в точках B_1 и B_2, лежащих на расстоянии 120 и 240 км к востоку от точки А. Расстояние между точками не превышает размеров однородных районов из публикации [4] и сопоставимо с масштабами синоптической изменчивости, вызванной движением барических образований над Баренцевым морем. Дополнительно на рис. 3 приведены контуры равной повторяемости сочетаний высот волн в этих точках, возможных 1 раз в 10 и 100 лет. Ось абсцисс, соответствующая выборке годовых максимумов в точке А, градуирована также в периодах повторяемости (1 год, 10 и 100 лет). Видно, что значения высот волн редкой повторяемости в соседних точках существенно зависимы; при этом степень статистической связи естественным образом убывает при увеличении расстояния. Эта зависимость обусловливает то, что одинаковый период повторяемости может быть приписан различным сочетаниям экстремальных волн в рассматриваемых точках, лежащим на соответствующем контуре. Например, если в точке А высота волны, возможная 1 раз в 100 лет,

$h^A = 27{,}2$ м, то в точке B_1 этому значению соответствует $h = 25{,}4$ м, т.е. волна, возможная 1 раз в 50 лет. Важно также, что событие, возможное 1 раз в 100 лет для нескольких точек одновременно, не обязательно предполагает появление экстремума, возможного 1 раз в 100 лет хотя бы в одной из точек. Например, 1 раз в 100 лет возможен следующий набор событий:

- в точке А реализуется событие с высотой волны, возможной 1 раз в 10 лет ($h^A = 22{,}5$ м),
- в точке B_1 $h^{B|A} = 24{,}3$ м (т.е. 30-летняя волна),
- в точке B_2 $h^{B|A} = 25{,}1$ м (т.е. 45-летняя волна).

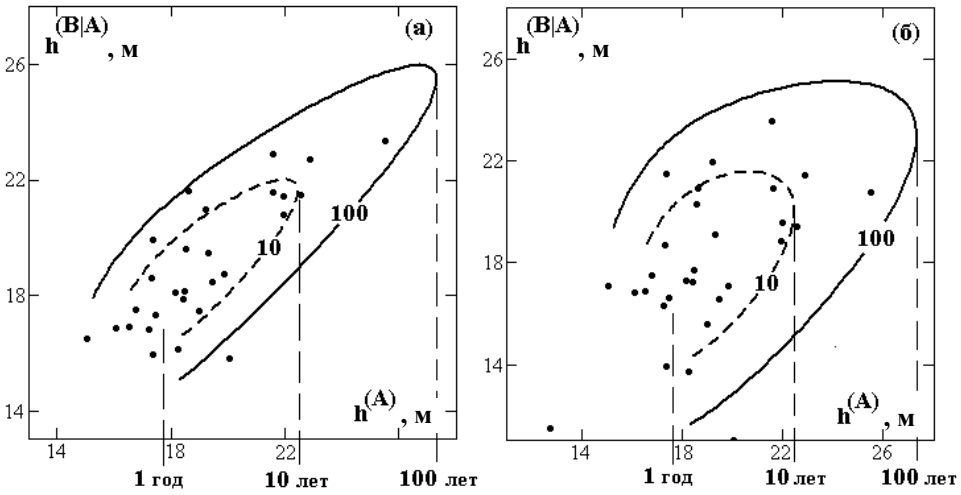


Рис. 3. Точечные диаграммы и изолинии повторяемости годовых максимумов высот волн 0.1% обеспеченности h^A в точке А и соответствующих им (условных) значений высот волн $h^{B|A}$ в точках B_1 (а) и B_2 (б) в одни и те же годы

Гидрометеорологические экстремумы по месяцам и направлениям

Высокопроизводительный программный комплекс [2] позволяет детализировать изменчивость экстремальных характеристик гидрометеорологических явлений с учетом их цикличности по месяцам (сезонам) и характерным направлениям θ. На рис. 4, а, приведена гистограмма повторяемости наибольших высот волн по направлениям и месяцам одновременно в Северной части Каспийского моря. Видно, что в зимний сезон наиболее волноопасное направление (ЮВ) прослеживается более отчетливо, чем летом, когда распределение экстремумов по направлениям становится практически равномерным. На рис. 4, б, приведены маргинальные значения повторяемости годового максимума высоты волн в синоптические сроки по месяцам в северной части Каспийского моря. Видно, что в разные годы он может наблюдаться с сентября по апрель, чаще всего – в феврале.

Детализация экстремумов подразумевает решение обратной задачи, т.е. согласование оценок наибольших высот волн, возможных 1 раз в T лет, с учетом и без учета отдельных месяцев и направлений. Процедура согласования сводится к тому, чтобы для каждого направления и месяца найти фиктивный период повторяемости $T(\theta, t)$, который бы показывал, какую обеспеченность во всем ансамбле имеет порядковая статистика обеспеченности $1 - 1/T(\theta, t)$, % условного распределения.

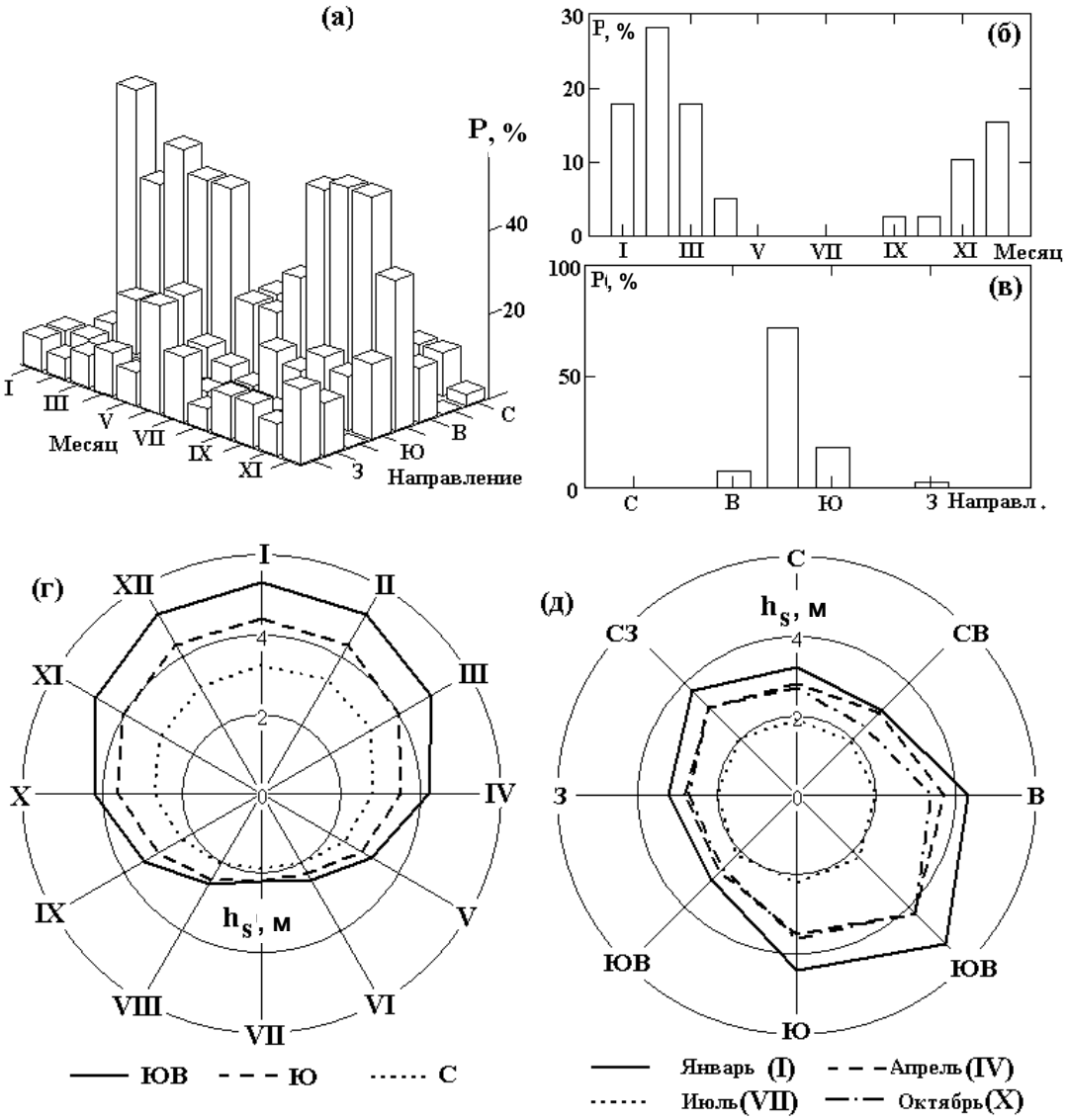


Рис. 4. Вероятностные характеристики условных экстремумов (по месяцам и направлениям) значительной высоты волны в Северном Каспии.
(а) – повторяемость, %, экстремумов по направлениям в различные месяцы; (б) – повторяемость, %, экстремумов по месяцам (все направления); (в) – повторяемость, %, экстремумов по направлениям (все месяцы); (г) – оценки высот волн 1 раз в 100 лет по месяцам (для разных направлений); (д) – оценки высот волн 1 раз в 100 лет по направлениям (для разных месяцев)

Например, для Северной части Каспийского моря экстремум, который появляется в феврале с южного сектора 1 раз в 100 лет, соответствует экстремуму 1 раз в 45 лет условного распределения, т.е. в предположении, если бы во все месяцы в течение 100 лет все волны распространялись только от юга. Полученные таким образом оценки условных экстремумов высот волн могут быть представлены как функция $h_T(\theta,t)$ двух переменных. На рис. 4, г–д, приведены сечения этой функции для оценки 100-летней высоты значительных волн в Северном Каспии по направлениям для разных месяцев и по месяцам для разных направлений. Видно, что в разные месяцы и по различным направлениям экстремальные характеристики могут отличаться в несколько раз, что представляет существенный практический интерес с точки зрения обеспечения безопасности морских сооружений.

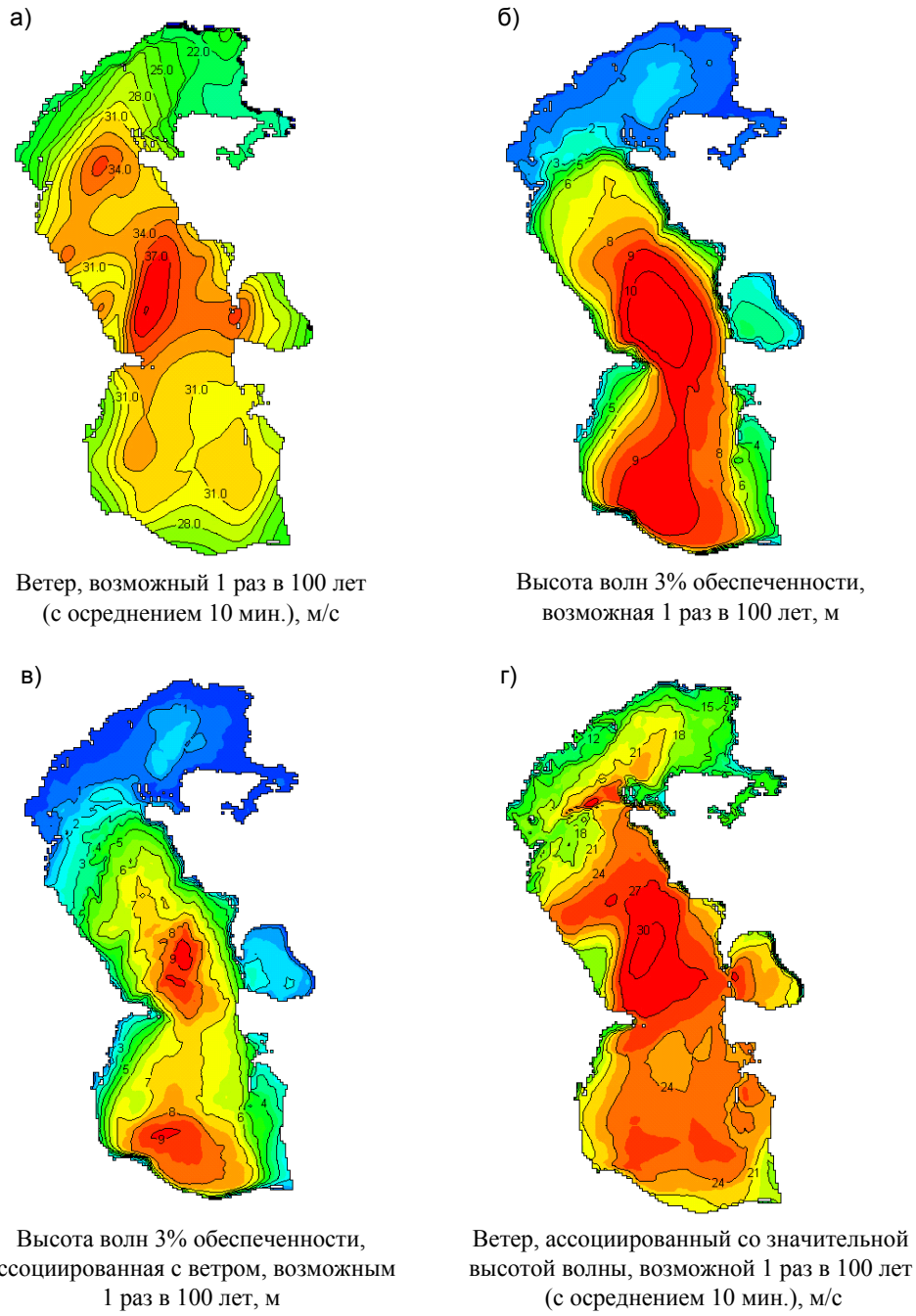
Многомерные гидрометеорологические экстремумы

Интерес к многомерным гидрометеорологическим экстремумам связан с усилением требований к надежности проектирования морских объектов и сооружений. Проектировщики сооружений на шельфе должны быть уверены, что объект сможет выдержать любые возможные комбинации воздействий внешней среды [8, 9]. В качестве иллюстрации на рис. 5 представлены примеры совместных характеристик гидрометеорологических экстремумов (скорости ветра и высот волн) во всем Каспийском море. Они получены путем расчетов за временной интервал с 1948 по 2007 гг. Видно, что наибольшая скорость ветра, возможная 1 раз в 100 лет, в центральной части Каспийского моря превышает 37 м/с. В то же время скорость ветра, соответствующая высоте волн, возможной 1 раз в 100 лет, в той же точке составляет чуть более 30 м/с. Таким образом, использование совместных характеристик экстремальных явлений вместо комбинации маргинальных экстремумов в точке позволяет существенно уточнить расчетные нагрузки, что позволяет повысить надежность проектируемой конструкции и избежать необоснованного завышения запаса прочности и следующих за этим финансовых затрат. Следует отметить, что в общем случае многомерные гидрометеорологические экстремумы характеризуются изолинией равной повторяемости сочетаний экстремальных характеристик, возможных 1 раз в T лет. Такая характеристика не дает однозначного ответа для выбора сочетания, наиболее опасного для того или иного объекта. Это требует дополнительных знаний об объекте проектирования. Если данная информация отсутствует, во многих практических случаях достаточно ограничиться так называемыми ассоциированными значениями, т.е. регрессией одной гидрометеорологической величины на другую (см. рис. 5).

Представление информации в электронных гидрометеорологических атласах и справочниках

Информация об экстремальных гидрометеорологических явлениях допускает различные формы представления, иллюстрирующие особенности пространственной и временной изменчивости. Ее непосредственная интерпретация затруднена в силу многомерности и многокомпонентности, что требует использования для доступа к таким данным специальных инструментальных средств – электронных гидрометеорологических атласов и справочников [10]. Реализация интерактивного атласа в виде web-сайта позволяет добиться достаточно высокого уровня доступности за счет стандартизации протоколов. Доступ к атласу возможен в двух режимах: непосредственная работа пользователя при помощи браузера с интерактивной версией и автоматическое взаимодействие приложений пользователя со справочным массивом посредством технологии Web-сервисов.

Доступ к Web-сервисам осуществляется на основе стандартного протокола SOAP. При этом получаемые данные представляют собой сообщения, оформленные с использованием языка XML. Для реализации интерактивного варианта справочника (работающего непосредственно в браузере клиента) использована технология Flash, позволяющая добиться большей гибкости, чем стандартные компоненты языка HTML, с меньшими трудозатратами. Применение данной технологии позволяет использовать богатый набор элементов управления и способов представления информации клиенту, организовать гибкий пользовательский интерфейс, отвечающий всем необходимым требованиям по работе с интерактивными картами.



Ветер, возможный 1 раз в 100 лет (с осреднением 10 мин.), м/с

Высота волн 3% обеспеченности, возможная 1 раз в 100 лет, м

Высота волн 3% обеспеченности, ассоциированная с ветром, возможным 1 раз в 100 лет, м

Ветер, ассоциированный со значительной высотой волны, возможной 1 раз в 100 лет (с осреднением 10 мин.), м/с

Рис. 5. Карты совместных экстремальных характеристик ветра и волнения в Каспийском море, полученные посредством компьютерного моделирования

Заключение

На основе предложенной в [1] концепции спроектирован и реализован высокопроизводительный программный комплекс [2], адаптированный к архитектуре суперкомпьютеров терафлопной производительности. Эффективность использования комплекса продемонстрирована на примере расчетов различных (в том числе качественно новых) видов экстремальных характеристик гидрометеорологических явлений в Каспийском и Баренцевом морях, включая экстремумы в точке, полевые экстремумы, экстремумы по месяцам и направлениям и многомерные экстремумы. Для анализа и интерпретации результатов расчетов разработан электронный атлас экстремальных гид-

рометеорологических явлений, реализованный в виде интерактивного web-сайта, в котором доступ к массивам справочных данных осуществляется посредством web-сервисов.

Литература

1. Бухановский А.В. и др. Высокопроизводительный программный комплекс моделирования экстремальных гидрометеорологических явлений. Часть I: Постановка задачи, модели, методы и параллельные алгоритмы // Научно-технический вестник СПбГУ ИТМО. – 2008. – Вып. 54. – С. 56–63.
2. Ковальчук С.В. и др. Высокопроизводительный программный комплекс моделирования экстремальных гидрометеорологических явлений. Часть II: Разработка и оценка программной архитектуры // Научно-технический вестник СПбГУ ИТМО. – 2008.– Вып. 54. – С. 64–71.
3. Ветер и волны в океанах и морях (1974). Справочные данные. – Морской регистр СССР. – Л.: Транспорт, 1974.
4. Справочные данные по режиму ветра и волнения Баренцева, Охотского и Каспийского морей. – СПб: Российский морской регистр судоходства, 2003. – 214 с.
5. Справочные данные по режиму ветра и волнения Балтийского, Северного, Черного, Азовского и Средиземного морей / Лопатухин Л.И. и др. – СПб: Российский Морской регистр судоходства. – 2006. – 450 с.
6. Tolman H.L. User manual and system documentation of WAVEWATCH-III version 2.22. NOAA / NWS / NCEP / MMAB. – 2002. – Technical Note 222. – 133 p.
7. Estimation of extreme wind wave heights / Lopatoukhin L.J. [et al] // Reports of World Meteorological Organization, WMO/TD. – 2000. – Vol. 1041. – 76 p.
8. Coles S.G., Tawn J. Statistical Methods for Multivariate extremes: an Application to structural design // Applied Statistics. – 1994. – Vol. 43. – №1. – P. 1–48.
9. Morton I.D, Bowers J. Extreme analysis in a multivariate offshore environment // Applied Ocean Res. – 1996. – Vol. 18. – P. 303–317.
10. Бухановский А.В., Давидан И.Н., Иванов Н.Е., Рожков В.А. Гидрометеорологический компьютерный справочник Балтийского моря // Труды ГОИН. – СПб: Гимиз, 2002. – Вып. 208. – С. 156–171.

Иванов Сергей Владимирович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, м.н.с., svivanov@mail.ifmo.ru

Ковальчук Сергей Валерьевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, м.н.с., kovalchuk@mail.ifmo.ru

Лопатухин Леонид Иосифович – Санкт-Петербургский государственный университет, д.г.н., профессор, leonid-lop@yandex.ru

Чернышева Екатерина Сергеевна – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, к.ф.-м..н., с.н.с., chereks@yandex.ru

Бухановский Александр Валерьевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, д.т.н., профессор, boukhanovsky@mail.ifmo.ru

УДК 004.4’22, 681.327.8

ИНСТРУМЕНТАЛЬНАЯ ТЕХНОЛОГИЧЕСКАЯ СРЕДА ДЛЯ СОЗДАНИЯ МАССОВЫХ МОБИЛЬНЫХ ОНЛАЙН-СЕРВИСОВ НОВОГО ПОКОЛЕНИЯ. ЧАСТЬ I: ПРИНЦИП ДЕЙСТВИЯ И ПРОГРАММНАЯ АРХИТЕКТУРА

О.А. Золотарев, Е.А. Гринина, А.В. Бухановский

Рассматривается проблема автоматизации проектирования и разработки распределенного программного обеспечения мобильной коммерции. Показано, что основным фактором, определяющим архитектуру и принцип действия мобильных сервисов, является специфика финансового регулирования в области электронных платежей. Разработана инструментальная среда поддержки процесса создания мобильных сервисов, пригодных для внедрения на российском рынке неголосовых услуг сотовой связи.
Ключевые слова: мобильные сервисы, электронная коммерция, программная архитектура.

Введение

Понятие *мобильной коммерции* (mCommerce) охватывает широкий спектр сервисов и услуг, которые предоставляются с помощью устройств мобильной связи [1]. Мобильная коммерция является разновидностью электронной коммерции как широкого спектра форм коммерческой деятельности, основанной на предоставлении услуг посредством телекоммуникационных технологий [2]. Проникновение технологий мобильной коммерции в России на настоящий момент столкнулось с рядом противоречий как технологического, так и нормативно-правового характера.

– Высокое проникновение услуг мобильной связи (включая неголосовые) практически во все слои населения (около 100% в крупных городах РФ) не соответствует качеству предоставляемых сервисов и контента (типовые развлекательные продукты – картинки, рингтоны, игры). Этот диссонанс уже с 2005 г. привел к «перегреву» рынка мобильного контента и стагнации доходов операторов связи и провайдеров [3].
– Возможности провайдеров по предоставлению мобильных сервисов нового поколения (мобильное телевидение, электронные билеты и пр.) в целом не обеспечены соответствующими технологиями их тарификации и оплаты. Как следствие, большинство провайдеров контента предоставляет свои услуги по стандартным схемам тарификации через операторов связи или через агрегаторов [4]. В общем случае это приводит к повышению себестоимости услуги в 2,5–3 раза, что негативно сказывается на потребительском спросе.
– Разнообразие технологических решений, предлагаемых производителями мобильных терминалов (включая специализированные SDK для конкретных моделей), осложняет задачу разработки и предоставления мобильных сервисов. Экстенсивный путь развития, ориентированный на платформенно-зависимые сервисы, порождает риски ответственности за некачественное оказание услуги.

В данной статье проводится критический анализ развития мобильной коммерции в различных регионах мира, на его основе формулируются требования к мобильным сервисам для российского рынка, а также рассматривается подход к автоматизации процесса их проектирования и разработки на основе инструментальной среды создания массовых мобильных онлайн-сервисов нового поколения.

Анализ международного опыта в области мобильной коммерции

В настоящее время в экономически развитых странах уровень проникновения сотовой связи превысил 100%, что создает предпосылки к интенсивному развитию рынка *массовых* мобильных услуг. В частности, в работе [5] показано, что *рост* доходов от

голосовых услуг для европейских операторов постепенно снижается, становясь отрицательным со второго квартала 2007 г., в то время как доходы от услуг передачи данных перманентно увеличиваются. Аналогичная тенденция отмечается и в США. Например, при снижении в 2007 г. голосового ARPU (как дохода в расчете на единичного пользователя) на 6% аналогичный показатель для передачи данных увеличился более чем на 47%. 16% суммарного ARPU в США составляют доходы, получаемые от мобильных сервисов. Весь мировой объем платежей, связанных с неголосовыми услугами, составил в 2007 г. более 2 млрд. долларов. По прогнозам аналитиков [5], к 2012 г. почти половина ARPU в мире будет формироваться за счет услуг передачи данных. Как следствие, это может быть обеспечено только за счет наращивания количества и ассортимента разнообразных мобильных сервисов, что связано не только с транспортной инфраструктурой, но и с расширением мультимедийной функциональности современных мобильных телефонов.

Детализация рынка мобильных услуг выявила существенную неоднородность по регионам мира. В настоящее время в мире существуют два крупнейших региональных рынка – Западная Европа (955 млн. USD) и Восточная Азия (702 млн. USD) [6]. Сопоставляя проекты мобильной коммерции в Западной Европе и Восточной Азии, становится очевидным, что успех внедрения и продвижения мобильных сервисов определяется, в первую очередь, *выбором рыночной модели*, который, в свою очередь, диктуется стилем финансового регулирования, принятого в данном государстве. Это связано с тем, что мобильный сервис (интерпретируемый как платная услуга посредством мобильной связи) предусматривает «дистанционную» идентификацию абонента и использование так называемых «электронных денег». Таким образом, вывод мобильных сервисов на рынок непосредственно связан с легализации системы оплаты мобильных услуг.

Специфика внедрения мобильных сервисов в России заключается в слабой банковской системе и недостаточном охвате населения банковскими услугами [3]. Как следствие, нормативно-правовая база, обеспечивающая легальное функционирование приемлемых моделей мобильной коммерции, находится в стадии формирования. Это породило однобокое и ограниченное развитие рынка мобильных услуг с преобладанием сервисов по предоставлению мобильного контента на основе так называемых «агрегаторских» схем [4], когда компания-агрегатор (КА) берет на себя установление множественных договоренностей с отдельными контент- и сервис-провайдерами, а также с операторами сотовой связи для облегчения процесса доставки контента пользователям. Таким образом, вместо заключения контрактов со многими операторами связи провайдеры вступают в договорные отношения только с КА. Современная российская модель работы КА существенно отличается от традиционных мировых схем. Например, в Западной Европе задача КА состоит в технологическом упрощении для контент-провайдеров выхода на рынок мобильных услуг. Всю маркетинговую деятельность, рекламу, продвижение собственной продукции берет на себя контент-провайдер, предоставляя продукцию от своего имени и под своей торговой маркой [5]. В России, напротив, большинство КА образовалось на базе крупных контент-провайдеров. Помимо технологической поддержки, они обеспечивают своим партнерам продвижение под единым брендом КА. Таким образом, КА по сути занимается перепродажей услуг и продукции отдельных провайдеров.

Основными недостатками агрегаторских схем являются увеличение нагрузки на биллинговую систему оператора связи, а также аспекты, связанные с финансово-юридическими вопросами оплаты посредством Premium SMS и удовлетворения претензий при некачественном оказании услуги. Агрегаторские схемы допускают весьма ограниченное применение и годятся в основном для продажи цифровых товаров и услуг (статического мобильного контента).

Выбор модели и способа реализации электронных платежей

Существующая в России нормативно-правовая база не дает возможности в полной мере привлекать операторов сотовой связи к процедурам расчетов посредством современных инфокоммуникационных систем, предполагающих использование безналичных финансовых инструментов без открытия банковского счета (переводы, предоплаченные карты). Это связано, во-первых, с чрезмерно усложненными процедурами идентификации, а во-вторых, с запретом некредитным организациям принимать денежные средства от покупателей с целью последующей передачи банку для осуществления перевода или пополнения банковской предоплаченной карты (за исключением частных случаев, предусмотренных ст. 13.1 Закона «О банках и банковской деятельности» [7]). Альтернативой является использование таких схем взаимодействия клиента и провайдера сервиса, в которых оператор связи предоставляет только транспортные услуги, а все взаиморасчеты выполняются через электронную платежную систему. В настоящее время можно выделить шесть основных групп платежных систем, потенциально конкурирующих на российском рынке. К ним относятся: (а) платежные системы банковских карт (Visa, MasterCard и др.); (б) Интернет-платежные системы на основе скрэтч-карт (E-Port, Рапида и пр.); (в) Интернет-платежные системы на основе удаленного управления счетом (WebMoney, Yandex-деньги и пр.); (г) системы мобильных платежей на основе управления банковским счетом (SimMP, «Мобильный банк»); (д) биллинговые системы операторов сотовой связи; (е) системы на основе предоплаченного финансового продукта (ПФП).

Таблица 1. Сравнительный анализ потребительских качеств платежных систем

Характеристика	**0**	**а**	**б**	**в**	**г**	**д**	**е**
Привычность для массового потребителя в РФ	+	-	-	-	-	-	-
Доступность (+ для более 50% населения в РФ)	+	-	-	-	-	-	+
Отсутствие передачи персональных данных потребителя	+	-	+	+	+	+	+
Независимость от банковских счетов	+	-	+	+	-	+	+
Охват небанковской аудитории	+	-	+	+	-	+	+
Равноправие участников платежей	+	-	+	+	-	-	+
Рентабельность микроплатежей	+	-	+	+	-	+	+
Поддержка офлайновых платежей	+	-	-	-	-	-	+
Отчисление части дохода оператору связи	-	-	-	-	-	+	+
Возможность платежей между пользователями системы	+	-	+	+	-	-	+
Возможность оплаты мобильного контента и услуг	-	-	-	-	-	+	+
Возможность оплаты товаров в розничной торговле	+	+	-	-	+	-	+
Осуществление платежей в автоматическом режиме	+	+	-	-	-	-	+
Осуществление платежей через Интернет	+	+	+	+	+	+	+

В табл. 1 приведен сравнительный анализ потребительских и технологических качеств платежных систем с точки зрения их использования в современных массовых мобильных онлайн-сервисах. Индексом «0» отмечен способ оплаты наличными деньгами. Из таблицы видно, что наиболее перспективными являются платежные системы на основе ПФП [8]. В настоящее время в России этот класс представлен исключительно продуктами банка «Таврический» на основе технологии ПэйКэш (мобильный кошелек Вымпелком, ПФП ПэйКэш [9]). Несмотря на то, что сама технология ПэйКэш является патентозащищенной, реализующая ее аппаратно-программная платформа эксплуатируется на коммерческой основе и поддерживает возможность подключения любых классов мобильных сервисов по открытым протоколам [10].

Обоснование архитектуры типового мобильного сервиса

Архитектура типового мобильного онлайн-сервиса как распределенного приложения является двухзвенной (клиент-провайдер). Провайдер поддерживает работу серверного приложения, обеспечивающего доступ как к собственным программным модулям, так и к внешним сервисам других провайдеров через стандартную коммуникационную среду. Пользователь взаимодействует с сервером провайдера, используя клиентское приложение, установленное на его мобильном телефоне. При этом процесс взаимодействия может осуществляться при помощи различных коммуникационных средств – SMS-сообщений, USSD-запросов, взаимодействия с сервером посредством сети Интернет (с использованием протоколов GPRS/EDGE, W-CDMA, беспроводных сетей локального доступа). Платежи за оказанные услуги выполняются в режиме реального времени посредством процессингового центра платежной системы, независимо взаимодействующей и с клиентом, и с провайдером. Этот снимает с операторов сотовой связи вопросы контроля качества предоставляемых провайдерами услуг. Поскольку факт платежной операции регистрируется в процессинговом центре, то клиент в случае невозможности использования продукта имеет право на возврат платежа на свой счет. При этом возврат может осуществляться в автоматическом режиме (например, если клиент не использовал электронный продукт до указанного срока) или в режиме рассмотрения претензии.

Дополнительно в состав мобильного сервиса может входить программное обеспечение аппаратных устройств контроля, например, POS-терминала или турникета доступа. Эти модули не взаимодействуют с платежной системой напрямую; обычно они подключены к серверу провайдера через Интернет, а для обмена данными с клиентом используются технологии локального взаимодействия, например, Bluetooth или NFC [11]. Провайдер сервиса также может поддерживать web-сайт, который предоставляет клиенту возможность конфигурации индивидуального набора сервисов и упрощает настройку репозитория; в бизнес-процессе оказания услуги сайт обычно не используется.

Примером мобильного онлайн-сервиса нового поколения, включающего в себя все вышеперечисленные компоненты, является сервис продажи электронных билетов на развлекательные мероприятия. Он позволяет клиенту оперативно связаться с поставщиком билетов, получить справочную информацию о мероприятии, выбрать параметры и количество билетов (например, свободные места в зале) и произвести оплату, получив при этом защищенный электронный билет непосредственно на свой мобильный телефон. В дальнейшем этот билет может быть использован путем предъявления (как изображения) с экрана мобильного телефона или считывания турникетом доступа по технологии NFC. В том случае, если клиент не посетил мероприятие, производится возврат оплаты за вычетом комиссии провайдера. Подробнее данный сервис описан в [12].

Инструментальная среда разработки мобильных сервисов

Несмотря на кажущуюся простоту двухзвенной архитектуры типового сервиса, его разработка и отладка требуют применения специальных навыков создания распределенных мобильных приложений и их сопряжения с платежной системой. Для упрощения этого процесса создана инструментальная технологическая среда (ИТС). ИТС рассчитана на три группы потенциальных пользователей, различающихся по квалификации и опыту разработки распределенных приложений, что соответствует различному порядку использования компонентов в рамках архитектуры, изображенной на рис. 1.

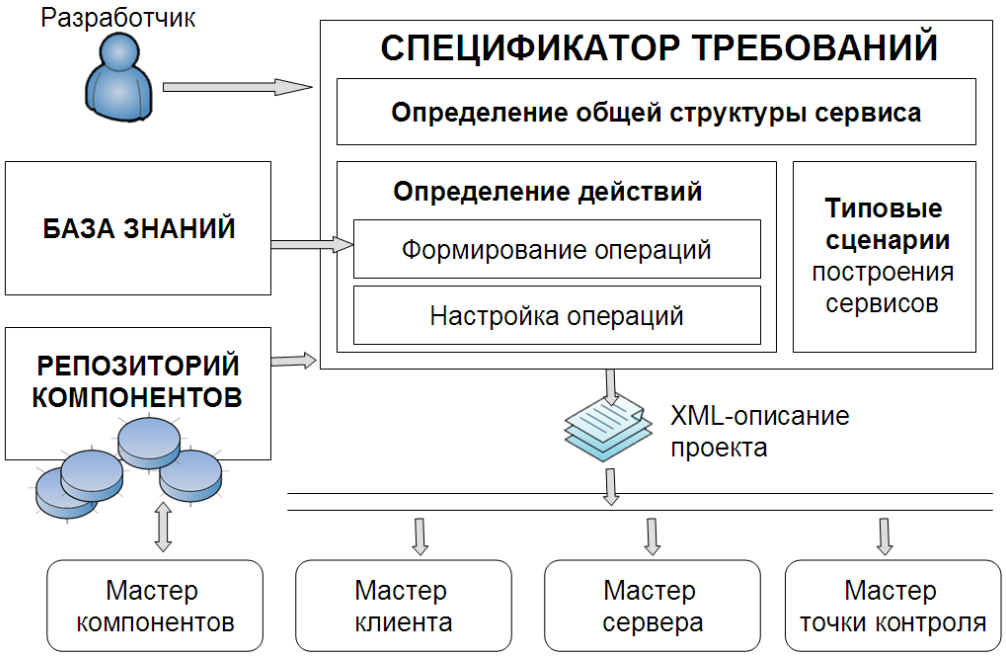


Рис. 1. Высокоуровневая архитектура инструментальной технологической среды

Наиболее важной частью ИТС является спецификатор требований (СТ). СТ обладает функциональностью экспертной системы и позволяет пользователю задавать описание мобильного сервиса в терминах бизнес-процессов предметной области посредством визуального редактора. Пользователь формирует общую архитектуру распределенного приложения, после чего настраивает его основные компоненты вручную, выбирая компоненты из репозитория, или использует типовые сценарии (эталонные сервисы), представленные в базе знаний. Результатом работы СТ является XML-документ, содержащий полное описание структуры распределенного приложения в терминах используемых компонентов.

Более низкоуровневым средством, чем СТ, являются мастера генерации кода (МГК). МГК на основании диалога с пользователем формируют архитектуру распределенного приложения и настраивают его основные модули, а также генерируют сами программные коды, используя готовые компоненты из репозитория. В отличие от СТ, взаимодействующего с пользователем в терминах бизнес-процессов, в МГК пользователь отражает, в первую очередь, технологические аспекты реализации сервиса. МГК могут использовать XML-документ проекта, создаваемый посредством СТ. В этом случае дублирующие вопросы в системе диалога МГК не используются; в лучшем случае МГК сводится к уровню транслятора документа проекта в исходный код приложения. В ИТС может использоваться несколько МГК, соответствующих разным классам модулей (например, клиента, сервера, точки контроля и создания новых компонентов). Программные коды «типовых» сервисов, построенных с помощью СТ и МГК, могут быть использованы опытными пользователями как каркас мобильных приложений с последующей «ручной» модификацией.

ИСТ реализована как надстройка над средой программирования Netbeans; при этом МГК играют роль шаблонов генерации кода на языке Java. СТ разработан при помощи технологии Windows Presentation Foundation, предоставляющей расширенные возможности для реализации пользовательского интерфейса. Создаваемое в ИСТ клиентское приложение использует платформу Java ME CLDC 1.0/MIDP 2.0, которая отличается достаточной степенью универсальности, что позволяет запускать приложения в операционных системах Windows Mobile или Symbian с использованием соответствующих виртуальных машин.

Заключение

В работе обоснована модель мобильной коммерции, позволяющая создавать массовые мобильные онлайн-сервисы нового поколения и активно продвигать их в условиях российского рынка неголосовых услуг сотовой связи. Разработана инструментальная технологическая среда, автоматизирующая процесс создания таких сервисов. Работа выполнена в рамках ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007–2012 годы», проект 2008-4-1.4-18-01-022.

Литература

1. Сарьян В.К., Золотарев О.А. Формирование сектора платежных услуг на сетях мобильной связи третьего поколения // Мобильные Системы. – 2004. – № 9.
2. Новомлинский Л. Электронная коммерция. Тенденции развития в мире и в России. – Режим доступа: www.tops.ru/publishing/pub_007.html, свободный.
3. Золотарев О.А., Кузнецов И.В., Крупнов А.Е, Скородумов А.И. Внедрение мобильных платежей как ключевой фактор развития рынка инфокоммуникационных услуг // Мобильные Системы. – 2007. – № 6.
4. Сети и телекоммуникации / Редакторская статья. – 2005.01.17. – Режим доступа: ngz.zniis.ru, свободный.
5. Arthur D. Little. Telecom Operators: In the eye of the telecom media storm. Analytic report – February, 2008. – 96 p.
6. Mobile Payments Strategies & Markets 2007–2011. Juniper Research. Analytic report. – 2007. – 150 p.
7. Федеральный Закон РФ от 21.03.2002 № 31–ФЗ «О банках и банковской деятельности».
8. Положение Центрального Банка России от 24 декабря 2004г. № 266–П «Об эмиссии банковских карт и об операциях, совершаемых с использованием платежных карт».
9. Крупнов Ю.С. О природе электронных денег // Бизнес и Банки. – 2003. – № 5.
10. Описание технологии PayCash и подключения к платежной системе. – Режим доступа: www.paycash.ru, свободный.
11. Near Field Communications (NFC), ABI Research, Analytic report. – 2007. – 120 p.
12. Гринина Е.А. и др. Инструментальная технологическая среда для создания массовых мобильных он-лайн-сервисов нового поколения. Часть II: Технологии локального взаимодействия // Научно-технический вестник СПбГУ ИТМО. – 2008.– Вып. 54. – С. 86–91.

Золотарев Олег Анатольевич – ООО «Технологии процессинга», генеральный директор, zolotarev@tprs.ru

Гринина Екатерина Александровна – ООО «Технологии процессинга», руководитель научного отдела, к.ф.-м.н., grinina@tprs.ru

Бухановский Александр Валерьевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, д.т.н., профессор, boukhanovsky@mail.ifmo.ru

УДК 004.4’22, 681.327.8

ИНСТРУМЕНТАЛЬНАЯ ТЕХНОЛОГИЧЕСКАЯ СРЕДА ДЛЯ СОЗДАНИЯ МАССОВЫХ МОБИЛЬНЫХ ОНЛАЙН-СЕРВИСОВ НОВОГО ПОКОЛЕНИЯ. ЧАСТЬ II: ТЕХНОЛОГИИ ЛОКАЛЬНОГО ВЗАИМОДЕЙСТВИЯ

Е.А. Гринина, О.А. Золотарев, И.А. Пименов, Д.А. Насонов, А.В. Бухановский

Рассмотрены вопросы применения технологий локального взаимодействия в мобильных онлайн-сервисах нового поколения. Обсуждаются особенности перспективной технологии NFC и аспекты ее использования в составе мобильного сервиса электронной продажи билетов, реализуемого посредством технологической инструментальной среды.
Ключевые слова: мобильный сервис, технология NFC, локальное взаимодействие.

Введение

Современный мобильный телефон представляет собой многоцелевой инструмент, обеспечивающий своему владельцу широкие возможности взаимодействия с реальным миром посредством разнообразных интерфейсов. Наравне с традиционными средствами взаимодействия, такими, как голосовая связь и Интернет, в настоящее время значительную роль начинают играть технологии *локального* взаимодействия [1]. Они обеспечивают адекватное отображение реальных объектов в виртуальное окружение, которым пользователь может управлять *over-the-air* [2]. В настоящее время существуют различные беспроводные технологии установления локальных соединений [3]: IrDa, Bluetooth, Wi-Fi, NFC и др. Они различаются по принципу функционирования, имеют различный радиус действия и подходят для решения совершенно разных задач взаимодействия мобильных устройств. Наиболее прогрессивной на настоящий момент считается технология NFC, основанная на принципе взаимной магнитной индукции [4]. Технология NFC подразумевает малый (до 10 см) радиус действия, что обеспечивает локальное взаимодействие двух устройств, в котором перехват сигнала третьей стороной максимально затруднен, но сеанс связи при этом конфигурируется просто и быстро. Эти качества нашли свое применение в задачах создания персональных систем мониторинга состояния здоровья [5] и продвижения технологий мобильной коммерции [6]. В данной статье рассматриваются вопросы использования технологии NFC в мобильных онлайн-сервисах нового поколения на примере сервиса электронной продажи билетов, который реализуется посредством технологической инструментальной среды [7].

Характеристики технологии NFC

NFC (Near Field Communication) – технология беспроводной связи на ультракоротких (до 10 см) расстояниях на основе эффекта взаимной магнитной индукции. Технология работает в нерегулируемом диапазоне 13,56 МГц, обеспечивает двунаправленное взаимодействие двух устройств, оснащенных интерфейсом NFC, и поддерживает скорости передачи данных 106, 212 и 424 кбит/с [8]. В основу NFC положены существующие технологии бесконтактных смарт-карт Mifare и Felica, разработанные NXP Semiconductors и Sony. Технология NFC является открытой платформой, регламентирована стандартами ISO 18092(NFCIP-1), ISO 21481(NFCIP-2) и совместима с ISO 14443, ISO 15693. Это позволяет NFC-устройству эмулировать бесконтактные карты, функционировать как ридер/райтер (reader/writer) бесконтактных карт упомянутых стандартов, RF- или NFC-меток, а также взаимодействовать с другим NFC-устройством в пиринговом режиме.

Технология NFC отличается простотой использования – два NFC-устройства инициируют сеанс связи автоматически при сближении на заданное расстояние; при этом время инициализации не превышает 0,1 с. Преимуществами NFC по сравнению, например, с Bluetooth и IrDa, являются меньшие энергоемкость и стоимость внедрения устройства в мобильный телефон. В табл. 1 приведен сравнительный анализ технологий NFC, Bluetooth и IrDa по набору технологических и потребительских критериев.

Таблица 1. Сравнительный анализ технологий локального взаимодействия

Характеристика	Bluetooth	IrDa	NFC
Сеанс связи в пиринговом режиме	-	+	+
Автоматическая инициализация сеанса связи	-	-	+
Высокая скорость инициализации сеанса связи	-	+	+
Конфиденциальность сеанса связи	-	-	+
Возможность быстрой транзакции	-	+	+
Высокая скорость передачи данных	+	-	+
Безопасность на аппаратном уровне	-	-	+
Безопасность на программном уровне	+	-	+
Эмуляция бесконтактных смарт-карт	-	-	+
Низкая стоимость внедрения в телефон	-	+	+
Наличие стандартных интерфейсов ПК	+	+	-

Из табл. 1 следует, что технология NFC с точки зрения задач мобильной коммерции является наиболее перспективной по совокупности факторов. Отсутствие на настоящий день стандартных (встроенных) интерфейсов NFC в персональных компьютерах обусловлено низкой распространенностью технологии; на данный момент эта проблема может быть решена применением интерфейсов-конверторов NFC-Bluetooth [9].

Классификация NFC-приложений для мобильного телефона

Реализация возможностей локального взаимодействия порождает новые классы программного обеспечения для мобильных телефонов, различающиеся по способам применения технологии NFC, рассмотренным в предыдущем разделе.

- Приложения, эмулирующие бесконтактную смарт-карту, например, транспортные, платежные, идентификационные сервисы, а также сервисы, связанные с картами лояльности и организацией доступа. Они реализуют бизнес-процессы, связанные с необходимостью предъявления электронных билетов. Большая часть таких сервисов характеризуется повышенными требованиями к безопасности коммуникаций.
- Приложения, поддерживающие режим ридер/райтер. К ним относятся, прежде всего, информационные сервисы, позволяющие получать данные с RF- или NFC-меток и смарт-постеров, а также загружать в мобильный телефон электронные билеты и купоны для последующего предъявления. Режим ридер/райтер может рассматриваться как вспомогательный для различных классов сервисов. Например, при покупке транспортного билета по технологии NFC можно дополнительно загрузить схему проезда или расписание работы транспорта.
- Приложения, требующие пирингового режима работы NFC-устройства. К ним относятся сервисы, связанные с необходимостью обмена данными с другим телефоном, компьютером или предметом бытовой электроники. Пиринговый режим открывает возможности для создания совершенно новых классов мобильных сервисов, так как предоставляет широчайшие возможности для осуществления бесконтактного взаимодействия двух электронных устройств [1].

Реализация NFC в мобильном телефоне – элемент безопасности

NFC-приложения реализуются в мобильном телефоне посредством элемента безопасности. Элемент безопасности представляет собой динамическую среду, в которой производится загрузка, удаление, персонализация и управление NFC-приложениями. Различные NFC-приложения в одном телефоне функционируют и управляются независимо друг от друга. Реализация мобильных сервисов на базе технологии NFC подразумевает повышенные требования к информационной безопасности в соответствии с ISO 14980. Элемент безопасности обеспечивает безопасное хранение данных, криптографическое шифрование, а также формирует защищенную среду для исполнения программного кода.

Программные средства, обеспечивающие работу с NFC в мобильном телефоне

В мобильном телефоне приложения с использованием NFC в силу необходимости унификации распределенного взаимодействия управляются *runtime*-системой. Наиболее часто в качестве *runtime*-системы применяется Java Virtual Machine, реализуемая посредством Java ME (MicroEdition), разработанной компанией Sun. Производители мобильных телефонов, устанавливающие как стандартные операционные системы, так и «черные ящики», разрабатывают платформенно-ориентированные версии JVM и JavaME. В настоящее время существует стандартизованный Java-интерфейс для реализации бесконтактных приложений в мобильных NFC устройствах, использующих специфику элемента безопасности. Это Security and Trust Services API для J2ME (JSR177). Он состоит из четырех пакетов, которые позволяют осуществлять взаимодействие с элементом безопасности посредством APDU (Application Program Data Units), предоставляют инструментарий JCRMI (Java Card Remote Methode Invokation), позволяющий J2ME приложениям использовать методику обращения к удаленному объекту Java Card, обеспечивают API для работы с инфраструктурой открытых ключей и содержат набор криптографических функций. Если для взаимодействия с элементом безопасности используется SATSA-APDU пакет, то не имеет значения, как именно архитектурно элемент безопасности реализован в мобильном устройстве. JCRMI может использоваться только в том случае, если элемент безопасности находится под управлением Java Card ОС. Отличия этих интерфейсов невелики, но разработчикам Java-приложений, по-видимому, более комфортно использовать RMI интерфейс для взаимодействия с Java Card апплетом. Еще одним важным API, необходимым для реализации NFC-приложений в мобильном телефоне, является JSR257 (Contactless Communication API). Данный программный интерфейс обеспечивает контроль работы NFC-модуля в мобильном устройстве. Он также позволяет использовать APDU. Отличие между APDU и JSR257 состоит в том, что JSR 257 может отсылать APDU-команды на внешнюю смарт-карту, на которой доступ к резидентному чипу осуществляется только посредством JSR177. Описанные выше API предоставляют разработчику полный доступ к NFC-функциональности мобильного устройства и элементу безопасности в наиболее распространенных моделях мобильных телефонов, поддерживающих NFC (например, Nokia 6131).

Сервис электронной продажи билетов с использованием технологии NFC

В качестве иллюстрации возможностей технологии NFC рассмотрим мобильный онлайн-сервис нового поколения, реализующий процесс продажи и использования электронных билетов (например, на развлекательные мероприятия). Архитектура сервиса и схема выполнения мобильного платежа подробно рассмотрены в [7]. В общем виде мобильный сервис реализует следующий бизнес-процесс.

- Пользователь (как клиент сервиса) на визуальном уровне получает информацию о предстоящем развлекательном мероприятии, например, в форме афиши. Афиша может быть снабжена специальной NFC-меткой, поднеся телефон к которой, пользователь может узнать дополнительную информацию о мероприятии и адрес в сети Интернет, с которого можно загрузить электронный билет.
- Осуществляется обращение к серверу провайдера (продавца билетов) через предустановленное клиентское приложение (автоматически или по запросу). Пользователю предоставляется интерактивная карта свободных мест (например, в зале), которую он может просмотреть на экране своего мобильного телефона и осуществить свой выбор, информируя провайдера о готовности оплатить билет.
- На счету пользователя в платежной системе резервируется сумма, эквивалентная стоимости билетов. В том случае, если пользователь захочет впоследствии отказаться от билетов (посредством того же сервиса), эта сумма остается на его счету, за исключением комиссии за бронирование.
- Пользователь, придя на мероприятие, должен осуществить аутентификацию с билетным турникетом посредством своего мобильного терминала (NFC). При успешной аутентификации (совпадении информации, хранящейся на терминальном аппарате пользователя, с базой данных билетов) пользователь допускается на мероприятие, а с его счета в платежной системе списывается соответствующая сумма.
- В том случае, если пользователь, купивший билет, не посетил мероприятие, соответствующая сумма на его счете деблокируется; при этом снимается комиссия за неиспользование билета.

Для использования сервиса пользователь должен иметь мобильный телефон, поддерживающий технологию NFC, и открытый счет в системе электронных платежей, поддерживающей работу с предоплаченным финансовым продуктом [7]. На телефон должно быть установлено клиентское приложение, которое можно скачать с сайта провайдера. Провайдер, помимо сайта, должен иметь выделенный сервер, подключенный в Интернет и осуществляющий управление базой данных билетов, NFC-турникет и соответствующий счет в системе электронных платежей. На рис. 1 приведена структурная схема мобильного сервиса.

Сервис представляет собой трехкомпонентное распределенное приложение, включающее в себя собственно сервисную часть (приложение провайдера), мобильный терминал и точку контроля. Сервис провайдера включает в себя модуль обработки клиентских запросов, компонент взаимодействия с платежной системой, который отвечает за выполняемые транзакции, и базу данных о текущих местах, проданных билетах, сеансах в целом. Сервис провайдера поддерживает минимальный графический интерфейс, необходимый для его настройки и мониторинга администратором. Мобильный терминал реализуется как Java MIDlet. Его основной функционал заключается в обеспечении интерфейса пользователя в целях выбора и покупки билета, а также получения информации о развлекательном мероприятии. Мобильный терминал включает в себя модуль визуализации, поддерживающий интерактивную карту мест, NFC-модуль для технического обеспечения процесса аутентификации, и модуль сетевого взаимодействия для связи с сервисом провайдера. Точка контроля реализована в виде *stand-alone*-приложения с графическим интерфейсом настройки и отслеживания статуса билетов с использованием интерфейса NFC. Она включает в себя пользовательский интерфейс (в форме индикатора аутентификации), модуль сетевого взаимодействия с серверной частью и NFC-модуль, необходимый для валидации билета.

Связи между функциональными частями сервиса осуществляются по протоколу HTTP поверх протокола TCP/IP. В отличие от устоявшейся процедуры покупки электронных билетов через Интернет, этот сервис сочетает в себе высокий уровень интерактивности с возможностями современных телекоммуникационных технологий.

Предложенная архитектура мобильного сервиса является двухзвенной: услуга предоставляется непосредственно пользователю, оператор сотовой связи играет роль исключительно посредника при предоставлении услуги. Это обеспечивает независимость от конкретных технологических платформ операторов связи.

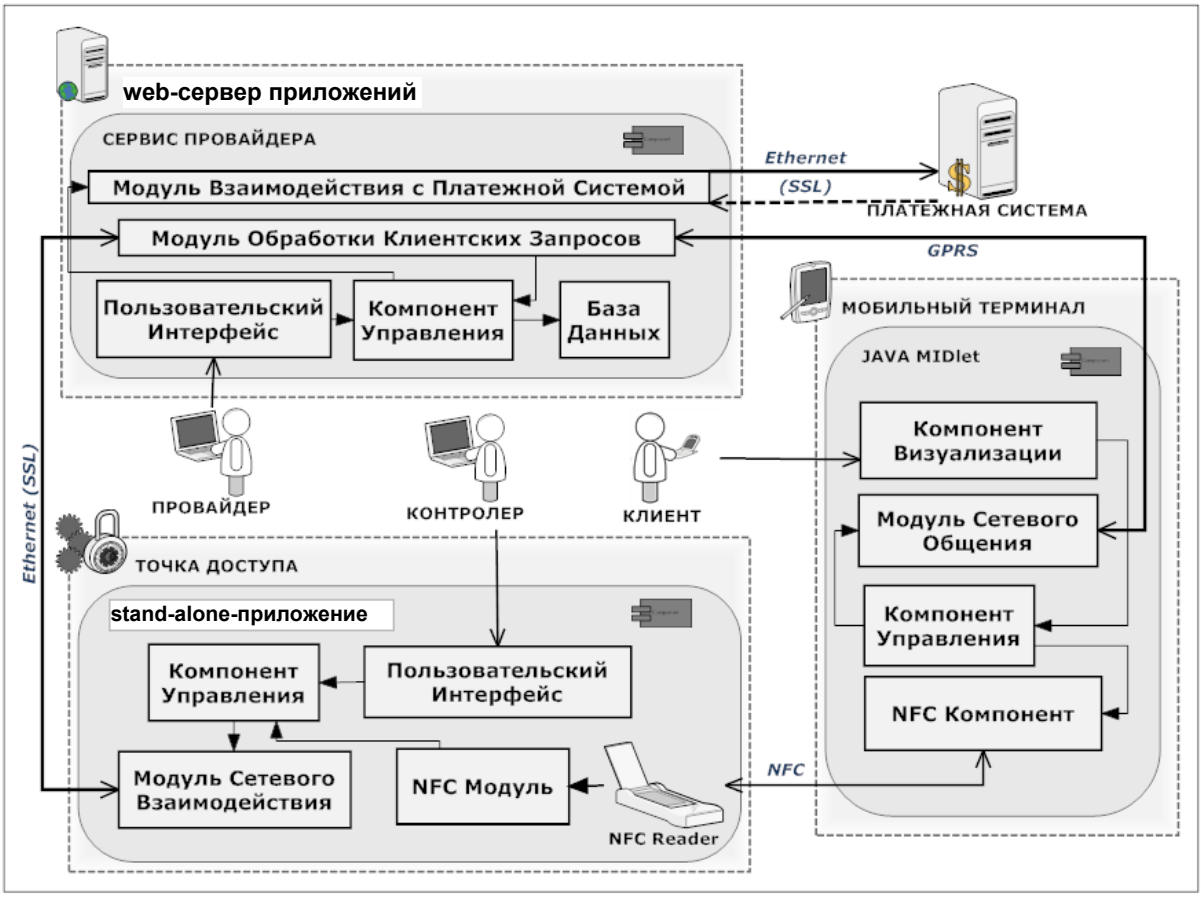


Рис. 1. Структурная схема мобильного сервиса

Для повышения удобства разработки компонентов сервиса допустимо применять инструментальную технологическую среду создания массовых мобильных онлайн-сервисов нового поколения. При помощи разработанной среды разработчик имеет возможность проектировать распределенное приложение путем описания в терминах бизнес-процессов предметной области и на его основе осуществлять автоматическую генерацию кодов, включающих типовые реализации компонент, отраженных на рис. 1. Подробнее вопросы использования инструментальной технологической среды рассмотрены в [7].

Заключение

Мобильные сервисы на основе технологий локального взаимодействия являются нововведением для отечественной отрасли мобильных услуг. Технологическая новизна рассмотренного сервиса продажи билетов определяется двухзвенной организационной схемой и применением прогрессивной технологии локального взаимодействия NFC для предъявления электронного билета в точке контроля. При этом часть функционала сервиса требует удаленного взаимодействия с сервером провайдера услуги, в то время как другие операции могут выполняться без использования каналов сотовой связи (например, аутентификация с билетным турникетом). Массовость применения такого сервиса обеспечивается не только обширной целевой аудиторией пользователей, но и большим

количеством потенциальных провайдеров, заинтересованных в предоставлении данной услуги. Работа выполнена в рамках проекта 2008-4-1.4-18-01-022 ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007–2012 годы».

Литература

1. Ailisto H., et al. Bridging the physical and virtual worlds by local connectivity-based physical selection // Personal and Ubiquitous Computing – 2006. – Vol. 10. – № 6. – P. 333–344.
2. Anokwa Y., et al. A User Interaction Model for NFC Enabled Applications // Proceedings of Fifth IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW'07). – 2007. – P. 357–361.
3. Kostakos V., et al. Building Common Ground for Face to Face Interactions by Sharing Mobile Device Context // Lecture Notes in Computer Science. – 2006. – Vol. 3987. – P. 222–238.
4. NFC Forum. – Режим доступа: www.NFC-forum.org, свободный
5. Morak J., et al. Feasibility and Usability of a Home Monitoring Concept based on Mobile Phones and Near Field Communication (NFC) Technology // Medinfo 2007: Proceedings of the 12th World Congress on Health (Medical) Informatics; Building Sustainable Health Systems. – 2007. – P. 112–116.
6. Choi Y.B., et al. The state-of-the-art of mobile payment architecture and emerging issues // International Journal of Electronic Finance. – 2006. – Vol. 1. – № 1. – P. 94–103.
7. Золотарев О.А., Гринина Е.А., Бухановский А.В. Инструментальная технологическая среда для создания массовых мобильных онлайн-сервисов нового поколения. Часть I: Принцип действия и программная архитектура // Научно-технический вестник СПбГУ ИТМО. – 2008. – Вып. 54. – С. 80–85.
8. ETSI EN 302 190. V.1.1.1 (2005–06), Near Field Communications; Interface and Protocol (NFCIP – 1) / Техническая спецификация ETSI.
9. Leong, C.Y., et al. Near Field Communication and Bluetooth Bridge System for Mobile Commerce // Industrial Informatics. 2006 IEEE International Conference. Singapoure, 16–18 Aug. 2006. – P. 50–55.

Гринина Екатерина Александровна – ООО «Технологии процессинга», руководитель научного отдела, к.ф.-м.н., grinina@tprs.ru

Золотарев Олег Анатольевич – ООО «Технологии процессинга», генеральный директор, zolotarev@tprs.ru

Пименов Илья Андреевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, студент, ilya.pimenov@gmail.com

Насонов Денис Александрович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, студент, denis.nasonov@gmail.com

Бухановский Александр Валерьевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, д.т.н., профессор, boukhanovsky@mail.ifmo.ru

2 ИЗБРАННЫЕ ПУБЛИКАЦИИ I СЕССИИ НАУЧНОЙ ШКОЛЫ «ТЕХНОЛОГИИ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ И СИСТЕМ»

УДК 681.3.069, 681.324

ОСОБЕННОСТИ АДАПТАЦИИ ВЫЧИСЛИТЕЛЬНЫХ АЛГОРИТМОВ ПОД ПАРАЛЛЕЛЬНУЮ АРХИТЕКТУРУ ГРАФИЧЕСКИХ АКСЕЛЕРАТОРОВ

С.В. Ковальчук, С.М. Вишняков

В работе обсуждаются вопросы отображения вычислительных алгоритмов на параллельную архитектуру GPU-акселератора. Описываются важнейшие характеристики рассматриваемой аппаратной архитектуры и их влияние на эффективность работы параллельных алгоритмов. Предлагается модель оценки производительности параллельного алгоритма, использующего вычислительные ресурсы графического акселератора. В качестве примера рассматривается задача оптимизации параметрической аппроксимации двумерной функции методом случайного поиска.

Ключевые слова: графический акселератор, GPGPU, параллельные алгоритмы.

Введение

В последние годы интенсивное развитие получила специфическая отрасль высокопроизводительных вычислений – расчеты на системах с параллельными акселераторами как дополнительными устройствами, принимающими на себя существенную часть вычислительной нагрузки работающего приложения [1]. К таким устройствам относятся, в частности, GPU (Graphic Processor Unit)-устройства, предназначенные для работы с графикой. Современные GPU, по сути, являются многопроцессорными системами SIMD-архитектуры с достаточно высокой (до 0,5 Тфлопс) пиковой производительностью. По сравнению с традиционными архитектурами (например, кластерами), они обладают несопоставимо низкой характеристикой «цена/производительность», что стимулирует интерес к использованию GPU не только для обработки графической информации, но и для решения произвольных вычислительных задач [2]. В частности, к таким задачам относятся сортировки больших объемов данных [3], решения систем линейных уравнений [4], уравнений математической физики [5–7] и пр.

Параллельное программирование для GPU традиционно использует графический программный инструментарий, например, на основе библиотеки OpenGL [8] или DirectX [9]. Для удобства работы с ними применяются пакеты более высокого уровня, например, Gg [10], HLSL [11] и OpenGL Shading Language [12]. Однако реализация вычислительных задач общего плана требует использования более универсальных средств, облегчающих отображение задач различного типа на архитектуру GPU-устройств и позволяющих абстрагироваться от особенностей работы графических алгоритмов. Например, компанией nVidia разрабатывается набор библиотек CUDA SDK [13], предназначенный для использования вычислительных мощностей видеокарт и, в перспективе, специализированных устройств для решения задач общего назначения.

Несмотря на высокую производительность, вычислительные акселераторы на базе GPU являются специфическим классом многопроцессорных систем, характеризуемым существенными ограничениями на масштабируемость, использование памяти и управление данными и задачами. Потому представляет интерес изучение способов отображения различных типов вычислительных алгоритмов на GPU-архитектуру с целью выявления ключевых особенностей этого процесса и факторов, влияющих на получаемую

производительность. В данной работе рассматриваются особенности этого процесса на примере алгоритма параметрической оптимизации методом случайного поиска.

Особенности архитектуры GPU-устройств и средства отображения вычислительных алгоритмов

Акселератор на базе GPU представляет собой набор вычислительных узлов (мультипроцессоров), состоящих из некоторого числа арифметико-логических устройств (АЛУ). В любой момент времени все АЛУ одного мультипроцессора выполняют одинаковую последовательность инструкций над разными наборами данных, расположенных в памяти GPU (т.е. мультипроцессоры имеют SIMD-архитектуру). В данной работе в качестве примера рассматривается акселератор nVidia GeForce 8800 GTX, который имеет 16 мультипроцессоров по 8 АЛУ в каждом.

Поскольку в SIMD-устройствах каждый мультипроцессор конфигурируется определенным образом для выполнения заданной последовательности команд на множестве входных данных, то перед разработчиком стоит задача формирования последовательности инструкций, решающей поставленную задачу конфигурации устройства и передачи в устройство потока входных данных. При использовании nVidia CUDA SDK эта процедура выглядит следующим образом: разработчик описывает ядро (процедуру, которая будет исполняться на GPU) и при запуске ядра на GPU задает ему конфигурацию на основе описанной ниже иерархии. Каждый поток, физически выполняющийся на АЛУ мультипроцессора, исполняет инструкции, описанные в ядре. При этом благодаря SIMD-архитектуре на каждом мультипроцессоре несколько потоков параллельно выполняют одну и ту же последовательность инструкций. Логически потоки объединяются в блоки, ограничивающие возможность обмена данными между потоками (обмен данными через общую память только внутри одного блока). Потоки из одного блока выполняются на одном и том же мультипроцессоре. Таким образом, перед запуском ядра разработчик задает его конфигурацию (grid) – размер блока и общее число блоков. Для более удобного структурирования конкретных задач в CUDA предусмотрена возможность конфигурирования блоков и потоков в двумерные сетки. Данная возможность обеспечивает удобство при обработке структурированных данных. Отметим еще один важный структурный элемент конфигурации ядра – основу (warp). Под основой (в терминологии nVidia) подразумевается набор потоков, инструкции которых выполняются на мультипроцессоре одновременно. Размер основы определяется архитектурой конкретного GPU-устройства, и размер блока, определяемый при конфигурировании ядра, должен быть пропорционален размеру основы.

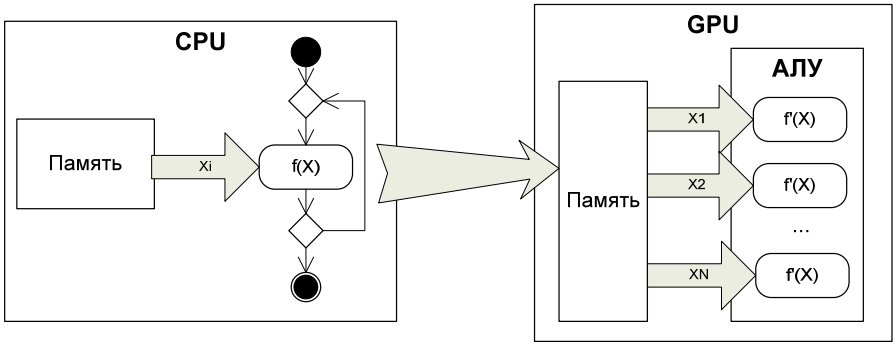


Рис. 1. Схема отображения вычислительного алгоритма на GPU-устройство

Ключевым моментом отображения вычислительного алгоритма на GPU-архитектуру является определение ядра, соответствующего решению поставленной задачи. При этом следует учитывать, что каждый из потоков, исполняющих код ядра, для получения входных данных должен обращаться к своей области памяти устройства. На

рис. 1 представлена общая схема преобразования последовательного циклического алгоритма в его GPU-реализацию.

При исполнении на CPU-системе алгоритм многократно последовательно вычисляет некоторую функцию $f(X)$ на различных наборах данных X_i. При исполнении на GPU указанная функция преобразуется к виду $f'(X)$, удобному для исполнения в рамках одного потока, и затем вычисляется на GPU параллельно, на разных наборах данных. Разумеется, данный подход является эффективным лишь в том случае, когда итерации цикла, вычисляющего $f(X)$, независимы. Заметим также, что при описании вычислительного ядра необходимо учитывать ряд важных особенностей работы GPU-устройств. Во-первых, SIMD-архитектура накладывает ограничения на использование ветвящихся конструкций (if/else). Это связано с тем, что потоки, исполняющиеся на одном и том же мультипроцессоре, должны оперировать одной и той же последовательностью инструкций, что может нарушаться при несовпадении условий ветвления у разных потоков. При нарушении этого ограничения может происходить сильное ухудшение производительности. Во-вторых, имеет место особенность работы GPU-устройств с памятью – определение ядра таким образом, чтобы одновременно выполняющиеся потоки читали данные из соседних участков памяти, существенно увеличивает производительность. Кроме того, необходимо учитывать трудоемкость обращения к памяти GPU-устройства. В-третьих, многое зависит от архитектуры конкретного GPU-устройства, на котором выполняется вычислительная задача. Более подробно эти и остальные особенности GPU-архитектуры рассмотрены в [13].

Отображение алгоритма параметрической оптимизации методом случайного поиска на архитектуру GPU-устройства

В качестве примера вычислительного алгоритма, отображаемого на GPU-архитектуру, в работе была выбрана задача параметрической аппроксимации двумерной функции с использованием метода случайного поиска [15].

В процессе работы данного алгоритма оптимизации производится случайный выбор направления в пространстве поиска, $V \in R^n$. В работе используется вариация метода адаптивного случайного поиска с линейной тактикой:

$$V_n = V_{n-1} + \Delta V_n \xi^{(j)}, \tag{1}$$

где $\xi^{(j)}$ – единичный орт в случайном направлении j. В этом случае шаг ΔV_n в случайно выбранном направлении повторяется до тех пор, пока он не перестанет приводить к уменьшению значения целевой функции $J(V)$:

$$\Delta V_n = \begin{cases} a\xi, \text{ если } \Delta J_{n-1} \geq 0, \\ \Delta V_{n-1}, \text{ если } \Delta J_{n-1} < 0. \end{cases} \tag{2}$$

Здесь a – начальный размер шага для нового направления. Для изменения длины шага в процессе работы алгоритма используется параметр релаксации. Решение об изменении шага принимается в зависимости от успешности предыдущей операции:

$$\left|\Delta V_n\right| = \begin{cases} \gamma_1 \left|\Delta V_{n-1}\right|, \text{ в случае успеха,} \\ \gamma_2 \left|\Delta V_{n-1}\right|, \text{ в противном случае.} \end{cases} \tag{3}$$

Для определения параметров релаксации используется следующее соотношение:

$$\gamma_1^p \gamma_2^{(1-p)} = 1. \tag{4}$$

Алгоритм имеет широкое применение, в частности, при решении задачи аппроксимации спектров морского волнения [16]. Одной из особенностей этой задачи является необходимость обработки больших массивов входных данных, что, с одной стороны,

приводит к большой трудоемкости вычислений, а с другой – позволяет эффективно использовать распараллеливание по данным. В результате эффективно используются вычислительные мощности GPU-устройств, поскольку для каждого набора входных данных решается одна и та же задача.

Анализ параллельной производительности

Анализ параллельной производительности проведен для алгоритма, основанного на декомпозиции по данным, при котором в каждом потоке выполняется подсчет значения целевой функции для своего набора входных данных. Предметом исследования являлось влияние конфигурации системы на получаемое ускорение, а также оценка составных частей времени работы алгоритма. Кроме того, был исследован эффект применения различных типов оптимизаций вычислительного ядра. Для оценки ускорения время работы с использованием GPU-устройства nVidia GeForce 8800 GTX сравнивалось со временем обработки тех же данных на CPU Intel Core 2 Duo 2.3 GHz.

На рис. 2 представлена зависимость получаемого ускорения от конфигурации GPU-устройства (число потоков в блоке, умноженное на число блоков) ядра.

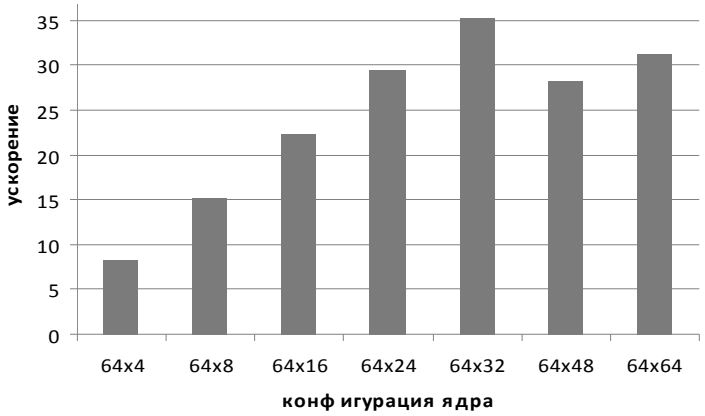


Рис. 2. Зависимость ускорения от конфигурации ядра

Рост ускорения при увеличении числа блоков от 4 до 16 объясняется тем, что в исследуемой системе 16 мультипроцессоров. Таким образом, при конфигурации в 4 и 8 блоков часть мультипроцессоров остается бездействующей. При дальнейшем увеличении числа блоков все мультипроцессоры загружены работой, тем не менее, ускорение продолжает расти. Этот факт объясняется особенностью обращения мультипроцессоров к памяти: если на одном мультипроцессоре, в соответствии с конфигурацией, выполняется одновременно несколько блоков, то, в то время как процессы одного блока ждут данные из памяти, на мультипроцессоре могут выполняться арифметические операции других блоков. Схема на рис. 3 поясняет данное утверждение: первоначально при увеличении числа блоков (а, значит, и количества входных данных) общее время работы практически не меняется.

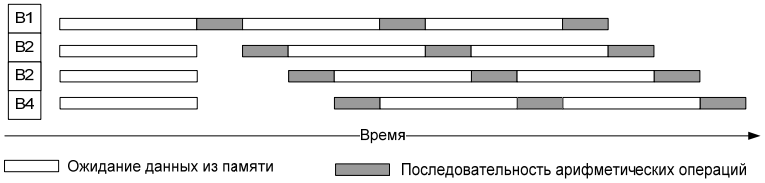


Рис. 3. Одновременное выполнение блоков на мультипроцессоре GPU-устройства

Аналогичный эффект имеет место при последовательности арифметических вычислений, в процессе которых результат одной операции используется в последую-

щей [13]. Задержки, возникающие в данной ситуации, компенсируются аналогичным образом. В тот момент, когда число блоков достигает некоторого критического значения, определяемого максимальным числом блоков, которое возможно запустить на одном мультипроцессоре при заданном размере блока и заданном ядре, время работы ядра резко увеличивается. Это число зависит от того, сколько ресурсов мультипроцессора используется одним потоком заданного ядра. Зависимость максимального числа блоков от числа регистров и размера разделяемой памяти, используемой ядром, можно найти в *nVidia Occupancy Calculator* [13]. На рис. 5 представлены результаты, полученные в процессе измерения составных частей времени работы алгоритма на GPU-устройстве:

$$T = T_{malloc}+T_{memcpyin}+T_{kernel}+T_{memcpyout}+T_{free}. \quad (5)$$

Время T состоит не только из времени выполнения ядра T_{kernel}, но и времени, затрачиваемого на выделение T_{malloc} и освобождение T_{free} памяти на устройстве и копирования данных $T_{memcpyin}$, $T_{memcpyout}$.

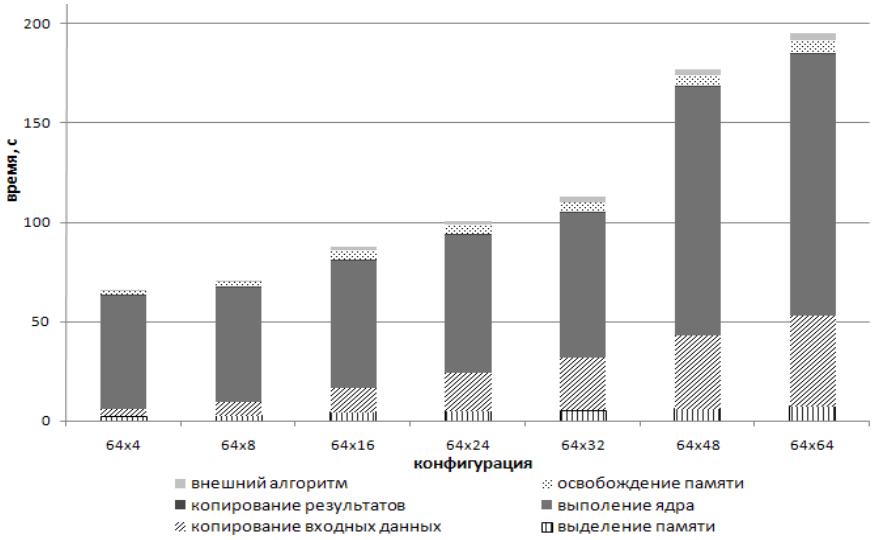


Рис. 5. Составные части времени работы

Время исполнения ядра практически не меняется с увеличением числа блоков от 4 до 32 – сначала из-за неполной загруженности устройства, затем – из-за описанного выше эффекта, в то время как при увеличении числа блоков до 48 происходит резкий скачок. Время, затрачиваемое на работу с памятью, растет при этом линейно (так как с увеличением числа блоков увеличивается и количество входных и выходных данных). Заметим, что это время в данном случае составляет существенную часть от общего времени. Таким образом, можно сделать вывод о том, что в задачах, использующих повторяющиеся наборы данных, следует избегать повторного копирования и выделения памяти там, где это возможно.

Изучим более подробно влияние конфигурации ядра на производительность алгоритма и попробуем получить теоретическую модель производительности. Для этого упростим ситуацию, исключив из рассматриваемого выше алгоритма недетерминированность, сведя его к многократному подсчету интеграла методом прямоугольников, и проведем ряд измерений с различными параметрами конфигурации ядра.

На рис. 6, а, представлена зависимость производительности ядра (количества обработанных данных, отнесенного ко времени работы ядра) от числа блоков в конфигурации ядра для различного числа потоков на блок. Вид зависимости производительности обусловливается ступенчатым характером зависимости времени работы ядра от тех же параметров (рис. 6, б). Ступенчатый характер этой зависимости объясняется описанным выше эффектом: при увеличении общего числа блоков время работы ядра не

изменяется до определенного момента благодаря тому, что операции добавляемых блоков выполняются во время «простоя» мультипроцессоров.

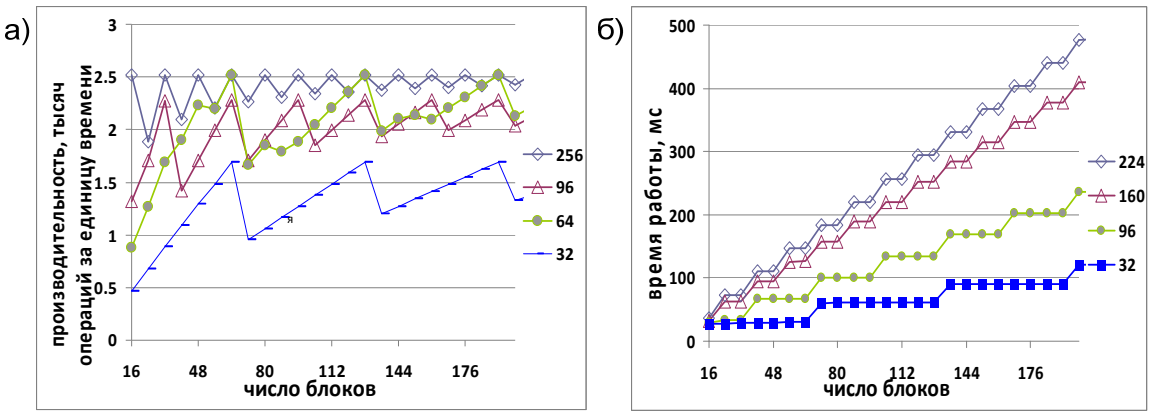


Рис. 6. Характеристики ядра

Попытаемся вывести зависимость, представленную на рис. 7, аналитически, учитывая тот факт, что «насыщение» мультипроцессоров происходит в первый раз в тот момент, когда общее число блоков, приходящееся на мультипроцессор, требует больше ресурсов, чем имеется на мультипроцессоре. Иначе говоря, в этот момент все блоки не могут работать на мультипроцессоре одновременно. Таким образом, становится верным неравенство

$$\frac{nblocks * (blockSize / warpSize)}{M} > mWarps * occupancy, \qquad (6)$$

где *nblocks* – общее число блоков в конфигурации ядра; *blockSize* – число потоков в блоке, *warpSize* – число потоков в основе (определяется архитектурой GPU-устройства); *mWarps* – максимальное число основ, которое может одновременно работать на одном мультипроцессоре GPU-устройства данной архитектуры; *occupancy* – отношение максимального числа основ, которое можно запустить на одном мультипроцессоре, исходя из использования данным ядром ресурсов и количества доступных ресурсов на мультипроцессоре, к *mWarps*. Параметр *occupancy* можно подсчитать при помощи *nVidia Occupancy Calculator* [13], зная количество регистров и разделяемой памяти, используемых ядром.

Исходя из приведенных выше соображений, можно предложить следующую аналитическую формулу для времени работы ядра:

$$T_{kernel} = t_0(blockSize) \cdot \frac{N}{nblocks \cdot blockSize} \cdot \left\lceil \frac{nblocks \cdot blockSize}{M \cdot mWarps \cdot occupancy \cdot warpSize} \right\rceil = \\ = t_0(blockSize) \cdot workPerThread \cdot \left\lceil \frac{warpsPerMP}{mWarps \cdot occupancy} \right\rceil, \qquad (7)$$

где t_0 – «эффективное» время работы ядра, которое вычисляется как время работы ядра при полной загрузке мультипроцессора перед насыщением (фактически – высота первой ступени графика), отнесенное к числу элементарных операций, выполняемых ядром; N – общее число элементарных операций, выполняемых ядром; *workPerThread* – число элементарных операций, приходящихся на один поток; *warpsPerMP* – количество основ, приходящихся на один мультипроцессор при данной конфигурации ядра.

Вычислив значения $t_0(blockSize)$ на основе данных, представленных на рис. 7, можно, используя формулу (7), построить теоретические графики зависимости времени работы ядра от его конфигурации для другого числа элементарных операций, выполняемых ядром. Сравнение теоретической зависимости с зависимостью, полученной на практике, представлено на рис. 7.

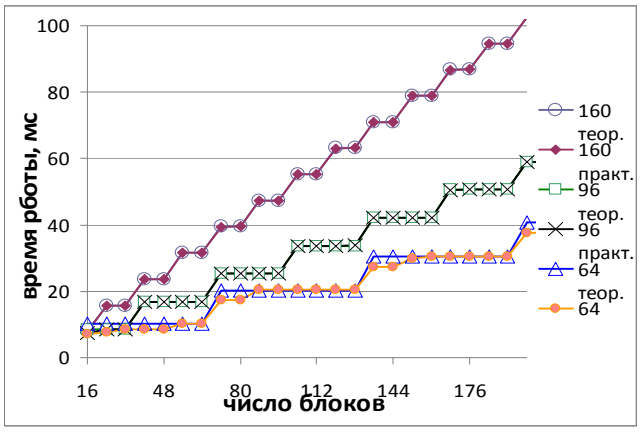


Рис. 7. Сравнение теоретической и практической зависимостей времени работы ядра от конфигурации

Как видно, теоретическая зависимость довольно хорошо совпадает с экспериментальными данными. Расхождения наблюдаются лишь в началах «ступеней» графика в случаях, когда блок содержит четное число основ. Причина этого расхождения является открытым вопросом.

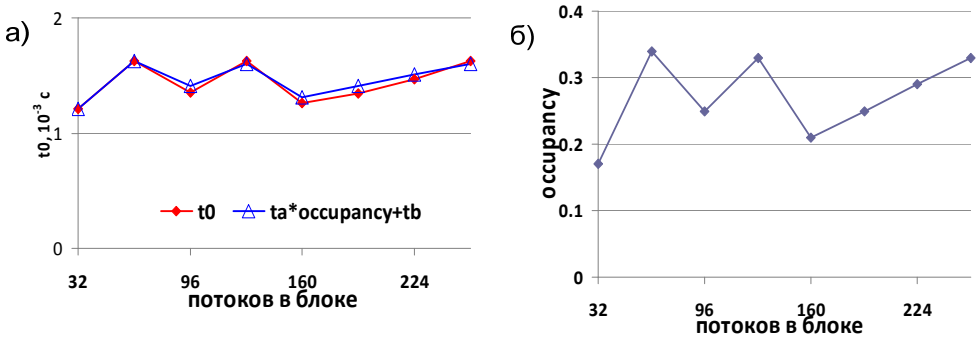


Рис. 8. Зависимость t_0(*blockSize*) и ее теоретический вид (а), зависимость *occupancy*(*blockSize*) (б)

Экспериментальные данные показывают, что параметр t_0 в формуле (7) зависит от размера блока. Соответствующая экспериментальная зависимость представлена на рис. 8, а. Заметим, что зависимость параметра *occupancy* от размера блока (рис. 8, б) носит похожий характер, согласно [13]. Этот факт позволяет предположить, что параметры t_0 и *occupancy* связаны линейным соотношением: $t_0 = t_a \cdot occupancy + t_b$. Приняв это предположение и вычислив значения t_a и t_b из экспериментальных данных, можно сравнить теоретическую зависимость t_0(*blockSize*) с полученной экспериментально.

Заключение

Результаты вычислительных экспериментов на задаче аппроксимации климатических спектрах морского волнения показывают, что адаптация алгоритма параметрической оптимизации к архитектуре GPU позволяет получить ускорение более 30 раз по сравнению с реализацией без использования GPU. Учитывая, что стоимость простейшей кластерной системы на основе стандартных комплектующих, обеспечивающих такое же ускорение, примерно в 30 раз превышает стоимость рассматриваемого GPU-устройства, это подтверждает целесообразность применения GPU-устройств при решении вычислительных задач подобного класса. В планах дальнейших исследований – разработка методов отображения других алгоритмов на архитектуру GPU-устройств.

Литература

1. Hasle G., Lie K.-A., Quak E. Geometric Modelling, Numerical Simulation, and Optimization: Applied Mathematics at SINTEF. – Springer, 2007. – 558 p.
2. General-Purpose Computation Using Graphics Hardware. – Режим доступа: http://gpgpu.org/, свободный.
3. Purcell T.J., Donner C., Commarano M., Jensen H.W., Hanrahan P. Photon mapping on programmable graphic hardware // Proceeding of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware. – Eurographics Association. – 2003. – P. 41–50.
4. Göddeke M., Strzodka R., Turek S. Accelerating Double Precision FEM Simulations with GPUs // Proceeding of ASIM 2005. – 18th Symposium on Simulation Technique.– 2005. – P. 139–144.
5. Hagen T.R., Henriksen M.O., Hjelmervik J.M., Lie K.-A. Using the graphic processor as a high–performance computational engine for solving system of hyperbolic conservation low // Geometric Modelling, Numerical Simulation, and Optimization: Applied Mathematics at SINTEF. – Springer, 2007. – P. 211–264.
6. Hagen T.R., Hjelmervik J.M., Lie K.–A., Natvig J.R., Henriksen M.O. Visual simulation of shallow–water waves // Simulation Practice and Theory. Special Issue on Programmable Graphics Hardware. – 2005. – V.13. – № 9. – P. 716–726.
7. Hagen T.R., Lie K.-A., Natvig J.R. Solving the Euler equation on graphical processing units // Computational Science – ICCS 2006: 6th International Conference, Reading, UK, May 28–31, 2006, Proceedings, Part IV, volume 3994 of Lecture Notes in Computational Science (LNCS). – Springer Verlag, 2006. – P. 220–227.
8. OpenGL – The Industry Standard for High Performance Graphics. – Режим доступа http://www.opengl.org/, свободный.
9. Microsoft's DirectX site. – Режим доступа: http://www.microsoft.com/directx, свободный.
10. Fernando R., Kilgard M.J. The Cg Tutorial: The Definitive Guide to Programmable Real–Time Graphics. – Adisson–Wesley Longman Publishing Co., 2003.
11. GPUBench: How much does your GPU bench. – Режим доступа: http://graphics.stanford.edu/projects/gpubench/, свободный.
12. Rost R.J. OpenGL Shading Language. – Adisson–Wesley Longman Publishing Co, 2004.
13. NVIDIA CUDA Compute Unified Device Architecture Programming Guide. Ver 1.0. June 2007. – NVIDIA Corporation, 2007.
14. Адинец А.В., Сахарных Н.А. О программировании вычислений общего назначения на графических процессорах // Научный сервис в сети Интернет: многоядерный компьютерный мир. 15 лет РФФИ: Труды Всероссийской научной конференции.– М.: Издательство МГУ, 2007. – С. 249–256.
15. Растригин Л.А. Адаптация сложных систем – Рига: Зинатне, 1981. – 375 с.
16. Boukhanovsky A.V., Lopatoukhin L.J., Guedes Soares C. Spectral wave climate of the North Sea // Applied Ocean Research. – 2007. – Vol. 29. – P. 146–154.

Ковальчук Сергей Валерьевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, м.н.с., kovalchuk@mail.ifmo.ru

Вишняков Сергей Михайлович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, студент, sergey.vishnyakov@gmail.com

УДК 004.021, 681.3.069

ИДЕНТИФИКАЦИЯ ПАРАМЕТРИЧЕСКИ СВЯЗАННЫХ МОДЕЛЕЙ СЛОЖНЫХ СИСТЕМ

С.В. Иванов

В работе рассмотрены подходы к идентификации моделей сложных систем в условиях параметрической связи моделей, описывающих различные диапазоны изменчивости и неполноты исходных данных. Эффективность предложенного подхода исследована на примере численных экспериментов различных моделей сложных систем. Приведены схемы идентификации параметров двух моделей в системе «океан–атмосфера» и модели популяционной динамики ВИЧ.
Ключевые слова: идентификация моделей, сложные системы.

Введение

Развитие методов и технологий компьютерного моделирования стимулирует интерес исследователей к изучению так называемых сложных систем (*complex systems*). Под сложной системой подразумевается [1] система, которая (а) состоит из большого числа компонентов; (б) допускает «дальние» связи между компонентами; (в) обладает многомасштабной (в том числе пространственно-временной) изменчивостью. Перечисленные свойства порождают ряд специфических проблем компьютерной реализации моделей сложных систем. Так, большое число компонентов отражается на ресурсоемкости вычислительных процедур и повышенных требованиях к объемам оперативной памяти для хранения структур данных. Многомасштабность сложных систем требует использования параметрически связанных моделей, описывающих иерархию смежных диапазонов изменчивости. Учет дальних связей, когда каждый компонент системы может взаимодействовать с каждым, приводит к необходимости предварительного решения проблемы связности при применении параллельных вычислений, что затрудняет или даже приводит к невозможности распараллеливания формальными методами. В силу условности выделения границ диапазонов изменчивости процедуры связывания или замыкания таких моделей традиционно используют эмпирические закономерности и коэффициенты, значения которых определяются посредством идентификации. Задача идентификации моделей заключается в определении их структуры и параметров по результатам наблюдений над входными и выходными переменными реальной системы и решается методами оптимизации [2].

Особенности идентификации моделей сложных систем

Идентификация моделей сложных систем тесно связано со спецификой самих моделей и с особенностями представления данных измерений. Так, можно отметить следующие особенности измерений:

- *фрагментарность,* т.е. невозможность наблюдения над всеми состояниями системы в каждый момент времени, что объяснимо в первую очередь большим числом элементов системы и экономической нецелесообразностью или даже технической невозможностью проведения таких измерений.
- *неоднородность***,** которая возникает в силу различий в постановках экспериментов, методов и средств измерения.
- *разномасштабность*, состоящая в том, что данные измерений относятся к разным масштабам изменчивости. Эта особенность в большей степени относится к сложности выбора и правильной интерпретации тех данных, которые предполагается использовать в рамках процедуры идентификации, а также к выбору данных для независимой проверки полученных результатов (верификации и валидации модели).

В итоге данные измерений можно описать в виде набора векторов $\text{ℰ} = \left(\text{ℰ}_{ij}(v)\right), i \in 1\ldots N, j \in 1\ldots M$, где ℰ_{ij} – разрозненные данные измерений по N источникам и M масштабам изменчивости, а v – пространственная и (или) временная координата. При этом особенности представления моделей связаны со способом представления их параметров. Так, вектор неизвестных (настраиваемых) параметров удобно представить как $\Xi = (\Xi_1, \Xi_2, \ldots, \Xi_N)$, где Ξ_i – параметры i-ой модели, которые в общем случае также являются вектором. При моделировании сложных систем в результате иерархической структуры моделей выходные величины моделей более высокого уровня используются как параметры моделей более низкого уровня. Данную особенность сложных систем удобно представить в следующей форме:

$$\begin{cases} y_N = F_N(v, \Xi_N), \\ y_{N-1} = F_{N-1}(v, y_N, \Xi_{N-1}), \\ \ldots \\ y_1 = F_1(v, y_N, \ldots, y_2, \Xi_1). \end{cases} \quad (1)$$

В итоге общая задача идентификации модели сложной системы состоит в минимизации невязки между выходами моделей разного уровня $F_1(\Xi_1), \ldots, F_N(\Xi_N)$ и разрозненными наблюдениями ℰ за поведением реальной системы $J(F_1, \ldots, F_N, \text{ℰ}) \xrightarrow[\Xi]{} \min$ при заданном наборе ограничений $G(\Xi_i) \in \Omega_i, i \in 1 \ldots N$.

Функциональная схема процедуры идентификации, реализующей описанную логику, приведена на рис. 1.

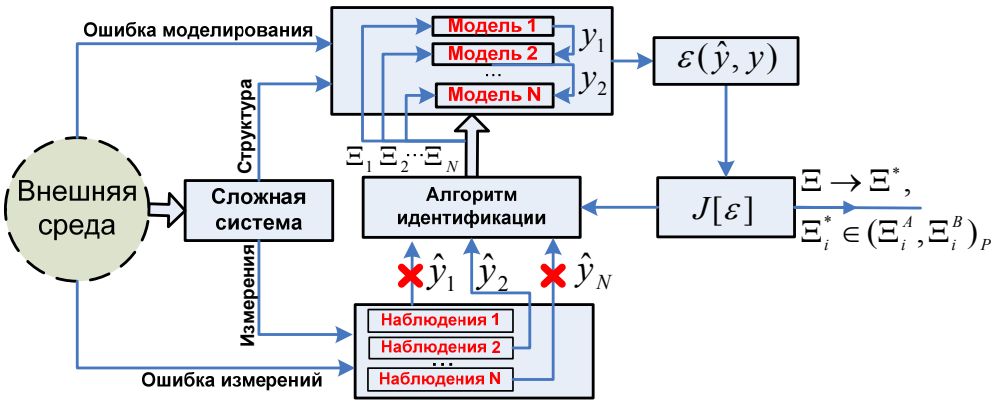


Рис. 1. Функциональная схема процедуры идентификации модели сложной системы

Особенностью данной схемы является неполнота данных наблюдений, относящихся к моделям разного уровня. Результатом процедуры идентификации являются оценки вектора параметров Ξ^*, принадлежащих некоторому вероятностному интервалу. В качестве основного алгоритма идентификации для описанных в работе моделей был выбран адаптивный алгоритм случайного поиска с переменным шагом, который легко реализуется и модифицируется под конкретные задачи, имеет линейную зависимость вычислительной сложности от числа параметров и позволяет работать с произвольными ограничениями, в том числе нелинейными.

Идентификация параметров модели приводного ветра

Расчет полей приводного ветра описывается рядом моделей, работающих в различных диапазонах изменчивости. Приведем основные физические соотношения, лежащие в основе модели.

Соотношение сил Кориолиса и градиента атмосферного давления P определяет движение атмосферы, называемое *геострофическим ветром*. В случае прямолинейных изобар компоненты геострофического ветра могут быть определены по формуле

$$(u_g, v_g) = \frac{1}{f_k \rho}\left(-\frac{\partial P}{\partial y}, \frac{\partial P}{\partial x}\right), \qquad (2)$$

где $f_k = 2\Omega\sin(\varphi)$ – параметр Кориолиса; ρ – плотность воздуха, Ω – угловая скорость вращения Земли; φ – широта места, u_g и v_g – компоненты геострофического ветра. Поля давления на уровне моря для глобальных массивов гидрометеорологических данных записываются на регулярной широтно-долготной сетке. Однако применение (2) требует использования метрической сетки, по которой рассчитываются разностные аналоги производных и вычисляются радиусы кривизны изобар. В настоящее время не существует надежных рекомендаций по выбору шага регулярной сетки, поскольку он существенно зависит от источника данных, поэтому шаг регулярной сетки по долготе δx и широте δy (т.е., по сути, размер ячейки) является предметом идентификации.

Как правило, атмосферные течения не являются прямыми, а двигаются вдоль криволинейных траекторий. Это подразумевает дополнительные ускорения вдоль радиусов кривизны, т.е. дополнительную центростремительную силу, что порождает градиентный ветер. Модуль градиентного ветра можно вычислить по формуле $Gr = \frac{f_k R}{2}\left[-1 + \sqrt{1 + \frac{4G}{f_k R}}\right]$, где f_k – параметр Кориолиса, $G = \sqrt{u_g^2 + v_g^2}$ – скорость геострофического ветра, R – радиус кривизны изобары в интересующей точке.

Для перехода от скорости градиентного ветра Gr к скорости приводного ветра на высоте 10 м G_{10} необходимо знать коэффициенты $\beta = \frac{G_{10}}{Gr}$ и α – угол отклонения направления ветра от изобары. В теоретических исследованиях существуют значительные расхождения в определении значений параметров β и α. Как следствие, эти параметры также являются предметом идентификации. Параметры $(\delta x, \delta y)$ характеризуют синоптический диапазон изменчивости, а параметры (α, β), будучи интегральными характеристиками, соответствуют климатическому диапазону, включающему в себя масштабы сезонной и межгодовой изменчивости. Их совместное определение требует рассмотрения всей системы масштабов совокупно.

Идентификация параметров модели, по сути, разбивается на два этапа (рис. 2).

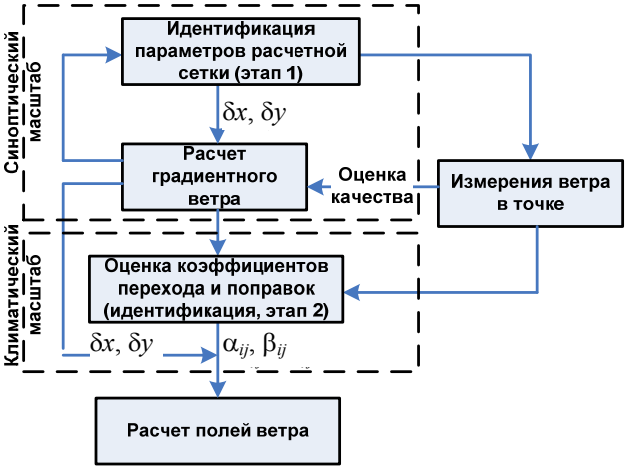


Рис. 2. Общая схема идентификации модели приводного ветра в синоптическом и климатическом диапазонах изменчивости

На первом этапе определяются параметры расчета прямоугольной сеточной области $(\delta x, \delta y)$. В качестве критерия качества идентификации предлагается использовать векторный коэффициент корреляции между геострофическим ветром и измерениями приводного ветра. Вторым этапом как в алгоритме моделирования, так и в процедуре идентификации является оценка коэффициентов перехода от градиентного ветра к приводному ветру, а также углов отклонения направления ветра от изобары в зависимости от стратификации атмосферы. При расчете этих коэффициентов необходимо, помимо значения модуля и направления смоделированного градиентного ветра, использовать значения температуры воды и воздуха в данной точке акватории (стратификация атмосферы). Коэффициенты перехода (α, β) определяются через средние многолетние значения отношения измеренного и смоделированного градиентного ветра в соответствующих диапазонах разностей температур воды и воздуха и силы градиентного ветра. Предложенный подход к идентификации параметров модели приводного ветра был использован для расчета ветра в акватории Атлантического океана. В качестве измерений были использованы данные с буя 44004, имеющего координаты 38,47° (с.ш.) и 70,56° (з.д.).

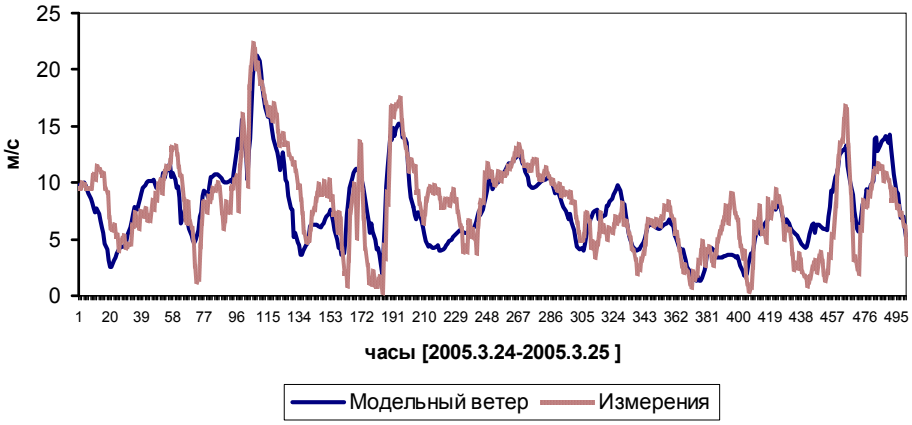


Рис. 3. Фрагменты реализаций моделей ветра для акватории Атлантического океана

Уточнение параметров модели посредством идентификации позволило повысить качество результирующих данных (векторный коэффициент корреляции составляет 0,82 по отношению к измерениям; для данных реанализа NCEP/NCAR эта величина составляет 0,74).

Идентификация параметров модели климатических спектров

Под климатическим спектром волн понимают характерный элемент ансамбля спектров, имеющий определенную вероятность и обусловливающий некоторые характерные волновые условия на заданной акватории. Функционально подобные классы спектров могут состоять из нескольких волновых систем и представлять собой хорошо известные аппроксимации, что позволяет представить спектр $S(\omega,\theta)$ в виде $S(\omega,\theta,\Xi)$, где Ξ – набор параметров. В данной работе рассматривается аппроксимация спектра вида JONSWAP [3], классическая запись которого в частотной области ω имеет вид

$$S(\omega) = \frac{\alpha g^2}{\omega^5}\exp\left[-1{,}25\left(\frac{\omega}{\omega_p}\right)^{-4}\right]\gamma^{\delta},\quad \delta = \exp\left[-\frac{(\omega-\omega_p)^2}{2\sigma^2\omega_p^2}\right]. \qquad (3)$$

Здесь α – так называемый параметр Филипса (параметр масштаба спектра), γ – параметр пиковатости, σ – параметр формы, ω_p – частота максимума спектра.

Даже при постоянном и устойчивом ветре волны распространяются не в одном фиксированном направлении, а в некотором секторе. Частотно-направленные спектры ветрового волнения и зыби можно представить в форме $S(\omega,\theta)=S(\omega)Q(\theta)$, где $S(\omega)$ – частотный спектр волнения, а $Q(\theta)$ – функция углового (θ) распределения энергии

$$Q(\theta)=C(s)[\cos(\frac{\theta-\overline{\theta}}{2})]^s,\ C(s)=\left[\int_{-\pi/2}^{\pi/2}\cos(\theta)^S d\theta\right]^{-1}. \quad (4)$$

Поскольку объектом анализа является спектральная плотность случайного процесса волнения (как квадратичная характеристика энергии волн), критерием качества для процедуры идентификации является сумма квадратов невязки модельного спектра и измерений $S(\omega,\theta)$. Таким образом, расчетная процедура сводится к нелинейной оптимизации функционала

$$J(\Xi_1\ldots\Xi_N)=\int_0^{\infty}\int_0^{\pi}\left[S(\omega,\theta)-\sum_{i=1}^{N}S_i(\omega,\theta,\Xi_i)\right]^2 d\omega d\theta \quad (5)$$

относительно набора параметров $\Xi_{1\ldots N}$ (N – неизвестное число волновых систем).

В результате идентификации параметров модели спектра выполняется снижение мерности массива данных – сведение $S(\omega,\theta,x,y,t)$ к набору параметров $\Xi(x,y,t)$ существенно меньшей мерности, чем исходные данные, что позволяет производить классификацию спектров относительно полученных параметров.

На рис. 4 приведена схема идентификации модели климатических спектров морского волнения. Ее отличием от аналогичной схемы для расчета полей приводного ветра (рис. 2) является применение процедуры идентификации к наблюдениям синоптического диапазона для получения характеристик в климатическом диапазоне, в то время как для модели расчета полей ветра наблюдения как раз климатического диапазона использовались для настройки модели в синоптическом диапазоне.

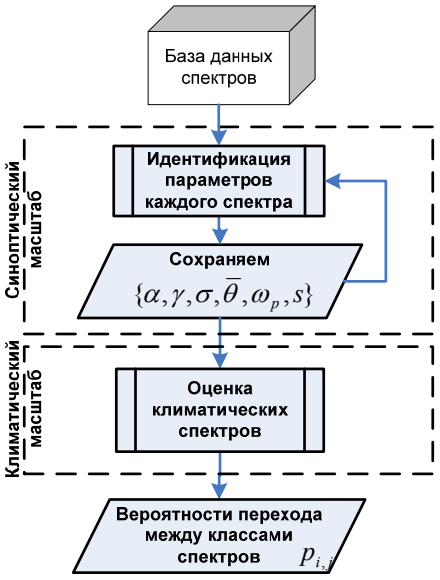


Рис. 4. Общая схема идентификации модели климатических спектров

Модель (3) различает одно-, двух- и многопиковые (по переменным ω и θ) спектры. Выделены пять классов климатических спектров: ветровые волны; зыбь (k=2); ветровые волны и близкая по частоте «свежая» зыбь (k=3); ветровые волны и «свежая» зыбь, различающиеся и по частоте, и по направлению (k=4); ветровые волны и зыбь, близкие (плохо различающиеся) по частоте и направлению (k=5). Каждый класс соответствует устойчивому состоянию k, следовательно, синоптическая изменчивость вол-

нения может быть представлена как марковская цепь $k = k(t)$ с матрицей переходных вероятностей $p_{ij}^{(t,t+1)} = P\{k^{(t+1)} = i \mid k^{(t)} = j\}, \quad i, j = \overline{1,m}$, и вектором предельной вероятности $\pi_j = P\{k^{(t)} = j\}, j = \overline{1,m}$. Оценка коэффициентов в матрице переходных вероятностей и векторе предельной вероятности также является частью процедуры идентификации параметров модели.

В табл. 1 показана перемежаемость климатических спектров в течение года для центральной точки Тирренского моря.

Таблица 1. Вероятностные характеристики (%) перемежаемости климатических спектров морского волнения. ВЕСЬ ГОД

Класс	Переходные вероятности из класса в класс (%)					Повторяемость по классам (%)
	I	II	III	IV	V	
I	61	-	36	-	3	11
II	-	66	30	3	1	3
III	8	2	75	1	14	46
IV	1	-	23	51	25	4
V	1	-	14	4	81	36

Идентификация параметров эпидемиологической модели ВИЧ

Несмотря на то, что модель эпидемиологической динамики ВИЧ относится к специфической предметной области и использует принципиально иной математический аппарат, при идентификации ее параметров допустимо использовать те же самые подходы, что и в задачах, описанных в предыдущих разделах.

Эпидемиологическая динамика ВИЧ описывается моделью комплексной сети, в которой каждый узел представляет собой отдельного человека, находящегося в одном из трех основных состояний: восприимчивый к вирусу (S), зараженный (I), удаленный из сети в результате диагностики СПИД или смерти (R). Сетевые связи (ребра графа) указывают на наличие контактов между людьми, которые могут приводить к новым случаям заражения. Для корректного описания динамики вируса внутри отдельного человека между состояниями *зараженный* и *удаленный* вводится ряд дополнительных промежуточных состояний, описываемых нестационарной марковской моделью специального вида [4]. Нестационарность модели определяется наличием факторов, оказывающих влияние на вероятность заражения, эффектом диагностики и лечения.

Комплексная сеть задается набором $\{G, \Im\}$, где G – ориентированный граф с множествами вершин V и ребер E, а $\Im$ – эволюционный оператор, определяющий изменение сети за интервал времени t: $\langle V, E\rangle_{t+1} = \Im\langle V, E\rangle_t$. Эволюционный оператор может быть представлен как композиция нескольких операторов $\Im = \bigotimes_k \Im_k$, соответствующих различным динамическим факторам (процессу распространения инфекции, динамике ребер и динамике узлов). В общем случае такие операторы являются некоммутативными.

Процедура моделирования эпидемиологической динамики ВИЧ задается следующей последовательностью шагов.

1. Сгенерировать сеть в соответствии с выбранным распределением степеней вершин и начальным числом инфицированных ρ_0.
2. Заразить узлы, имеющие общие ребра с инфицированными узлами, с вероятностью λ для каждого узла независимо.

3. Для каждого инфицированного узла применить правило развития СПИД через коэффициенты переходных вероятностей $P_{i,j}$. Узлы, достигшие стадии СПИД, из сети удаляются.
4. Применить демографическое правило (часть узлов удаляется из сети, а часть добавляется в соответствии с демографическим балансом).
5. Зафиксировать текущее состояние узлов и сгенерировать новую сеть.
6. Повторять (2)–(5) необходимое число раз.

Все шаги моделирования основаны на генерации случайных чисел, поэтому метод моделирования относится к классу Монте-Карло. Однако данная модель включает в себя набор неизвестных параметров, которые могут быть оценены только на основе экспериментальных данных и процедуры идентификации. В частности, к ним относятся вероятность заражения λ, начальное число инфицированных ρ_0 и демографический коэффициент d. Это позволяет сформулировать задачу идентификации модели комплексной сети как задачу оптимизации, на каждом шаге которой выполняются шаги алгоритма (2)–(5),

$$\sum_{t=1981}^{2007} \left[r(t) - AIDS_t \right]^2 \underset{(\lambda, d, \rho_0)}{\longrightarrow} \min . \qquad (6)$$

Здесь $AIDS_t$ – ежегодное число официально регистрируемых случаев СПИД в конкретной стране; в силу специфики $AIDS_t$ как интегральной характеристики по большой выборке и центральной предельной теоремы в (6) допустимо применять евклидову метрику для критерия качества. Общая схема идентификации, иллюстрирующая предложенный подход, приведена на рис. 6.

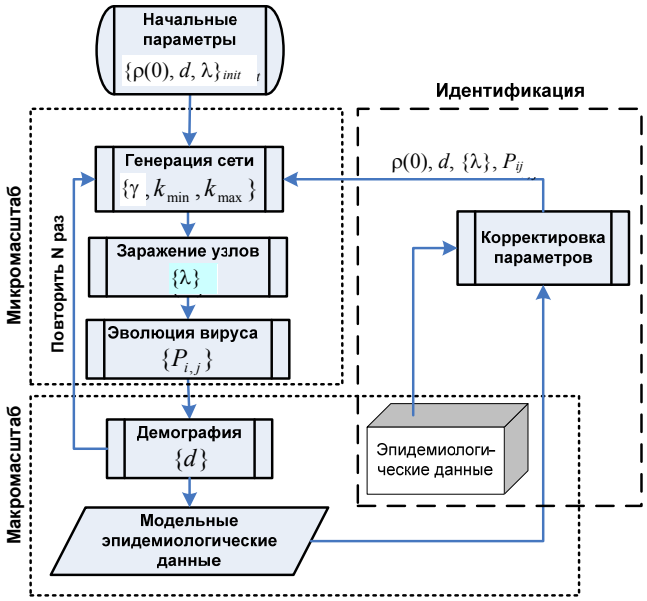
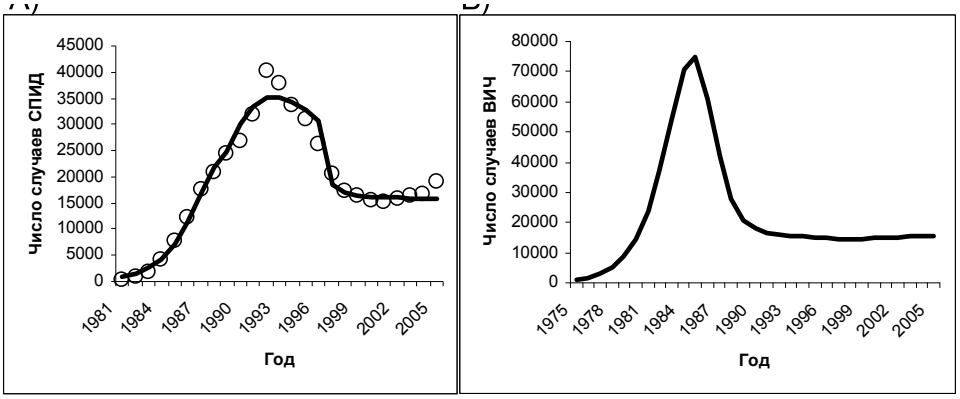


Рис. 6. Общая схема идентификации параметров модели эпидемиологической динамики ВИЧ

Особенностью схемы, приведенной на рис. 6, является необходимость проведения всего цикла моделирования для идентификации неизвестных параметров, что порождает параметрическую зависимость между моделями. При этом неизвестные параметры относятся как к микромасштабу (вероятность заражения индивидуума), так и к макромасштабу (демографический коэффициент, размер начальной популяции) модели. При этом данные наблюдений (как интегральное количество заболевших) относятся исключительно к макромасштабу. Для идентификации параметров модели были использова-

ны данные по случаям СПИД в США (доступны с 1981 г.). На рис. 7, а–б, приведены интегральные характеристики эпидемиологической ситуации – количество заболевших СПИД и количество ВИЧ-инфицированных.



а б

Рис. 7. Результат моделирования динамики ВИЧ в США:
а – результат моделирования и данные измерений пол случаям СПИД;
б – модельная кривая случаев ВИЧ (для гомосексуальной популяции)

Стоит отметить, что число случаев ВИЧ невозможно измерять прямыми методами, так как многие носители вируса не знают о своем статусе. Даже в случае диагностики восстановить по клиническим данным точное время заражения невозможно, однако моделирование позволяет не только сделать оценки реального числа случаев ВИЧ в масштабе популяции к текущему моменту, но и восстановить ход развития эпидемии в прошлом. Таким образом, данную модель сложной системы можно рассматривать как виртуальную измерительную систему.

Выводы

Основные результаты работы состоят в следующем. Предложена общая схема идентификации сложных систем в условиях параметрической связи моделей, описывающих различные диапазоны изменчивости. В рамках предложенной схемы разработано программное обеспечение идентификации моделей сложных систем на примере задач расчета полей приводного ветра, оценивания климатических спектров морского волнения и моделирования эпидемиологической динамики ВИЧ.

Литература

1. Boccara N. Modeling complex systems. – Springer, 2004. – 397 p.
2. Эйкхофф П. Основы идентификации систем управления. – М.: Мир, 1975.
3. Hasselmann K., et al. Measurements of wind-wave growth and swell decay during the Joint North Sea Wave Project (JONSWAP) // Dtsch. Hydrogr. Z. – 1973. – P. 1–95.
4. Sloot P.M.A., Ivanov S.V., Boukhanovsky A.V., Van De Vijver D.A.M.C. and Boucher C.A.B. Stochastic simulation of HIV population dynamics through complex network modeling // International Journal of Computer Mathematics. – 2008. – P. 1175–1187.

Иванов Сергей Владимирович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, м.н.с, svivanov@mail.ifmo.ru

УДК 004.021

ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ АСИНХРОННЫХ КЛЕТОЧНО-АВТОМАТНЫХ АЛГОРИТМОВ

К.В. Калгин

В статье предлагается алгоритм распараллеливания асинхронного клеточного автомата, основанный на его вероятностных свойствах. Приводятся результаты тестирования на кластере МСЦ МВС-100000.
Ключевые слова: параллельный алгоритм, методы Монте-Карло, асинхронный клеточный автомат, клеточный автомат.

Введение

В настоящее время известно много клеточно-автоматных моделей природных явлений, которые, в зависимости от назначения, различаются алфавитом состояний, структурой дискретного пространства, правилами переходов и режимами функционирования [1]. Имеются два базовых режима работы клеточных автоматов (КА) – синхронный и асинхронный. В некоторых случаях также используют смешанные синхронно-асинхронные режимы [2] – блочно-синхронный и упорядоченный.

КА моделирование реальных процессов требует больших вычислительных мощностей, поскольку для выявления в них каких-либо свойств необходимо оперировать большим количеством клеток (10^{10}–10^{12}) в течение длительного времени (10^3–10^5 итераций). Основные свойства КА – естественная мелкозернистая параллельность и локальность взаимодействий – делают эту модель привлекательной для использования при моделировании на мультикомпьютерах. Однако асинхронные КА не поддаются столь же легкому распараллеливанию, как синхронные.

Асинхронные клеточные автоматы (АКА), рассматриваемые в данной статье, применяются в основном для моделирования кинетических процессов в наносистемах. Известны результаты их применения для изучения поверхностных химических реакций на катализаторах [3], адсорбции, сублимации и диффузии атомов при эпитаксиальном росте кристаллов [4], при исследовании процессов диссоциации водорода в пористых электродах водородного энергетического элемента, а также при проектировании режимов напыления металлов для получения твердых и антикоррозийных покрытий [2].

Известны два варианта [5] асинхронного режима – пошаговый (step-driven) и повременной (time-driven). В первом случае итерация разбивается на шаги, и на каждом шаге с равной вероятностью может быть выбрана любая клетка для изменения состояния, а во втором – время следующего изменения определяется экспоненциально распределенной случайной величиной. В статье рассматриваются АКА первого типа, они требуют существенно меньших накладных расходов на организацию вычислений, в отличие от АКА второго типа, и при этом на больших размерах клеточной области эквиваленты им. Результаты распараллеливания повременного (time-driven) АКА представлены в [6].

Асинхронный клеточный автомат

Асинхронный клеточный автомат, как и синхронный, определяется набором $\langle Z_d, A, V, \varphi \rangle$, где Z^d – конечное подмножество дискретного d-мерного пространства, которое определяет *множество местоположений клеток*, A – множество возможных состояний клеток, V – *шаблон соседства* – набор векторов, определяющий соседство клетки $x \in Z^d$, $V(x) = \{x + v \mid v \in V\} \subset Z^d$, а $\varphi : A^{|V|} \to A$ – *правило перехода* в новое состояние. Под *срабатыванием* клетки далее понимается изменение ее состояния по правилу перехода.

Клеткой будем называть пару $(x,a) \in Z^d \times A$, где $a \in A$ – состояние клетки, а $x \in Z^d$ – ее *координата*. Множество клеток $\Omega = \{(x,a)\} \subset Z^d \times A$, в котором нет клеток с одинаковыми координатами, будем называть *клеточным массивом*. Далее будем отождествлять клетку и ее координаты, поскольку в клеточном массиве между ними существует взаимно однозначное соответствие.

В работе рассматривается двумерная прямоугольная область $Z^d = Z^2$ размера $N_x \times N_y$, а в качестве шаблона соседства – окрестность фон Неймана $V = \{(-1,0),(0,-1),(1,0),(0,1),(0,0)\}$.

Процесс работы АКА разбивается на итерации, которые состоят из $|Z^2| = N_x N_y$ шагов. *Шагом* будем называть единичное применение функции перехода к некоторой случайно выбранной клетке. Таким образом, в ходе одной итерации АКА реализуется некоторая случайная последовательность клеток $C = \{c_1,...,c_{|z^2|} \mid c_i \in Z^2\}$.

Вероятностные свойства АКА

При распараллеливании клеточный массив делится между процессами на непересекающиеся домены D_1, D_2, …, D_p. Множество *граничных клеток* домена D_k будем обозначать $B_k = \{c \in D_k \mid \exists j \neq k : V(c) \cap D_j \neq \varnothing\}$. Рассмотрим поведение АКА при моделировании одним процессом, но при логическом делении клеточного массива на домены.

Пусть $C = \{c_1,...,c_{|z^2|} \mid c_i \in Z^2\}$ – упорядоченное множество клеток, которое определяет порядок срабатывания во время одной из итераций (рис. 1). Пусть $in_k = C \cap (D_k \setminus B_k)$ – упорядоченное множество внутренних срабатываний, а $bound_k = C \cap B_k$ – граничных срабатываний домена D_k.

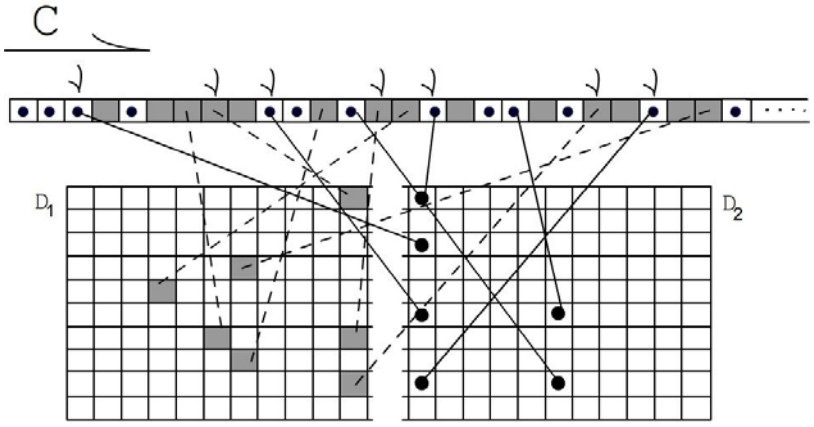


Рис. 1. Последовательность срабатываний одной итерации

Пусть $L_k = \{l_0^k, l_1^k, ..., l_r^k, ..., l_{|bound_k|}^k\}$, где $l_0^k = 0$, а l_i^k – номера шагов, на которых происходят срабатывания клеток из множества $bound_k$.

Пусть $I_r^k = \{c_i \in in_k \mid l_{r-1}^k < i < l_r^k\}$ – упорядоченное множество клеток, срабатывания которых происходят внутри домена D_k между двумя срабатываниями на его границе. Для лучшего понимания на рис. 2 графически отображены множества L_k и I_r^k.

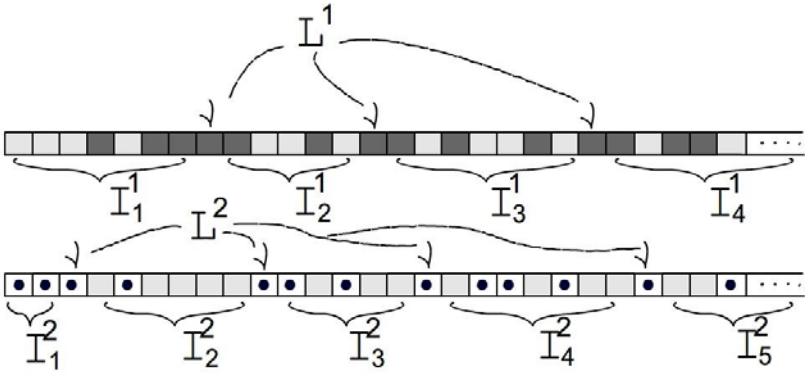


Рис. 2. Разбитый на подмножества порядок срабатывания

По C и по разбиению Z^2 на домены однозначным образом определяется множество троек

$$\Upsilon(C)=\left\{\left\langle L_k, bound_k, I_r^k\right\rangle \mid 1\le k\le p, 1\le r\le \left|bound_k\right|\right\},$$

но обратное неверно – далеко не всегда по $\Upsilon(C)$ порядок срабатывания восстанавливается однозначно.

Рассмотрим другой порядок C' с таким же разбиением, $\Upsilon(C')=\Upsilon(C)$. Тогда порядки C и C' могут отличаться лишь в упорядочивании клеток из I_r^k, относящихся к различным доменам, а упорядочивание клеток внутри и на границах, а также между границами доменов сохраняется. Следовательно, исполнения C и C' на одинаковых начальных клеточных массивах приводят к одним и тем же результатам – в этом случае эти два порядка будем называть *эквивалентными*.

Таким образом, для исполнения очередной итерации вместо порядка C достаточно сформировать некоторое разбиение $\Upsilon(C)$. Поскольку на каждой итерации порядок формируется по определенным правилам, то и $\Upsilon(C)$ должно формироваться в соответствии со своими вероятностными характеристиками элементов множеств L_k, размеров множеств I_k и $bound_k$:

$$P(l_r^k - l_{r-1}^k = t \mid t>0) = (1-\beta_k)^{t-1}\cdot\beta_k \tag{1}$$

где $\beta_k = \left|B_k\right| / \left|Z^2\right|$, а

$$P(\left|l_r^k\right| = t) = \frac{\Delta!}{t!(\Delta-t)!}\alpha_k^t(1-\alpha_k^t)^{\Delta-t} \tag{2}$$

– биномиальное распределение, где $\Delta = l_r^k - l_{r-1}^k$, а $\alpha_k = \left|D_k \setminus B_k\right| / \left|Z^2 \setminus B_k\right|$;

$$\left|bound_k\right| = \left|L_k\right| - 1 \tag{3}$$

Алгоритм формирования $\Upsilon(C)$ кратко можно описать следующим образом.

1. В соответствии с (1) генерируются величины l_r^k, тем самым определяя $\left|bound_k\right|$ и $\left|L_k\right|$. Пусть x – экспоненциально распределенная случайная величина с параметром $\lambda = -\log(1-\beta_k)$, тогда $P(\lfloor x \rfloor = t) = (1-\beta_k)^{t-1}\cdot\beta_k$, откуда получаем

   $l_r^k = l_{r-1}^k + \left[randExp(-\log(1-\beta_k))\right]$,

   где $randExp(\lambda)$ – генератор псевдослучайных чисел экспоненциального распределения.

1. Пользуясь генератором псевдослучайных чисел биномиального распределения $randBin(\Delta,\alpha)$, в соответствии с (2) получаем
   $\left|I_r^k\right| = randBin(\Delta,\alpha_k)$.
2. Размеры множеств $bound_k$ и I_r^k уже определены, а их элементы (координаты клеток) получаются с использованием генератора псевдослучайных чисел равномерного распределения.

Стоит заметить, что разбиение $\Upsilon(C)$, сформированное по такому алгоритму, определяет множество эквивалентных порядков, у каждого из которых общее число срабатываний во всем клеточном массиве является случайной величиной с математическим ожиданием $\left|Z^2\right|$.

Реализация параллельного алгоритма

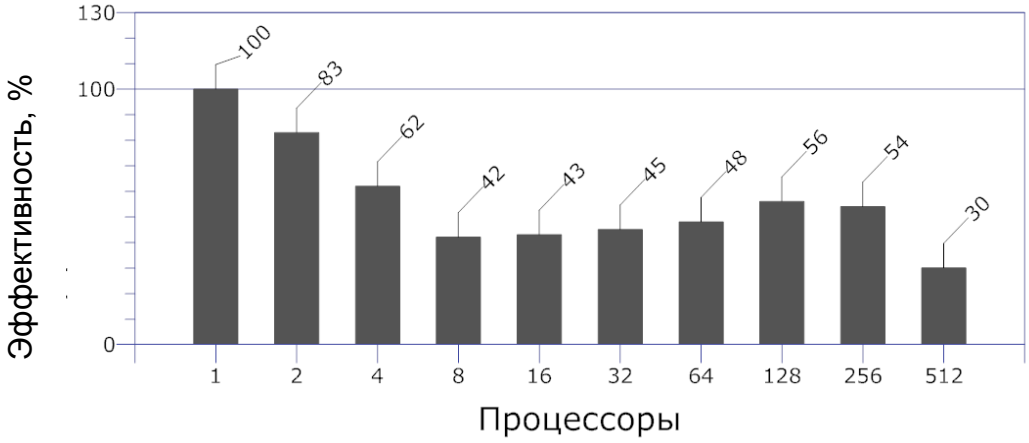
1. Построение $\Upsilon(C)$. Для каждого домена D_k соответствующим процессом строится набор $\left\langle L_k, bound_k, I_r^k \right\rangle$. Затем процессы, отвечающие соседним доменам D_k и $D_{k'}$, обмениваются множествами L_k, $bound_k$ и $L_{k'}$, $bound_{k'}$.
2. Исполнение итерации. Состояния клеток в каждом домене D_k последовательно изменяются в соответствии с полученными порядками обхода $bound_k$ и I_p^k. Если очередная клетка c лежит на границе, $c \in bound_k$, и ее номер из L_k есть l_j^k, то в соответствии с множествами $L_{k'}$ и $bound_{k'}$, полученными от соседних процессов, выполняется следующее:

для каждого соседнего домена $D_{k'}$ и для каждого срабатывания клетки $c' \in bound_{k'}$ с номером $l_i^{k'} \in L_{k'}$ такого, что $l_i^{k'} < l_j^k$, $c' \in V(c)$, причем состояние клетки c' еще не было получено после последнего изменения, получаем состояние клетки c' от процесса k';
применяем функцию перехода к клетке $c \in bound_k$;
для каждого соседнего домена $D_{k'}$ выполняем: если существует срабатывание клетки $c' \in bound_{k'}$ с номером $l_i^{k'} \in L_{k'}$, такое, что $l_j^k < l_i^{k'}$ и $c \in V(c')$, то отсылаем новое состояние граничной клетки c соответствующему процессу.

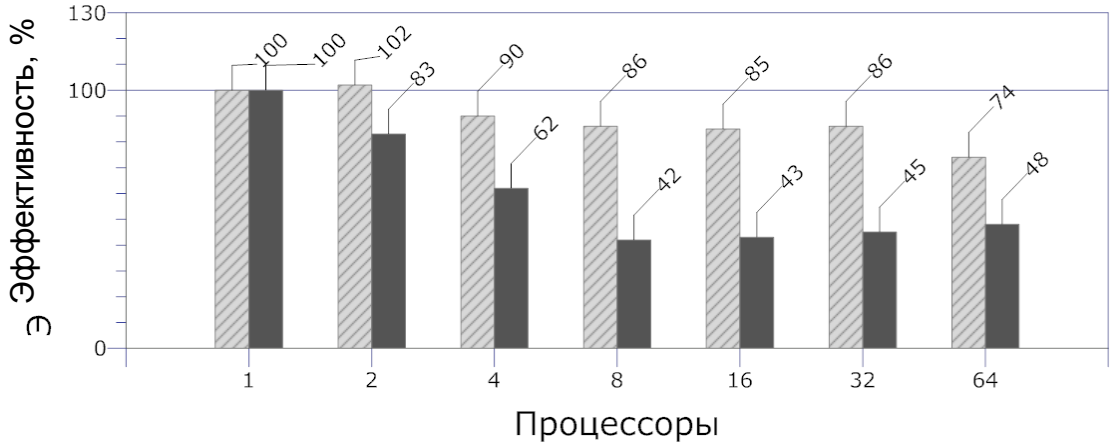
Результаты тестирования

В качестве КА модели для тестирования была выбрана модель Изинга – математическая модель намагничивания материала в статистической физике. Описание модели в терминах АКА можно найти в [6].

Тестирование проводилось при следующих условиях. Размер клеточной области $\left|Z^2\right| = N_x N_y$, где $N_x = 2^{15} = 32768$ и $N_y = 2^{14} = 16384$, а Z^2 делится на прямоугольники, по форме максимально приближенные к квадратам. Используемый кластер – МСЦ МВС-100000 (восемь процессорных ядер Xeon в одном узле). Эффективность вычисляется по формуле $e = T_1 / \left(p * T_p\right)$, где p – количество используемых процессоров, а T_k – время счета на k процессорах.



а



б

Рис. 3. а – эффективность работы при использовании всех процессоров всех узлов, б – сравнение эффективности работы при использовании одного (серые столбцы) или всех (черные столбцы) процессоров каждого из вычислительных узлов

Запуск производился в двух режимах – при использовании всех процессоров всех узлов (рис. 3, а) и при использовании лишь одного процессора из каждого узла. Сравнение эффективности работы в этих режимах (рис. 3, б) показало, что эффективность второго режима приблизительно в два раза больше. Это может быть объяснено тем, что скорость случайного доступа в память при восьми работающих процессорах на порядок меньше, чем при одном процессоре.

Заключение

В статье был предложен новый алгоритм крупноблочного распараллеливания АКА, основанный на его вероятностных свойствах. Представленные результаты тестирования реализации свидетельствуют о пригодности ее применении для моделирования больших АКА моделей реальных процессов на достаточно большом количестве процессоров.

Дальнейшая работа заключается в исследовании влияния на ускорение формы шаблона соседства V, сложности функции перехода φ и способах деления клеточной области Z^2.

Литература

1. Бандман О.Л. Клеточно-автоматные модели пространственной динамики // Сборник научных докладов «Системная информатика». – Новосибирск: Изд–во СО РАН, 2006. – Вып. 10.
2. Bandman O.L. Coarse-grained parallelization of cellular-automata simulation algorithms // Parallel Computing Technologies. – Heidelberg: LNCS Springer. – 2007. – Vol. 4671. – P. 370–384.
3. Elokhin V.I. Application of Statistical Lattice Models to the Analysis of Oscillatory and Autowave Processes on the Reaction of Carbon Monoxide Oxidation over Platinum and Palladium Surfaces // Kinetics and Catalysis. – 2003. – Vol. 44. – P. 672–700.
4. Neizvestny I.G. 3D-model of epitaxial growth on porous {111} and {100} Si surfaces // Computer Physics Communications. – 2002. – Vol. 147. – P. 272–275.
5. Schönfisch B., de Roos A. Synchronous and asynchronous updating in cellular automata // BioSystems. – 1999. – Vol. 51. – P. 123–143.
6. Overeinder B.J., Sloot P.M.A. Extensions to Time Warp Parallel Simulation for Spatial Decomposed Applications // Proceedings of the 5th annual conference of the Advanced School for Computing and Imaging (ASCI). – 1999.

Калгин Константин Викторович – Новосибирский государственный университет, студент, KalginKV@gmail.com

УДК 004.75

АРХИТЕКТУРА СЕРВИС-ОРИЕНТИРОВАННОЙ РАСПРЕДЕЛЕННОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ И ЕЕ РЕАЛИЗАЦИЯ НА ОСНОВЕ ФРЕЙМВОРКА OSGI

А.В. Гаврилов, И.И. Доровских, И.Д. Красинский

Создание распределенных вычислительных систем является актуальным для решения больших вычислительных задач. В ходе анализа существующих РВС и технологий их построения был выделен ряд недостатков. Предложена архитектура РВС, позволяющая избежать обозначенных проблем. Реализованная по ней РВС с применением фреймворка OSGi была использована для решения конкретных задач и показала свою эффективность.

Ключевые слова: распределенная вычислительная система, фреймворк, распределение ресурсов, планирование ресурсов, жизненный цикл, сервис-ориентированная система

Введение

Распределенная вычислительная система (РВС) [1] представляет собой систему, обеспечивающую инфраструктуру для запуска и выполнения задач на множестве входящих в нее узлов, работающих, например, на персональных компьютерах, объединенных сетью. В настоящее время такие системы составляют серьезную конкуренцию кластерам в силу активного распространения и роста компьютерных сетей. Для РВС адаптируют алгоритмы решения различных задач – расчета напряжений в сложных механизмах [2], расчета микро- и нанооптических элементов, обработки большеформатных изображений, различных задач математической физики. В основном это задачи, требующие сверхбольших вычислительных и временных затрат, например трехмерный рендеринг сложных деталей.

В данной работе описано видение сервис-ориентированной (СО) архитектуры РВС, которое положено в основу разработанной авторами системы. Результаты реализации показывают не только возможность такого подхода, но и некоторые его преимущества.

Архитектура РВС

Существующие подходы к реализации РВС. Существующие технологии позволяют разработчику самостоятельно реализовать РВС, например, с использованием функционального языка Erlang или библиотеки, реализующей стандарт MPI. Другой подход заключается в использовании реализованной РВС, например, Condor или X-COM. Обычно такая РВС предоставляет набор базовых интерфейсов программирования приложений (Application Programming Interface, API), позволяющий разработчику использовать инструменты РВС. В таком случае возможности разработчика ограничены предоставляемым API. Большие возможности для реализации РВС предоставляет набор инструментов Globus Toolkit: по сути, он является стандартом, предоставляющим набор инструментальных средств, а не комплект модулей. Однако стоит отметить большую сложность при организации и администрировании РВС, реализованной на основе Globus. Таким образом, при использовании существующих подходов возникают сложности адаптации РВС под конкретные условия эксплуатации.

Жизненный цикл работы РВС. Для эффективной работы РВС обычно используется много вычислительных узлов, на каждом из которых должна быть установлена соответствующая программа, играющая роль узла РВС. На одном вычислительном устройстве может быть запущено несколько узлов (рис. 1). Обычно выделяют узел (Master), который будет распределять задачи между другими узлами (Slave) [3]. Поступающие на узел Master задачи распределяются планировщиком по узлам РВС.

Рис. 1. Порядок запуска задачи на РВС

В процессе работы РВС могут происходить исключительные ситуации, чаще всего связанные с отказом узла, влияющие на выполнение вычислительных задач и функционирование РВС в целом. Это означает, что РВС должна поддерживать репликацию данных и миграцию модулей распределенных программ (РП). Эти и другие функции предопределяют роль РВС в качестве инфраструктуры для РП.

Жизненный цикл распределенной программы. Модули РП работают изолированно с точки зрения разных физических узлов (рис. 1). На одном узле могут быть одновременно запущены несколько разных модулей вычислительных программ, которые будут работать в некоторой среде, предоставляемой узлом РВС. Такую среду назовем *контейнером*. Контейнер обеспечивает жизненный цикл модулей РП, предоставляет им API РВС для получения начальных данных, коммуникаций между модулями РП, сохранения результата и позволяет мигрировать задачи в другой контейнер при отказе узла.

Особенности параллельных алгоритмов с точки зрения архитектуры системы. В литературе подробно описаны алгоритмы, предназначенные для выполнения на системах параллельных вычислений [4, 5]. Параллельные алгоритмы достаточно часто являются основой при разработке РП.

Для работы РП необходимо решить две задачи (рис. 2) – *вычислительную* и *коммуникационную* (запуск модулей РП на разных узлах, организация коммуникации между ними, передача начальных данных, сборку результата, организация топологической структуры [4]). Такая архитектура предопределена при работе параллельного алгоритма на системах параллельных вычислений, так как параллельная программа, формируя поток инструкций для процессора, определяет порядок вычислений и передачи данных между процессорами.

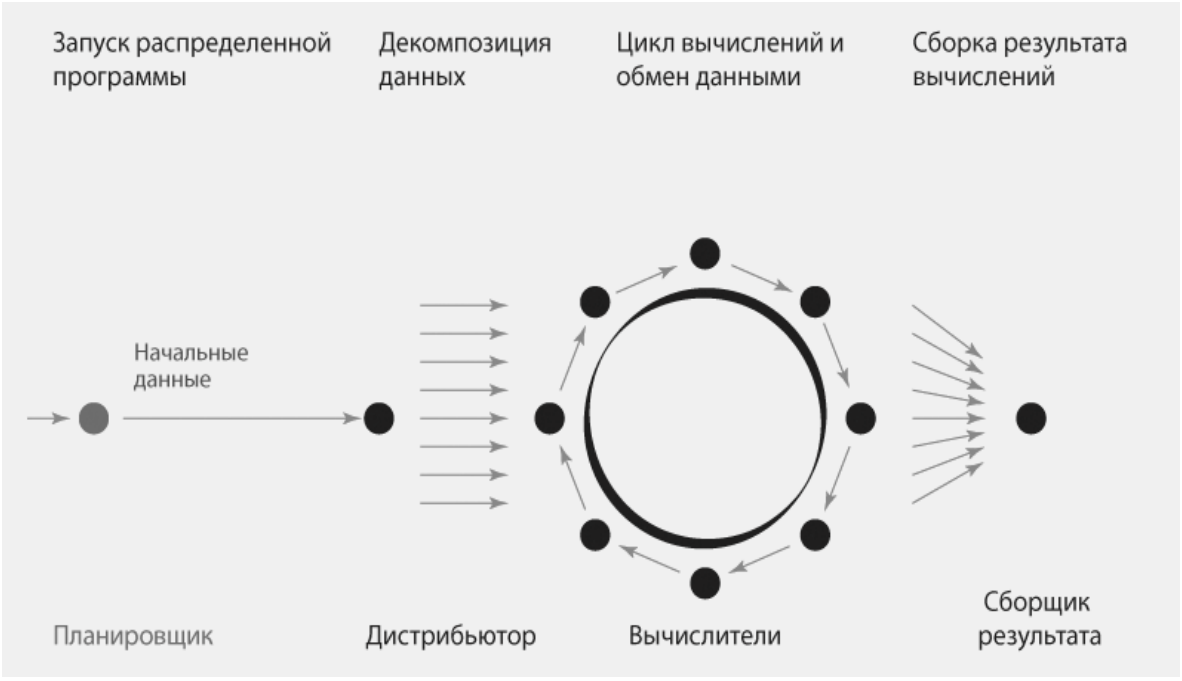


Рис. 2. Этапы работы распределенной программы

В силу того, что разные узлы РВС могут выходить из строя, задачи коммуникации представляют собой очевидную проблему, так как модули РП не знают, на каком физическом узле находятся необходимые им данные, где работают другие модули РП и как с ними обмениваться данными. Информацией о местонахождении данных и модулей распределенных программ владеет только РВС.

Решение обозначенных выше задач имеет смысл переложить с разработчиков РП на средства РВС. Предлагается подход, предусматривающий наличие в РВС *единой информационной среды* (ЕИС) – системы хранения и коммуникации данных, позволяющей отделить решение вычислительной задачи от решения коммуникационной. Все операции, связанные с хранением и передачей данных, а также коммуникацией между узлами (например, на основе событий), берет на себя ЕИС.

Роли модулей в жизненном цикле распределенной программы. На практике, кроме реализации вычислений, приходится выполнить ряд вспомогательных задач – декомпозицию начальных данных и их подготовку для вычислений, сборку результата и т.д. Эти операции можно выполнять асинхронно по отношению к операциям вычисления.

Как видно из рис. 2, структурно в РП можно выделить несколько взаимодействующих модулей, каждый из которых решает собственную задачу: модуль вычислений (Solver), модуль декомпозиции начальных данных (Distributor), модуль сборки результата (Combiner), модуль управления жизненным циклом задачи (Director). Структурно

различные модули абсолютно идентичны как для контейнера, так и для ЕИС. Это, во-первых, упрощает разработку распределенных программ за счет декомпозиции сложного параллельного алгоритма на несколько более простых частей (каждая из которых выполняет свою роль), и, во-вторых, данный подход позволяет разработчику сосредоточиться на решении вычислительной задачи, перекладывая решение коммуникационной задачи на РВС.

Архитектура РВС на основе СО-подхода. Как было сказано выше, РВС является инфраструктурой для распределенных программ, давая им богатые возможности и отвечая за стабильность их работы. Основные требования к РВС: предоставлять систему для хранения и передачи данных через протоколы коммуникации, среду для запуска и выполнения вычислительных задач, поддерживать репликацию данных, управлять жизненным циклом запущенных на ней вычислительных задач. Как видно, вышеизложенные требования достаточно независимы и могут решаться независимыми модулями – сервисами, согласованно взаимодействующими на каждом узле и во всей РВС [6].

Архитектурный слой, который занимается обеспечением жизненного цикла сервисов, принято называть *платформой*. Предлагается концепция СО-архитектуры РВС, позволяющая произвести декомпозицию частей РВС на самостоятельные архитектурные слои (наборы сервисов), что облегчает реализацию РВС и ее специализацию (путем специализации только сервисов). На рис. 3 представлены основные архитектурные слои, каждый из которых использует предоставляемые возможности предыдущего слоя. *Контейнер* обеспечивает запуск модулей вычислительных программ, жизненный цикл этих модулей, предоставление доступа к API РВС. *Единая информационная среда* реализует хранение вычислительных данных, метрик, событий, информации о модулях вычислительных программ, создание реплик данных, предоставление сервисам единого информационного пространства имен сервисов, данных, событий и метрик, доступа к данным, коммуникацию между модулями. *Платформа* обеспечивает жизненный цикл служебных и вычислительных сервисов: запуск, остановку, перезапуск, обновление, установку новых сервисов.

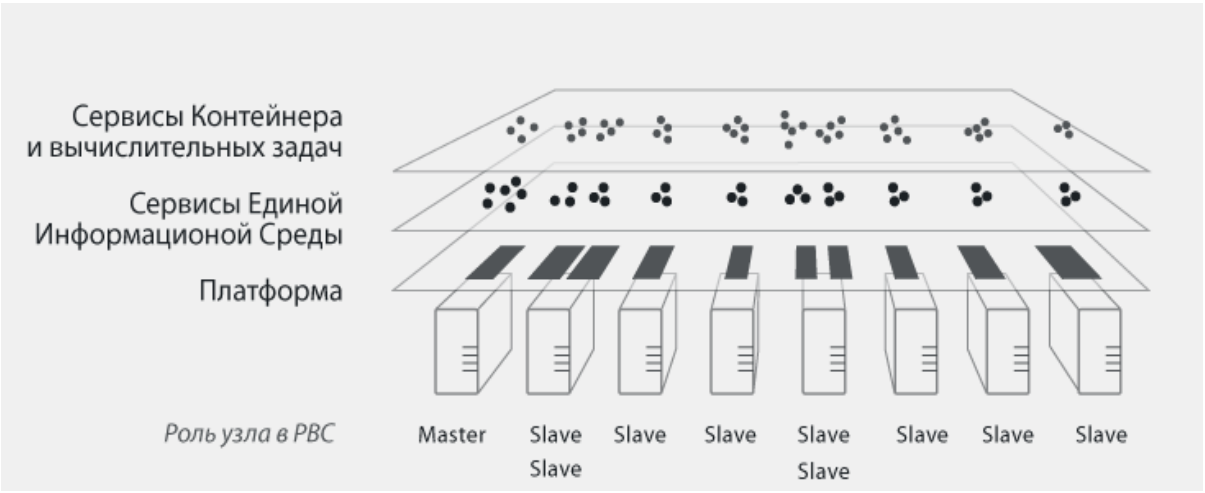


Рис. 3. Архитектура СО-РВС

Существующие подходы к созданию РВС в том или ином (обычно неявном) виде содержат данные архитектурные слои, однако предлагаемая архитектура в явном виде перераспределяет задачи, решаемые РВС, на слабо связанные и относительно независимые сервисы. Такой подход делает прозрачной архитектуру РВС и упрощает ее разработку, адаптацию, развитие.

Реализация РВС и пример ее использования

Реализация архитектуры проекта XPDC.Slavery. По рассмотренному выше принципу построения архитектуры был разработан экспериментальный фреймворк XPDC.Slavery (eXtensible Platform for Distributed Calculations), позволяющий решать вычислительные задачи в одноранговой локальной сети. Это ограничение определено выбором протоколов коммуникации (сервис Connector) и не является ограничением архитектуры РВС. XPDC.Slavery является кроссплатформенной РВС, за счет реализации на платформе Java позволяя объединять вычислительные узлы под управлением различных операционных систем.

Структурно следует выделить 3 архитектурных слоя: набор сервисов Slavery, реализующих контейнер и сервисы распределенных вычислительных алгоритмов; набор сервисов XPDC, реализующих ЕИС; сервис-ориентированная платформа на основе спецификации OSGi.

СО платформа OSGi. Платформа XPDC была разработана с применением технологии OSGi на базе ее реализации (фреймворка Equinox от Eclipse Foundation) с использованием сервисов, предоставляемых другой реализацией (Knoplerfish). Технология OSGi позволяет создавать программы из небольших модулей (bundles), пригодных для повторного использования. OSGi использует СО модель взаимодействия компонентов, а также позволяет подключать, отключать и изменять компоненты без перезагрузки системы [7].

Единая информационная среда. Уровень XPDC. В XPDC ЕИС реализована набором взаимодействующих сервисов Хранилища данных (Storage) и Коммуникаций (Connector), а также вспомогательных сервисов – сервиса Метрик (Metrics), сервиса Событий (Events), сервиса Репликаций (Replication), сервиса работы с базой данных (SQL-совместимой) и некоторых других.

Интерфейс, предоставляемый ЕИС для взаимодействия с данными, включает чтение, асинхронное чтение (предварительный запрос данных без блокировки), запись и удаление. Для работы с данными необходимо знать номер сессии, в рамках которой работает РП, и строковый идентификатор – имя объекта в Хранилище.

Сервис Коммуникаций реализует обработчики следующего набора протоколов: UDP, TCP, HTTP(S), LRMP, TRAM, RPC. Гибкость архитектуры позволяет реализовать обработчики для протоколов, не имеющих отношения к наиболее часто используемым протоколам TCP/IP и UDP, например пиринговых.

Контейнер. Уровень Slavery. Основными сервисами Контейнера являются Исполнитель вычислительных РП (Executor, представляет классы для разработки РП, механизмы запуска задач как из системы, так и из внешних клиентов, механизм сохранения контрольных точек для миграции вычислительного программы между узлами) и Часовой (Sentinel, отслеживает жизненный цикл РП и обеспечивающих миграцию сервисов). Таким образом, набор сервисов Контейнера реализует необходимые для работы РП инструменты и управляет их жизненным циклом.

Реализация распределенной версии параллельного алгоритма адаптивной обработки изображения. Рассмотрим реализованную РВС с точки зрения разработчика. В качестве алгоритма для демонстрации воспользуемся адаптивным алгоритмом устранения дефокусировки и смаза на изображении.

Алгоритм восстановления изображения [8, 9] состоит из следующих этапов:

1. преобразование цветового пространства;
2. выбор на искаженном изображении фрагментов, наиболее подходящих для формирования тестовых образцов;
3. формирование из фрагментов тестовых образцов (компьютерное ретуширование изображений на фрагментах);

4. идентификация параметров восстанавливающего фильтра по отобранным и отретушированным фрагментам;
5. обработка изображения фильтром;
6. обратное преобразование цветового пространства.

В описанном алгоритме могут быть реализованы параллельно (при этом каждой ветке алгоритма передается своя часть исходного изображения [10]) шаги 1, 2, 5 и 6. Исходя из этапов алгоритма, пользователь создает шесть наследников класса Solver. В каждом из них будут осуществляться действия:

1. загрузка данных;
2. вызов метода-обработчика из библиотеки;
3. сохранение обработанных данных.

На рис. 4 приведен фрагмент кода РП, отвечающий за исполнение одного из этапов.

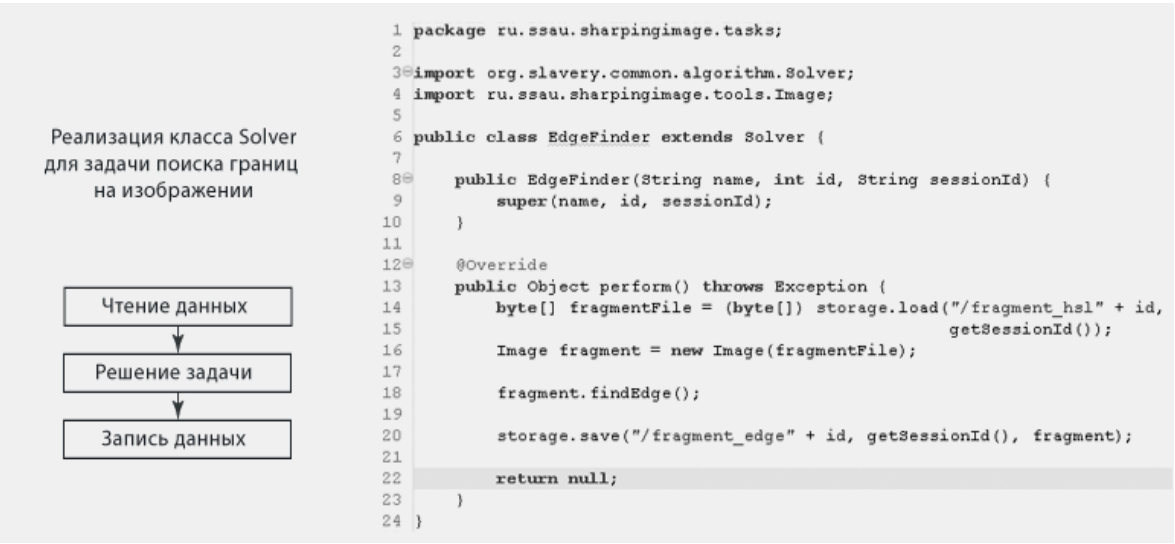


Рис. 4. Вычислительный модуль распределенной программы

Управление последовательностью шагов реализовано в наследнике класса Director, который запускает на предоставленных узлах экземпляры классов-наследников Solver. Процессы декомпозиции и сборки результата из класса Director удобно вынести в наследники классов Distributor и Combiner, что позволит запускать их асинхронно на разных узлах РВС.

Как видно из рис. 4, ЕИС позволяет разработчику не заботиться о физическом местонахождении данных и предоставляет быстрый и удобный доступ к ним. Таким образом, использование платформы XPDC.Slavery снимает с разработчика распределенных алгоритмов необходимость реализации инфраструктурных задач РВС.

Заключение

Исходя из общих предположений, не уменьшающих универсальность распределенной вычислительной системы, предложена архитектура РВС. Выделение нескольких архитектурных слоев разделяет сервисы по сферам их ответственности и облегчает их замену. Наличие единой информационной среды облегчает работу с данными и обеспечение их сохранности.

С использованием разработанной архитектуры была реализована РВС. Опыт создания распределенных программ для данной системы говорит о ее соответствии поставленным требованиям универсальности, гибкости и простоты.

Универсальность разработанной РВС дает возможность использовать ее не только в исследовательских, но и в производственных целях.

Работа выполнена при поддержке российско-американской программы BRHE (RUXO-014-SA-06), РФФИ (06-08-01024, 07-02-12134, 07-07-00210) и Программы фундаментальных исследований РАН (П-15.4.6).

Литература

1. Хорошевский В.Г. Архитектура вычислительных систем. – М.: МГТУ им. Н.Э. Баумана, 2005. – 410 с.
2. Soblewski M. Soblewski Sorcer: Computing and Metacomputing Intergrid // Texas Tech University. – 2007. – P. 74–85.
3. Седельников М.С. Алгоритм распределения набора задач с переменными параметрами по машинам вычислительной системы // Автометрия. – 2006. – № 1.– С. 68–75.
4. Голуб Дж., Ван Лоун Ч. Матричные вычисления. – М.: Мир, 1999. – 512 с.
5. Ортега, Дж. Введение в параллельные и векторные методы решения линейных систем. – М.: Мир, 1991.– 342 с.
6. Богданов А.Б., Станкова Е.Н., Мареев В.В. Сервис-ориентированная архитектура: новые возможности в свете развития grid технологий. – Институт высокопроизводительных вычислений и интегрированных систем, 2005. – 32 с.
7. The Dynamic Module System for Java. – Режим доступа http://www.osgi.org/Specifications/, свободный
8. Фурсов В.А. Идентификация моделей систем формирования изображений по малому числу наблюдений. – С.: СГАУ, 1998. –13 с.
9. Зимин Д.И., Фурсов В.А. Технология определения восстанавливающего фильтра и обработки цветных изображений // Компьютерная оптика. – 2005. – Вып. 27. – С. 170–176.
10. Доровских И.И., Попов С.Б., Фурсов В.А. Реализация типовых алгоритмов обработки изображений на мультиядерных процессорах // Материалы 7-ой Международной конф.-семинара «Высокопроизводительные параллельные вычисления на кластерных системах» . – 2007. – С. 306–313.

Гаврилов Андрей Вадимович – Институт систем обработки изображений РАН, м.н.с., Andrey.gavrilov.samara@gmail.com

Доровских Ирина Игоревна – Самарский государственный аэрокосмический университет им. академика С.П. Королева (СГАУ), аспирант, irinsmile@yandex.ru

Красинский Илья Дмитриевич – Самарский государственный аэрокосмический университет им. академика С.П. Королева (СГАУ), аспирант, IDKras@yandex.ru

УДК 519.6

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ИСПОЛЬЗОВАНИЯ ТАБУ-МАШИНЫ И НЕЙРОННЫХ СЕТЕЙ ХОПФИЛДА ДЛЯ РЕШЕНИЯ ЗАДАЧ ДИСКРЕТНОЙ ОПТИМИЗАЦИИ ИЗ ОБЛАСТИ РАСПРЕДЕЛЕННЫХ БАЗ ДАННЫХ

Э.А. Бабкин, М.Е. Карпунина

В статье исследуются преимущества нейросетевого подхода к решению задач дискретной оптимизации с использованием табу-поиска. Выявлены области значений параметров табу-машины, соответствующие более эффективному нахождению качественно лучших решений по сравнению с сетями Хопфилда.
Ключевые слова: табу-поиск, нейронные сети, генетические алгоритмы, задачи дискретной оптимизации, распределенные базы данных

Введение

Проблемы принятия оптимальных решений возникают в различных областях науки и техники, оказывая решающее влияние на развитие систем автоматизации проектирования, управления, научных исследований. Методы теории оптимизации активно используются для построения и реконфигурации сложных распределенных технических систем с множеством параметров и ограничений, таких как распределенные измерительные комплексы и базы данных (РБД).

Мы предлагаем новый подход к решению задач дискретной оптимизации, возникающих в ходе оптимизации логической структуры (ЛС) РБД. Наш метод основан на использовании табу-машины (ТМ) [1], структура которой аналогична структуре нейронной сети (НС) Хопфилда (СХ) [2].

В статье описываются детали работы ТМ при ее использовании для решения задачи синтеза оптимальной ЛС (ОЛС) РБД, модификации алгоритма табу-поиска, связанные с особенностями решаемой задачи, полученные результаты, их оценки и прогнозы.

Предметная область и постановка задачи оптимизации

РБД оказывают большое влияние на архитектурные и операционные свойства распределенных систем сбора/обработки данных, так как облегчают распараллеливание обработки запроса, повышают доступность данных, отказоустойчивость. Одной из главных проблем разработки РБД является синтез ОЛС. Ранее с использованием формальной модели РБД [5] в работе [4] мы предложили решение упрощенного варианта этой задачи с использованием СХ. Однако в общей постановке задача сложнее, и ее решение состоит из двух этапов.

1. Разбиение множества групп данных на типы логических записей (далее – *декомпозиция данных*) с использованием ограничений:
   a. на число групп в составе логической записи – $\sum_{i=1}^{I} x_{it} \le F_t, \forall t = \overline{1, t_0}$;
   b. на однократность включения групп в записи – $\sum_{t=1}^{t_0} x_{it} = 1, \forall i = \overline{1, I}$;
   c. на требуемый уровень информационной безопасности системы – $x_{it} x_{i't} = 0$ для заданных d_i и $d_{i'}$.

Кроме этих ограничений, синтез типов логических записей должен проходить с максимальным учетом семантической смежности групп данных.

2. Безызбыточное размещение типов логических записей по узлам вычислительной сети (ВС) (далее – *размещение записей*) при ограничениях:

   d. на безызбыточность размещения – $\sum_{r=1}^{R_0} y_{tr} = 1, \forall t = \overline{1,T}$;

   e. на длину логической записи на узлах ВС – $\sum_{i=1}^{I} x_{it} y_{tr} \rho_i \psi_0 \le \theta_{tr}$, $\forall t = \overline{1,T}, \forall r = \overline{1,R_0}$;

   f. на общее число записей на сервере *r*-го узла ВС – $\sum_{t=1}^{t_0} y_{tr} \le h_r, \forall r = \overline{1,R_0}$;

   g. на объем доступной внешней памяти серверов сети для хранения локальных баз данных – $\sum_{t=1}^{t_0}\sum_{i=1}^{I} \psi_0 \rho_i \pi_i x_{it} y_{tr} \le \eta_r^{ВЗУ}, \forall r = \overline{1,R_0}$;

   h. на суммарное время обслуживания оперативных запросов на серверах – $\sum_{r=1}^{R_0}\sum_{t=1}^{t_0} z_{pr}^t \cdot (t_r^n + t_r) \le T_p, \forall p = \overline{1,P_0}$.

Оптимизация ЛС РБД проводится по критерию минимума общего времени последовательной обработки множества запросов пользователей:

$$\min_{\{x_{it},y_{tr}\}} \sum_{k=1}^{K_0}\sum_{p=1}^{P_0} \xi_{kp}^3 \cdot \varphi_{kp}^3 \cdot \{\sum_{r_1=1}^{R_0} \nu_{kr_1} \cdot [\sum_{r_2=1}^{R_0} z_{pr_2} \cdot (t^{раз} + t_{r_1r_2}^{сер} + t_{r_1r_2}^{пер} \cdot (1 + \sum_{t=1}^{t_0} z_{pr_2}^t)) + t^{сб}] + \sum_{r_2=1}^{R_0}\sum_{t=1}^{t_0} z_{pr_2}^t \cdot (t_{r_2}^n + t_{r_2})\}.$$

Использование табу-машины для решения задачи синтеза оптимальной логической структуры РБД

Для оценки возможностей и преимуществ табу-поиска над ранее предложенными авторами решениями на основе СХ [4, 6] или комбинации НС и генетических алгоритмов (ГА) [7] рассмотрим основанные на нем алгоритмы для декомпозиции данных и размещения записей и их особенности по сравнению с [1].

Для **декомпозиции данных** используется ТМ из одного слоя нейронов, соединенных полными двунаправленными связями. Число нейронов в слое равняется n^2, где $n = I$ – число групп данных на входе. Функция энергии сети для декомпозиции данных после формулировки ограничений в терминах ТМ имеет вид

$$E = -\frac{1}{2} \cdot \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{x=1}^{n}\sum_{y=1}^{n} w_{xi,yj} \cdot OUT_{xi} \cdot OUT_{yj} + \sum_{i=1}^{n}\sum_{x=1}^{n} \theta_{xi} \cdot OUT_{xi} + \frac{C}{2} \cdot n^2,$$

где $w_{xi,yj} = -A \cdot \delta_{xy} \cdot (1 - \delta_{ij}) + B \cdot \delta_{ij} \cdot (1 - \delta_{xy}) \cdot (2 \cdot a_{xy}^{\varepsilon} - 1) - C - E \cdot \delta_{ij} \cdot (incomp\_gr_{xy} + incomp\_gr_{yx})$ – веса нейронов, а $\theta_{xi} = (\frac{B}{2} \cdot \sum_{y \ne x} (a_{xy}^{\varepsilon})^2 - C \cdot n + \frac{D}{2 \cdot F_i})$ – пороги нейронов.

Улучшение качества решения. В «критерий желания» ТМ, наряду с проверкой уменьшения энергии нового состояния, введена проверка уменьшения значения функции, построенной так, что ее минимальное значение, равное нулю, достигается при выполнении всех ограничений этапа, а семантическая связность групп данных полностью учтена.

Увеличение производительности ТМ. Предложен грамотный пересчет изменений энергии $\Delta E(S_i)$ сети при изменении состояния нейрона *i*. Здесь требуется пересчет только одного выхода сети. Выходы НС трансформируются в матрицу размерности $n \times n$, показывающую распределение групп данных по типам записей. Значит, изменение состояния одного нейрона – изменение значения одного элемента матрицы. Идея оптимизации заключается в использовании предыдущего значения $\Delta E(S_i)$ для подсчета

последующего. Выделив функцию вычисления добавка $Diff(S_i)$, вносимого в $\Delta E(S_i)$ изменением состояния нейрона *i*, имеем изменение энергии

$$\Delta E(S_i)_{next} = \Delta E(S_i)_{previous} - Diff(S_{i\ before\ change}) + Diff(S_{i\ after\ change}),$$

где $Diff(S_{i\ before\ change})$ − значение $Diff(S_i)$, если нейрон *i* находится в прежнем состоянии, $Diff(S_{i\ after\ change})$ − значение $Diff(S_i)$, если нейрон *i* − в новом состоянии. Новый вариант пересчета содержит меньшее число операций, работает быстрее.

Структура ТМ для **размещения записей** аналогична таковой для декомпозиции данных, но число нейронов в слое равно $T \cdot R_0$, где T − число синтезированных типов записей, R_0 − число узлов ВС. После формулировки ограничений в терминах ТМ получена функция энергии ТМ для размещения записей:

$$E = -\frac{1}{2} \cdot \sum_{r_1=1}^{R_0} \sum_{r_2=1}^{R_0} \sum_{t_1=1}^{T} \sum_{t_2=1}^{T} w_{t_1 r_1, t_2 r_2} \cdot OUT_{t_1 r_1} \cdot OUT_{t_2 r_2} + \sum_{r_1=1}^{R_0} \sum_{t_1=1}^{T} \theta_{t_1 r_1} \cdot OUT_{t_1 r_1} + B \cdot T^2,$$

где $w_{t_1 r_1, t_2 r_2} = -A \cdot \delta_{t_1 t_2} \cdot \left(1 - \delta_{r_1 r_2}\right) - B$ − веса нейронов, а $\theta_{t_1 r_1} = -B \cdot T + \frac{C \cdot \psi_0}{2 \cdot \theta_{t_1 r_1}} \cdot \sum_{i=1}^{n} (x_{it_1} \cdot \rho_i) +$

$+ \frac{D}{2 \cdot h_{r_1}} + \frac{E \cdot \psi_0}{2 \cdot \eta_{r_1}^{B3Y}} \cdot \sum_{i=1}^{n} (\rho_i \cdot \pi_i \cdot x_{it_1}) + \frac{F \cdot (t_{r_1}^n + t_{r_1})}{2} \cdot \sum_{p=1}^{P_0} \left(\frac{SN_{pt_1}}{T_p} \right)$ − пороги нейронов; здесь $n = I$ − число групп данных, $z_{pr_1}^{t_1} = OUT_{t_1 r_1} \cdot SN_{pt_1}$ и SN_{pt_1} вводится как нормализованная сумма, т.е. $SN_{pt_1} = 1$, если $\sum_{i=1}^{n} \omega^3 \cdot x_{it_1} \geq 1$, и $SN_{pt_1} = 0$, если $\sum_{i=1}^{n} \omega_{pi}^3 \cdot x_{it_1} = 0$, где ω_{pi}^3 − матрица $(P_0 \times n)$ использования групп данных при выполнении запросов [5].

Улучшение качества решения. В «критерий желания» ТМ, наряду с проверкой уменьшения энергии нового состояния, введена проверка уменьшения значения функции, построенной так, что ее значение складывается из суммы значений целевой функции на текущей конфигурации решения и некоторой искусственной функции, минимальное значение которой, равное нулю, достигается при выполнении всех ограничений этапа, причем значение 1-го слагаемого всегда на порядок меньше значения 2-го. Следовательно, в процессе оптимизации ТМ прежде всего стремится к состоянию, удовлетворяющему ограничениям этапа. Затем среди множества таких состояний производится поиск квазиоптимального состояния, доставляющего минимум функции энергии. Благодаря обработке удаленных состояний (ОУС) в ТМ множество уже найденных состояний, удовлетворяющих ограничениям, будет не оставаться стационарным, а расширяться в процессе решения, давая возможность еще большего уменьшения целевой функции задачи. Степень этой «расширяемости» определяется значениями параметров ТМ и прямо пропорциональна времени работы алгоритма.

Увеличение производительности ТМ. Идея оптимизации ТМ здесь аналогична подходу, применяемому при декомпозиции данных.

Результаты экспериментов и их анализ

ТМ позволяет избежать стабилизации в локальных минимумах энергии, что увеличивает качество решения по сравнению с обычными СХ. Рассмотрим влияние параметров $< l, C, \beta >$ ТМ на качество решения и время его нахождения.

Качество решения при декомпозиции данных оценим степенью учета синтезированными типами записей семантической смежности групп данных, их составляющих:

$S_q = \sum_i \sum_x \sum_{y \neq x} OUT_{xi} \cdot (OUT_{yi} - a^{\hat{}}_{xy})^2$, где $A^{\hat{}} = \{a^{\hat{}}_{xy}\}$, $x, y = \overline{1,n}$ − матрица семантической смежности групп данных. Семантическая смежность полностью учтена при $S_q = 0$. Чем ближе S_q к 0, тем выше качество полученного решения.

Декомпозиция проводилась для числа групп данных $n = 10;\ 20;\ 40$ сначала с помощью НС-ГА-алгоритма [4], а затем − с ТМ. Рис. 1 показывает степень близости полученных решений к решению с $S_q = 0$, а рис. 2 − зависимость времени решения задачи от ее размерности для ТМ до и после улучшения производительности и для СХ. Плата за улучшение качества решения ТМ − большее время работы алгоритма по сравнению с СХ-подходом. Но после оптимизации ТМ производительность повысилась настолько, что при увеличении размерности задачи результат был получен быстрее, чем при использовании НС-ГА-алгоритма.

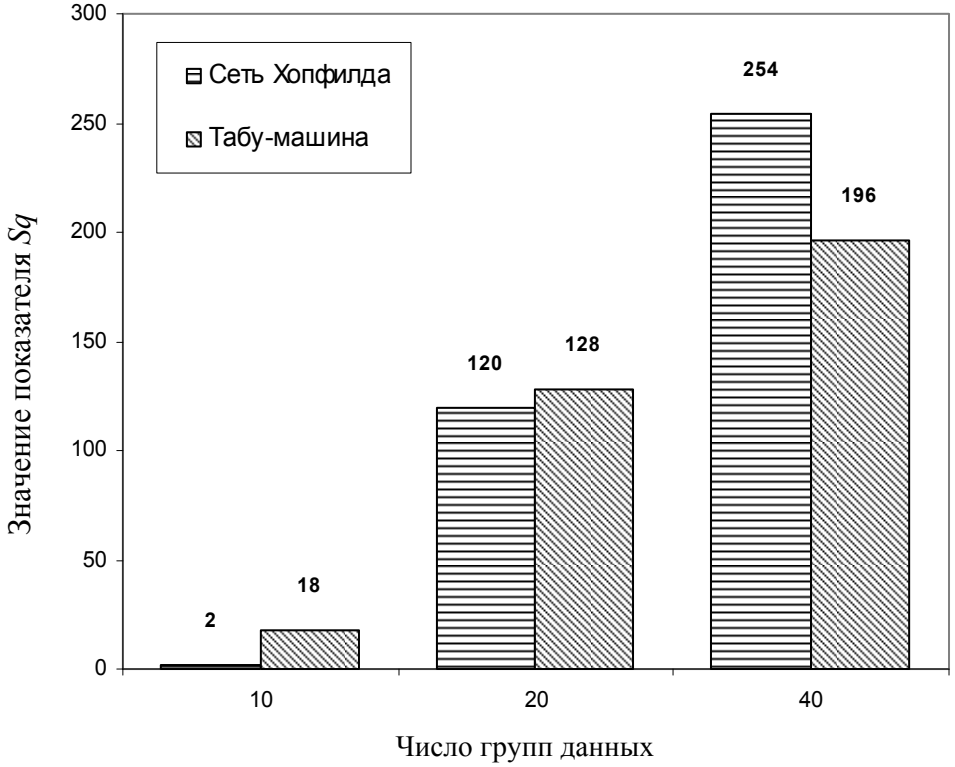


Рис. 1. Степень близости решений, полученных с помощью табу-машины и с помощью сети Хопфилда, к решению, полностью учитывающему семантическую смежность групп данных: значение показателя S_q

Безызбыточное размещение записей проводилось также для $n = 10;\ 20;\ 40$ сначала с помощью НС-ГА-алгоритма, а затем с помощью ТМ.

Определяющим для ТМ является набор параметров $< l, C, \beta >$: l − табу-размер, C − число вызовов ОУС, $\beta > 0$ − заданный коэффициент ТМ. Для исследования их влияния на качество решения и скорость его получения было положено: (1) $C = 10 \cdot q,\ q = \overline{1,5}$; (2) $l = \left[\frac{5 \cdot k}{100} \cdot T \cdot R_0 \right],\ k = \overline{0,20}$; (3) $\beta = \frac{l}{T \cdot R_0} + 0{,}1$ − выражение, рассматриваемое как некое минимальное значение β при заданном l. Большие значения β обеспечивают большую широту поиска, увеличивая время решения задачи. Поэтому значение β было зафиксировано у его нижней границы при заданном l, обеспечив некоторое ускорение при проведении экспериментов. Для каждой из конфигураций зада-

чи было получено $5 \cdot 21 = 105$ пробных решений при различных значениях $< l, C, \beta >$. Основной упор был сделан на значения l и C.

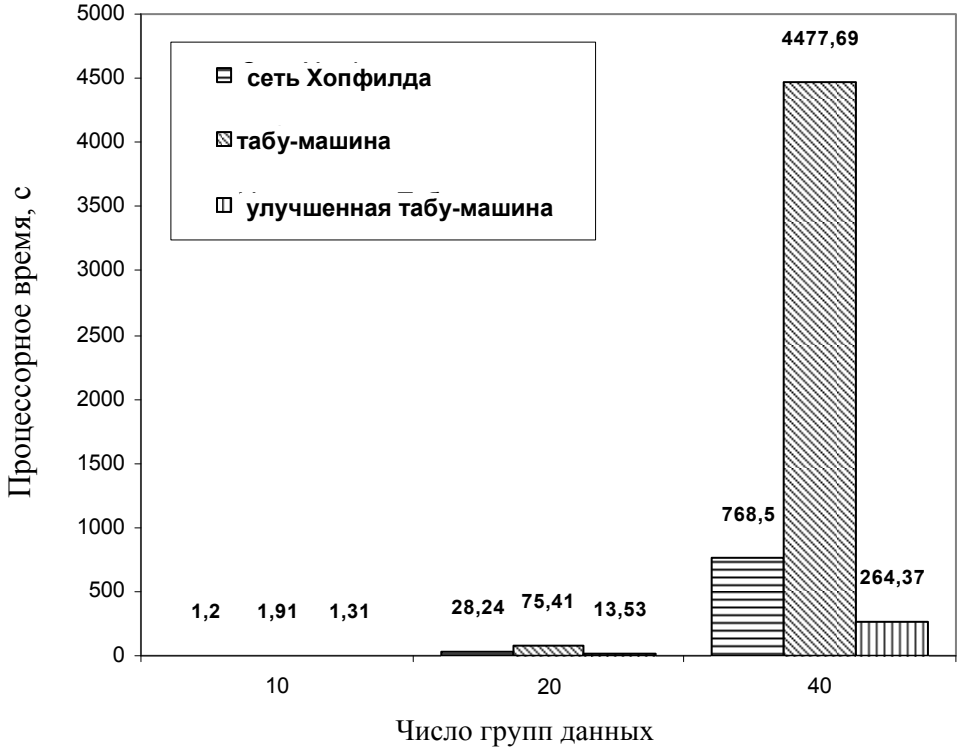
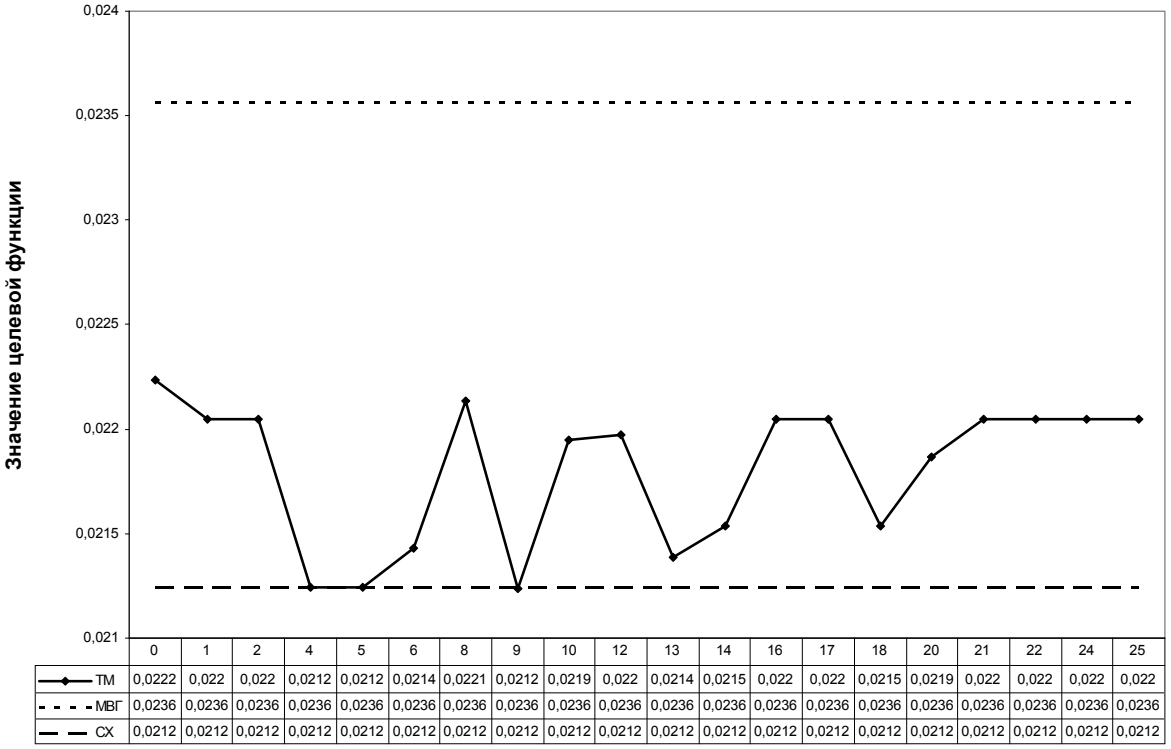


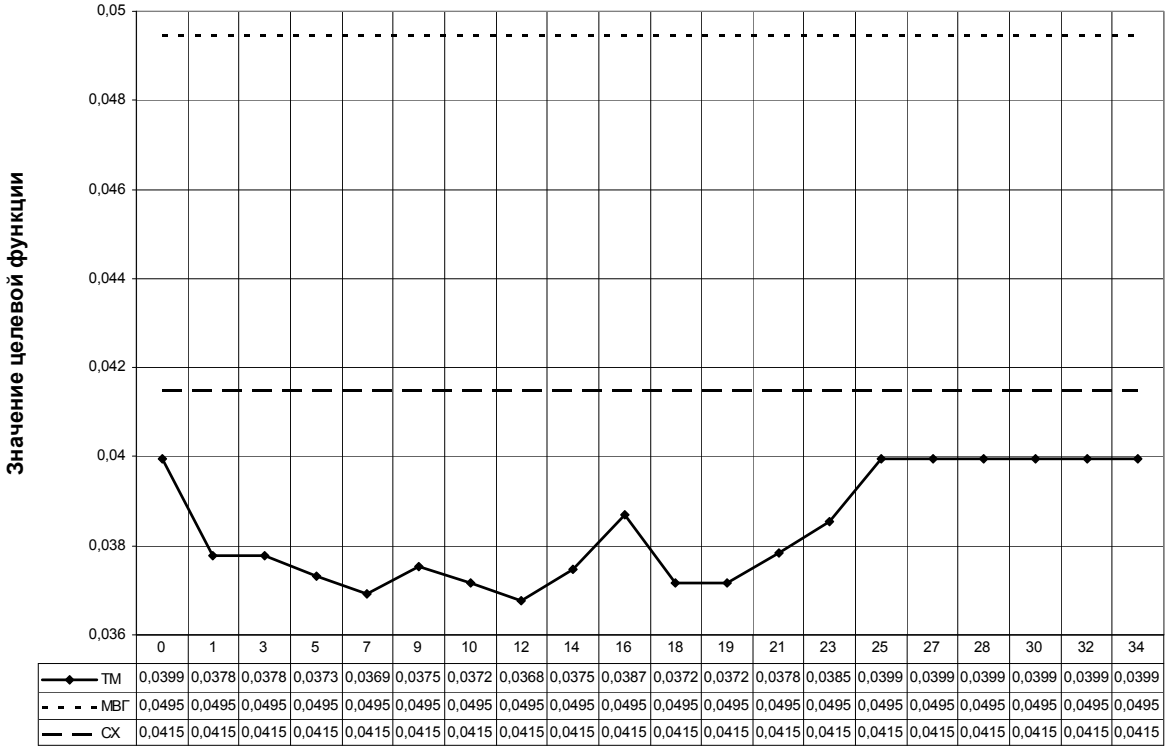
Рис. 2. Процессорное время, потраченное табу-машиной до и после улучшения производительности и сетью Хопфилда на декомпозицию данных



	0	1	2	4	5	6	8	9	10	12	13	14	16	17	18	20	21	22	24	25
ТМ	0,0222	0,022	0,022	0,0212	0,0212	0,0214	0,0221	0,0212	0,0219	0,022	0,0214	0,0215	0,022	0,022	0,0215	0,0219	0,022	0,022	0,022	0,022
МВГ	0,0236	0,0236	0,0236	0,0236	0,0236	0,0236	0,0236	0,0236	0,0236	0,0236	0,0236	0,0236	0,0236	0,0236	0,0236	0,0236	0,0236	0,0236	0,0236	0,0236
СХ	0,0212	0,0212	0,0212	0,0212	0,0212	0,0212	0,0212	0,0212	0,0212	0,0212	0,0212	0,0212	0,0212	0,0212	0,0212	0,0212	0,0212	0,0212	0,0212	0,0212

Значение коэффициента *l* (при *C*=10, β=1/*n*+0,1)

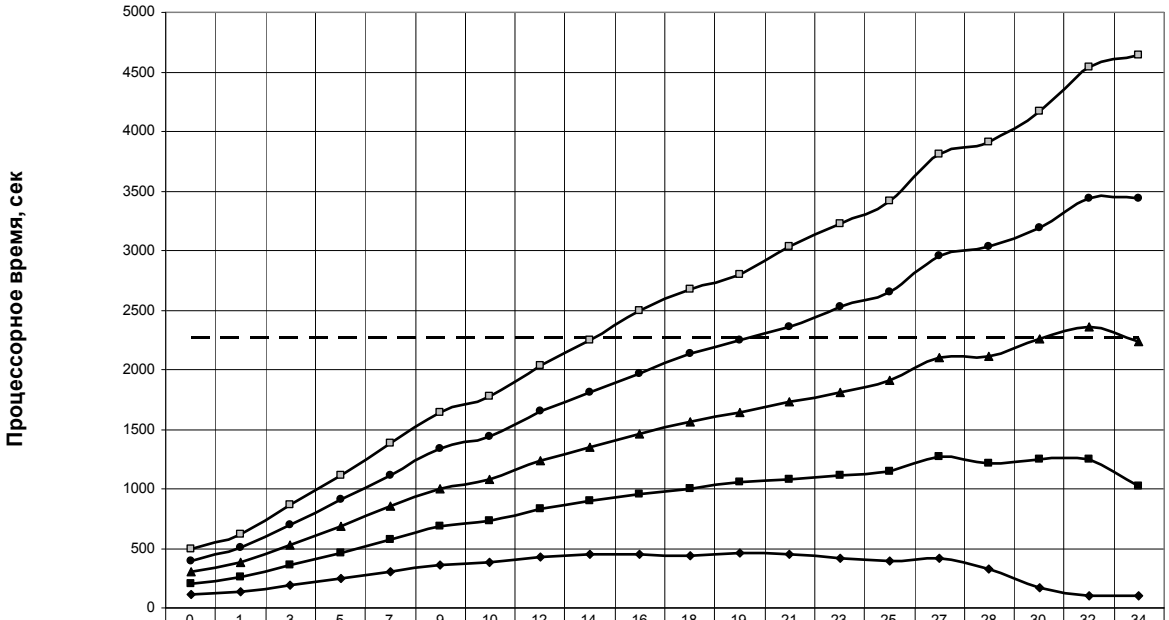
Рис. 3. Качество размещения записей по узлам. Конфигурация задачи: *n* = 20; *C* = 10



	0	1	3	5	7	9	10	12	14	16	18	19	21	23	25	27	28	30	32	34
ТМ	0,0399	0,0378	0,0378	0,0373	0,0369	0,0375	0,0372	0,0368	0,0375	0,0387	0,0372	0,0372	0,0378	0,0385	0,0399	0,0399	0,0399	0,0399	0,0399	0,0399
МВГ	0,0495	0,0495	0,0495	0,0495	0,0495	0,0495	0,0495	0,0495	0,0495	0,0495	0,0495	0,0495	0,0495	0,0495	0,0495	0,0495	0,0495	0,0495	0,0495	0,0495
СХ	0,0415	0,0415	0,0415	0,0415	0,0415	0,0415	0,0415	0,0415	0,0415	0,0415	0,0415	0,0415	0,0415	0,0415	0,0415	0,0415	0,0415	0,0415	0,0415	0,0415

Значение коэффициента *l* (при *C*=10, β=1/*n*+0,1)

Рис. 4. Качество размещения записей по узлам. Конфигурация задачи: *n* = 40; *C* = 10



	0	1	3	5	7	9	10	12	14	16	18	19	21	23	25	27	28	30	32	34
ТМ: C=10	108,24	133,42	189,27	243,53	299,58	355,14	384,05	431,92	445,2	451,25	443,36	464,38	443,97	419,44	388	420,58	327,5	164,5	103	102,92
ТМ: C=20	205	255,13	356,64	463,61	573,09	684,33	733,3	830,52	897,86	952,58	998,84	1058,7	1080,2	1112,6	1143,3	1267,5	1212,5	1250,3	1250,4	1017,7
ТМ: C=30	298,41	379,67	527,59	685,78	849,42	1002,5	1081,3	1234,5	1352,5	1461,6	1563,2	1639	1726,7	1809,8	1913,8	2097,3	2111,3	2254,1	2362,1	2231,8
ТМ: C=40	395,5	502,98	701,08	908,55	1113,8	1342,5	1434,5	1651,9	1807,3	1971,6	2139,6	2248,5	2354	2522,9	2652,2	2952,4	3034,4	3190	3436,5	3442,8
ТМ: C=50	489,42	623,58	866,77	1114	1380,2	1642	1771,5	2032,1	2247,5	2490,6	2671,1	2801,2	3031,1	3228,6	3421,3	3805,5	3907,4	4169,6	4534,8	4641,8
СХ	2275	2275	2275	2275	2275	2275	2275	2275	2275	2275	2275	2275	2275	2275	2275	2275	2275	2275	2275	2275

Значение коэффициента *l* (при *C*=10, β=1/*n*+0,1)

Рис. 2. Процессорное время, потраченное на размещение записей по узлам

Экспериментально получены зависимости качества решения и времени его нахождения от значения параметра l. Наиболее значимые результаты показаны на рис. 3, 4 (качества решений, полученных для аналогичных конфигураций задач с помощью метода ветвей и границ (МВГ) [3]) и на рис. 5 (процессорное время, затраченное на размещение записей). В случае оценки как качества, так и временных характеристик ТМ наилучшие значения $< l, C, \beta >$ – те, при которых график, соответствующий ТМ, лежит ниже или на уровне с графиком НС-ГА-алгоритма.

Таким образом, с увеличением размерности задачи решения, полученные с помощью ТМ, качественно превосходят СХ-решения. Поэтому прогноз об улучшении качества ТМ-решения по сравнению с альтернативным методом СХ при увеличении размерности задачи особенно ценен, так как специфика задачи синтеза ОЛС РБД делает ее актуальной именно при большом числе групп данных.

Заключение

К преимуществам ТМ можно отнести независимость решения от относительных весов A, B, C, D, E, F термов функции энергии сети. Эксперименты показали, что при больших значениях C удается получить больше качественных решений при различных значениях l. Для конфигураций задачи с 10 группами данных ТМ не было найдено качественно лучших решений, чем полученные с помощью НС-ГА-алгоритма. Для задач с 20 и 40 группами данных ситуация обстоит иначе. Наихудшим с точки зрения качества решения является наименьшее значение, $C = 10$. По рис. 3, 4 видно, что для 20 групп данных наилучшее решение со значением целевой функции 0,021236 получено при $l = 9$, а для 40 групп данных – со значением 0,036781 при $l = 12$. В проведенных экспериментах ТМ в 1-ом случае состояла из $T \cdot R_0 = 9 \cdot 3 = 27$ нейронов, а во 2-ом – из $T \cdot R_0 = 12 \cdot 3 = 36$ нейронов. Таким образом, наилучшим значением параметра l является величина, равная количеству типов записей, синтезированных при декомпозиции данных.

Исследования ТМ как альтернативного метода решения задачи безызбыточного размещения типов записей по узлам ВС выявили возможности улучшения качества решения и увеличения производительности алгоритма.

Литература

1. Mingne S., Nemati H.R. Mingne Sun Tabu Machine: A New Neural Network Solution Approach for Combinatorial Optimization Problems // Journal Of Heuristics. – 2003.
2. Джонс М.Т. Программирование искусственного интеллекта в приложениях / Пер. с англ. Осипов А.И. – М.: ДМК Пресс, 2004. – 312 с.
3. Кульба В.В. [и др.] Теоретические основы проектирования оптимальных структур распределенных баз данных / Серия «Информатизация России на пороге XXI века».– М.: СИНТЕГ, 1999. – 660 с.
4. Бабкин Э.А., Карпунина М.Е. Об одном методе синтеза структуры распределенной базы данных, основанном на использовании искусственных нейронных сетей Хопфилда // Известия Академии инженерных наук им. А.М. Прохорова. Прикладная математика и механика. – Н.Новгород: НГТУ, 2005. – Т 12. – С. 37–46.
5. Babkin E., Petrova M. Application of genetic algorithms to increase an overall performance of artificial neural networks in the domain of synthesis DDBs optimal structures // Information Technology and Control. – 2006. – Vol. 35. – № 3A. – P. 285–294.
6. Babkin E., Petrova M. Towards application of neural networks for optimal synthesis of distributed database systems // Proc. of the 12th IEEE Intrl. Conference on Electronics, Circuits, and Systems, Gammarth, Tunisia. – 2005. – P. 486–490.

7. Карпунина М.Е. Использование генетических алгоритмов для повышения эффективности работы искусственных нейронных сетей // Известия Академии инженерных наук им. А.М. Прохорова. Бизнес-информатика. – Н.Новгород: НГТУ.– 2005.

Бабкин Эдуард Александрович – Государственный университет – Высшая школа экономики (ГУ-ВШЭ), зав. каф., к.т.н., доцент, babkin@hse.nnov.ru

Карпунина Маргарита Евгеньевна – Государственный университет – Высшая школа экономики (ГУ-ВШЭ), аспирант, levati@yandex.ru

УДК 004.315

СПЕЦИАЛИЗИРОВАННОЕ ВЫЧИСЛИТЕЛЬНОЕ УСТРОЙСТВО ДЛЯ ОБРАБОТКИ РАДИОЛОКАЦИОННОЙ ИНФОРМАЦИИ

А.И. Грушин, М.Л. Ремизов, А.В. Ростовцев, Д.Д. Николаев, Чинь Куанг Киен

Одной из характерных задач обработки радиолокационной информации является вычисление комплексной матрицы в режиме реального времени. В работе рассмотрена аппаратная реализация алгоритма рекурсивного вычисления комплексной матрицы 64×64.

Ключевые слова: вычислительное устройство, матрица, комплексное умножение с накоплением, числа с плавающей запятой, ПЛИС

Введение

В [1] показано, что при выделении сигналов на фоне интенсивных помех наиболее трудоемкой в вычислительном отношении является процедура вычисления в режиме реального времени комплексной обратной матрицы $R^{-1}{}_n$, зависящей от выборки векторов y_n и константы μ:

$$R_0^{-1} = I\ ;\ R_n^{-1} = R_{n-1}^{-1} - \frac{1}{\mu + \tilde{y_n} z_n} z_n \tilde{z}_n\ ,\ z_n = R_{n-1}^{-1} y_n\ ,\ n = 1, 2, \ldots, 128, \qquad (1)$$

где R_n^{-1} – комплексная матрица размерностью 64×64, I – единичная матрица, μ – целая положительная константа, y_n – 64-мерный комплексный вектор, коэффициенты действительной и мнимой частей компонента – 12-разрядные целые числа со знаком, ˜ обозначает комплексное сопряжение и транспонирование, $N = 128$ – количество векторов y_n в одной выборке.

Значение матрицы R^{-1} вычисляется на выборке из 128 векторов y_n, затем она передается в другое устройство для принятия решения об обнаружении полезного сигнала. За 5 с (рабочий цикл радиолокационной системы) необходимо выполнить вычисление матрицы для 576 различных выборок векторов с тремя значениями μ.

Вычисления программным способом на компьютере (Core 2 Duo, 2,66 ГГц, 1 Гбайт) занимают более 43 мин., т.е. не обеспечивают требуемую скорость, поэтому принято решение реализовать вычисления на аппаратуре с параллельной архитектурой с созданием прототипа на ПЛИС.

Анализ формулы и результатов моделирования позволил установить, что $\tilde{y_n} z_n$ – положительная величина, и вычисления в формате single [2] обеспечивают необходимую точность и диапазон вычислений.

Схема вычисления и формат чисел

Для вычисления нового значения матрицы нужно провести 128 итераций по формуле (1). Итерацию разделим на этапы (табл. 1).

Таблица 1. Этапы вычислений

	Этап	Операции и объем вычислений
1	$z_n = R_{n-1}^{-1} y_n$	64×64 MAC
2	$\alpha = \mu + \tilde{y}_n z_n$	1×64 MAC
3	$k = \frac{1}{\alpha}$	1DIV
4	$w_n = -kz_n$	64 MUL
5	$R_n^{-1} = R_{n-1}^{-1} - w_n \tilde{\mathbf{z}}_n$	64×64 MAC

Операции деления DIV и умножения MUL выполняются над действительными числами, а умножение с накоплением MAC выполняется над комплексными числами.

Моделирование показало, что 62-разрядные числа с фиксированной запятой обеспечивают необходимый диапазон вычислений и точность до 10-го двоичного знака по сравнению с вычислениями программным способом в формате single, но производительность при этом меньше требуемой. Использование чисел с плавающей запятой формата single позволяет автоматически решить проблему диапазона и точности вычислений. В табл. 2 приведено сравнение двух реализаций вычислителя.

Таблица 2. Два варианта реализации

Характеристика	Фиксированная запятая 62 разряда	Плавающая запятая single
Процент занимаемой площади кристалла	12 %	55 %
Рабочая частота [МГц]	300	200–250
Время вычисления R^{-1} для одного μ, мс	5,6	1
Трудозатраты на разработку, чел.-мес.	~6	~15

Использование чисел с фиксированной запятой предполагает более простые алгоритмы, меньшие трудозатраты, меньшую площадь кристалла, но требование по производительности повлекло за собой выбор чисел с плавающей запятой.

Исследование алгоритма вычисления обратной матрицы показало, что для обеспечения необходимого диапазона вычислений достаточно 7-разрядного порядка, поэтому в работе предложен формат чисел с плавающей запятой, в котором скрытый бит мантиссы представлен в явном виде (рис.1).

Рис. 1. Формат чисел с плавающей запятой, в котором скрытый бит мантиссы представлен в явном виде

В устройстве не поддерживаются некоторые требования стандарта [2], это упрощает аппаратуру и повышает быстродействие при сохранении той же точности.

Основные узлы вычислительного устройства

Узел преобразования целого в вещественное CI2F. На входе – 12-разрядное целое число Y в прямом коде со знаком. На выходе – вещественное число, содержащее знак, 7-разрядный порядок и 24-разрядную мантиссу, младшие 13 разрядов мантиссы всегда нулевые.

Узел комплексного умножения с накоплением MAC. Узел вычисляет следующую функцию (рис. 2, а):

$AC - BD + E + (AD + BC + F)i$,

где $A + Bi$, $C + Di$, $E + Fi$ – комплексные операнды, коэффициенты A, B, C, D, E, F – числа с плавающей запятой.

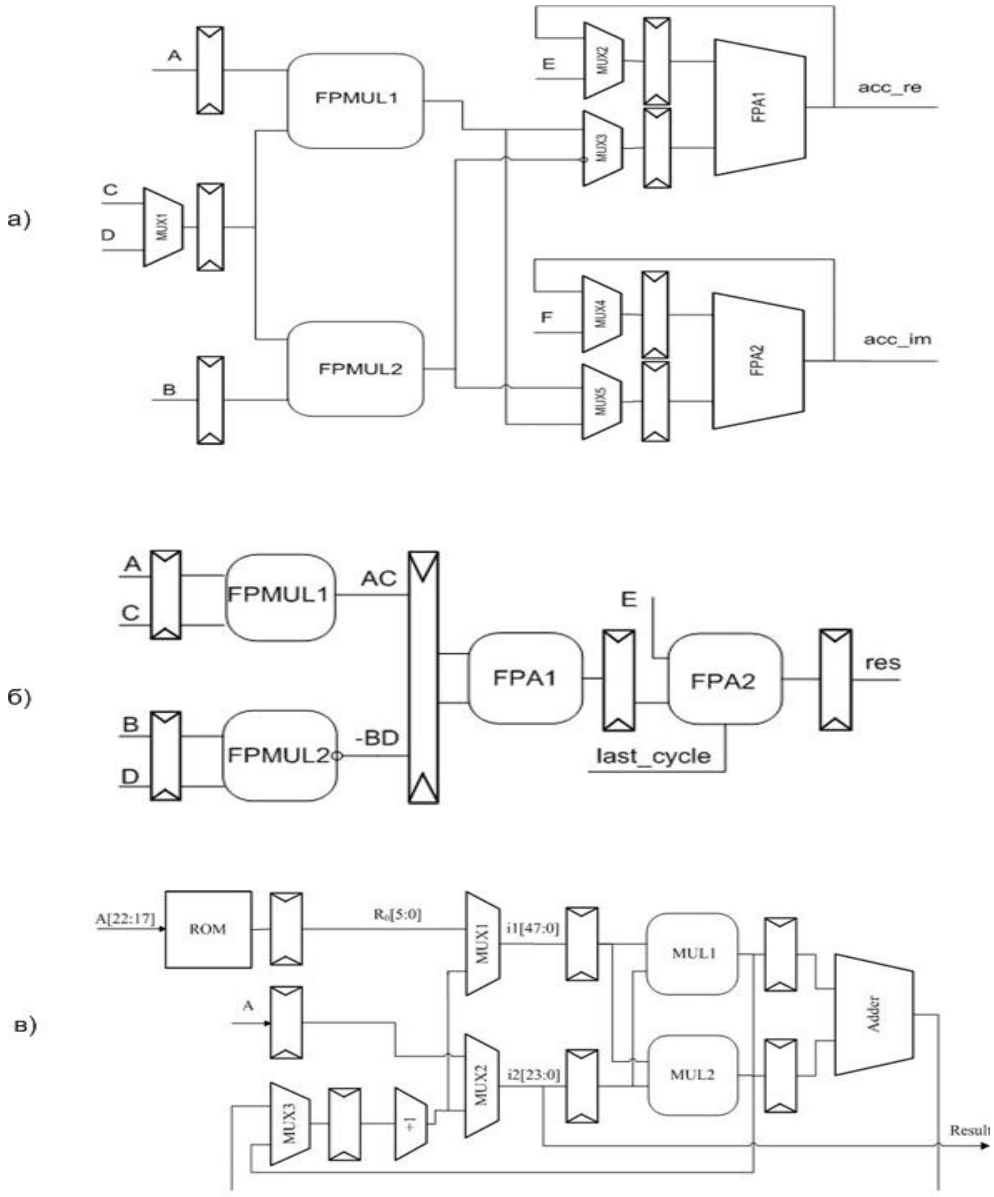


Рис. 2. Функция, вычисляемая узлом комплексного умножения с накоплением MAC

Сначала коммутатор MUX1 пропускает C во входной регистр. Узел умножения чисел с плавающей запятой FPMUL1 вычисляет произведение AC, а узел FPMUL2 – BC. AC проходит через MUX3 на вход узла сложения чисел с плавающей запятой FPA1,

на второй вход которого через коммутатор MUX2 проходит *E*. *BC* через MUX5 попадает на вход FPA2, где складывается с *F*. Затем через MUX1 в узлы умножения проходит *D*, в FPMUL1 вычисляется произведение *AD*, в FPMUL2 вычисляется *BD*. *AD* через MUX5 попадает в FPA2, где складывается с *BC* + *F*, которое через MUX4 подается с выхода acc_re. BD через MUX3 попадает в FPA1, где вычитается из *AC* + *E*, которое через MUX2 подается с выхода acc_im.

В режиме накопления через внешние коммутаторы на входы *E* и *F* подается ранее вычисленный комплексный результат. Таким образом, выход acc_re = *AC* – *BD* + *E*, acc_im = *AD* + *BC* + *F*.

Время выполнения операции MAC составляет 14 тактов, новые операнды на вход MAC можно подавать каждые 8 тактов.

Узел комплексного умножения с накоплением MACR. На вход узла MACR (рис. 2, б) поступают два комплексных числа *A* + *Bi* и *C* + *Di* и действительное число *E*. Результат является положительным числом с плавающей запятой res = *AC* – *BD* + *E*. Мантиссы чисел A и B содержат 24 разряда, а мантиссы чисел *C* и *D* – 11 разрядов.

MACR состоит из двух узлов умножения чисел с плавающей запятой FPMUL1 и FPMUL2 и двух узлов сложения чисел с плавающей запятой FPA1 и FPA2. Для повышения точности разность двух полных 35-разрядных произведений *AC* – *BD* округляется до 35 разрядов к ближайшему. Потом к этому прибавляется *E*, и сумма округляется до 35 разрядов. Когда сигнал last_cycle = 1, происходит округление до 24 разрядов, т.е. большего из форматов исходных чисел.

Время выполнения операции MACR – 12 тактов, новые операнды на вход MACR можно подавать каждые 4 такта.

Узел вычисления обратной величины Recipr. Узел использует алгоритм Ньютона–Рафсона [3], итерации проводятся по формуле

$$R_i = R_i \cdot (2 - A \cdot R_{i-1}),$$

где R_{i-1} – предыдущее приближение, R_i – следующее приближение, A – мантисса числа, обратная величина которого вычисляется.

В качестве начального приближения R_0 используется значение из таблицы, хранящейся в памяти ROM (рис. 2, в), которое выбирается по шести старшим разрядам дробной части мантиссы числа, подающегося на вход узла.

Коммутатор MUX1 пропускает множимое R_0, MUX2 пропускает A, в умножителе MUL1 в следующем такте вычисляется произведение R_0*A[23:0]. Разность $2 - A \cdot R_0$ вычисляется как дополнительный код произведения $A \cdot R_0$.

Второе умножение первой итерации выполняется сразу на двух умножителях MUL1 и MUL2, два произведения складываются в сумматоре Adder. Произведение $R_0 \cdot (2 - A \cdot R_0)$ округляется до 24 разрядов с помощью инкрементора. Таким образом, в результате первой итерации мы получаем новое приближение мантиссы результата R_1.

Две следующие итерации выполняются аналогичным образом.

Устройство выдает результат через 19 тактов после получения операнда.

Структура вычислительного устройства

На рис. 3, а, изображена структурная схема специализированного вычислительного устройства. Основной объем вычислений производится в блоке из 64 узлов MAC1–MAC64, каждый из которых вычисляет *AC* – *BD* + *E* + (*AD* + *BC* + *F*)*i*. Он позволяет быстро выполнять действия с матрицами, предусмотренные табл. 1. Блок MAC совершает вычисления одновременно над всеми компонентами одного столбца матрицы R_n^{-1}, что обусловливает организацию памяти для хранения матрицы R_n^{-1}. Это память с глу-

биной 64 (количество столбцов) и разрядностью шины данных 64×2×32. В состав устройства также входят узел преобразования целого в вещественное CI2F, узел MACR, узел вычисления обратной величины, узел управления Control Unit и другие узлы.

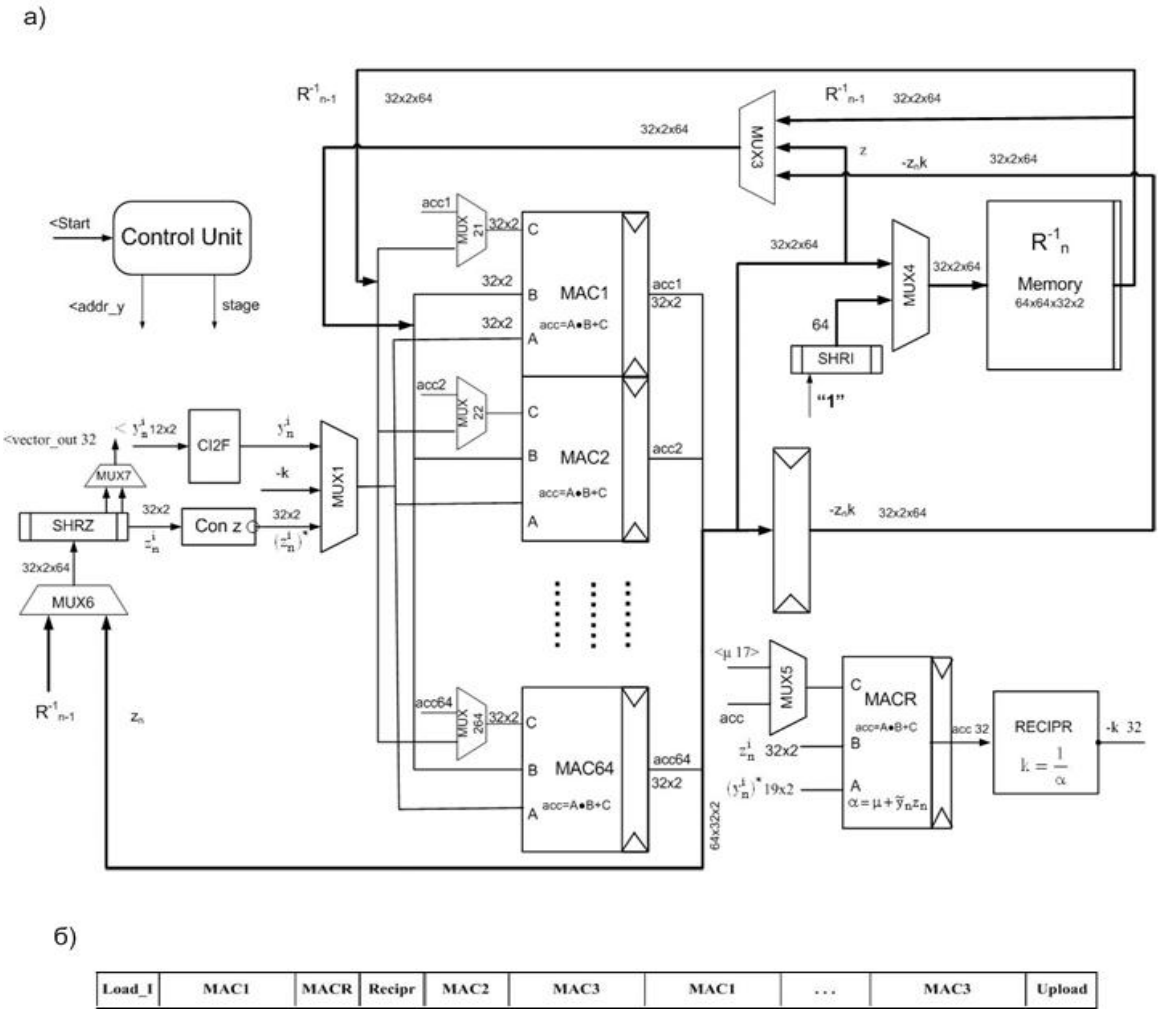


Рис. 3. Структурная схема специализированного вычислительного устройства (а) и временная диаграмма его работы (б)

Интерфейс вычислительного устройства

Входной сигнал Start запускает работу устройства, параметр μ определяет возможность различать близко находящиеся цели, от него зависит точность вычислений, результат с фазированной антенной решетки после аналогово-цифрового преобразователя дает вектор y_n. Адрес *addr_y* используется для чтения компоненты вектора y_n из внешней памяти, младшие 6 разрядов определяют номер компоненты, старшие 7 разрядов определяют номер вектора. В памяти хранятся 128 векторов выборки. 32-разрядный выход vector_out попеременно выдает коэффициент действительной и мнимой частей элемента матрицы, через него во внешнюю память для дальнейшей обработки передается вычисленная матрица.

Описание работы вычислителя

Вычисления начинаются с подачи сигнала Start, который вырабатывается один раз для обсчета всех 128 векторов y_n выборки. По сигналу Start обнуляются счетчики

узла управления, и устройство начинает работать в соответствии с временной диаграммой (рис. 3, б).

Фаза Load_I (начальная загрузка). В память обратной матрицы R_n^{-1} записывается вещественная единичная матрица *I*. Для этого на соответствующий вход коммутатора MUX4 подается коэффициент мнимой части 0, коэффициент действительной части – вещественная единица. В течение 64 тактов прописываются все 64 столбца памяти. В первом столбце вещественная единица записывается в первый (верхний) элемент, во втором столбце – во второй элемент, и т.д. Для этого в сдвиговом регистре SHRI каждый такт на один разряд сдвигается единица, указывающая на нужный элемент столбца.

Фаза MAC1 (вычисление вектора z_n). Длительность фазы – 64 цикла работы MAC. Коммутатор MUX1 пропускает компоненту y_n, которая поступает в MAC1–MAC64 на вход множителя, MUX2 при подаче нулевой компоненты вектора y_n пропускает 0, затем acc (результат с выхода MAC), MUX3 – столбец из памяти матрицы R_n^{-1}. Каждый цикл работы MAC1 меняются разряды адреса чтения y_n *addr_y* [5:0].

За 64 цикла работы блок MAC вырабатывает 64 компоненты вектора z_n, которые параллельно принимаются в сдвиговый регистр SHRZ, где преобразуются в последовательный код во время фазы MACR.

Фаза MACR (вычисление знаменателя α). Длительность фазы – 64 цикла работы MACR. На вход множителя поступает 19 разрядов компоненты $(y_n^1)^*$ (знак, 7 разрядов порядка, 11 старших разрядов мантиссы) с выхода преобразователя целого в вещественное CI2F, на вход множимого поступает компонента z_n из сдвигового регистра SHRZ. Через MUX5 на вход слагаемого в начальный момент времени поступает константа μ с входа спецвычислителя, затем через этот коммутатор проходит acc (результат предыдущей операции MACR).

При каждом цикле работы MACR меняются младшие 6 разрядов адреса чтения y_n и происходит сдвиг на одну компоненту в регистре SHRZ. В конце фазы на выходном регистре находится величина α, которая используется в фазе Recipr.

Фаза Recipr (вычисление обратной величины). За три итерации вычисляется обратная величина $1/\alpha$, длительность фазы – 19 тактов.

Фаза MAC2 (вычисление произведения $-k \cdot z_n$. Длительность фазы – 1 цикл работы MAC. С выхода узла Recipr через MUX1 на вход множителя MAC1–MAC64 поступает величина $-k$. На вход множимого через MUX3 поступают компоненты вектора z_n, на вход слагаемого через MUX2 подается 0. Произведение записывается в регистр RgZK.

Фаза MAC3 (вычисление нового значения матрицы R_n^{-1}). Длительность фазы – 64 цикла работы MAC. Через MUX1 в блок MAC1–MAC64 на вход множителя поступает компонента вектора z_n^*, она получается в узле комплексного сопряжения ConZ. На входы множимого MAC1–MAC64 через MUX3 поступают компоненты вектора $-k \cdot z_n$. На вход слагаемого MAC1–MAC64 через MUX2 из памяти матриц поступает столбец старого значения матрицы R_{n-1}^{-1}. На выходе MAC1–MAC64 получается новое значение столбца матрицы, которое записывается в память матрицы через MUX4.

При каждом цикле работы MAC в память матрицы записывается новое значение столбца, в сдвиговом регистре SHRZ происходит сдвиг на одну компоненту вектора z_n (тем самым в младших 64 разрядах регистра оказывается следующая компонента вектора z_n), из памяти считывается новый столбец. За 64 цикла работы MAC в память записывается новое значение матрицы. По завершении фазы MAC3 значение старших семи разрядов адреса чтения компоненты вектора y_n *addr_y* [12:6] увеличивается на 1.

Фаза Upload (передача матрицы R^{-1} во внешнюю память, происходящая по окончании обработки выборки векторов). Передача всей матрицы занимает 8192 такта. Из памяти считывается столбец, через MUX6 он записывается в сдвиговый регистр SHRZ, где раз в 2 такта происходит сдвиг вправо на одну компоненту. С младших 64 разрядов регистра SHRZ через MUX7 каждый такт считывается половина элемента матрицы: в первом такте – 32 разряда мнимой части, во втором такте – 32 разряда действительной части. Эти действия повторяются 64 раза, по числу столбцов матрицы R^{-1}. По окончании фазы устройство ожидает нового сигнала Start.

Фазы Load_I и Upload выполняются один раз за время обработки выборки векторов y_n, остальные фазы повторяются 128 раз.

Верификация и создание прототипа

Все узлы вычислительного устройства прошли верификацию в автономном режиме на случайных направленных тестах. Одинаковые воздействия подавались на RTL-модель узла на языке Verilog и на функциональную модель этого узла, написанную на языке C++. Реакции обеих моделей сравнивались. Функциональная модель не описывает все подробности алгоритмов обработки чисел с плавающей запятой, используемых в узле, а опирается на операции, выполняемые микропроцессором Pentium 4, что упростило написание функциональной модели и ускорило ее отладку и работу.

После автономной верификации узлы собирались в устройство, которое также верифицировалось на случайных направленных тестах.

Устройство синтезировано на ПЛИС типа Virtex-5 xc5vlx330 [4].

Прототип устройства занимает 54% ресурсов кристалла, рабочая частота – 200 МГц. Вычисление матрицы для одной выборки занимает менее 10^{-3} сек, производительность ~6,5 gigaFLOPS.

Заключение

В работе описано вычислительное устройство с параллельной архитектурой для аппаратной реализации алгоритма рекурсивного вычисления комплексной матрицы 64×64. Проанализирована и преобразована исходная формула, разработаны функциональные модели, проведено исследование точности вычислений, предложено представление информации, облегчающее проектирование, разработаны и верифицированы узлы устройства, разработана структура специализированного вычислительного устройства, проведена отладка устройства, синтезирован прототип.

Производительность и точность устройства удовлетворяют требованиям.

Повышение точности вычислений улучшает возможность различать близко расположенные цели, для этого можно увеличить разрядность мантиссы используемого формата и уменьшить количество округлений при вычислениях за счет использования метода MAF (multiply add fused).

Производительность устройства можно увеличить за счет полной конвейеризации узлов, использования более быстрых алгоритмов умножения и вычисления обратной величины и использования нескольких узлов MACR. Реализация этих возможностей, наряду с использованием заказной интегральной схемы отечественного производства с технологией 180 нм, позволяет увеличить производительность в 5–10 раз по сравнению с прототипом.

Литература

1. Черемисин О.П. Адаптивные алгоритмы обработки сигналов в многоканальных приемных системах с антенными решетками // Радиотехника и электроника. – 2006.– Т. 51. – № 9. – С. 1087–1098.

2. IEEE Standard for Binary Floating-Point Arithmetic / ANSI/IEEE Standard No. 754. – American National Standards Institute. – Washington, DC, 1985.
3. Ercegovac M., Lang T. Digital Arithmetic. – San Francisco: Morgan Kaufmann Publishers, 2004.
4. Сайт FPGA and CPLS solutions from xilinx. – Режим доступа: http://www.xilinx.com/support/documentation/virtex-5.htm, свободный.

Грушин Анатолий Иванович – Институт точной механики и вычислительной техники РАН, ведущий научный сотрудник, к.т.н., aigrushin@ipmce.ru

Ремизов Максим Леонидович – Институт точной механики и вычислительной техники РАН, инженер-конструктор, mlremizov@ipmce.ru

Ростовцев Артем Вадимович – Институт точной механики и вычислительной техники РАН, инженер-конструктор, avrostovtsev@ipmce.ru

Николаев Дмитрий Дмитриевич – Московский физико-технический институт (государственный университет), студент, ddnikolaev@ipmce.ru

Чинь Куанг Киен – Московский физико-технический институт (государственный университет), студент, kkchin@ipmce.ru

УДК 004.45

ТЕХНОЛОГИЯ ПОСТРОЕНИЯ ВИРТУАЛЬНЫХ ИСПЫТАТЕЛЬНЫХ СТЕНДОВ В РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СРЕДАХ

Г.И. Радченко, Л.Б. Соколинский

В работе представлена технология построения виртуальных испытательных стендов в распределенных вычислительных средах на основе технологии CAEBeans. Дано описание структурной организации оболочек CAEBeans, описаны базовые принципы реализации системы CAEBeans.

Ключевые слова: CAEBeans, Грид, распределенные вычисления, Грид-сервисы, иерархии проблемно-ориентированных оболочек

Введение

При разработке новых технологических линий и производственных технологий важную роль играют экспериментальные комплексы и стенды. В качестве примера можно привести аэродинамические трубы, стенды по испытанию резьбы труб и др. В настоящей работе рассматривается технология построения виртуальных испытательных стендов путем создания и анализа математических моделей разрабатываемой продукции на высокопроизводительных вычислительных системах с помощью инженерных пакетов. Такой подход позволяет существенно повысить точность анализа проектных вариантов продукции, значительно сократить стоимость опытно-конструкторских работ и проводить виртуальные эксперименты, которые в реальности выполнить невозможно.

Для проведения экспериментов в области современного компьютерного моделирования требуются значительные вычислительные ресурсы. Опыт использования суперкомпьютерных систем показывает, что максимальная эффективность использования вычислительных ресурсов может быть достигнута при объединении таких систем в вычислительные Грид-сегменты [1].

В качестве перспективного подхода к решению задач организации виртуальных испытательных стендов в распределенных вычислительных средах предлагается применение технологии CAEBeans [2]. В основе технологии CAEBeans лежит обеспечение сервисно-ориентированного предоставления программных ресурсов базовых компонентов CAE-систем и формирование деревьев проблемно-ориентированных оболочек, инкапсулирующих процедуру постановки и решения конкретного класса задач. Технология CAEBeans автоматизирует декомпозицию задач на типовые подзадачи, поиск вычислительных ресурсов, постановку задач соответствующим базовым компонентам CAE-систем, мониторинг хода решения задач, предоставление результатов решения задачи пользователю.

Модель задачи инженерного моделирования

При постановке задачи посредством проблемной оболочки CAEBean пользователю предоставляется возможность оперировать терминами той проблемной области, в рамках которой проводится решение задачи. Каждая конкретная задача может быть описана некоторым индивидуальным набором значений входных параметров, необходимых для реализации алгоритма решения данного класса задач. В качестве примера таких параметров можно привести скорость поступательного движения и скорость вращения трубы при моделировании процесса закалки [3]. С точки зрения системы CAEBeans, каждый параметр решения задачи имеет определенную семантику, определяя свойство проблемной области задачи или отражая особенности процесса решения задачи. *Постановкой задачи*, с точки зрения системы CAEBeans, является указание пользователем значений всех параметров, определяющих конкретную инженерную задачу. Таким образом, проблемно-ориентированную постановку задачи можно описать так называемым *полным дескриптором задачи*, представляющим собой множество пар «параметр–значение».

Для решения задачи инженерного моделирования пользователь должен пройти сложный технологический цикл, который может включать в себя следующие этапы: подготовка геометрии модели исследуемой области; построение вычислительной сетки; определение физики протекающих процессов; решение поставленной задачи моделирования; визуализация и анализ полученных результатов. Таким образом, мы можем составить *логический план* решения каждого определенного класса задач компьютерного моделирования, который представляет собой ориентированный граф, в вершинах которого могут находиться блоки двух типов: подзадачи, выполняемые отдельными базовыми компонентами CAE-пакетов, и специальные служебные операции управления потоком решения задачи (ветвление, распараллеливание и т.п.), что обеспечивает возможности гибкого исполнения логического плана. В процессе решения полный дескриптор задачи распределяется между блоками операций логического плана решения задачи.

В соответствии с этим, в зависимости от определяемой семантики, все множество параметров можно разбить на непересекающиеся подмножества категорий. С одной стороны, категории параметров отражают отдельные этапы технологического цикла (построение геометрии, визуализация и др.). С другой стороны, можно выделить отдельные категории параметров, ответственных за процесс решения задачи инженерного моделирования (например, выбор определенной среды генерации геометрии задачи или системы визуализации).

Дерево проблемных оболочек CAEBeans

В основе технологии CAEBeans лежит процесс классификации типовых задач, для решения которых используется CAE-пакет, и построения дерева проблемных оболочек

CAEBeans над CAE-пакетом на основе этой классификации. Корневой элемент дерева проблемных оболочек представляется проблемной оболочкой, соответствующей наиболее обобщенному описанию определенного класса решаемых задач. Процедуру формирования дерева проблемных оболочек можно описать как движение от корневого элемента к листьям. Технология CAEBeans позволяет дочерним проблемным оболочкам конкретизировать классы задач, решаемые их родительской оболочкой, путем выделения и инкапсуляции групп параметров, значения которых фиксируются на данном уровне абстракции.

Первоначально для решения конкретной инженерной задачи предлагается использовать проблемную оболочку с листовой вершины соответствующего дерева проблемных оболочек. Данная оболочка имеет максимально простой пользовательский интерфейс. Однако ее область применения ограничена и допускает решение узкого класса задач. Если инженеру необходимо охватить более широкий класс задач, он осуществляет переход к родительской оболочке, переходя, тем самым, на более высокий уровень абстракции. Родительская оболочка расширяет класс задач, который может быть решен посредством проблемной оболочки предыдущего уровня абстракции. Таким образом, пользователь может дойти до корневого CAEBean, который предлагает максимальные возможности по решению конкретного класса задач.

Слои архитектуры системы CAEBeans

Для поддержки технологии CAEBeans и разработанной модели инженерной задачи была разработана слоистая архитектура системы CAEBeans. Архитектура системы CAEBeans формируется из четырех слоев – концептуального, логического, физического и системного (рис. 1).

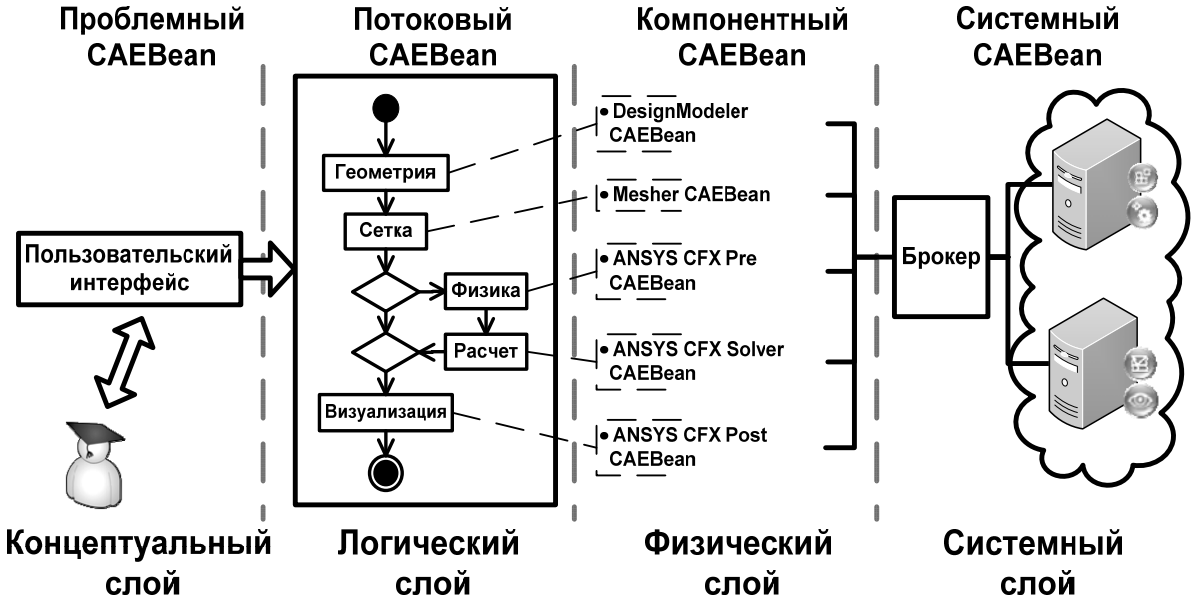


Рис. 1. Схема слоев архитектуры CAEBeans

Концептуальный слой системы формируется на основе оболочек CAEBeans, которые мы будем называть проблемными. Пользовательский интерфейс, предоставляемый *проблемным CAEBean*, является основным средством взаимодействия пользователя с системой CAEBeans, посредством которого пользователь может произвести постановку инженерной задачи, проследить за ходом решения поставленной задачи, получить требуемые результаты решения.

Процесс постановки задачи посредством проблемного CAEBean сводится к указанию тем или иным образом значений входных параметров, характеризующих задачи

соответствующего класса. Проблемный CAEBean скрывает от пользователя распределенный характер вычислительной среды, структуру аппаратных, программных и лицензионных ресурсов. Также технология CAEBeans предусматривает возможность формирования дерева проблемных CAEBeans. Это позволяет организовать подстройку проблемных CAEBeans под возможную конкретизацию базовой общей задачи в соответствии с требованиями конечных пользователей.

При постановке пользователем задачи соответствующий проблемный CAEBean формирует *полный дескриптор задачи*. При этом не делается различий между тем, указывал ли пользователь значение данного параметра явно, или же оно было тем или иным способом вычислено соответствующим проблемным CAEBean. Набор параметров, составляющих полный дескриптор задачи, одинаков для всех проблемных CAEBean, входящих в одно дерево проблемных оболочек.

Сформированный полный дескриптор задачи передается в *потоковый CAEBean*, представляющий логический слой системы CAEBeans. Потоковый CAEBean реализует *логический план* решения определенного класса задач компьютерного моделирования. При получении полного дескриптора задачи потоковый CAEBean распределяет массив переданных параметров по блокам операций логического плана решения задачи (рис. 2). В процессе решения задачи потоковый CAEBean, «руководствуясь» планом решения задачи, передает параметры подзадачи соответствующим *компонентным CAEBean* и получает от них результаты решения соответствующих подзадач.

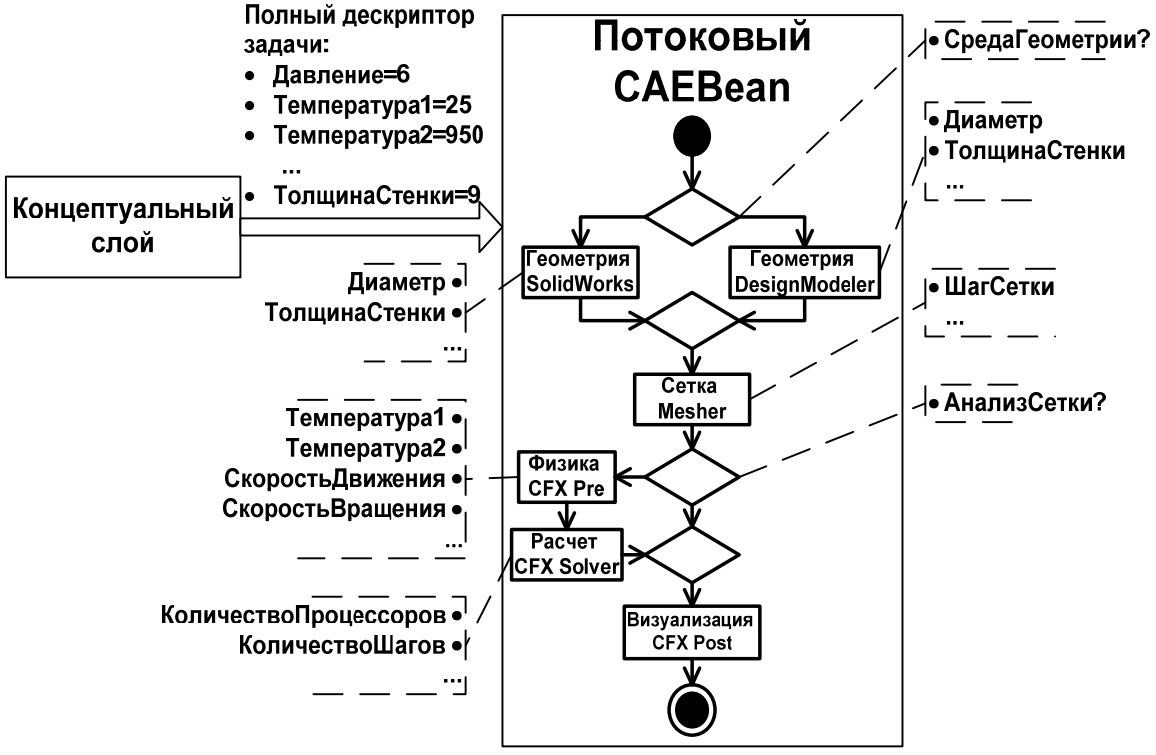


Рис. 2. Распределение параметров по блокам логического плана решения задачи

Компонентные оболочки CAEBean, составляющие физический слой системы CAEBeans, инкапсулируют процесс постановки и решения конкретной подзадачи компьютерного моделирования посредством определенного базового компонента. Компонентный CAEBean обладает информацией о методах взаимодействия с конкретным базовым компонентом. В зависимости от интерфейса автоматизированной постановки задач, поддерживаемого базовым компонентом, разрабатывается специальный генератор постановки данной подзадачи. Внедрение физического слоя в систему CAEBeans позволяет обеспечить универсальность применения совместимых базовых компонентов для решения одинаковых подзадач. Возможность применения того или иного базового

компонента для решения определенной подзадачи зависит только от возможности реализации компонентного CAEBean, транслирующего постановку подзадачи в соответствующую компонентно-ориентированную форму.

С точки зрения системной реализации, большинство современных CAE-систем может быть представлено как набор относительно функционально-независимых компонентов, каждый из которых отвечает за собственную ступень технологического цикла решения задачи компьютерного моделирования. С точки зрения CAEBeans, такие компоненты (*базовые компоненты*) – это изолированные приложения, которые представляются как «черный ящик», реализующий некий специфический интерфейс [3–5].

Системным CAEBean называется оболочка, инкапсулирующая функциональные возможности конкретного базового компонента и обеспечивающая сервисно-ориентированный подход к постановке задач и получению результатов. Системный CAEBean, реализованный для конкретного базового компонента, является его «представителем» в вычислительной Грид-среде. Практически во всех случаях современными инженерными системами поддерживается широкий спектр различных методов управления процессом постановки и решения задач.

Для решения подзадачи, поставленной на физическом слое компонентным CAEBean, необходимо обеспечить ее проекцию на набор ресурсов, предоставляемых системным слоем. Для организации этого процесса используется *брокер ресурсов* – автоматизированная система регистрации, анализа и предоставления ресурсов распределенной вычислительной среды. Брокер представляет собой промежуточное программное обеспечение, реализующее оптимальный выбор и виртуализацию доступа к наборам ресурсов в гетерогенной вычислительной среде. Компонентный CAEBean формирует запросы на предоставление вычислительных ресурсов, которые передает брокеру ресурсов. На основе этих запросов брокер выбирает те ресурсы, которые удовлетворяют критериям поиска, и передает адреса данных ресурсов компонентному CAEBean.

Реализация оболочек CAEBeans

Важной частью работы является применение разрабатываемой технологии для решения конкретных производственных задач. По заказу Челябинского трубопрокатного завода разработана проблемно-ориентированная оболочка для решения задачи «Моделирование овализации труб при закалке» [3]. Разработанная оболочка инкапсулирует процесс решения задачи в рамках функциональных возможностей пакета DEFORM и позволяет производить моделирование задачи вычислительными ресурсами суперкомпьютерного центра Южно-Уральского государственного университета.

Заключение

В статье представлены основные результаты разработки технологии CAEBeans, ориентированной на формирование виртуальных испытательных стендов в распределенных вычислительных средах. Архитектура представленной системы может быть разделена на четыре слоя – концептуальный, логический, физический и системный.

Можно выделить следующие основные этапы дальнейшего развития работы: программная реализация иерархии оболочек CAEBeans; разработка программных средств для поддержки технологии CAEBeans. В рамках апробации технологии CAEBeans необходимо продолжать разработку оболочек CAEBeans для решения реальных задач инженерного анализа.

Работа проводилась при финансовой поддержке Федерального агентства по науке и инновациям (грант 2007-4-1.4-20-01-026), программы СКИФ-ГРИД (грант СГ-1/07) и Фонда содействия развитию малых форм предприятий в научно-технической сфере (грант 7434).

Литература

1. Foster I., Kesselman C., Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations // International J. of Supercomputer Applications and High Performance Computing. – 2001. – Т. 15. – № 3. – P. 200–222.
2. Радченко Г.И., Соколинский Л.Б. CAEBeans: иерархические системы структурированных проблемно-ориентированных оболочек над инженерными пакетами // Тр. Всерос. науч. конф. «Научный сервис в сети Интернет: многоядерный компьютерный мир. 15 лет РФФИ». – Новороссийск, 2007. – С. 54–57.
3. Дорохов В.А., Маковецкий А.Н., Соколинский Л.Б. Разработка проблемно-ориентированной GRID-оболочки для решения задачи овализации труб при закалке // Тр. междунар. науч. конф. «Параллельные вычислительные технологии». – СПб, 2008. – 520 с.
4. Радченко Г.И., Соколинский Л.Б., Шамакина А.В. Разработка компонентно-ориентированных CAEBean-оболочек для пакета ANSYS CFX // Тр. междунар. науч. конф. «Параллельные вычислительные технологии». – СПб, 2008. – С. 438–443.
5. Насибулина Р.С. [и др.] Методы организации программных интерфейсов к инженерным пакетам в среде GPE // Тр. междунар. науч. конф. «Параллельные вычислительные технологии». – СПб, 2008. – С. 537.

Радченко Глеб Игоревич – Южно-Уральский государственный университет, аспирант, gleb.radchenko@gmail.com

Соколинский Леонид Борисович – Южно-Уральский государственный университет, д.ф.-м.н, профессор, sokolinsky@acm.org

УДК 519.685

РАЗРАБОТКА ПРИЛОЖЕНИЙ В СРЕДЕ PARJAVA

А.И. Аветисян, В.В. Бабкова, М.Д. Калугин

Статья посвящена вопросам разработки и настройки масштабируемых параллельных программ в рамках интегрированной среды ParJava. Рассмотрены основные трудности разработки и инструменты среды, помогающие их решать.

Ключевые слова: разработка масштабируемых параллельных приложений, высокопродуктивные кластерные вычисления.

Введение

При расчете сложных технологических систем объем вычислений настолько велик, что возникает необходимость в выполнении программ на высокопроизводительных вычислительных системах. Кластеры являются одним из распространенных типов таких систем.

Существует много инструментальных систем, таких как TAU [1] (Университет штата Орегон и Исследовательский центр Juelich из Лос-Аламоса) и Paradyn [2] (Университет штата Висконсин), которые поддерживают реализацию, отладку и доводку параллельных программ. Как правило, эти системы предоставляют программисту наборы таких инструментов для анализа параллельных программ, как профилировщик, трассировщик, инструменты для моделирования работы программы, визуализатор и др. Эти системы являются наборами различных инструментов, нацеленных на поддержку разработки параллельных программ. Однако в них не рассматриваются вопросы организации комплексного использования набора инструментов в рамках единой методологии, позволяющей разрабатывать параллельные программы гарантированного качества с использованием таких инструментов.

В настоящей работе, в отличие от рассмотренных систем, предложена итерационная модель процесса разработки параллельных программ в рамках интегрированной среды ParJava [3]. При этом не только использовались уже реализованные инструменты, но были разработаны новые (поддержка контрольных точек, Омега-тест, выявление возможности распараллеливания без обменов и др.).

Технологический процесс использования среды включает следующие этапы:

1. реализация последовательной версии разрабатываемой программы;
2. оценка максимального потенциально достижимого ускорения по доле последовательных вычислений в программе;
3. обеспечение возможности параллельного выполнения гнезд циклов:
   a. исследование гнезд циклов на возможность параллельного выполнения (вычисление вектора направлений и вектора расстояний между итерациями гнезда циклов с помощью Омега-теста);
   b. преобразование гнезд циклов к виду, допускающему параллельное выполнение (устранение зависимостей по данным между итерациями);
4. распределение данных по узлам вычислительной сети;
5. выбор операций обмена данными;
6. оценка границ области масштабируемости и времени счета на реальных данных;
7. использование механизма контрольных точек в случае необходимости.

Инструментальные средства среды ParJava позволяют увязать перечисленные этапы в единый технологический процесс, обеспечивающий итеративную разработку параллельной программы: если после очередного этапа выясняется невозможность достижения необходимого качества (масштабируемости), следует вернуться на предыдущий этап и скорректировать его результаты. Например, если после этапа 2 выяснится, что минимально возможная доля последовательных вычислений слишком велика, необходимо вернуться на этап 1 и, в связи со сказанным ранее, изменить последовательный алгоритм.

Отметим, что, вообще говоря, итеративная разработка предполагает достаточно большое число итераций, что может привести к большим накладным расходам по времени разработки и ресурсам. Итеративная разработка реальных программ, предлагаемая в данной работе, возможна благодаря тому, что большая часть процесса выполняется на инструментальном компьютере, в том числе за счет использования инструментов, базирующихся на интерпретаторе параллельных программ.

Определение распараллеливаемой части программы

Чтобы получить масштабируемую программу, надо свести к минимуму последовательную часть программы, так как даже небольшая доля последовательных вычислений существенно сокращает область масштабируемости параллельной программы (закон Амдала).

В ParJava реализован инструмент «Профилировщик», на котором вычисляется временной профиль последовательной программы. Временной профиль позволяет оценить потенциально достижимое ускорение.

Цикл распараллеливаем тогда, когда нет зависимостей между итерациями. Зависимость между двумя итерациями возникает тогда, когда они ссылаются на один и тот же элемент массива, причем как минимум одно из них ссылается по записи. Для выявления таких зависимостей в общем случае составляется система уравнений и неравенств относительно индексов и выясняется, имеет ли эта система решения. Уравнения возникают из условия равенства индексных выражений, а неравенства показывают границы изменения индексов, в пределах которых происходят обращения в программе.

Для определения зависимостей существует большое количество тестов. Прямое решение задачи требует значительных временных затрат даже в случае линейных ин-

дексных выражений, так как нахождение зависимостей сводится в итоге к решению системы диофантовых уравнений (или задаче целочисленного линейного программирования), т.е. к NP-полной задаче. Тесты для определения зависимостей можно условно разбить на простые, но не точные, и точные, но сложные. Компромиссным вариантом здесь является Омега-тест, который основан на последовательном применении набора тестов по мере увеличения сложности решения.

Следует отметить, что зависимости могут существовать как внутри одной итерации, так и между итерациями одного цикла. При распараллеливании цикла зависимости внутри итерации мало влияют на вид параллельного алгоритма, тогда как наличие зависимостей между итерациями не позволяет выполнять витки циклов в произвольном порядке, т.е. параллельно на нескольких процессорах.

В среде ParJava зависимости по данным можно проверить при помощи инструмента Loop Analyzer, в который на данный момент включены тесты расстояний и Омега-тест.

Если циклы имеют зависимости между итерациями, существуют способы так их преобразовать, чтобы эти зависимости оказались внутри итераций. Самый надежный и простой способ обойти зависимости – это использование дополнительных буферов для хранения копий массивов, однако он хорош до той поры, пока объем используемой памяти не критичен. Чтобы получить более эффективный вариант программы, применяют композицию примитивных преобразований, которые меняют индексное пространство гнезда циклов [4].

На данный момент в ParJava не реализовано инструментов аффинных преобразований циклов, так как только сам пользователь может определить, в какой последовательности применить эти преобразования, чтобы выявить параллелизм программы. Каждое преобразование гнезда циклов можно выразить через последовательность примитивных аффинных преобразований, каждое из которых соответствует внесению простого изменения в циклы на уровне исходного кода. Существует семь примитивных преобразований, самые значительные из которых – перестановка и скашивание [5]. Если удается так распределить данные по процессорам, что на каждый процессор попадают все данные, которые используются на этом процессоре, то процессоры могут работать независимо друг от друга. В этом случае синхронизации (пересылок данных) не требуется. В ParJava есть инструмент «Независимое распараллеливание», который определяет такие случаи и строит требуемое разбиение массива и его распределение по процессорам, используя известный алгоритм в [6].

Оптимизация пересылки данных в программе

При выполнении большей части программ возникает ситуация, когда данные, вычисляемые на одном процессоре, требуются на другом. В этом случае процессор, использующий данные, требует их у процессора, вычисляющего данные, и возникает не только обмен данными, но и синхронизация. Реализуя синхронизации в программе, необходимо учитывать временные затраты, которые они привносят. Поэтому после определения распараллеливаемого кода в программе возникает вторая проблема – каким образом распараллеливать последовательный код. Это задача оптимального распределения данных и реализации обменов.

Для сокращения времени на моделирование и эксперименты можно реализовать модельный пример программы, сохранив размер последовательной части, размер пересылок и время счета цикла. Такой модельный пример можно быстро переделывать и интерпретировать при помощи инструмента среды ParJava «Интерпретатор», чтобы лучше оценить, какого наибольшего ускорения можно добиться.

Для анализа происходящих в программе пересылок необходимо воспользоваться встроенным в ParJava анализатором трасс. Иллюстрация трасс программы может пока-

зать места разбалансировки и узкие места в программе, что поможет программисту выбрать операции пересылок и их оптимальное расположение в коде.

```
·//sending¶
·Send¶
·Recv¶
·//calculating¶
for·(i·=·beg_i;·i·<·end_i;·
i++)¶
·for·(j·=·0;·j·<·N;·j++)¶
··B[i][j]·=·f(A[i][j]);¶
```

```
//sending¶
ISend¶
IRecv¶
//calculating¶
for·(i·=·beg_i·+·1;·i·<·end_i·-·1;·
i++)¶
·for·(j·=·0;·j·<·N;·j++)¶
··B[i][j]·=·f(A[i][j]);¶
//waiting¶
Wait();¶
//calculating·last·columns¶
if·(myid·!=·0)¶
·for·(j·=·0;·j·<·N;·j++)¶
··B[0][j]·=·f(tempL[j]);¶
··if·(myid·!=·proc_size·-·1)¶
··for·(j·=·0;·j·<·N;·j++)¶
···B[N·-·1][j]·=·f(tempR[j]);¶
```

Рис. 1. Пример использования блокирующих и неблокирующих пересылок

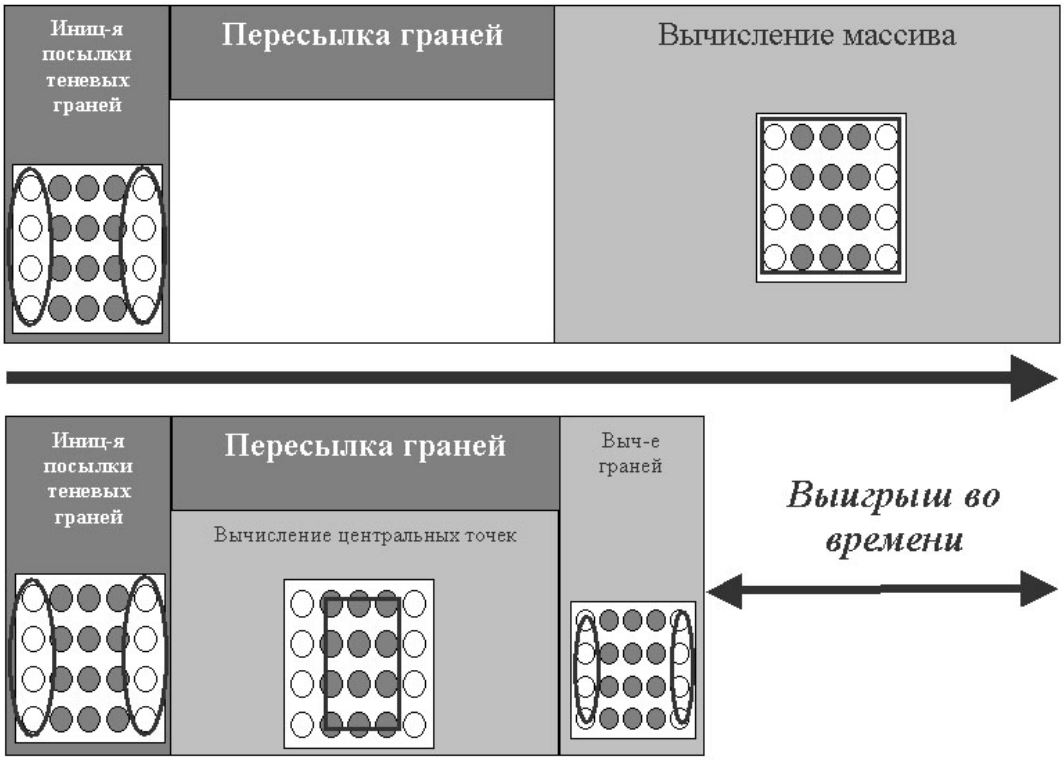


Рис. 2. Получение выигрыша во времени при использовании неблокирующих коммуникаций

Проиллюстрировать важность оптимального выбора операций пересылок может модельный пример (рис. 1). В программе, обрабатывающей массив данных, заменили блокирующие операции Send и Recv на неблокирующие. Схематично процесс пересылок и счета на временной прямой изображен на рис. 2. Совмещение процесса пересылки и счета центральных точек массива дало возможность получить выигрыш во времени. В результате такой замены можно получить большой рост ускорения всей параллельной программы, что демонстрирует график на рис. 3. Варьируя на модельном примере время счета и время пересылок путем изменения объема пересылаемых данных, можно посмотреть, какого ускорения можно в итоге добиться, а затем перенести изменения на большую программу.

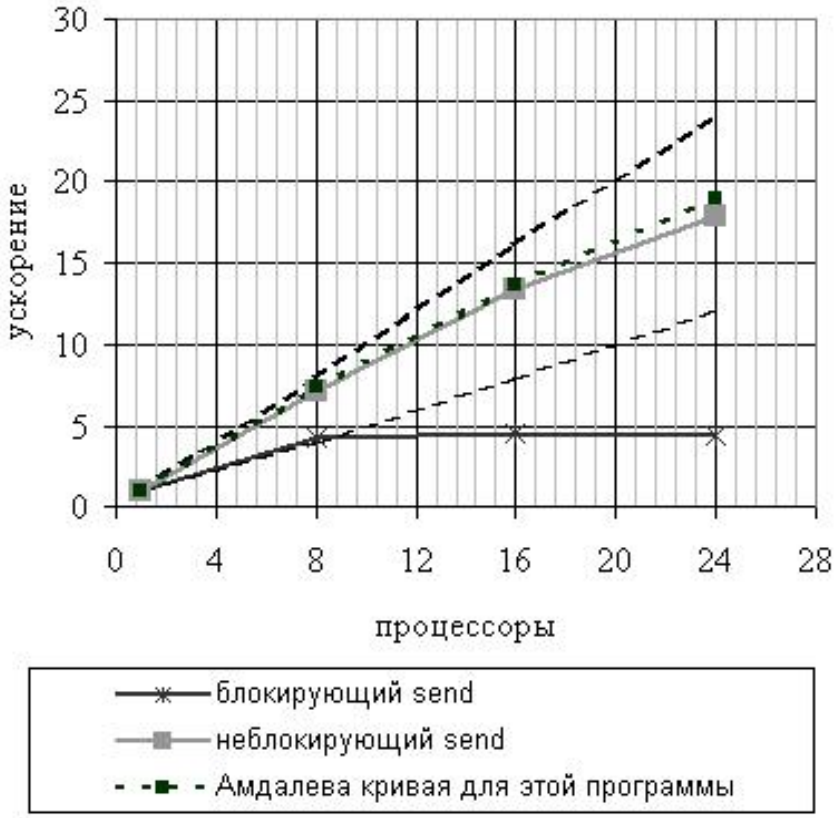


Рис. 3. Сравнение масштабируемости программы с блокирующими и неблокирующими пересылками

Выбрав методы пересылок и реализовав их в настоящей программе, необходимо произвести оценку ее масштабируемости для выбора оптимального количества процессоров. Для этого используется инструмент среды ParJava «Интерпретатор».

Результаты применения ParJava для разработки прикладной программы моделирования интенсивных атмосферных вихрей

Одним из приложений, разработанных в среде ParJava, является параллельная программа моделирования интенсивных атмосферных вихрей (ИАВ), построенная на теории мезомасштабных вихрей по В.Н. Николаевскому (программа разработана в Институте системного программирования РАН в сотрудничестве с Институтом физики Земли РАН) [7, 8]. ИАВ описываются нелинейной системой уравнений смешанного типа, численное решение такой системы в трехмерной сжимаемой сухоадиабатической атмосфере требует очень большого объема вычислений.

Было рассчитано несколько вариантов развития торнадо с учетом силы Кориолиса, в том числе был произведен расчет первых 5 минут развития торнадо, имеющего диаметр 1600 м и высоту 2000 м (он описывался трехмерным массивом размера 320×320×200). На фрагменте кластера МСЦ РАН (64 двухпроцессорных Power 2,2 GHz, 4 GB) вычисления потребовали 96 часов времени счета без учета накладных расходов на организацию контрольных точек. Результаты расчетов подтвердили теорию В.Н. Николаевского и получили признание у специалистов [9].

Заключение

Разработка приложения показала, что среда ParJava обеспечивает быстрое прототипирование и модификацию параллельных программ, что важно при использовании высокопроизводительных вычислений для расчетов сложных технологических систем. В этом случае после выполнения одной-двух серий вычислений по программе в нее вносятся существенные изменения (по существу, программа полностью изменяется). Среда ParJava позволяет сократить время на проектировании и реализацию эффективной параллельной версии программы на основе исходного последовательного алгоритма. Эта среда активно используется в ИСП РАН, ВЦ им. А.А. Дородницына РАН, МСЦ РАН и в учебном процессе ВМК МГУ и ФУПМ МФТИ.

Литература

1. Shende S., Malony A.D. Integration and Application of the TAU Performance System in Parallel Java Environments // Proceedings of the Joint ACM Java Grande ISCOPE Conference. – June, 2001. – P. 87–96.
2. Сайт The Paradyn project. – Режим доступа http://www.paradyn.org/index.html, свободный.
3. Иванников В.П. [и др.] Оценка динамических характеристик параллельной программы на модели // Программирование. – 2006. – №4. – Р. 21–37.
4. Pugh W. The Omega Test: a fast and practical integer programming algorithm for dependence analysis. –ACM, 1991.
5. Bacon D.F., Graham S.L., Sharp O.J. Compiler Transformations for High-Performance Computing // ACM Computing Surveys. – 1994. – Vol. 26. – № 4.
6. Aho A.V. [et al]. Compilers: Principles, Techniques, & Tools. – 2nd ed. – Pearson Education Inc., 2007. – 1000 p.
7. Аветисян А.И., Бабкова В.В., Гайсарян С.С., Губарь А.Ю. Рождение торнадо в теории мезомасштабной турбулентности по Николаевскому. Трехмерная численная модель в ParJava // Математическое моделирование. – 2008. – Т. 20. – № 8.– С. 28–40.
8. Аветисян А.И., Бабкова В.В., Губарь А.Ю Возникновение торнадо: трехмерная численная модель в мезомасштабной теории турбулентности по В.Н. Николаевскому // ДАН. Геофизика. – 2007. – Т. 419. – № 4. – С. 547–552.
9. Avetisyan A.I. [et al]. Intensive Atmospheric Vortices Modeling Using High Performance Cluster Systems // PaCT-2007. – LNCS, 2007. – P. 487–495.

Аветисян Арутюн Ишханович – Институт системного программирования РАН, ученый секретарь, к.ф.-м.н., arut@ispras.ru

Бабкова Варвара Вадимовна – Институт системного программирования РАН, м.н.с., barbara@ispras.ru

Калугин Михаил Дмитриевич – Институт системного программирования РАН, стажер-исследователь, shaman@ispras.ru

УДК 519.245

ЭФФЕКТИВНОЕ ИСПОЛЬЗОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ РЕСУРСОВ ДЛЯ ОПРЕДЕЛЕНИЯ СПРАВЕДЛИВЫХ ЦЕН ОПЦИОНОВ БЕРМУДСКОГО ТИПА

А.С. Горбунова, И.Б. Мееров, А.С. Никонов, А.В. Русаков, А.В. Шишков

В работе рассматривается вопрос эффективной реализации одного из алгоритмов определения справедливых цен опционов бермудского типа. Приводится описание подходов к оптимизации вычислений как в последовательном, так и в параллельном варианте, а также результаты вычислительных экспериментов.
Ключевые слова: метод Монте-Карло, нахождение справедливой цены опциона, параллельное программирование, оптимизация вычислений.

Введение

Задача определения рациональных цен опционов является актуальной в связи с их популярностью на финансовых рынках. Опционы американского типа [1, 2], которые могут быть предъявлены к исполнению в произвольный момент времени в течение срока действия контракта, получили широкое распространение на фондовых биржах. На практике такая задача часто решается для дискретного аналога опционов американского типа – опционов бермудского типа, которые могут быть предъявлены к исполнению в любой из конечного количества зафиксированных в контракте моментов времени. Получение аналитического решения в общем случае является затруднительным. Популярный подход к решению данной задачи состоит в применении методов Монте-Карло и средств имитационного моделирования. Успех применения этих методов существенно зависит от качества используемых генераторов случайных чисел и, как правило, требует больших вычислительных затрат. По этой причине разработка параллельных реализаций расчетных алгоритмов для решения задач такого типа оказывается особенно актуальной.

В работе [3] рассматривается рекурсивная реализация одного из наиболее популярных алгоритмов для решения подобных задач – Broadie–Glasserman Random Trees [2]. Авторами настоящей статьи разработана нерекурсивная последовательная реализация, производительность которой в среднем на 18% выше. Отличительной особенностью данной реализации является ориентация на распараллеливание на базе технологий OpenMP и MPI, позволяющее эффективно использовать имеющиеся вычислительные ресурсы. Указанные особенности позволят применить результаты работы при создании многопроцессорных/многоядерных аппаратно-программных комплексов и систем для оптимального принятия решений на финансовых рынках в условиях, когда выигрыш во времени решения задачи дает большое преимущество в конкурентной борьбе.

Постановка задачи

Рассмотрим *N*-мерный финансовый рынок, эволюционирующий в непрерывном времени [1]:

$$\begin{aligned} &dB_t = rB_t dt, \quad B_0 > 0; \\ &dS_t^i = S_t^i((\theta^i - \delta^i)dt + \sigma^i dW_t^i), \quad S_0^i > 0, \end{aligned} \tag{1}$$

где процессы $B = (B_t)_{t\geq 0}$ и $S^i = (S_t^i)_{t\geq 0}$, $i = 1, ..., N$, описывают поведение облигации B и i-ой акции в каждый момент времени $t \geq 0$ соответственно.

Поведение облигации B задается процентной ставкой r, изменение стоимости акции S^i определяется волатильностью σ^i и нормой возврата θ^i, δ^i – ставка дивиденда.

Случайные возмущения в цене акции S^i описываются *i*-ой компонентой векторного винеровского процесса $W^i = (W_t^i)_{t\geq 0} (W = W^1,...,W^N)$.

Предположим, что σ^i и $\mu^i = \theta^i - \delta^i, i = 1, ..., N$ являются функциями времени, а ковариационная матрица $COV = (\mathrm{cov}_{ij})_{i,j=1,...,N}$ отражает зависимости между ценами рисковых активов финансового рынка в каждый момент времени.

Рассмотрим одну из часто встречающихся разновидностей опциона max-call – опциона, для которого функция выплаты $h(t, S_t)$ имеет следующий вид:

$$h(t, S_t) = (\max_{i=1...N}(S_t^i) - P)^+, \qquad (2)$$

где положительная константа *P* определяется опционным контрактом, $x^+ = \max(x, 0)$.

Решим задачу вычисления стоимости опциона бермудского типа, который может быть предъявлен к исполнению в любой из заранее фиксированных моментов $t \in Time = \{t_0 = 0, t_1, t_2,..., t_d = T\}$, где момент времени T задает истечение срока действия опциона.

Описание алгоритма

Математически поиск рациональной цены опциона сводится к решению следующей стохастической оптимизационной задачи [2]: $Q = \max_{\tau \in Time} E[\frac{h(S_\tau)}{B_\tau}]$.

Пусть $Q(t,x) = \max(h(t,x), E\left\{\frac{Q(t+1, S_{t+1})B_t}{B_{t+1}} \middle| S_t = x\right\})$, $Q(T, S_T) = h(t, S_T)$. Тогда стоимость опциона бермудского типа определяется как $C = Q(0, S_0)$ [2].

Алгоритм Broadie–Glasserman Random Trees основан на методе Монте-Карло и динамическом программировании. Первым шагом его работы является построение дерева цен акций, каждый *i*-ый уровень которого соответствует моменту времени t_i. Второй шаг алгоритма состоит в обратном динамическом программировании – вычислении верхней и нижней оценок справедливой цены опциона методом их пересчета от листьев дерева к корню. После многократного повторения шагов 1 и 2 вычисляется доверительный интервал для параметра *C*. Подробно алгоритм и его теоретическое обоснование описаны в [2].

Программная реализация и подходы к оптимизации вычислений

Некоторые подходы к реализации алгоритма описаны в работе [3]. В настоящей статье мы ограничимся описанием основных усовершенствований, внесших наибольший вклад в оптимизацию вычислений по скорости, а также подходов к распараллеливанию алгоритма для многоядерных и кластерных систем.

Обход дерева цен акций. Главной особенностью реализации алгоритма является обход дерева цен акций в глубину (рис. 1). Корень дерева содержит *N*-мерный вектор цен акций в начальный момент времени, а каждая вершина дерева хранит *N*-мерный вектор цен акций в соответствующий момент времени t_i из множества *Time*. Таким образом, каждый уровень дерева соответствует некоторому моменту исполнения опциона t_i. Количество уровней соответствует количеству моментов исполнения опциона (*d*), а количество ветвей дерева (*K*) является параметром алгоритма и влияет на точность вычислений. Значения цен акций в узлах рассчитываются путем численного интегрирования системы стохастических дифференциальных уравнений (1). В отличие от первона-

чальных реализаций, основанных на обходе дерева цен акций в ширину, описываемая реализация использует обход дерева в глубину.

Для навигации по дереву используется стек, содержащий номера вершин в порядке посещения. Максимальная длина стека – *d*size_of(int)* байт – соответствует максимальной длине пути в дереве от корня до листа. Для адресации памяти вводится нумерация вершин дерева таким образом, что корень получает номер 0, а, начиная с *K*, номера присваиваются по порядку потомкам только тех узлов, которые лежат на верхней ветви (рис. 1).

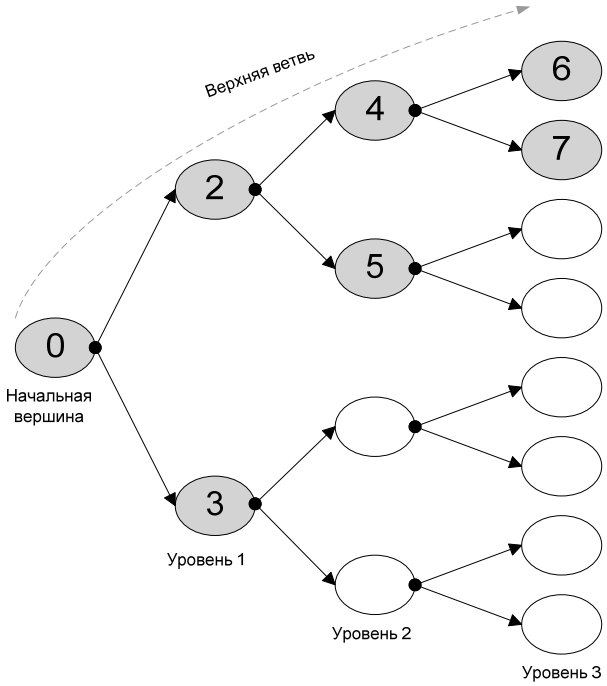


Рис. 1. Пример обхода дерева стоимостей акций для *N*=1, *K*=2, *d*=4

На каждом шаге алгоритма номер текущей вершины сравнивается с номером предыдущей. Рассмотрим логику работы алгоритма, исходя из того, что особым будет случай, когда вершина является листом дерева, а также случай, когда вершина является *K*-м потомком (т.е. данных достаточно для построения оценок для узла-предка).

1. Если номер предыдущего узла меньше номера текущего узла (направление движения по дереву от корня):
   если текущий узел является листом дерева (рис. 2, а):
   a. моделируем цену акций для текущего узла;
   b. оцениваем стоимость опциона;
   c. делаем шаг назад;
   если текущий узел является внутренним узлом дерева (рис. 2, б):
   a. моделируем цену акций для текущего узла;
   b. вычисляем номер узла для шага вперед и осуществляем переход.
3. Если номер предыдущего узла больше номера текущего узла (направление движения по дереву к корню), вычисляем, каким по счету потомком является текущий узел:
   a. если узел не является последним из потомков (не *K*-й), переходим к следующему потомку (рис. 2, в);
   b. если узел является последним из потомков (*K*-й): оцениваем стоимость опциона для предка текущего узла и делаем шаг назад (рис. 2, г).

Результатом работы алгоритма являются нижняя и верхняя оценки стоимости опциона.

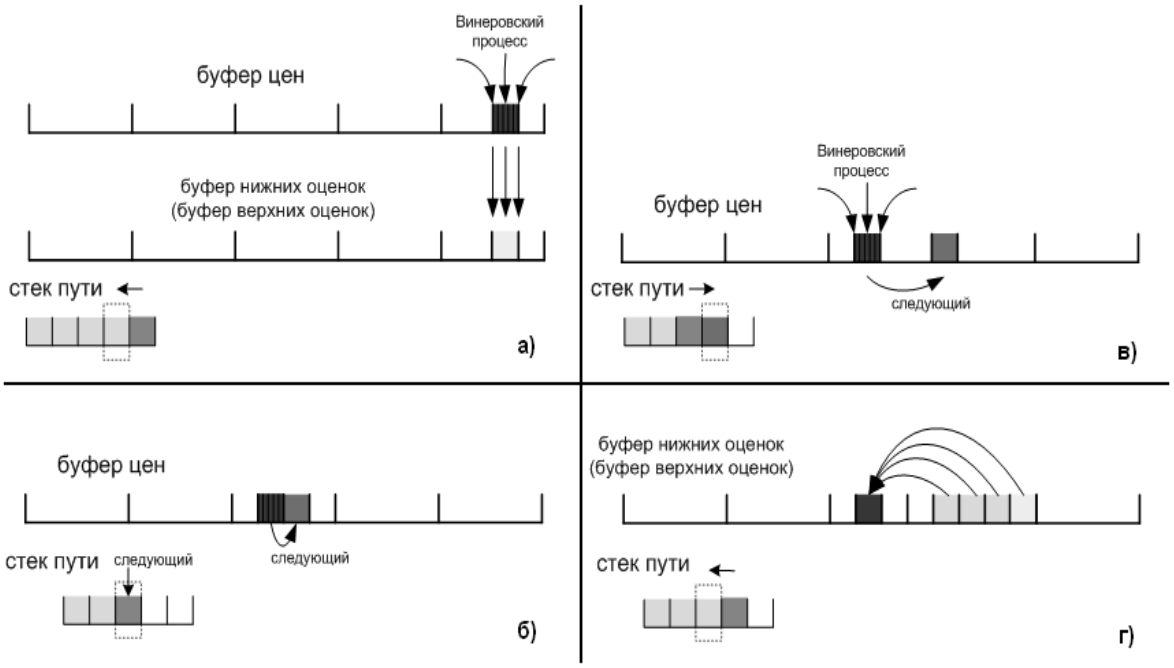


Рис. 2. Реализация обхода дерева

Нерекурсивная реализация. В работе [3] был реализован рекурсивный алгоритм обхода дерева цен акций. Так как задача обладает высокой вычислительной трудоемкостью, нерекурсивная реализация алгоритма целесообразна для уменьшения времени работы. Немаловажным преимуществом такой реализации является также то, что моделирование цены акций занимает меньше времени, так как, благодаря изменению порядка вычислений, появляется возможность векторизовать функцию вычисления экспонент применительно к массиву достаточно большого размера.

Важно заметить, что для хранения цен акций при использовании данного алгоритма необходимо *N*K*d*size_of(double)* байт, а для хранения оценок – *2*d*K*size_of(double)*. Алгоритм строит дерево акций в глубину и одновременно с этим подсчитывает оценки стоимости опциона.

Схема распараллеливания. В основе алгоритма лежат стохастические математические модели. Для их реализации используется метод Монте-Карло. Потому для получения результатов, наиболее приближенных к действительности, необходимо проведение как можно большего числа запусков с целью получения среднего значения. Исходя из этого, очевиден наиболее простой способ распараллеливания алгоритма – запуск функций вычисления цены опциона на каждом вычислительном устройстве независимо и сбор результатов на главном вычислительном устройстве.

Распараллеливание с помощью технологии MPI проводилось двумя способами – на уровне обхода дерева и на уровне независимых испытаний метода Монте-Карло. Основная идея распараллеливания (рис. 3) базируется на том факте, что генерация ветвей в дереве стоимостей акций происходит независимо друг от друга. Следующий за этим подсчет верхних и нижних оценок начинается с последнего уровня, далее проходит все уровни дерева до первого, используя на каждом шаге лишь результаты предыдущего шага (работает схема динамического программирования). Лишь на нулевом уровне (в корне дерева) происходит агрегация результатов, собранных на различных ветвях.

Схема распараллеливания выглядит следующим образом.

1. Головной процесс формирует первый уровень дерева.
2. Построение ветвей дерева равномерно распределяется между процессами (в том числе и головным) в соответствии с рис. 3.

3. По окончании построения дерева каждый процесс пересчитывает оценки от листьев дерева к корню вплоть до первого уровня.
4. Получив оценки первого уровня, процессы пересылают результат головному процессу.
5. Головной процесс агрегирует полученные результаты и вычисляет верхнюю и нижнюю оценки стоимости опциона.

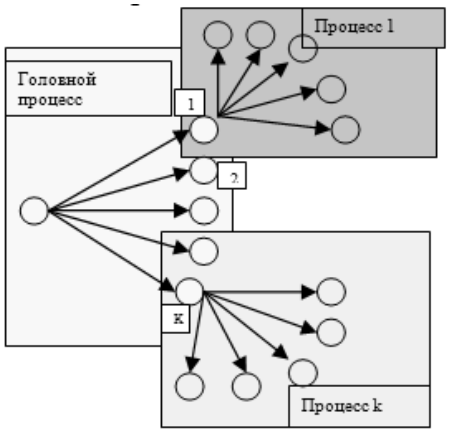


Рис. 3. Схема распараллеливания на уровне обхода дерева

Кроме того, также как и с помощью OpenMP, было проведено распараллеливание на уровне независимых испытаний метода Монте-Карло на основе MPI.

Результаты вычислительных экспериментов

Реализация описанного выше алгоритма проводилась на языке программирования C с использованием библиотек Intel® Math Kernel Library (MKL) 10.0.012, OpenMP 2.0, Microsoft MPI и оптимизирующего компилятора Intel® C/C++ Compiler 10.0.026 for Windows. Результаты вычислительных экспериментов получены на системе Intel® Core(TM) 2 Quad (Kentsfield, 4 core × 2.4 GHz, L2 Cache 2 × 4096 KB). Для экспериментов были выбраны тестовые значения параметров, перечисленные в табл. 1.

Таблица 1. Значения параметров

N	5
K	50
D	5, 6 в разных экспериментах
T (дни)	365
R	5%
S_0	(90; 100; 110; 120; 130)
P	100
Коэффициент корреляции	0,3
μ	(-0,05; -0,05; -0,05; -0,05; -0,05)
σ	(0,2; 0,2; 0,2; 0,2; 0,2)

Эксперимент демонстрирует (рис. 4) время работы алгоритма для 5 и 6 моментов исполнения опциона в сравнении с реализацией из [3]. Ниже приведены результаты распараллеливания для многоядерных/многопроцессорных систем с использованием OpenMP (табл. 2) и MPI (табл. 3).

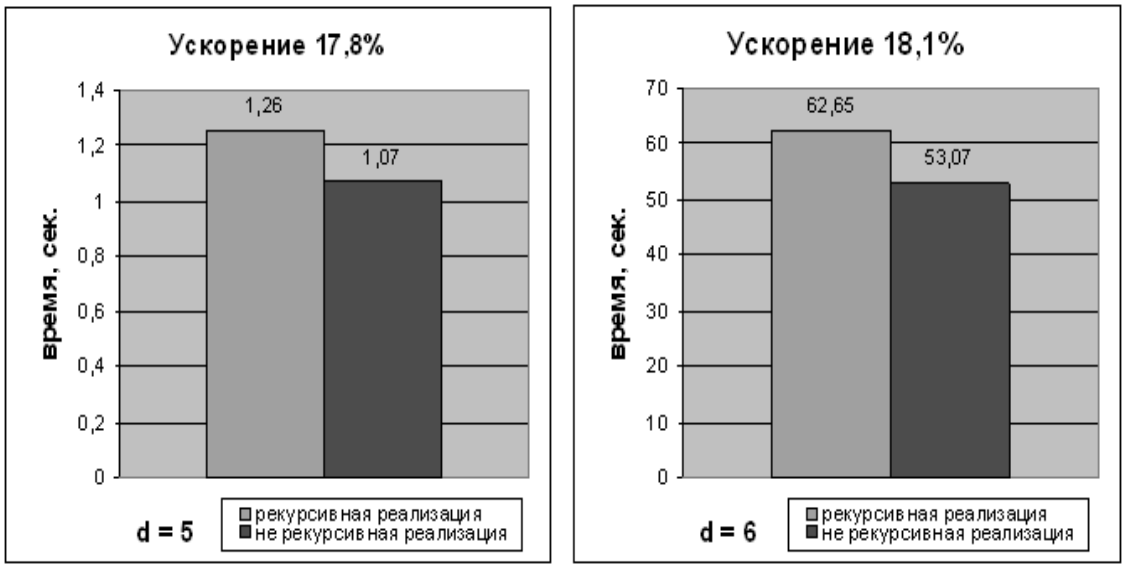


Рис. 4. Сравнение с рекурсивной реализацией

Таблица 2. Ускорение при распараллеливании с использованием OpenMP

Количество ядер	1	2	4
Ускорение	1	1,98	3,93

Таблица 3. Ускорение при распараллеливании с использованием MPI

Схема распараллеливания / кол-во процессоров	1	2	4
На уровне обхода дерева	1	1,97	3,96
На уровне независимых итераций	1	1,99	3,99

Результаты вычислительных экспериментов, основанных на использовании технологии MPI, получены на системе Intel® XEON 5150 Dual Core (4 core x 2.66 GHz, 4Gb DDR2) с межузловым соединением 1Gb Ethernet.

Как можно заметить, созданная программная реализация показывает практически линейное ускорение в зависимости от числа ядер. При использовании технологии MPI также были получены подобные результаты.

Заключение

Разработана эффективная реализация алгоритма Broadie–Glasserman Random Trees нахождения цены опциона Бермудского типа, позволившая получить прирост производительности в 15–18% относительно реализации [2]. Программа эффективно выполняется на многоядерных архитектурах, показывая практически линейное ускорение как на многоядерных, так и на многопроцессорных вычислительных системах.

Полученные результаты могут быть использованы в рамках создания аппаратно-программных комплексов для оптимального принятия решений при анализе финансовых рынков.

Литература

1. Ширяев А.Н. Основы стохастической финансовой математики. – В 2 тт. – М.: Фазис, 1998.

2. Broadie M., Glasserman P. Pricing American-style securities using simulation // Journal of Economic Dynamics and Control. – 1997.– Vol. 21. – P. 1323–1352.
3. Горбунова А.С. и др. Параллельная реализация одного алгоритма нахождения цены опционов бермудского типа // Материалы 5-го Международного научно-практического семинара «Высокопроизводительные параллельные вычисления на кластерных системах»; Под ред. проф. Р.Г. Стронгина. – Н. Новгород: Изд-во Нижегородского госуниверситета, 2005. – С. 60–67.

Горбунова Анна Сергеевна – Нижегородский государственный университет им. Н.И. Лобачевского, аспирант, anna_nn@mail.ru

Мееров Иосиф Борисович – Нижегородский государственный университет им. Н.И. Лобачевского, доцент, к.т.н., mib@uic.nnov.ru

Никонов Андрей Сергеевич – Нижегородский государственный университет им. Н.И. Лобачевского, студент, andrey.nikonov@itlab.unn.ru

Русаков Андрей Валентинович – Нижегородский государственный университет им. Н.И. Лобачевского, студент, andrey.rusakov@itlab.unn.ru

Шишков Александр Валерьевич – Нижегородский государственный университет им. Н.И. Лобачевского, студент, alekcac@gmail.com

УДК 519.6

СРЕДА МОДЕЛИРОВАНИЯ ДЛЯ РЕШЕНИЯ ПЕРКОЛЯЦИОННЫХ ЗАДАЧ

Е.Н. Головченко, Д.В. Петров

Процесс подготовки и запуска заданий на современных вычислительных системах коллективного пользования сопряжен с рядом трудностей. Для его упрощения авторами была создана среда моделирования. С ее использованием произведено моделирование заводнения месторождения нефти. Месторождение аппроксимировалось перколяционными решетками, содержащими 10^6–10^{10} узлов. Исследованы стационарный и нестационарный режимы воздействия на месторождение. Получены графики распределений нефти, воды, давления и вакансии по месторождению.

Ключевые слова: математическое моделирование, параллельное программирование, теория перколяции.

Введение

В настоящее время многопроцессорные вычислительные системы перестали быть редкостью, и пользователи зачастую имеют доступ к нескольким вычислительным ресурсам. Также в последнее время большое внимание уделяется развитию GRID-технологий, что дает многим пользователям возможность вычислять задачи в рамках распределенных систем. Однако процесс подготовки и запуска заданий для каждой системы имеет свои особенности, а операции, связанные с проектированием, не всегда интуитивно понятны обычному пользователю. При моделировании практических задач чаще всего требуется провести несколько десятков вычислений одной и той же задачи с разными или одними и теми же наборами входных данных. В результате пользователю приходится выполнять десятки раз повторяющиеся действия. Операции, производимые при решении задачи, могут быть нацелены на различные кластеры или хранилища данных, что требует запоминания и многократного ввода различных логинов и паролей. Для упрощения процесса взаимодействия пользователя с вычислительными системами была создана специализированная среда моделирования.

С использованием данной среды было проведено моделирование заводнения месторождения нефти, которое аппроксимировалось перколяционными решетками, содержащими 10^6–10^{10} узлов. Была исследована эффективность стационарного и нестационарного режимов воздействия на месторождение. Получены графики распределений нефти, воды, давления и вакансии по месторождению.

Созданная среда моделирования может использоваться при управлении большими вычислительными комплексами, а вместе с модулем, описывающим инвазионную перколяцию, – для проектирования очистных систем и сооружений, а также химических реакторов.

Среда моделирования

Основными операциями, требующимися для решения задач на вычислительных системах, являются передача файлов на вычислительную систему, компиляция заданий, их запуск, отмена и передача результатов на рабочую станцию пользователя. При запуске нескольких десятков заданий эти операции вызывают такие трудности, как многократный набор пароля при копировании файлов, необходимость ожидания окончания выполнения предыдущей команды и т.д. Входные и выходные данные могут находиться в местах, отличных от места запуска задания, в результате чего приходится проводить авторизацию с различными логинами и паролями. Среда моделирования, изображенная на рис. 1, позволяет упростить процесс взаимодействия пользователя с вычислительными системами.

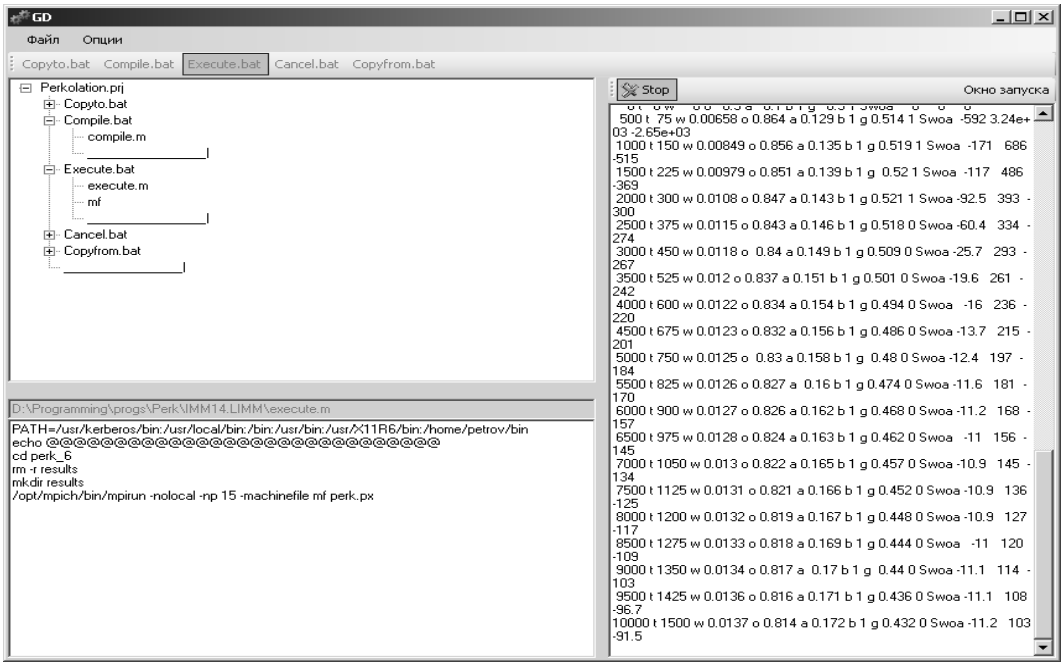


Рис. 1. Среда моделирования

Интерфейс среды состоит из панели инструментов, автоматически формируемой из дерева заданий, находящегося в одном из окон, окна редактирования файлов заданий и окна вывода информации о запуске заданий. Логин и пароль пользователя вводятся один раз для каждой операции, и в дальнейшем среда сама отслеживает выбор логина и пароля и передает их при каждом входе в систему. Использование файлов заданий позволяет редактировать сразу несколько команд, а после запуска ожидать выполнения всех операций.

После настройки среды на конкретную систему дальнейшее ее использование предполагает работу только с панелью инструментов, что позволяет скрыть особенности взаимодействия с конкретной вычислительной системой.

Модель инвазионной перколяции

При решении задачи заводнения месторождения нефти использовалась трехмерная динамическая перколяционная модель [1]. Подразумевается инвазионная перколяция, когда одна жидкость вытесняет другую [2]. Согласно теории перколяции, пласт аппроксимируется кубической решеткой, узлы которой отождествляются с порами пласта, а ребра – с капиллярными каналами. Предполагается, что компоненты (нефть, вода и вакансия) находятся только в поровом пространстве, а доля открытых для протекания каналов равна p ($0 \le p \le 1$).

Для численного моделирования использовалась следующая аппроксимация:

$$\frac{\overline{G}_m - G_m}{\tau} = -\sum_{r \in \gamma_m} k_{r,m}\left(G_m - G_r\right), \tag{1}$$

$$\frac{\overline{F}_m^{\alpha} - F_m^{\alpha}}{\tau} = -\sum_{r \in \gamma_m} k_{r,m}\left(\Psi_{r,m} F_r^{\alpha} F_m^{0} - \Psi_{m,r} F_r^{0} F_m^{\alpha}\right), \tag{2}$$

$$\Psi_{r,m} = \frac{1}{2}\left[\left|G_r - G_m\right| - \left(G_r - G_m\right)\right], \tag{3}$$

где m – индекс узла, $G, F^{\alpha}, \overline{G}, \overline{F}^{\alpha}$ – значения сеточных функций давления и насыщенности флюидов в моменты времени t и $t+\tau$, τ – шаг по времени, $k_{r,m}$ – проводимость ребра, γ_m – множество индексов узлов, смежных с узлом m, различные α соответствуют различным компонентам.

По пласту расставлены нагнетательные и добывающие скважины. Узлы, соответствующие нагнетательным скважинам, заполнены водой под некоторым давлением. Узлы, соответствующие добывающим скважинам, заполнены вакансией. В остальных узлах доля нефти – 0,9, доля вакансии – 0,1.

Моделирование заводнения месторождения нефти

Различают стационарный и нестационарный режимы воздействия на месторождение. При стационарном воздействии подача воды на нагнетательные скважины происходит непрерывно, при нестационарном – с некоторым периодом. Отношение длительности периода воздействия к периоду нагнетания называется *скважностью воздействия*.

Для сравнения эффективности стационарного и нестационарного режимов было проведено исследование зависимости заводнениния месторождения нефти от доли открытых ребер и скважности воздействия. Для аппроксимации месторождения использовалась перколяционная решетка размером 100×100×100. Нагнетательные и добывающие скважины были расставлены равномерно в шахматном порядке (сетка из 4×4 скважин). На основе полученных данных построены поверхности зависимости доли остаточной нефти от доли открытых ребер и скважности воздействия. На рис. 2 представлена поверхность для времени моделирования, равного 50000 единиц, и ее фрагмент. Доля открытых ребер менялась от 0,1 до 1,0, скважность – от 1 до 6.

На начальном этапе стационарное воздействие дает больший эффект, чем нестационарное, но через 50000 единиц при долях открытых ребер от 0,4 до 0,7 включительно происходит снижение доли остаточной нефти при скважности 3. Поверхность на рис. 2 показывает выигрыш нестационарного воздействия над стационарным, а ее

фрагмент для долей открытых ребер от 0,4 до 0,5 и скважностей воздействия от 2 до 6 отражает выигрыш нестационарного режима при скважности 3.

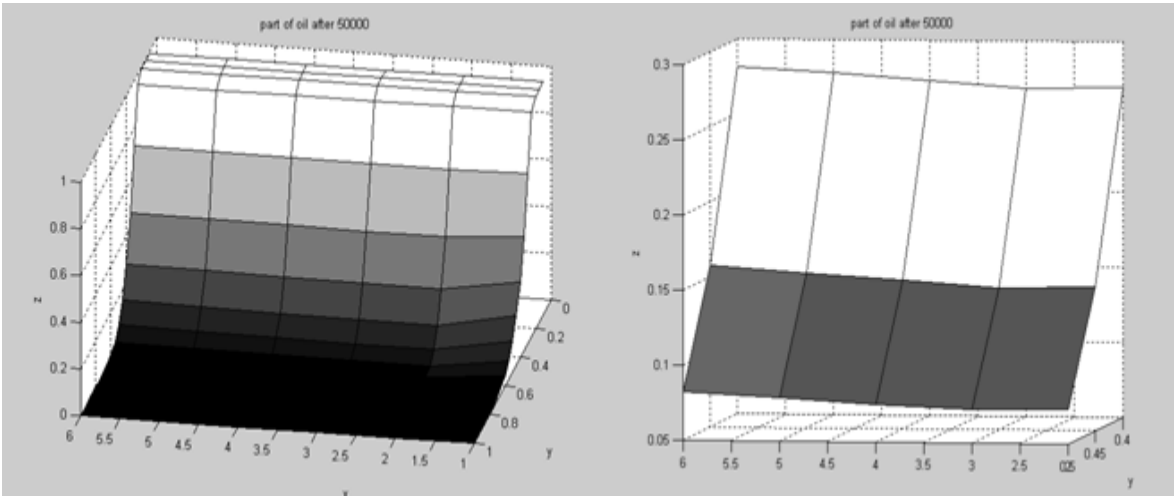


Рис. 2. Поверхность зависимости доли остаточной нефти от доли открытых ребер и скважности воздействия для времени 50000 единиц и ее фрагмент

На рис. 3 изображены кривые зависимости доли остаточной нефти от скважности воздействия для доли открытых ребер 0,4 и времен 40000, 50000 и 100000 единиц.

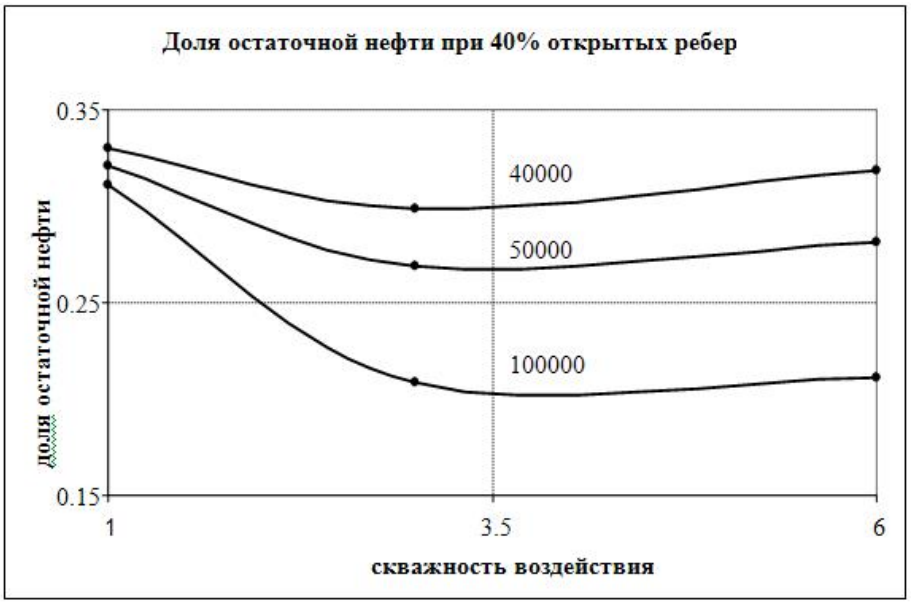


Рис. 3. Зависимости доли остаточной нефти от скважности воздействия при различных временах

Через 100000 единиц, по сравнению с 40000 и 50000, увеличивается выигрыш от нестационарного воздействия (увеличивается разница между скважностями 3 и 1 и между скважностями 6 и 1). С другой стороны, выигрыш скважности 3 над скважностью 6 постепенно уменьшается. Это позволяет предположить, что на больших временах самой оптимальной станет уже скважность 6, а не 3.

Распределения флюидов и давления

В ходе исследования были получены распределения флюидов и давления на различных временах. Рис. 4 иллюстрирует изменение распределения нефти. Как видно из

рисунка, нефть постепенно вымывается из пласта: через 15000 единиц остались только небольшие участки между скважинами.

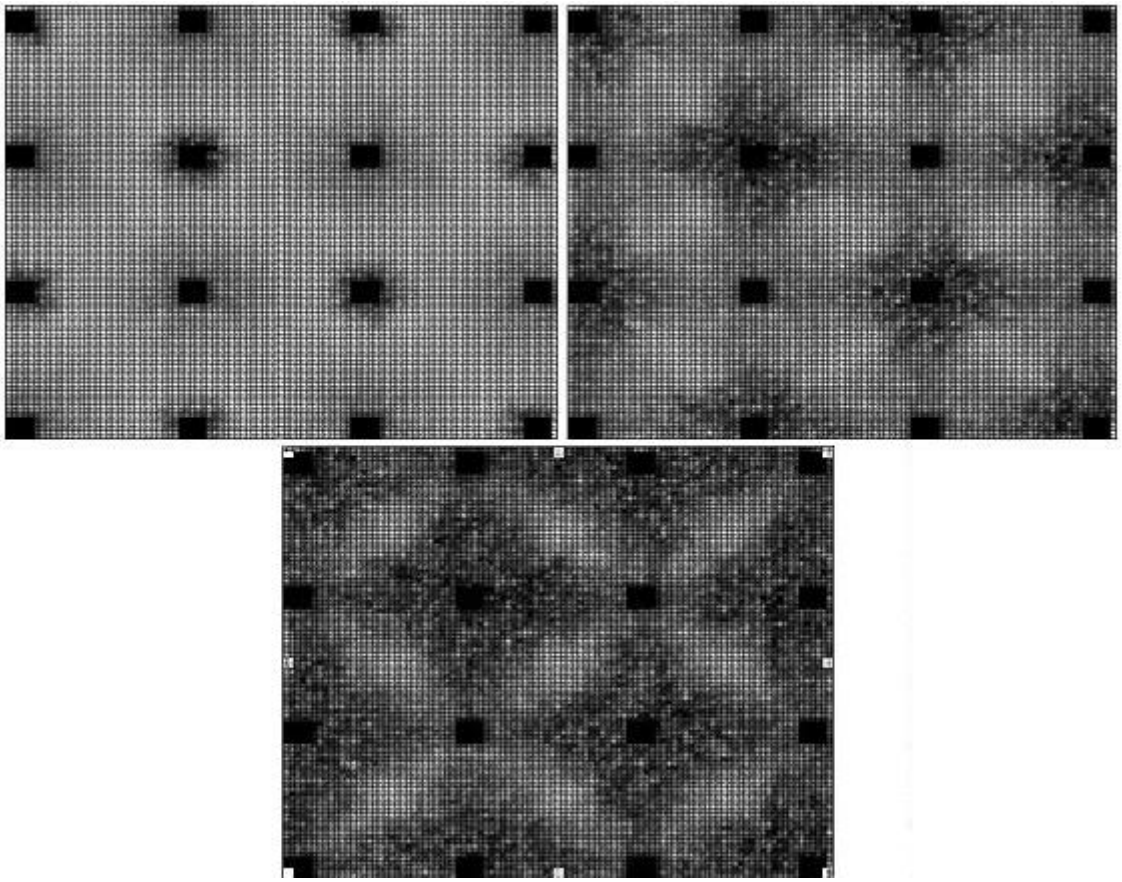


Рис. 4. Распределение нефти через 3000, 9000 и 15000 единиц

Моделирование на больших перколяционных решетках

Моделирование заводнения месторождения до 50000 единиц на перколяционной решетке размером 10^6 на 110 процессорах занимало около 90 минут. Моделирование на перколяционной решетке с 10^9 вершин до 4000 единиц на 500 процессорах заняло около 500 мин., решетки с 10^{10} вершин до 600 единиц на 500 процессорах – около 700 мин.

Заключение

Для упрощения процесса взаимодействия пользователя с вычислительными системами была создана среда моделирования. При помощи данной среды было проведено моделирование заводнения месторождения нефти и сравнение стационарного и нестационарного режимов воздействия на месторождение. Получены графики распределений нефти, воды, давления и вакансии по месторождению. Месторождение аппроксимировалось сетками до 10^{10} вершин включительно.

Созданная среда моделирования может использоваться при управлении большими вычислительными комплексами, а вместе с модулем, описывающим инвазионную перколяцию, – для проектирования очистных систем и сооружений, химических реакторов.

Работа выполнялась при поддержке РФФИ (грант 07-01-12079-офи).

Литература

1. Тарасевич Ю.Ю. Перколяция. Теория, приложения, алгоритмы: Учебное пособие. – М.: Едиториал УРСС, 2002.– 112 с.
2. Wilkinson D., Wlemsen J.F. Invasion percolation: A new form of percolation theory // J. Phys. – 1983. – A16.– P. 365–376.

Головченко Евдокия Николаевна – Институт математического моделирования РАН, м.н.с., ge03@imamod.ru

Петров Дмитрий Владиславович – Московский государственный технологический университет «СТАНКИН», аспирант, bito@mail.ru

УДК 004.021

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ МОДЕЛИРОВАНИЯ КОНКУРЕНТНОГО РЫНКА НА КЛАСТЕРНЫХ СИСТЕМАХ

М.Ю. Нестеренко, Д.В. Леонов, Е.В. Болгова, А.С. Кириллов

В статье рассматривается подход к решению вычислительно-сложной задачи построения оптимальной коалиции и распределения выигрыша в кооперативных играх для n игроков. Представлена схема параллельного алгоритма решения поставленной проблемы и проанализированы полученные результаты работы программной реализации алгоритма.

Ключевые слова: кооперативные игры, параллельное программирование.

Введение

Теория игр, которая начала развиваться совсем недавно, может описать практически любую ситуацию. В настоящее время разработано большое количество видов игровых ситуаций, которые нашли свое применение в таких областях, как приборостроение, математика, разработка программного обеспечения, экономика, социология и др.

Среди направлений теории игр особое место занимают кооперативные игры, например, разработка компьютерных систем, оптимальных стратегий безопасности систем и др. В них игроки имеют возможность принимать совместные решения и, в некоторых случаях, перераспределять выигрыши между собой; под выигрышем может пониматься эффективность, надежность и т.д. В кооперативной игре задаются возможности и предпочтения различных групп игроков (коалиций), для которых строятся оптимальные стратегии и задаются распределения суммарных выигрышей между ними. Особого внимания заслуживают задачи моделирования конкурентных рынков. Примером подобной задачи может быть моделирование конкурентного рынка IT, где игроками являются производители компьютерной техники и программного обеспечения, имеющие возможность объединяться в коалиции, конкурирующие за соответствующие ниши рынка.

Существуют различные способы решения кооперативных игр, например, решение на основе функции полезности Неймана–Моргенштерна, решение игры в виде C-ядра и в стратегиях угроз. Однако эти подходы не определяют оптимальных стратегий игроков внутри коалиций, а решение игры в виде C-ядра дает целое множество допустимых решений, не определяя оптимального. Попытки построения количественного подхода к решению кооперативной игры предприняты И.Д. Протасовым [2]. В работе предлагается задавать кооперативные игры в виде совокупности биматричных игр. Для задания игры n игроков требуется $2^n - 1$ биматричных игр, каждая из которых должна быть решена, что обусловливает необходимость применения высокопроизводительных технологий.

Таким образом, задачами исследования является разработка математического обеспечения решения кооперативной игры на кластере и моделирование с помощью разработанного программного обеспечения конкурентного рынка.

Модель конкурентного рынка в виде кооперативной игры

Постановка задачи моделирования конкурентного рынка подробно рассмотрена в [2, 3], но проблема ресурсоемкости не затронута, является актуальной и может быть решена с использованием высокопроизводительных технологий. В общем виде для двух игроков модель конкурентного рынка рассмотрена В.А. Колемаевым в [4]. При увеличении количества игроков до трех требуется задание уже не двух, а шести матриц следующего вида [1]:

$$A_{X/Y} = \begin{matrix} \\ K \\ Cm \end{matrix} \begin{matrix} \begin{matrix} K & \quad Cm \end{matrix} \\ \begin{pmatrix} \frac{bX_0^2}{16} - d & \frac{bX_0^2}{36} - d \\ \frac{bX_0^2}{12} - d & \frac{b3X_0^2}{100} - d \end{pmatrix} \end{matrix}, \quad B_{Y/X} = \begin{matrix} \\ K \\ Cm \end{matrix} \begin{matrix} \begin{matrix} K & \quad Cm \end{matrix} \\ \begin{pmatrix} \frac{bX_0^2}{16} - d & \frac{bX_0^2}{12} - d \\ \frac{bX_0^2}{36} - d & \frac{b3X_0^2}{100} - d \end{pmatrix} \end{matrix},$$

где матрица $A_{X/Y}$ ($B_{Y/X}$) – матрица выигрышей игрока *X* в игре с игроком *Y* (*Y* в игре с игроком *X*) в том случае, если игроки могут действовать по двум стратегиям – Курно и Стакельберга, $X_0 = (a–c)/b$ – величина общего выпуска, при котором прибыль каждой фирмы отрицательна и равна (-*d*) (*c* – предельные издержки, *b* – падение цены при увеличении на единицу выпуска, *d* – постоянные издержки). Выражение для прибыли конкурирующих фирм принимает вид

$$\Pi_i(X_1, X_2, X_3) = bX_i(X_0 - (X_1 + X_2 + X_3)) - d.$$

При этом модель значительно упрощена тем, что издержки фирм и цена на продукцию являются линейными функциями. Таким образом, игра *n* игроков может быть представлена C_n^2 биматричных игр, что уже является вычислительно емким показателем.

Обозначим через *N* множество всех игроков, $N = \{1, 2, ..., n\}$, а через *K* – любое его подмножество. Пусть игроки из *K* договариваются между собой о совместных действиях и, таким образом, образуют одну коалицию. Образовав коалицию, множество игроков *K* действует как один игрок против остальных игроков, и выигрыш этой коалиции зависит от применяемых каждым из *n* игроков стратегий.

Обозначим функцией *v* характеристическую функцию, ставящую в соответствие каждой коалиции *K* наибольший уверенно получаемый ею выигрыш *v*(*K*). Всего характеристических функций может быть 2^n, причем коалиция с номером 0 является пустой коалицией, и ее выигрыш всегда равен нулю.

Обозначим через $X=\{x_1, x_2, \ldots, x_n\}$ дележ, являющийся распределением выигрышей игроков.

В бескоалиционных играх исход формируется в результате действий игроков, которые в этой ситуации получают свои выигрыши. Исходом в кооперативной игре является дележ, возникающий не как следствие действий игроков, а как результат их соглашений. Поэтому в кооперативных играх сравниваются не ситуации, как это имеет место в бескоалиционных играх, а дележи, и сравнение это носит более сложный характер.

Мы предлагаем следующий алгоритм решения кооперативной игры: коалиционная игра задается в виде $C_n^2 = n \cdot (n-1)$ матриц размера $k \times k$, где *n* – количество игроков, *k* – количество стратегий у игроков, имеющих вид

$$A_{xy}=\begin{pmatrix} a_{11} & a_{12} & ... & a_{1k} \\ a_{21} & a_{22} & ... & a_{2k} \\ ... & ... & ... & ... \\ a_{k1} & a_{k2} & ... & a_{kk} \end{pmatrix},$$

где $a_{ij}(i, j = \overline{1,k})$ – выигрыши игрока *x* в игре с игроком *y*. Элементы матриц вычисляются по аналогии с игрой двух игроков, более подробно подход к вычислению выигрышей игроков в зависимости от выбора ими той или иной стратегии приведен в [4]. Стратегии коалиции определяются выбором игроков-участников коалиций. Таким образом, у коалиции из *m* игроков будет k^m чистых стратегий, где k – число выборов у каждого игрока.

При построении матрицы выигрышей коалиции следует исходить из наихудших условий, т.е. предположить, что остальные участники игры образовали коалицию с намерениями минимизировать выигрыши противостоящей коалиции. Количество стратегий у коалиции определяется числом участников коалиции и количеством выборов у каждого участника. Пусть коалиция состоит из *m* игроков и каждый имеет по *k* выборов, тогда матрица коалиции будет иметь размер $k^m \times k^{n-m}$, где *n* – общее число игроков.

Элементы матрицы выигрышей коалиции вычисляются как сумма выигрышей каждого игрока-участника коалиции у всех игроков (в зависимости от условий игры возможно включение в суммарный выигрыш коалиции и выигрыш между игроками внутри коалиции).

Каждая игра решается сведением к задаче линейного программирования, которая, в свою очередь, решается с помощью симплекс-метода Дж. Данцига. Сведение к задаче линейного программирования осуществляется следующим образом: если первый игрок придерживается своей оптимальной смешанной стратегии *X*, а второй игрок придерживается свой чистой *j*-ой стратегии, то выполняются следующие неравенства:

$$\begin{cases} x_1 \cdot a_{11} + ... + x_m \cdot a_{m1} \geq v, \\ ... \\ x_1 \cdot a_{1n} + ... + x_m \cdot a_{mn} \geq v, \end{cases}$$

где $a_{ij}(i, j = \overline{1,k})$ – выигрыши игрока *x* в игре с игроком *y*, x_i – вероятность использования первым игроком *i*-ой стратегии, $\sum_{i=1}^{m} x_i = 1$, *v* – цена игры, максимальный гарантированный выигрыш, который получит первый игрок при выборе оптимальной смешанной стратегии, $v > 0$.

Если в неравенствах поделить обе части на *v* и обозначить $\frac{x_i}{v}$ за t_i, так как x_i – вероятность использования *i*-ой стратегии, то получим следующее:

$$F = \sum_{i=1}^{m} t_i = \sum_{i=1}^{m} \frac{x_i}{v} = \frac{1}{v} \to \min,$$

$$\begin{cases} t_1 \cdot a_{11} + ... + t_m \cdot a_{m1} \geq 1, \\ ... \\ t_1 \cdot a_{1n} + ... + t_m \cdot a_{mn} \geq 1, \end{cases}$$

что, в свою очередь, является задачей линейного программирования и решается с помощью симплекс-метода Дж. Данцига.

Для нахождения дележа используется принцип оптимальности распределения выигрыша между игроками в задачах теории кооперативных игр, называемый вектором Шепли, который вычисляется по следующей формуле:

$$\Phi(v)_i = \sum_{i \in N} \frac{(k-1)!(n-k)!}{n!} \cdot \left[v(N) - v(N \setminus i) \right],$$

где n – количество игроков, k – количество участников коалиции N.

Анализ подходов к распараллеливанию решения кооперативной игры

Для решения кооперативных игр применяется сведение множества биматричных игр к множеству стратегических игр и решение каждой из них симплекс-методом для нахождения выигрыша каждой коалиции. Как было отмечено раньше, всего коалиций может быть 2^n, где n – количество игроков, следовательно, необходимо решить 2^n стратегических игр симплекс-методом Дж. Данцига, который обладает экспоненциальной сложностью [7]. При достаточно больших n это занимает длительное время. Кроме того, большую вычислительную сложность имеет процесс формирования матриц, которыми задаются коалиции. Трудоемкость этого процесса составляет $O(n^2)$. Так как матрицы, описывающие коалиции не зависят друг от друга, а зависят только от матриц, задающих саму игру, то генерация матриц может осуществляться параллельно.

Исходя из сказанного, можно сделать вывод, что формирование матриц, характеризующих коалиции, и их расчет симплекс-методом являются независимыми участками кода, которые можно выполнять параллельно.

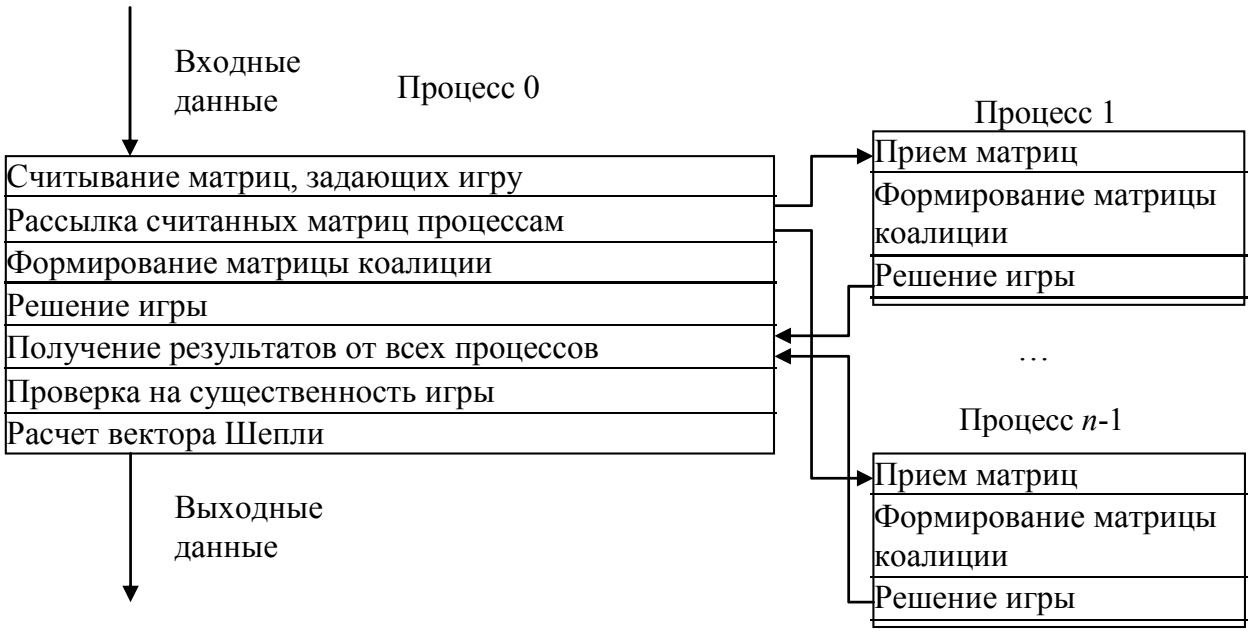


Рис. 1. Схема работы параллельного алгоритма

Суть разработанного параллельного алгоритма (рис. 1) состоит в следующем.

Нулевым процессом выполняется считывание матриц, задающих игру, после чего они рассылаются остальным незанятым работающим процессам, которые, в свою очередь, формируют матрицы, характеризующие коалиции. Номера игроков, входящих в коалицию, определяются по соответствию позиций единичных элементов двоичного представления порядкового номера формируемой матрицы. После этого полученная матрица сводится к задаче линейного программирования и решается симплекс-методом. Результат отсылается нулевому процессу, который проверяет игру на сущест-

венность, и, если игра существенна, то вычисляется дележ по Шепли. Затем все результаты выводятся в выходной файл.

Алгоритм реализован на языке C с использованием библиотеки MPI (Message Passing Interface). В приведенном выше алгоритме доля последовательного участка подчиняется закону $f=\frac{k}{k^{n-1}+k}$, где n – количество игроков, k – количество стратегий у игроков, из которого видно, что при возрастании n или k доля последовательных вычислений стремится к нулю.

Тестирование программы производилось на кластере Оренбургского государственного университета, построенного на базе двухпроцессорных Xeon 3 Ггц под управлением операционной системы SuSe Linux. Тестирование производилось для 7 игроков, и у каждого игрока имелось по 4 стратегии поведения.

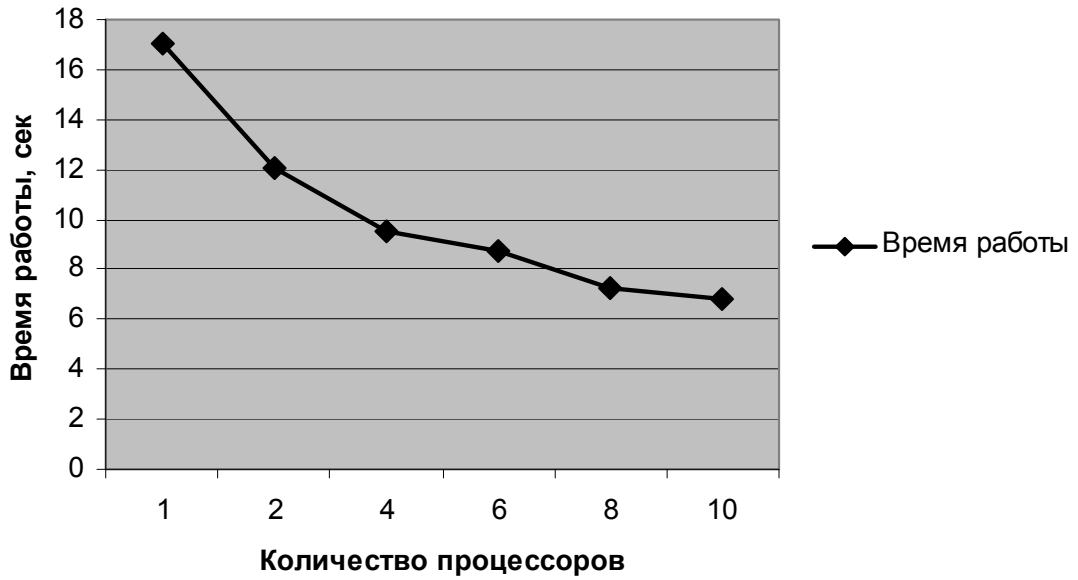


Рис. 2. График зависимости времени работы от количества процессоров

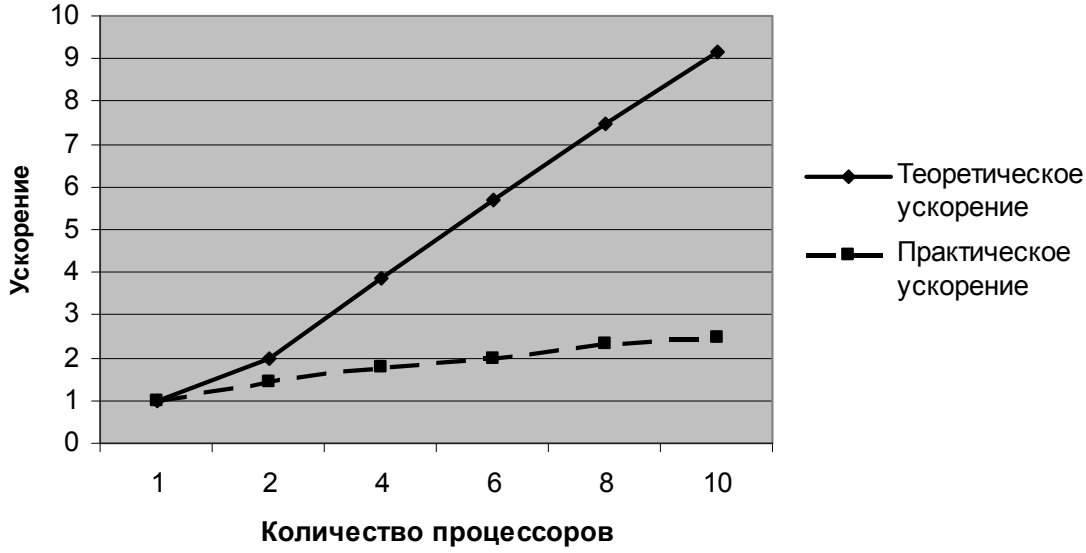


Рис. 3. График зависимости теоретического и практического ускорения от количества процессоров

Зависимость времени работы программы от количества процессоров представлена на рис. 2. Для наглядного представления построен график зависимости теоретической оценки

ускорения по закону Амдала и ускорения работы программы, рассчитанного по результатам тестирования в зависимости от числа процессоров, представленный на рис. 3.

Заключение

В результате анализа подходов к решению кооперативных игр выявлена необходимость построения количественного метода их решения. В статье предложен метод построения характеристической функции на основе задания кооперативной игры множеством биматричных игр. Выявлена высокая вычислительная сложность данного метода, что обусловливает необходимость применения высокопроизводительных технологий. В результате распараллеливания алгоритма, реализующего метод, удалось достичь ускорения около 2,5 раз. Большое разногласие между теоретическим и практическим ускорением указывает на перспективность данной оптимизации параллельного алгоритма.

Литература

1. Нестеренко М.Ю. Игровое моделирование рынка одного товара // Материалы региональной научно–практической конференции. – Оренбург, 2004.
2. Протасов И.Д. Теория игр и исследование операций: Учебное пособие. – М.: Гелиос АРВ, 2003. – 368 с.
3. Крушевский А.В. Теория игр. – Киев: Вища школа, 1977. – 216 с.
4. Колемаев В.А., Калинина В.Н. Теория вероятностей и математическая статистика: Учеб. для вузов. – 2–е изд., перераб. и доп. – М.: Юнити, 2003. – 352 с.
5. Антонов А.С. Параллельное программирование с использованием технологии MPI: Учебное пособие. – М.: Изд–во МГУ, 2004. – 71 с.
6. Антонов А.С. Введение в параллельные вычисления: Методическое пособие. – М.: Изд–во МГУ, 2002. – 69 с.
7. Данциг Дж. Линейное программирование, его применения и обобщения. – М.: Прогресс, 1966.– 600 с.

Нестеренко Максим Юрьевич – Оренбургский государственный университет, к.т.н., nmu@mail.osu.ru

Леонов Денис Владимирович – Оренбургский государственный университет, м.н.с., d_leonov@list.ru

Болгова Екатерина Владимировна – Оренбургский государственный университет, студент, koten-ka@inbox.ru

Кириллов Алексей Сергеевич – Оренбургский государственный университет, студент, kas-56@yandex.ru

3 ПЕРСПЕКТИВНЫЕ ИССЛЕДОВАНИЯ В ОБЛАСТИ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ И КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ. КРАТКИЕ НАУЧНЫЕ СООБЩЕНИЯ

УДК 519.642

МОДЕЛИРОВАНИЕ КРУПНОМАСШТАБНЫХ ВИХРЕВЫХ СТРУКТУР НА ВЫЧИСЛИТЕЛЬНЫХ КОМПЛЕКСАХ ПАРАЛЛЕЛЬНОЙ АРХИТЕКТУРЫ

А.В. Бабаков, П.А. Новиков

В работе исследуются вопросы моделирования нестационарного вихревого следа за спускаемыми космическими аппаратами на многопроцессорных вычислительных системах.
Ключевые слова: газодинамика, спускаемый космический аппарат, параллельные алгоритмы.

Введение

Численное моделирование вихревой структуры ближнего следа связано с определенными трудностями, в первую очередь, со сложной структурой потока, а также с отсутствием адекватной модели турбулентности. С вычислительной точки зрения математическое моделирование аэродинамики пространственных объектов характеризуется сложностью построения и большим объемом вычислительной области, размерность которой достигает нескольких миллионов ячеек разностной сетки.

Основной целью исследования являлось изучение на вычислительных комплексах параллельной архитектуры вопросов зарождения и динамики крупномасштабных вихревых структур в ближнем следе, изучение их влияния на аэродинамические характеристики аппаратов.

Методика и результаты численного моделирования

В качестве модели среды выбрана полная нестационарная модель невязкого сжимаемого газа – модель Эйлера. Основаниями для использования данной модели являются выводы [1], где показывается, что мелкомасштабная турбулентность не оказывает принципиального влияния на крупномасштабные вихревые структуры, которые, в основном, являются следствием волновых процессов, не связанных непосредственно с величиной вязкости. Математическая модель выписывается в виде конечно-разностного аналога законов сохранения, записанных в интегральной форме для конечного объема вычислительной ячейки для каждой аддитивной характеристики среды. Численное моделирование течения осуществлялось на основе нестационарного варианта консервативного метода потоков [2], являющегося методом сквозного счета и позволяющего рассчитывать параметры и структуру потока во всей области интегрирования единым образом без выделения особенностей.

Для численного моделирования использовались параллельные алгоритмы, реализованные на многопроцессорных вычислительных системах кластерной архитектуры. Использовался суперкомпьютер МВС-15000ВМ с пиковой производительностью 8,13 TFLOPS и процессорами на решающем поле 3 ГГц.

Моделирование течений в широком диапазоне скоростей определяет разные виды и масштабы вычислительных сеток, что обусловливает различное геометрическое разбиение расчетной области для более эффективного распределения работы между вычислительными узлами. Эффективность расчета на 40–50 процессорных узлах (в зависимости от объема

вычислительной сетки) для варианта программы с объемом области около одного миллиона ячеек достигала в среднем 0,06 с для расчета одного временного слоя.

Анализ результатов численного исследования для околозвуковых режимов показывает, что нестационарные процессы происходят, в основном, в донной области и связаны с формированием и сходом вихрей с обтекаемой поверхности. Диффузия вихревых структур вниз по потоку определяет нестационарность течения, распространяющуюся в ближнем следе, что является причиной осцилляции газодинамических параметров. На рис. 1 в качестве примера показана структура потока в следе за двумя формами спускаемых аппаратов, визуализированная по методике [3].

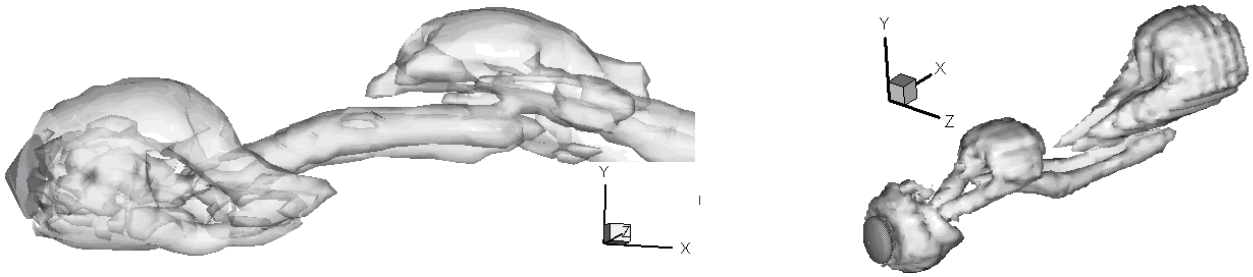


Рис 1. Структура потока в следе за двумя формами спускаемых аппаратов

Заключение

Реализация параллельных алгоритмов метода потока на вычислительных комплексах кластерной архитектуры позволяет эффективно осуществлять математическое моделирование пространственных течений около моделей космических аппаратов сложной формы, проводить исследования нестационарного вихревого следа и учитывать влияние нестационарности на аэродинамические характеристики в широком диапазоне скоростей.

Литература

1. Belotserkovskii O.M. Turbulence and instabilities. – MIPT, 1999 .– 460 p.
2. Бабаков А.В. Численное моделирование задач внешней аэродинамики в широком диапазоне скоростей на основе консервативных разностных схем // Научная диссертация. – М. Наука, 1992.
3. Hussain A.K. Coherent structure and turbulence // J. Fluid Mech. – 1986. – Vol. 173. – P. 303–356.

Бабаков Александр Владимирович — Институт автоматизации проектирования РАН, зав. отделом, д.ф.-м.н., с.н.с., babakov@icad.org.ru

Новиков Павел Александрович — Институт автоматизации проектирования РАН, м.н.с., pavel_novikov@mail.ru

УДК 519.63

АДАПТИВНЫЕ ДЕКАРТОВЫ СЕТКИ

А.А. Сухинов

Рассматриваются методы реализации локально-перестраиваемых декартовых сеток, динамически адаптирующихся к получаемому решению. Описаны структура сетки, критерий адаптации, способ параллельной реализации.
Ключевые слова: адаптивные сетки, высокопроизводительные вычисления.

Введение

Численное решение задач математической физики обычно требует наличия сетки, покрывающей расчетную область. Точность используемых при этом численных методов (конечных элементов/объемов/разностей) может быть повышена с помощью схем высокого порядка аппроксимации (которые зачастую приводят к появлению осцилляций) или путем уменьшения размеров элементов сетки. Уменьшение размеров ячеек во всей области значительно увеличивает вычислительную сложность задачи. Адаптивные сетки – метод, позволяющий сгустить сеточные элементы в областях, где они наиболее необходимы.

Одно из важных приложений адаптивных сеток – математическое моделирование процессов в полупроводниковых приборах [1], где около границ раздела материалов возникают большие градиенты физических величин. Адаптивные сетки также могут быть применены для приближенного представления пространственно-распределенных величин с целью экономии памяти или уменьшения количества данных, передаваемых по каналам связи.

Изложение ведется на примере двумерных сеток, однако разработанные алгоритмы могут быть легко обобщены на любое число измерений.

Структура сетки

Для формализации задачи вводятся следующие предположения.

1. Вначале сетка состоит из единственной прямоугольной *корневой ячейки* C_0.
2. Каждая ячейка $C_i (0 \le i < N)$ хранит величину V_i (скалярную или векторную), описывающую среднее значение вычисляемой функции в пределах C_i.
3. Ячейка может быть разбита (разделена) на четыре дочерние ячейки одинакового размера. Существуют также подходы, при которых ячейка разбивается пополам вдоль одной из координатных осей [2].
4. Объединены могут быть лишь те ячейки, которые составляли одну ячейку.
5. Ячейка не должна иметь в своей окрестности ячеек, которые отличаются от нее по размерам более чем в два раза (это более сильное ограничение, чем в других работах [3]). *Окрестность ячейки* – это прямоугольник, по размерам в три раза больший ячейки, центр которого совпадает с центром ячейки.

Двумерная адаптивная сетка может храниться в виде четверичного дерева [3], при этом для вычислений используются только листовые ячейки. *Разделение* ячейки означает создание четырех ее потомков без удаления самой ячейки; *объединение* ячейки означает удаление всех ее потомков.

Нахождение соседних ячеек в дереве упрощается, если в каждой ячейке хранить дополнительную информацию, называемую *адресом ячейки*. В одномерном случае адрес – это последовательность бит. У корневой ячейки адрес пустой. Адрес левого потомка получается из адреса родителя добавлением нулевого младшего бита; адрес правого потомка получается добавлением единичного младшего бита. Если рассматривать

адрес как двоичное число, то адреса соседей могут быть вычислены вычитанием и прибавлением единицы от/к адресу текущей ячейки. В многомерном случае для хранения адреса используется несколько последовательностей бит.

Для аппроксимации дифференциальных уравнений на сетке должна быть построена *интерполяция*. Для этого в каждой ячейке C_i располагаются 9 интерполяционных точек (в углах, в серединах сторон, в центре). Интерполяционная функция между точками восполняется билинейным способом. Значения функции $f_0^i,...,f_8^i$ в интерполяционных точках вычисляются так, чтобы среднее значение интерполяционной функции по ячейке C_i было равно V_i. При разбиении ячейки ее потомки получают величины, равные средним арифметическим значений соответствующих четырех интерполяционных точек. При объединении среднее значение потомков будет присвоено родительской ячейке. При таком подходе повторяющиеся разбиения/объединения ячеек не приведут к накоплению погрешности.

Для аппроксимации частных производных второго порядка (например, методом конечных объемов) требуются непрерывные частные производные интерполяционной функции. Однако задача нахождения непрерывно дифференцируемой интерполяционной функции обычно сводится к трудоемкой процедуре решения системы линейных уравнений. Поэтому мы пошли другим путем: в каждой интерполяционной точке, помимо значения интерполяционной функции, вычисляются оценки ее частных производных. Частные производные восполняются билинейным способом и используются для консервативной аппроксимации производных второго порядка.

Адаптация сетки

Для получения высокой точности решения сетка должна адаптироваться к сеточной функции, изменяющейся на каждом временном шаге. В качестве критерия адаптации мы используем скалярную величину, называемую *вариацией*, которая для листовой ячейки C_i определяется выражением

$$D_i = \max_{0\le j,k\le 8}\left|f_j^i - f_k^i\right| \overset{\text{(в скалярном случае)}}{=} \max_{0\le j\le 8}\left(f_j^i\right) - \min_{0\le j\le 8}\left(f_j^i\right). \qquad (1)$$

Цель алгоритма адаптации сетки – минимизация максимальной вариации при ограниченном числе листовых ячеек.

Параллельная реализация

Задача, решаемая на адаптивной сетке, может быть распараллелена методом разбиения области. Расчетная область разбивается на одинаковые по размерам прямоугольные блоки, называемые *кластерами*, которые распределяются между процессорами как неделимые единицы для балансировки загрузки и минимизации обмена данными. Каждый кластер содержит поддерево адаптивной сетки; в кластерах может быть различное число ячеек. Для достижения балансировки загрузки каждый кластер должен быть назначен какому-либо процессору в соответствии со следующим критерием:

$$\max_{i=1,...,n}\left(N_i + \alpha_1 \cdot L_i\right) + \alpha_2 \cdot \max_{i=1,...,n} D_i \to \min. \qquad (2)$$

Здесь n – число процессоров; N_i – число ячеек на i-м процессоре; L_i – суммарная длина границы между кластерами процессора i и кластерами других процессоров (измеряется в ячейках); D_i – число ячеек, которые должны быть переданы к/от процессора i для достижения требуемого распределения кластеров на следующем временном шаге

(накладные расходы), α_1, α_2 – коэффициенты, зависящие от скорости передачи данных и решаемой задачи.

Заключение

Тестирование выполненной автором реализации описанных алгоритмов показало, что машинное время решения тестовой задачи переноса с неявной аппроксимацией на адаптивной сетке примерно в 1,5 раза больше, чем время решения задачи на равномерной сетке с тем же числом ячеек (накладные расходы на интерполяцию и адаптацию). Однако для достижения той же точности решения равномерная сетка должна иметь примерно в 16 раз больше ячеек, чем адаптивная. Поэтому, при той же точности, решение на адаптивной сетке может быть получено в 10 раз быстрее, чем на равномерной.

Метод адаптивных сеток значительно уменьшает время вычислений и снижает требуемый объем оперативной памяти по сравнению с равномерными сетками при той же точности получаемого решения. Разработана параллельная реализация алгоритмов с динамической балансировкой загрузки.

Литература

1. Тихомиров П. Система SENTAURUS TCAD компании Synopsys: новое поколение приборно-технологических САПР / П. Тихомиров, П. Пфеффли, М. Зорзи // Электроника НТБ. – 2006. – № 7. – С. 89–95.
2. Berger M.J. Aspects (and Aspect Ratios) of Cartesian Mesh Methods / M.J. Berger, M.J. Aftosmus // Proc. of the 16th Int. Conf. on Numerical Methods in Fluid Dynamics, (6–10 July, 1998, Arcachon, France). – Springer-Verlag, Heidelberg, Germany, 1998. – P. 1–12.
3. Hannoun N. Issues in Adaptive Mesh Refinement Implementation / N. Hannoun, V. Alexiades // Electronic Journal of Differential Equations: Sixth Mississippi State Conference on Differential Equations and Computational Simulations. – 2007. – P. 141–151.

Сухинов Антон Александрович — Московский физико-технический институт (государственный университет), аспирант, Soukhinov@gmail.com

УДК 004.4'23

ИНТЕРПРЕТАЦИЯ КАК СРЕДСТВО ИССЛЕДОВАНИЯ ДИНАМИЧЕСКИХ СВОЙСТВ ПАРАЛЛЕЛЬНОЙ ПРОГРАММЫ НА ИНСТРУМЕНТАЛЬНОМ КОМПЬЮТЕРЕ

М.С. Акопян

В работе рассматривается среда ParJava, поддерживающая разработку параллельных Java-программ для вычислительных систем с распределенной памятью. В среде ParJava строится модель параллельной программы, позволяющая оценивать ее динамические параметры не на целевой вычислительной системе, а на персональном компьютере разработчика.

Ключевые слова: среда разработки, Java, параллельное программирование.

При проектировании и конструировании сложных технических устройств обычно используются системы автоматизации проектирования, работающие на многопроцессорных вычислительных системах. Эффективность САПР во многом определяется качеством параллельных программ, выполняющих необходимые вычисления. Отсутствие методов компиляции параллельных программ вынуждает оптимизировать их вручную,

используя средства низкого уровня, что приводит к замедлению разработки и увеличению ее стоимости.

В данной работе рассматривается среда ParJava [1] разработки параллельных Java-программ для вычислительных систем с распределенной памятью, в состав которой входят инструментальные средства, облегчающие разработку параллельной программы. В среде ParJava используется модель параллельной программы, позволяющая оценивать эффективность и масштабируемость разрабатываемой параллельной программы, интерпретируя ее модель. Таким образом, большая часть отладки и доводки параллельных программ выполняется не на целевой вычислительной системе (высокопроизводительном кластере), а на инструментальном компьютере (персональном компьютере разработчика).

Среда ParJava предоставляет прикладному программисту набор инструментов, поддерживающих разработку параллельных программ: интерпретатор модели (позволяет исследовать поведение программы на инструментальной машине), редактор модели (позволяет вносить изменения в модель, при этом модель не перестраивается с нуля), анализатор модели (позволяет исследовать различные характеристики модели, например распараллеливаемость цикла) и т.д. Для поддержки параллельных программ на языке Java была разработана и реализована библиотека mpiJava, который представляет собой привязку к существующим реализациям библиотеки MPI [2] из среды Java.

В модели параллельной программы явным образом выделяются базовые блоки (линейные последовательности вычислений) и поток управления. Для сокращения времени интерпретации и обеспечения возможности выполнения интерпретации на инструментальном компьютере в среде ParJava моделируются только поток управления процессов моделируемой программы и операции обмена данными между процессами. Значения времени выполнения базовых блоков определяются на целевой вычислительной системе до начала интерпретации модели. Интерпретация модели лишь распространяет значения исследуемых атрибутов на остальные узлы модели.

Процессы параллельной программы моделируются с помощью логических процессов. Логический процесс представляет собой последовательность действий (примеры действий – выполнение базового блока, выполнение операции обмена и т.п.), имеющих определенную продолжительность. В логическом процессе определено понятие модельных часов. Показание модельных часов каждого логического процесса в момент начала интерпретации равно нулю. После интерпретации очередного действия к модельным часам соответствующего логического процесса добавляется значение продолжительности этого действия. Логические процессы реализованы в виде потоков языка Java.

Коммуникационные функции моделируются с помощью базовых операций обмена [1]. Время выполнения базовых операций оценивается с помощью тестов.

После интерпретации вершина модели может быть редуцирована. Редукция вершины модели заключается в том, что поддерево модели с корнем в рассматриваемой вершине заменяется одной вершиной типа «редуцированный базовый блок». В дальнейших интерпретациях модели эта часть программы интерпретироваться не будет, а значение времени выполнения данной вершины будет содержаться в редуцированном блоке, на который была заменена соответствующая часть программы во время редукции. В результате интерпретации модели можно получить частотные и временные профили программы, оценить время выполнения программы и ее частей, получить границы области масштабируемости. С помощью анализатора модели можно определить перекрытия вычислений и обменов и выдать рекомендации по оптимальной расстановке вызовов коммуникационных функций.

Среда ParJava используется в ИСП РАН при разработке параллельных прикладных программ, в частности, реализованы программа моделирования интенсивных атмосферных вихрей (торнадо) [3] и программа моделирования водно-ионной оболочки

ДНК [4]. Погрешность оценки динамических параметров составила менее 10%. В дальнейшем предполагается расширить интерпретатор модели возможностями сбора и анализа трассы параллельной программы с использованием модели.

Литература

1. Прогнозирование производительности MPI-программ на основе моделей / В.П. Иванников, А.И. Аветисян [и др.] // Автоматика и телемеханика. – 2007. – № 5. – С. 8–17.
2. MPI: The complete Reference / Marc Snir, Steve Otto, [et al]. – Vol. 1, 2. – 2nd edition. – The MIT Press, 1998.
3. Аветисян А.И. Возникновение торнадо: трехмерная численная модель в мезомасштабной теории турбулентности по В.Н. Николаевскому / А.И. Аветисян, В.В. Бабкова и А.Ю. Губарь // ДАН. Геофизика. – 2007. – Т. 419. – № 4.– С. 547–552.
4. Аветисян А.И., Гайсарян С.С. Разработка параллельного алгоритма компьютерного моделирования водно-ионной оболочки ДНК. Информационные и математические технологии в науке и управлении // Тр. XIII Байкальской Всероссийской конференции «Информационные и математические технологии в науке и управлении». Часть I. – Иркутск: ИСЭМ СО РАН, 2008. – С. 195–206.

Акопян Манук Сосович — Институт системного программирования РАН, м.н.с., manuk@ispras.ru

УДК 519.642

ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ РАСПРОСТРАНЕНИЯ УПРУГИХ ВОЛН В СРЕДАХ, ХАРАКТЕРНЫХ ДЛЯ ГРЯЗЕВЫХ ВУЛКАНОВ

Д.А. Караваев

Кратко изложен метод решения задачи численного моделирования распространения упругих волн в трехмерно неоднородных средах. На основе приведенного алгоритма создан комплекс программ. Предложен способ распараллеливания программы для численного расчета средствами OpenMP и MPI.

Ключевые слова: упругие волны, грязевые вулканы, параллельные алгоритмы.

Введение

Проблема генезиса грязевых вулканов является дискуссионной. В работах [1–4] предложен вибросейсмический метод мониторинга магматических структур с контролируемым сейсмическим источником, который позволит получить новые знания о строении вулканов, их происхождении и динамике поведения дилатантных структур живущих вулканов. Предлагаемый программный комплекс фактически является инструментарием для проведения экспериментальных работ на местности.

Комплекс программ позволяет проводить численное моделирование распространения упругих волн в трехмерных моделях упругих сред и определять параметры и место расположения системы возбуждения и системы наблюдения относительно вулканической структуры для получения оптимальных экспериментальных данных при проведении натурных геофизических экспериментов. Результаты численного моделирования могут быть использованы при интерпретации данных вибросейсмических зондирований грязевых вулканов.

Постановка задачи и метод решения

Численное моделирование распространения сейсмических волн в сложно построенных упругих средах проводится на основе полной системы уравнений теории упругости, записанной в скоростях перемещений и напряжений с нулевыми начальными и граничными условиями. Решение поставленной задачи основано на использовании конечноразностного метода [5]. Схема построена с учетом интегральных законов сохранения [5]. Конечноразностная схема имеет второй порядок аппроксимации по времени и пространству [5]. В связи с тем, что область расчета ограничена, необходимо использовать метод поглощающих границ (Perfectly Matched Layers, PML) [6].

Построение модели трехмерной упругой среды

Модель среды строится из криволинейных параллелепипедов, в которых задаются параметры среды, непрерывные внутри каждого блока, а затем методом конечных элементов происходит интерполяция параметров среды на сетку, на которой производится расчет. Возможно включение в слоистую модель цилиндрической, конической, эллипсоидальной и др. подобластей различной геометрии со своими параметрами среды. Можно моделировать присутствие трещин и газовых пузырей.

Параллельная реализация

Для трехмерных разностных схем имеет смысл применять гибридную технологию распараллеливания: внутри каждого вычислительного «узла» для распараллеливания применять OpenMP, а между «узлами» – MPI. Это значительно сокращает количество обменов информацией между узлами. Для обоих вариантов технологически удобным способом разбиения расчетной области является разбиение на слои вдоль координаты *Z*. Количество узлов в каждом слое определяется в зависимости от количества возможных вычислительных ядер. Каждый вычислительный узел будет рассчитывать количество слоев, равное числу процессорных ядер, имеющихся на нем. Для версии MPI будет необходим обмен данными между узлами в граничном слое.

Результат численного моделирования

Моделируется среда с находящимся в ней цилиндром, которые обладают следующими параметрами:

– вмещающая среда – V_p=2,0 км/с, V_s=1,0 км/с, ρ = 1 г/см^3;
– цилиндр – V_p =1,0 км/с, V_s =0,7 км/с, ρ = 1 г/см^3.

Результаты моделирования представлены на рис. 1.

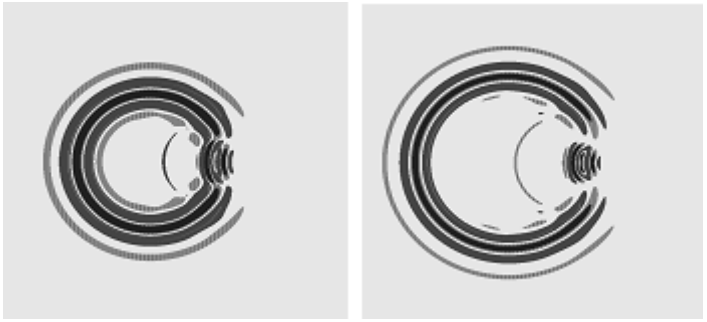


Рис. 1. Снимки Uz компоненты волнового поля, плоскость Oxy.
Источник типа «центр давления», несущая частота 4 Гц

Литература

1. Глинский Б.М., Собисевич А.Л., Хайретдинов М. Опыт вибросейсмического зондирования сложно построенных геологических структур (на примере грязевого вулкана Шуго) // Докл. РАН. – 2007. – Т. 413. – № 3. – С.398–402.
2. Глинский Б.М. Фатьянов А.Г. Численно-аналитическое моделирование волновых полей в разномасштабных зонах вулканической деятельности // Всероссийская конференция по вычислительной математике «КВМ–2007», Новосибирск, 18–20 июня 2007.
3. Глинский Б.М., Фатьянов А.Г. Вибросейсмический мониторинг живущих вулканов // Материалы 2-го межд. симпозиума «Активный геофизический мониторинг вулканов». – Новосибирск, 2005. – С. 57–61.
4. Глинский Б.М., Фатьянов А.Г. Изучение и мониторинг грязевых вулканов активными сейсмическими методами // Материалы 2–го межд. симпозиума «Активный геофизический мониторинг вулканов». – Новосибирск, 2005. – С. 52–57.
5. Bihn M. A Stable Discretization Scheme for the Simulation of Elastic Waves / M. Bihn, T. Weiland // Proceedings of the 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics. – 1997. – Vol. 2. – P. 75–80.
6. Collino F. Application of the PML absorbing layer model to the linear elastodynamic problem in anisotropic heterogeneous media / F. Collino, C. Tsogka // Geophysics. – 2001. – Vol. 66. – № 1. – P. 294–307.
7. Clemens M. Descrete electromagnetism with the finite integration technique / M. Clemens, T.Weiland // Progress In Electromagnetics Research. – 2001. – Vol. 32. – P. 65–87.

Караваев Дмитрий Алексеевич — Институт вычислительной математики и математической геофизики СО РАН, магистр, dmitry1985@ngs.ru

УДК 004.4'23

ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ С ИСПОЛЬЗОВАНИЕМ МИКРОПРОЦЕССОРОВ СЕМЕЙСТВА CELL

С.М. Вишняков, А.С. Мордвинцев

Целью работы является исследование применимости архитектуры CBEA к различным типам вычислительных задач, в частности – к задаче трассировки лучей через сцену, имеющую воксельное представление. Для этого предполагается разработать демонстрационное приложение, позволяющее пользователю перемещаться по трехмерному миру и взаимодействовать с ним.

Ключевые слова: микропроцессор Cell, воксельная графика.

Процессор Cell BE

В настоящее время в связи с невозможностью дальнейшего роста тактовой частоты процессоров разработчики вычислительных машин в качестве способа увеличения производительности рассматривают использование параллельных вычислительных систем. В данной области можно выделить несколько сложившихся тенденций:

- использование многоядерных и многопроцессорных систем традиционной (x86) архитектуры;
- использование параллельных вычислительных акселераторов, встраиваемых в системы традиционной архитектуры;
- использование параллельных вычислительных систем «альтернативных» архитектур.

Примером «альтернативной» архитектуры является архитектура Cell Broadband Engine Architecture (CBEA), разрабатываемая компанией IBM. Данная архитектура уже нашла свое применение в игровых приставках Sony Playstation 3 и активно продвигается компанией IBM в качестве платформы для высокопроизводительных вычислений различного характера.

Первой реализацией CBEA является процессор Cell Broadband Engine (CBE, Cell). Центральным элементом процессора Cell является процессорное ядро Power Processor Element (PPE), базирующееся на 64-битной архитектуре Power Architecture. Ядро PPE отвечает за работу операционной системы, ввод/вывод данных и распределение вычислений между остальными элементами системы – восемью ядрами Synergetic Processor Element (SPE). Поскольку ядра SPE предназначены исключительно для вычислительных задач и, по сути, являются параллельными вычислительными акселераторами для главного процессорного ядра (PPE), их архитектура серьезным образом отличается от архитектуры обычных процессоров.

В современных вычислительных системах узким местом зачастую оказывается обращение к памяти. Поэтому тот факт, что при программировании Cell загрузка данных из общей памяти вычислительного устройства в локальную память SPE регулируется программистом, дает определенные преимущества в плане производительности. Операции обращения к памяти являются асинхронными – следовательно, при правильном подходе время ожидания новых данных из памяти не уходит впустую, а тратится на обработку предыдущего блока данных. Таким образом, при оптимальной организации работы ядер SPE система может достигать производительности в 200 ГФлопс (при работе с числами ординарной точности).

Воксельная графика

Основная идея воксельной графики заключается в представлении сцены в виде трехмерного массива кубических ячеек, ребра которых выровнены по осям координат. Таким образом, воксельную сцену можно рассматривать как трехмерный аналог двухмерного растрового изображения. Одной из наиболее распространенных областей применения такого представления трехмерных данных является томография. Результатом сканирования является трехмерный массив, каждый элемент которого содержит численное значение, обозначающее какую-нибудь характеристику одной из ячеек исследуемого объема. В зависимости от типа сканера это может быть плотность, прозрачность для определенного вида излучения, содержание некоторого вещества и т.д.

Интерес для визуализации представляют только ячейки, находящиеся на поверхности объектов. Если хранить только их, то объем требуемой при фиксированном разрешении памяти будет пропорционален площади присутствующих в сцене объектов, которая, в свою очередь, пропорциональна второй степени линейных размеров сцены.

Для хранения вокселей поверхности в данном проекте используется представление сцены в виде разреженного октарного воксельного дерева (sparse voxel octree). Весь объем сцены помещается в куб, соответствующий корню дерева. Затем этот объем разбивается на восемь дочерних кубов. Кубы, целиком лежащие вне или внутри объектов сцены, отбрасываются. Для кубов, содержащих участки поверхностей объектов, операция повторяется. Когда на очередном уровне детализации объем пространства, соответствующий узлу, становится достаточно маленьким, в нем сохраняется цвет и вектор нормали к поверхности на этом участке, и дальнейшее разбиение прекращается.

Для построения изображения в данной работе используется обратная трассировка лучей. Лучи трассируются из глаза наблюдателя через каждый экранный пиксель до первого пересечения с поверхностью объектов сцены. Цвет пикселя определяется цветом вокселя, с которым пересекся луч, нормалью к поверхности и настройками освещения.

Текущие результаты

В рамках работы над проектом были разработаны несколько реализаций алгоритмов построения и визуализации воксельных сцен для различных архитектур. Для процессора Cell с использованием 16 SPU удалось достичь производительности 180 мс на кадр разрешением 1024×768. Многопотоковая реализация, использующая только PPU, работает более 2 с. Можно отметить, что текущая реализация имеет потенциал для увеличения производительности: во-первых, имеет смысл реализовать более эффективную схему кэширования запросов процессоров SPU к глобальной памяти, во-вторых, более эффективно использовать SIMD-возможности процессора, а также использовать при рендеринге ядра PPU совместно с SPU.

Литература

1. Интервью Джона Кармака (id Software). – Режим доступа: http://www.pcper.com/article.php?aid=532, свободный.
2. Эксперементальный воксельный движок, разработанный Кеном Сильверманом. – Режим доступа: http://www.advsys.net/ken/voxlap.htm, свободный.
3. Aaron Knoll, Ingo Wald, Steven Parker, and Charles Hansen. Interactive Isosurface Ray Tracing of Large Octree Volumes // Proceedings of the IEEE Symposium on Interactive Ray Tracing, Salt Lake City, 2006. – Режим доступа: http://www.cs.utah.edu/~knolla/octiso-rt06.pdf, свободный.

Вишняков Сергей Михайлович — Санкт-Петербургский государственный университет информационных технологий, механики и оптики, студент, sergey.vishnyakov@gmail.com

Мордвинцев Александр Сергеевич — Санкт-Петербургский государственный университет информационных технологий, механики и оптики, студент, zzznah@gmail.com

УДК 004.4'23

СТОХАСТИЧЕСКОЕ МОДЕЛИРОВАНИЕ КОМПЛЕКСНЫХ СЕТЕЙ

И.И. Колыхматов

Исследование посвящено разработке и исследованию параллельных алгоритмов для стохастического моделирования комплексных сетей с учетом их неоднородности и нестационарности. Предложенные алгоритмы применены для решения ряда прикладных задач математической эпидемиологии.

Ключевые слова: комплексные сети, параллельные алгоритмы.

Введение

Комплексные сети, структура которых нерегулярна, сложна и динамически эволюционирует во времени, являются частью математического аппарата для описания, изучения свойств и поведения сложных систем в нескольких диапазонах пространственной и временной изменчивости [1, 2]. Комплексные сети применяются для моделирования объектов и систем, другие способы исследования которых (с помощью наблюдений и активного эксперимента) не представляются целесообразными или возможными. В общем виде динамическая модель на основе комплексной сети представляет собой случайный граф, закон взаиморасположения ребер и вершин для которого задается распределением вероятностей.

Параллельные алгоритмы прямого моделирования

После изучения структурных особенностей комплексных сетей была сформулирована вероятностная модель комплексной сети и предложено семейство параллельных алгоритмов [2], позволяющих воспроизводить комплексную сеть как стохастический граф с заданными вероятностными свойствами. В алгоритме генерации комплексных сетей по заданному распределению степеней вершин распараллеливание осуществляется за счет одновременной обработки независимых блоков данных в ходе построения допустимого совершенного паросочетания на разных процессорах. Предложенные в работе [2] алгоритмы основываются на естественных свойствах распараллеливания структуры графов и соответствуют особенностям вычислительной архитектуры – с общей памятью (SMP), кластерной или P2P. Приводятся результаты экспериментальных исследований производительности параллельных алгоритмов на многоядерных вычислительных системах и кластерной системе TForge-Mini. Работоспособность алгоритмов проиллюстрирована на примере моделирования эпидемиологических комплексных сетей, описывающих распространение ВИЧ.

В качестве основной модели динамики ВИЧ предлагается использовать комплексную сеть, в которой каждый узел представляет собой человека, находящегося в одном из 3 основных состояний: восприимчивый к инфекции, зараженный, удаленный из сети. Состояние «инфицированный» имеет ряд дополнительных признаков, связанных с уровнем содержания вируса в крови, который определяется временем с момента заражения и наличием лечения. Связи между узлами характеризуют контакты между людьми, которые могут приводить к заражению. Структура эпидемиологической сети может иметь несколько типов связей; каждый тип характеризуется своим видом распределения с некоторым набором параметров. Ситуация осложняется тем, что необходимо знать и уметь моделировать не только структуру связей в заданный момент времени, но и динамику сети, характеризующую конкретный тип связей за достаточно долгий период времени (например, за период жизни одного поколения).

Программная реализация

В ходе структуризации и классификации сервисов для типовой архитектуры высокопроизводительного программного комплекса моделирования сложных систем была построена онтология сервисов [3]. Реализация системы базовых сервисов, позволяющих гибко и эффективно проектировать программное обеспечение для моделирования сложных систем, выполнена в ходе разработки демонстрационного приложения – системы моделирования эпидемиологической динамики ВИЧ на основе аппарата комплексных сетей [4, 5]. Для оценки производительности параллельных алгоритмов были написаны реализации как последовательных, так и параллельных алгоритмов на языке C++ в среде Microsoft Visual Studio 2005 с использованием технологий OpenMP и MPI.

Результатом работы программной системы является собственно сеть с динамически меняющимися свойствами – «виртуальная популяция», с помощью которой можно изучать распространение инфекции. Статистический анализ этой популяции позволяет не только восстанавливать скрытые параметры эпидемиологической ситуации, но и предсказывать развитие эпидемии (в случае ВИЧ – на несколько десятков лет вперед), а также выявлять возможные факторы превентивного управления эпидемией [6].

Заключение

В результате исследования изучен подход эволюционного моделирования динамики сложных систем с применением аппарата комплексных сетей на параллельных

вычислительных архитектурах на примере эпидемиологической динамики вируса иммунодефицита человека.

В ходе работы были проанализированы математические модели комплексных сетей, сформулирована вероятностная модель комплексной сети, разработаны параллельные алгоритмы прямого моделирования динамики эпидемиологических комплексных сетей. Для данных алгоритмов построены аналитические зависимости параллельного ускорения и выполнено отображение данных алгоритмов на целевую вычислительную архитектуру.

Алгоритмы моделирования эволюционной динамики эпидемиологических комплексных сетей [2, 4–6] могут применяться для решения следующих задач:

1. моделирование стохастической контактной сети заданного объема и свойств на основе интегральных вероятностных характеристик и расчет ее эволюции под действием внешних эволюционных факторов (спланированного воздействия, эпидемии);
2. прогнозирование экстремальных сценариев поведения сообщества и выявление факторов для превентивного противостояния этим сценариям.

Литература

1. Newman M.E.J. The Structure and Function of Complex Networks // SIAM Review.– 2003.– Vol. 45. – № 2. – P. 167–256.
2. Параллельные алгоритмы моделирования комплексных сетей / С.В. Иванов, И.И. Колыхматов, А.В. Бухановский // Известия вузов. Приборостроение. – 2008. – Т. 51. – № 10. – С. 5–11.
3. Ковальчук С.В., Иванов С.В., Колыхматов И.И., Бухановский А.В. Особенности проектирования высокопроизводительных программных комплексов для моделирования сложных систем // Информационно-управляющие системы. – 2008. № 3. – С. 10–18.
4. Высокопроизводительные технологии синтеза, эволюционного моделирования и визуализации комплексных сетей в среде Microsoft Compute Cluster Server / И.И. Колыхматов, С.В. Иванов [и др.] // Технологии Microsoft в теории и практике программирования. Материалы конференции (Нижний Новгород, 19–20 марта 2008 г.). – Нижний Новгород: Изд-во НнГУ, 2008. – С. 174–177.
5. Иванов С.В., Колыхматов И.И., Бухановский А.В. Моделирование популяционной динамики ВИЧ на основе механизма комплексных сетей: математическая модель и параллельный алгоритм // Материалы 6 Международного научно-практического семинара «Высокопроизводительные параллельные вычисления на кластерных системах», Санкт-Петербург, 12–17 декабря 2006.– Т. 1.– С. 208–209.
6. Stochastic simulation of HIV population dynamics through complex network modelling / P.M.A. Sloot, S.V. Ivanov [et al.] // International Journal of Computer Mathematics. – 2008. – Vol. 85.– Issue 8. – P. 1175–1187.

Колыхматов Илья Игоревич — Санкт-Петербургский государственный университет информационных технологий, механики и оптики, студент, kolyhmatov@rain.ifmo.ru

УДК 004.021

МЕТРОЛОГИЧЕСКОЕ ОБЕСПЕЧЕНИЕ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ И СИСТЕМ

А.В. Семенов, А.В. Бухановский

Рассматривается современное состояние проблемы метрологического обеспечения технологий распределенных вычислений и систем. С точки зрения обеспечения единства измерений анализируются объекты, методы и средства измерений, применяемые в данной предметной области.
Ключевые слова: метрологическое обеспечение, распределенные вычисления.

Введение

Неотъемлемой характеристикой компьютерной программы является время ее исполнения. Оно совокупно зависит от особенностей реализуемого алгоритма, специфики данных и характеристик компьютера, на котором производятся вычисления. На процесс выполнения программы влияет большое количество различных факторов, затрудняющих точное измерение и прогноз времени выполнения; особенно ярко это свойство проявляется на многопроцессорных и распределенных вычислительных системах. Извлечение количественной информации о свойствах объектов с заданной точностью и достоверностью является классической задачей метрологии [1]. За время ее существования было выработано большое количество разнообразных подходов и методов измерений различных параметров, могущих быть примененными к высокопроизводительным вычислениям. Одним из важнейших понятий метрологии является *единство измерений* – состояние измерений, при котором их результаты выражены в узаконенных единицах величин и погрешности не выходят за установленные границы с заданной вероятностью. Единство измерений необходимо для того, чтобы можно было сопоставить результаты измерений, выполненных в разных местах и в разное время, с использованием разных методов и средств измерений. В данной работе производится анализ текущего состояния метрологического обеспечения высокопроизводительных вычислений применительно к основным категориям понятий – объектам, процедурам и средствам измерений.

Объекты измерений

В настоящее время не существует однозначной классификации объектов измерений в области высокопроизводительных вычислений. Наиболее часто рассматривается такая характеристика, как срок окончания выполнения задачи (*deadline*); типизация способов задания *deadline* рассмотрена в [2]. Сроки окончания измеряются в единицах измерения времени и являются абсолютной характеристикой выполнения задачи.

В ряде случаев удобнее рассматривать фрагментарные временные характеристики задачи. Например, в [3] рассматривается такая характеристика, как время отклика приложения, в [4] – время ожидания начала выполнения, в [5] – время создания потоков операционной системой, и пр. Фрагментарные характеристики могут рассматриваться как параметры модели, описывающей время выполнения задачи в целом.

Для описания масштабируемости задачи используются относительные временные характеристики, например, параллельное ускорение и параллельная эффективность [6]. В работе [7] измеряется такая характеристика приложения, как изоэффективность. Эти величины целесообразно применять для определения целесообразности распараллеливания задачи на заданное число узлов или для сопоставления нескольких способов распараллеливания одного и того же алгоритма. Для сравнения различных вычислительных алгоритмов и/или различных вычислительных систем они не применяются. Понятие масштабируемости тесно связано с интегральными характеристиками параллель-

ных приложений, такими, например, как профиль параллельности, средняя параллельность и абсолютная параллельность [8].

Отдельный класс объектов измерений составляют специфические параметры, характеризующие не столько саму параллельную задачу, сколько особенности многопользовательской системы, на которой она выполняется. К ним относятся количество работающих задач, коэффициент утилизации, количество пользователей, получивших свою задачу в срок, и пр. [9]. В работе [10] сделан шаг в направлении создания единой системы объектов измерений как характеристик, определяющих распределенную систему.

Процедуры измерений

Основными методами измерения производительности задач, выполняющихся на распределенной вычислительной системе, являются прототипирование, аналитический расчет, прогнозирование на основе «исторической информации», обучение в реальном времени, а также гибридный метод, сочетающий в себе особенности одного или нескольких вышеупомянутых подходов. Широко распространенными методами оценки производительности параллельных программ при помощи аналитического расчета являются законы Амдала и Густафсона [11].

Как известно [1], один из способов определения погрешности измерительной процедуры заключается в вычислении разности измеренного и эталонного значения величины. В области высокопроизводительных вычислений известны исследования, направленные на построение некоторых эталонных объектов измерения. Например, в [12] построена модель для генерации задач, имеющих параметры, близкие к реально исполняемым задачам (с целью тестирования различных механизмов управления суперкомпьютерами высокого уровня занятости). В работе [13] рассмотрен способ уменьшения необходимого для улучшения конфигурации системы количества итеративных измерений; здесь близость к эталону интерпретируется в терминах сходства между соседними итерациями.

Основным способом организации измерительной процедуры в настоящее время является расстановка точек останова с последующим сбором фрагментарных измерительных данных. Методы статического и динамического анализа производительности программы, а также способы оптимизации расстановки точек останова описаны в [14].

Средства измерений

В высокопроизводительных вычислениях применяются два типа средств измерений: первый тип измеряет характеристики системы, на которой работают задачи (производительность, количество процессоров и др.), а второй – характеристики задачи, работающей на системе (время работы, параллелизм и др.). К первой группе относятся такие инструменты, как NAS Parallel benchmarks, Linpack, SPEC. В работе [15] описываются общие принципы построения программных и аппаратных *мониторов* – средств измерений, позволяющих узнать производительность системы, принципиальное отличие систем друг от друга, а также сформулировать пути последующего улучшения производительности системы.

Для измерения временных характеристик задачи, выполняющейся на параллельной вычислительной системе, применяются два метода, отличающиеся по использованию средств измерений. Первый – внедрение в исходный код программы вызовов некоторых функций измерения времени, наподобие *gettimeofday()* в UNIX, компиляция и запуск программы несколько раз подряд. Способы фрагментации исходного кода программы для более точного измерения времени описаны в [14], там же дано подробное описание данного средства, описано влияние таких факторов, как заполненность кэша,

и вычислена погрешность измерений. Второй метод – использование сторонних программ, которые внедряются в процесс работы измеряемой программы и проводят измерения, например, [16].

Заключение

Проведенный в работе анализ показал, что, несмотря на актуальность проблемы измерения характеристик параллельных приложений и вычислительных систем, в настоящее время еще не сложилось общепринятых подходов к ее решению. Разными авторами предлагаются различные метрики количественной оценки задач и производительности программных и аппаратных систем, разработаны многочисленные методы измерений и специальные измерительные средства, ориентированные в большинстве своем на узкие классы прикладных задач. Можно утверждать, что в настоящий момент применяемые методы и технологии не находятся в состоянии единства измерений, что оставляет конъюнктурный вопрос создания системы метрологического обеспечения высокопроизводительных вычислений открытым.

Литература

1. Цветков Э.И. Основы математической метрологии. – СПб: Политехника, 2005. – 510 с.
2. Kao B., Garcia–Molina H. Deadline Assignment in a Distributed Soft Real–Time System // IEEE Transactions on Parallel and Distributed Systems. – 1997. – Vol. 8. – №12. – P. 1268–1274.
3. Dror G. Feitelson, Larry Rudolph, Uwe Schwiegelshohn et al. Theory and Practice in Parallel Job Scheduling // Proceedings of the Job Scheduling Strategies for Parallel Processing, April 05, 1997. – P. 1–34.
4. Iosup A., Epema D.H.J., Franke et al. On grid performance evaluation using synthetic workloads // Lecture Notes in Computer Science. – Springer, 2006. – Vol. 4376. – P. 232–255.
5. Kejariwal A. et al. Tight analysis of the performance potential of thread speculation using spec CPU 2006 // Proceedings of the 12th ACM SIGPLAN symposium on Principles and practice of parallel programming. March 14–17, 2007. San Jose, California, USA. – P. 215–225.
6. Bertasi P., et al. Obtaining Performance Measures through Microbenchmarking in a Peer–to–Peer Overlay Computer // Proceedings of the First International Conference on Complex, Intelligent and Software Intensive Systems. – 2007. – P. 285–290.
7. Grama A., Gupta A., Kumar V. Isoefficiency: Measuring the Scalability of Parallel Algorithms and Architectures // IEEE Parallel & Distributed Technology: Systems & Technology. – 1993. – Vol. 1. – № 3. – P. 12–21.
8. Overeinder B.J. Distributed Event–driven Simulation: Scheduling Stategies and Resource Management: Ph.D. thesis. – Universiteit van Amsterdam, 2000. – 222 p.
9. Welsh M., Culler D. Adaptive overload control for busy internet servers // Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems. March 26–28, 2003, Seattle, WA. – P. 4.
10. Dan A., Dumitrescu C., Ripeanu M. Connecting client objectives with resource capabilities: an essential component for grid service managent infrastructures // Proceedings of the 2nd international conference on Service oriented computing. November 15–19, 2004. New York, NY, USA. – P. 57–64.
11. Shi Y. Reevaluating Amdahl's Law and Gustafson's Law. – Temple Un., Oct 1996. – P. 17. – Режим доступа: http://www.cis.temple.edu/~shi/docs/amdahl.ps, свободный.

12. Dinda P., O'Hallaron D.R. Realistic CPU Workloads through Host Load Trace Playback // Selected Papers from the 5th International Workshop on Languages, Compilers, and Run–Time Systems for Scalable Computers. May 25–27, 2000. – P. 246–259.
13. Osogami T., Kato S. Optimizing system configurations quickly by guessing at the performance // Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems. June 12–16, 2007. San Diego, California, USA. – Vol. 35.– Issue 1 (June 2007). – P. 145–156.
14. Топорков В.В. Модели распределенных вычислений. – М.: ФМЛ, 2004. – 320 с.
15. Carlson G. How to save money with computer monitoring // Proceedings of the ACM annual conference. August 01–01, 1972. Boston, Massachusetts, United States. – P. 1018–1023.
16. Miller M., et al. The Paradyne Parallel Performance Measurement Tools // IEEE Computer. – 1995. – Vol. 28. – № 11. – P. 37–46.

Семенов Александр Владимирович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, аспирант, avsemyonov@gmail.com

Бухановский Александр Валерьевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, д.т.н., профессор, boukhanovsky@mail.ifmo.ru

УДК 533.6.011

ПРИМЕНЕНИЕ ГРАФИЧЕСКИХ СОПРОЦЕССОРОВ НА СУПЕРКОМПЬЮТЕРЕ МВС-ЭКСПРЕСС ДЛЯ РАСЧЕТА ЗАДАЧ АЭРОГАЗОДИНАМИКИ

А.А. Давыдов

Использование графических сопроцессоров для вычислений общего назначения испытывает значительный подъем благодаря появлению новых средств программирования. Однако параллельные вычисления на графических процессорах на сегодняшний день встречаются достаточно редко. На примере макетного суперкомпьютера МВС-ЭКСПРЕСС показывается перспективность данного направления.
Ключевые слова: аэрогазодинамика, графический сопроцессор.

Введение

Применение видеокарт NVidia GeForce 8800 GTX [1] для расчета задач газовой динамики позволяет ускорить расчет в 10–15 раз по сравнению с серверными процессорами, входящими в состав типичных кластерных установок. Однако относительно небольшой объем оперативной памяти таких устройств (до 1 ГБ) создает ощутимые ограничения на размер решаемых задач. Параллельное использование графических процессоров накладывает очень жесткие требования на коммуникационную среду, выполнение которых не под силу большинству современных кластерных установок. В ИПМ им. М.В. Келдыша РАН построен и передан в опытную эксплуатацию макет гибридного суперкомпьютера из 6 узлов, объединенных каналами PCI-Express, с пиковой производительностью около полутора терафлопс. В состав каждого узла данного макета, помимо четырех универсальных процессоров (ядер), входит один графический сопроцессор GeForce 8800 GTX (рис. 1). В работе проводится исследование эффективности параллельного использования графических сопроцессоров для расчета задач газовой динамики.

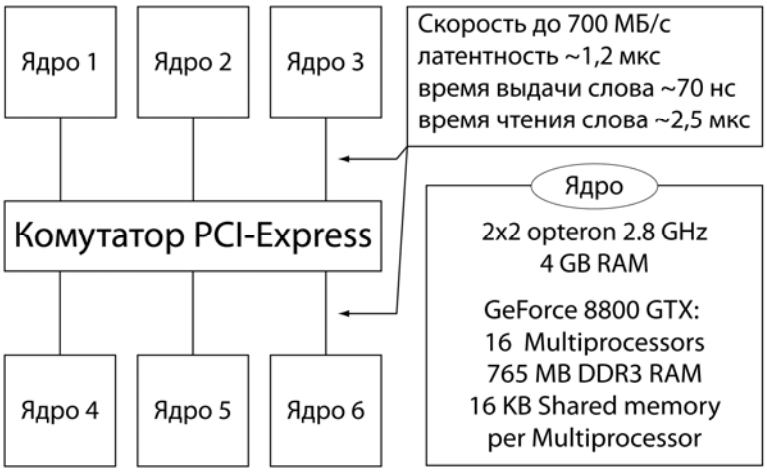


Рис. 1. Схема установки МВС-ЭКСПРЕСС

Графические сопроцессоры и параллельные вычисления

При организации работы параллельной программы приходится не только постоянно пересылать данные с графического устройства в память процессора, но и обмениваться данными между узлами. Скорость типичной для современных кластерных установок коммуникационной среды настолько мала, что делает малоэффективным использование графических сопроцессоров в составе узла кластерной установки. Эту проблему удается решить, лишь используя новые высокоскоростные интерфейсы, такие как PCI-Express.

МВС-Экспресс

Установка МВС-ЭКСПРЕСС представляет собой 6 узлов, объединенных каналами PCI-Express через шину PCI-Express. Каждый узел снабжен двумя двуядерными процессорами Opteron с тактовой частотой 2,8 ГГц, оперативной памятью 4 ГБ и графической картой NVidia GeForce 8800 GTX (рис. 1). Для связи узлов друг с другом применяются коммутаторы взаимного прямого доступа в память с очень низкими показателями задержки при обращении к сети.

В рамках опытной эксплуатации системы МВС-ЭКСПРЕСС написан многопроцессорный программный комплекс для расчета двумерных плоских и осесимметричных течений идеального газа на многоблочных регулярных сетках по явной разностной схеме С.К. Годунова [2]. Комплекс позволяет проводить расчет как на универсальных процессорах, входящих в состав узла, так и на графических сопроцессорах. В табл. 1 приведено сравнение времени расчета на трех процессорах Opteron (2,8 MHz) и трех графических картах NVidia GeForce 8800 GTX соответственно.

Таблица 1. Сравнение времени расчета 5000 шагов на сетке из 7 блоков

	Время расчета, с	Ускорение
3×Opteron 2,8 MHz	60	1
3×GeForce 8800 GTX	8,6	6,9

Заключение

Коммуникационная среда, примененная на установке МВС-ЭКСПРЕСС, позволяет эффективно использовать графические ускорители NVidia GeForce 8800 GTX для

параллельных вычислений, что представлялось невозможным на типичных на сегодняшний день кластерных установках. Учитывая, что в задачах газовой динамики характерное время расчета на многопроцессорной установке составляет сотни и даже тысячи часов, такой подход представляется вполне оправданным, тем более, что стоимость такого графического ускорителя сравнима со стоимостью серверного процессора. Стоит также отметить, что задачи газовой динамики отнюдь не являются единственным полем для применения такого рода систем.

Литература

1. NVIDIA Corporation. – Режим доступа: http://nvidia.com, свободный.
2. Годунов С.К., Забродин А.В., Иванов М.Я., Крайко А.Н., Прокопов Г.П. Численное решение многомерных задач газовой динамики. – М.: Наука, 1976.

Давыдов Александр Александрович – ИПМ им. М. В. Келдыша РАН, м.н.с., alexander.a.davydov@gmail.com

УДК 004.074

АНАЛИЗ БЫСТРОДЕЙСТВИЯ ОЗУ И ПОСТРОЕНИЕ МОДЕЛИ ПРОИЗВОДИТЕЛЬНОСТИ ЭВМ

К.С. Солнушкин

В работе исследовано быстродействие ОЗУ отдельно взятой ЭВМ как функция от параметров подсистемы памяти – частоты шины и режима работы контроллера памяти. Предложена и экспериментально подтверждена модель производительности, устанавливающая линейную зависимость производительности от быстродействия ОЗУ.

Ключевые слова: модели производительности, быстродействие ОЗУ

Сфера использования высокопроизводительных вычислений сегодня постоянно расширяется. Однако анализ производительности ЭВМ до сих пор остается сложной задачей, далекой от окончательного решения. К факторам, оказывающим значительное влияние на производительность ЭВМ, относится быстродействие оперативного запоминающего устройства (ОЗУ). Несовершенство ОЗУ либо несбалансированность его скоростных характеристик с аналогичными характеристиками центрального процессора приводит к низкой производительности ЭВМ.

В [1] введены понятия *производительность* и *быстродействие* в применении к ЭВМ. Там же введены понятия номинального и эффективного быстродействия. *Эффективным быстродействием* ОЗУ мы будем называть быстродействие (в операциях в единицу времени), наблюдаемое с помощью программ-тестов. В различных тестах измерения организованы по-разному, и понятие операции с ОЗУ размывается. Распространенным вариантом операции является чтение (запись) крупных блоков данных с измерением затраченного времени. Количество байт, перемещенных из ОЗУ в регистры центрального процессора (ЦП) или обратно за единицу времени, будем называть *эффективной пропускной способностью*.

Для исследования эффективной пропускной способности ОЗУ применим тест STREAM [2]. Тест обладает понятными правилами измерения быстродействия, простой интерпретацией результатов. Накоплена значительная статистика по разным типам ЭВМ. Будем исследовать ЭВМ класса IBM PC на основе ЦП AMD Athlon XP 2600+. В данной ЭВМ набор микросхем (так называемый «чипсет») модели nForce2 обеспечивает для контроллера ОЗУ два канала доступа к памяти, которые могут функционировать

одновременно (параллельно). Пользователь может настроить ЭВМ на использование как одного, так и обоих каналов доступа к памяти.

При исследовании варьировались два фактора – частота шины памяти и режим работы ОЗУ (одно- или двухканальный). Результаты измерений приведены в табл. 1. Анализ показывает, что увеличение пропускной способности нельзя аппроксимировать линейно. С ростом частоты шины прирост быстродействия монотонно уменьшается; иными словами, быстродействие выходит на асимптоту.

Теперь поставим следующую задачу: выяснить, как эти же параметры ЭВМ (частота шины и режим работы контроллера памяти) влияют на ее производительность в реальных задачах. В качестве такой задачи рассмотрим численное моделирование с помощью программной системы FLUENT [3] процесса сгорания метана в горелке при наличии турбулентных потоков; расчетная сетка содержит 353 тыс. элементов (тестовая задача FL5M3, поставляемая с системой FLUENT).

Таблица 1. Пропускная способность ОЗУ на тесте STREAM (ядро Copy)

Частота шины памяти, МГц	Пропускная способность ОЗУ, МБайт/с	
	Режим работы контроллера памяти	
	Одноканальный	Двухканальный
166	686	988
200	795	1064
266	986	1135
333	1094	1212
400	1138	1188

Если разделить количество секунд в сутках на астрономическое время счета в секундах, мы получим количество задач данного типа, которое ЭВМ способна решить за сутки, без учета ввода/вывода. Эта величина в терминах системы FLUENT называется рейтингом (англ. *rating*) и официально используется для сравнения производительности ЭВМ. Рейтинг – это, фактически, производительность ЭВМ на данной задаче, измеренная в задачах за сутки (в «штуках за единицу времени»). Размерность рейтинга, таким образом, $с^{-1}$. В результате эксперимента получены данные, характеризующие зависимость рейтинга от исследуемых параметров (табл. 2).

Таблица 2. Рейтинг на задаче FL5M3

Частота шины памяти, МГц	Рейтинг, $с^{-1}$	
	Режим работы контроллера памяти	
	Одноканальный	Двухканальный
166	99,2	110,8
266	112,9	117,3
333	117,9	121,4
400	117,7	120,5

Производительность ЭВМ, как и ранее – быстродействие ОЗУ, нелинейно зависит от частоты шины и с повышением частоты выходит на асимптоту. Возникает гипотеза, согласно которой производительность ЭВМ, возможно, линейно зависит от быстродействия ОЗУ.

Чтобы подтвердить эту гипотезу, по данным табл. 1, 2 построим график производительности ЭВМ в зависимости от быстродействия ОЗУ (см. рис. 1). Как видно из ри-

сунка, экспериментальные данные хорошо согласуются с выдвинутой гипотезой. Обе линии на рис. 1 являются отрезками прямых. Это означает, что производная от производительности ЭВМ по быстродействию ОЗУ постоянна, т.е. темп роста производительности при увеличении быстродействия является постоянной величиной. Кроме того, отрезки параллельны. Это означает, что темпы роста производительности не только постоянны, но и одинаковы для обоих режимов работы контроллера памяти.

Как следует из модели, производительность ЭВМ на реальной вычислительной задаче увеличивается линейно с ростом быстродействия ОЗУ. Это позволяет прогнозировать производительность будущих ЭВМ на имеющихся задачах в процессе проектирования ЭВМ. В качестве направления дальнейших исследований можно указать проверку найденных закономерностей для других типов ЭВМ и вычислительных задач.

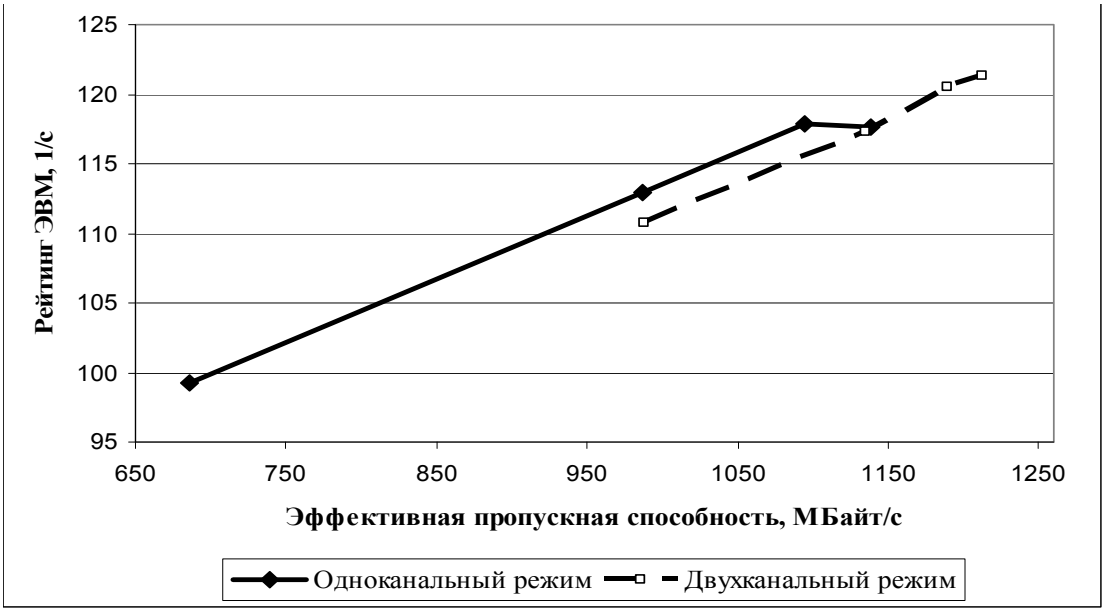


Рис. 1. Зависимость производительности ЭВМ от быстродействия ОЗУ

Литература

1. Солнушкин К.С. О значении терминов «производительность» и «быстродействие» в применении к ЭВМ // Научно-технические ведомости СПбГПУ. – СПб: Изд-во СПбГПУ. – 2008. – Т. 59. – № 3.– С. 297–300.
2. McCalpin J. D. Memory Bandwidth and Machine Balance in Current High Performance Computers // IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter. – December 1995. – P. 19. – Режим доступа: http://tab.computer.org/tcca/NEWS/ DEC95/DEC95.HTM, свободный.
3. FLUENT. CFD Flow Modeling Software and Solutions from Fluent. – Режим доступа: http://www.fluent.com/software/fluent/, свободный.

Солнушкин Константин Сергеевич – Санкт-Петербургский государственный политехнический университет, ведущий программист, ks@spbstu.ru

SUMMARY

HIGH-PERFORMANCE SOFTWARE FOR NANO-DIMENSIONAL ATOM-MOLECULE SYSTEMS MODELING

V.N. Vasilyev (vasilev@mail.ifmo.ru), A.V. Boukhanovsky (boukhanovsky@mail.ifmo.ru), S.A. Kozlov (kozlov@mail.ifmo.ru), V.G. Maslov (maslov@sp.ru), N.N. Rosanov (rosanov@nr3748.spb.edu)

Approach for building of next-generation high-performance software aimed at nanostructure and nanocomplex modeling within the bounds of iPSE concept is proposed. Proof of high-level architecture of this software is given; structure and purposes of main problem-oriented services are defined; issues of human-computer interaction organization, data- and knowledge management, results visualization are considered.
Keywords: nanostructure modeling, high-performance computing, scientific software architecture

CONCEPTION AND METHODOLOGICAL BASIS FOR INTELLIGENT GRID SYSTEMS BUILDING

Y.I. Nechaev (petr_oleg@mail.ru), A.V. Boukhanovsky (boukhanovsky@mail.ifmo.ru), V.N. Vasilyev (vasilev@mail.ifmo.ru)

The problem of intelligent Grid systems is discussed. The main attention is focused on formulating of conceptual basis and principals of data processing in complex dynamic environment. Formal description of logic space for knowledge model integration, including Neuro-Fuzzy systems and multi-agent technologies, is produced. Features of computing intelligence are considered, choice model, main definitions and axioms of intelligent Grid systems are formulated. Interpretation of Grid system as parallel computing architecture is given.
Keywords: Grid-computing, intelligent system, Fuzzy-logic

HIGH-PERFORMANCE GRID-APPLICATIONS DEVELOPMENT ENVIRONMENT. PART I: PRINCIPLES

A.V. Larchenko (aleksey.larchenko@gmail.com), A.V. Dunaev (anton.dunaev@gmail.com), A.V. Boukhanovsky (boukhanovsky@mail.ifmo.ru)

Grid-service-based high-performance applications development principles are discussed. A new design approach based on a special type of workflow is proposed. The use of Grid-applications developer decision support is well founded. Basic principles of intelligent decision support system working are described.
Keywords: grid, software design, intelligent system, high-performance computing

HIGH-PERFORMANCE GRID-APPLICATIONS DEVELOPMENT ENVIRONMENT. PART II: ARCHITECTURE, IMPLEMENTATION AND APPLICATION

A.V. Larchenko (aleksey.larchenko@gmail.com), A.V. Dunaev (anton.dunaev@gmail.com), A.V. Boukhanovsky (boukhanovsky@mail.ifmo.ru)

Issues to practical implementation of the decision support system for high-performance applications developer in PEG software family are discussed. Basic principles of the system working are described. Also assessments of composite applications effectiveness using task of climatic spectra approximation are presented.
Keywords: grid, software design, intelligent system, high-performance computing, development environment

HIGH-PERFORMANCE GRID-APPLICATIONS DEVELOPMENT ENVIRONMENT. PART III: KNOWLEDGE ACQUISITION AND FORMALIZATION

A.V. Dunaev (anton.dunaev@gmail.com), A.V. Larchenko (aleksey.larchenko@gmail.com), A.V. Boukhanovsky (boukhanovsky@mail.ifmo.ru)

Acquisition, knowledge representation and formalization of Grid services performance is discussed in this paper. Hierarchy of analytical models as determinate function of random variable (parameters of Grid environment) that represents the application working time is considered. Indirect approach to knowledge acquisition about performance using simulation modeling is introduced. Identifying method of parallel speedup probabilistic capabilities is presented. It is based on the knowledge about application working time distribution.
Keywords: grid, application performance, parallel speedup, knowledge engendering

HIGH-PERFORMANCE SOFTWARE FOR HYDROMETEOROLOGICAL EXTREME EVENTS SIMULATION. PART I: PROBLEM STATEMENT, MODELS, METHODS AND PARALLEL ALGORITHMS

A.V. Boukhanovsky (boukhanovsky@mail.ifmo.ru), O.I. Zilberstein (lmpi@yandex.ru), S.V. Ivanov (svivanov@mail.ifmo.ru), S.V. Kovalchuk (kovalchuk@mail.ifmo.ru), L.I. Lopatoukhin (leonid-lop@yandex.ru), S.K. Popov (lmpi@yandex.ru), M.M. Chumakov (lmpi@yandex.ru)

Issues of new concept implementation based on computer simulation for new extreme hydrometeorological events characteristics producing are discussed. General stages of calculation process, main features of modern approaches, numerical methods and parallel algorithms for hydrodynamic and statistic simulation of wind, waves, currents and sea level fields are described. Main calculation operations performance estimation is considered, a proof for supercomputer architecture choice is given.

Keywords: extreme events, hydrometeorological modeling, parallel algorithms

HIGH-PERFORMANCE SOFTWARE FOR HYDROMETEOROLOGICAL EXTREME EVENTS SIMULATION. PART II: SOFTWARE ARCHITECTURE DESIGN AND ASSESSMENT

S.V. Kovalchuk (kovalchuk@mail.ifmo.ru), A.V. Boukhanovsky (boukhanovsky@mail.ifmo.ru), S.V. Ivanov (svivanov@mail.ifmo.ru)

High-performance software for hydrometeorological extreme events simulation design and development peculiarities are considered within the bounds of Service-Oriented Architecture paradigm (SOA). It is shown by investigation using prototype software and parametric performance model, that parallel decomposition type for each of calculation modules of developing software is defined by task features and parallel hardware and software architecture specific character. The approach for dynamic performance-efficient organization of parallel software architecture by using intelligent management service is proposed. Results of architecture assessment using computing experiments are presented.

Keywords: high-performance computing, parallel software architecture, parallel performance management

HIGH-PERFORMANCE SOFTWARE FOR METOCEAN EXTREME EVENTS SIMULATION. PART III: CALCULATION RESULTS INTERPRETATION AND APPLICATION

S.V. Ivanov (svivanov@mail.ifmo.ru), S.V. Kovalchuk (kovalchuk@mail.ifmo.ru), L.I. Lopatoukhin (leonid-lop@yandex.ru), E.S. Chernysheva (chereks@yandex.ru), A.V. Boukhanovsky (boukhanovsky@mail.ifmo.ru)

Forms and ways of extreme hydrometeorological characteristics representation based on computer simulation are discussed. General types of extreme are considered: in point, in field, by months and by directions, multidimensional characteristics. Extreme wave height and wind speed estimation for the Barents Sea and the Caspian Sea is described.

Keywords: hydrometeorological extreme events, extreme characteristics atlas

INSTRUMENTAL ENVIRONMENT FOR MASS MOBILE NEW-AGE ONLINE-SERVICES. PART 1. OPERATING PRINCIPLE, SOFTWARE ARCHITECTURE

O.A. Zolotarev (zolotarev@tprs.ru), E.A. Grinina (grinina@tprs.ru), A.V. Boukhanovsky (boukhanovsky@mail.ifmo.ru)

The issue on distributed system software designing and engineering process automation for mobile commerce is considered. It is shown that mobile service architecture and operating principle concept is defined basically by e-Payment regulation character. Instrumental environment supporting construction process for mobile service, suitable for penetration into Russian mobile market, is developed.

Keywords: mobile service architecture, distributed system, instrumental environment

INSTRUMENTAL ENVIRONMENT FOR MASS MOBILE NEW-AGE ONLINE-SERVICES. PART 2. NFC TECHNOLOGY

E.A. Grinina (grinina@tprs.ru), O.A. Zolotarev (zolotarev@tprs.ru), I.A. Pimenov (ilya.pimenov@gmail.com), D.A. Nasonov (denis.nasonov@gmail.com), A.V. Boukhanovsky (boukhanovsky@mail.ifmo.ru)

This paper presents comprehensive study on issues concerning mobile online service based on NFC local communication technology. The mobile ticketing Id is considered as an example.
Keywords: mobile service, NFC, instrumental technological environment.

ADAPTATION OF CALCULATION ALGORITHMS TO PARALLEL GRAPHICS ACCELERATORS ARCHITECTURE

S.V. Kovalchuk (kovalchuk@mail.ifmo.ru), S.M. Vishnyakov (sergey.vishnyakov@gmail.com)

Issues on calculation algorithm mapping onto GPU-hardware parallel architecture are discussed in this work. Main features of considered hardware architecture and their impact on parallel algorithms performance are described. Performance estimation model for parallel algorithm using computational resources of graphics accelerator is proposed. Optimization of parametric function approximation using random search is taken as an example of calculation task.
Keywords: graphics accelerator, GPGPU, parallel algorithms

IDENTIFICATION OF PARAMETRIC BOUND MODELS OF COMPLEX SYSTEMS

S.V. Ivanov (svivanov@mail.ifmo.ru)

Approaches for complex systems models identification are considered in the paper. The distinctive features of models are parametric bound, multi-scalability and incompleteness of observations. Effectiveness of this approach is proved by several examples. Schemes of parameter identification for two models in system “ocean-atmosphere” and HIV population dynamic model are shown.
Keywords: model identification, complex systems

PARALLEL SIMULATION OF ASYNCHRONOUS CELLULAR AUTOMATA EVOLUTION

K.V. Kalgin (KalginKV@gmail.com)

For simulating physical and chemical processes on molecular level, asynchronous cellular automata with probabilistic transition rules are widely used being sometimes referred to as Monte-Carlo methods. The simulation requires a huge cellular space and millions of iterative steps for obtaining the CA evolution representing a real scene of the process. This may be attained by allocating the CA evolution program onto a multiprocessor system. We propose a new parallelization method of asynchronous CA based on its stochastic properties. The experiment results are presented.
Keywords: parallel algorithms, Monte-Carlo methods, asynchronous cellular automata, cellular automata

ARCHITECTURE OF SERVICE-ORIENTED DISTRIBUTED COMPUTED SYSTEM AND ITS REALIZATION ON THE BASE OF FRAMEWORK OSGI

A.V. Gavrilov (Andrey.gavrilov.samara@gmail.com), I.I. Dorovskih (irinsmile@yandex.ru), I.D. Krasinsky (IDKras@yandex.ru)

Development of distributed computing systems (DCS) is important for solving problems that require a great number of calculations. Analysis of existing widespread DCS has revealed several shortcomings. New DCS architecture is introduced that allows to avoid these shortcomings. The architecture is implemented in DCS based on OSGi framework. This system was used for solving particular problems and has shown its performance.
Keywords: distributed computing system, framework, resource distribution, resource planning, life-cycle, service-oriented systems

COMPARATIVE STUDY OF TABU-MACHINE AND HOPFIELD NEURAL NETWORKS APPLICATION TO SOLVE DISCRETE OPTIMIZATION PROBLEMS FROM DISTRIBUTED DATABASES DOMAIN

E.A. Babkin (babkin@hse.nnov.ru), M.E. Karpunina (levati@yandex.ru)

We propose a new neural networks approach based on tabu-search to solve discrete optimization problems arising during the synthesis of optimal distributed database logical structures. The solutions with a quality on

average greater by 3.8% than quality of the solutions, received with help of Hopfield networks, have been obtained. Thanks to optimization of Tabu-machine the operating time of data decomposition algorithm has been lowered in compare with an initial version of tabu-algorithm (on average more than twice) as well as with neural networks algorithm based on Hopfield networks (on average on 36%). In the course of the research the investigation of Tabu-machine parameters space has been carried out.
Keywords: tabu search, neural networks, genetic algorithms, discrete optimization problems, distributed databases.

DEDICATED NUMERATOR FOR RADAR SIGNAL PROCESSING

A.I. Grushin (aigrushin@ipmce.ru), M.L. Remizov (mlremizov@ipmce.ru), A.V. Rostovtsev (avrostovtsev@ipmce.ru), D.D. Nikolaev (ddnikolaev@ipmce.ru), Trinh Quang Kien (kkchin@ipmce.ru)

Real time complex matrix calculation is a typical radar signal processing problem. Implementation of recursive calculation of complex matrix 64x64 is considered in the paper.
Keywords: numerator, matrix, complex multiply add, floating point, FPGA

TECHNOLOGY OF VIRTUAL TEST DESKS CONSTRUCTING IN DISTRIBUTED COMPUTING ENVIRONMENT

G.I. Radchenko (gleb.radchenko@gmail.com), L.B. Sokolinsky (sokolinsky@acm.org)

We propose a technology of virtual test desks constructing in distributed computing environment based on a CAEBeans technology. We describe a structure organization of CAEbeans shells. We describe basic principles of CAEBeans system implementation.
Keywords: CAEBeans, Grid, distributed computing, Grid-services, hierarchies of problem-oriented shells

PARALLEL SOFTWARE DEVELOPMENT IN PARJAVA

A.I. Avetisyan (arut@ispras.ru), V.V. Babkova (barbara@ispras.ru), M.D. Kalugin (shaman@ispras.ru)

The goal of the paper is development of a scalable parallel program with the help of ParJava environment. The problems concerning program optimization in order to increase its scalability are discussed, and ParJava tools are presented.
Keywords: scalable parallel program development, high productivity cluster computing

EFFECTIVE UTILIZATION OF COMPUTATIONAL RESOURCES FOR BERMUDA OPTIONS RATIONAL PRICES ESTIMATION

A.S. Gorbunova (anna_nn@mail.ru), I.B. Meyerov (mib@uic.nnov.ru), A.S. Novikov (andrey.nikonov@itlab.unn.ru), A.V. Rusakov (andrey.rusakov@itlab.unn.ru), A.V. Shishkov (alekcac@gmail.com)

Our results in effective implementation of one algorithm of rational price determination for Bermuda options are reported in the paper. The approaches for optimization of calculations both in serial and parallel versions based on MPI and OpenMP are described. Computational results show linear speedup at clusters/SMP systems.
Keywords: Monte-Carlo methods, options pricing, parallel programming, optimization of calculations

SIMULATION INVIRONMENT FOR PERCOLATING PROBLEMS

E.N. Golovchenko (ge03@imamod.ru, ge03@imamod.ru), D.V. Petrov (bito@mail.ru)

Task preparation and launching on modern multiuser systems have a number of peculiar properties. Execution of cluster experiments is connected with necessity of multiple authorization, state control of files distributed among network resources, synchronization of stages of the sequences of commands execution and other difficulties. A simulation environment was created for simplification of the process of interaction with cluster systems. Using this environment water flooding of oilfield was carried out. Oilfield was approximated with percolating meshes with 10^6 – 10^{10} vertices. Steady and nonsteady states of influence on oilfield were explored. Diagrams of oil distribution, water, pressure and vacancy over the field were obtained.
Keywords: Mathematical simulation, parallel programming, percolation theory

SOFT DEVELOPMENT OF COMPETITIVE MARKET MODELING FOR COMPUTATIONAL CLUSTERS

M.Yu. Nesternko (nmu@mail.osu.ru), D.V. Leonov (d_leonov@list.ru), E.V. Bolgova (koten-ka@inbox.ru), A.S. Kirillov (kas-56@yandex.ru)

The approach to computational difficult problem – the finding of optimal coalition and gain distribution in n-players cooperative games is considered. The parallel algorithm of cooperative games solving is proposed. The testing results of algorithm program realization are analyzed.

Keywords: cooperative games, parallel programming

SIMULATION OF LARGE-SCALE VORTEX STRUCURES ON PARALLEL COMPUTERS

A.V. Babakov (babakov@icad.org.ru), P.A. Novikov (pavel_novikov@mail.ru)

The given work is devoted to studying of landing modules aerodynamics for supersonic and transonic modes of a running flow. The near wake structure, formation and interaction of large-scale vortical structures are investigated for hypersonic flows influence on aerodynamic characteristics of equilibrium and non-equilibrium chemical reactions.

Keywords: gas dynamics, landing spaceship, parallel algorithms

LOCALLY-REFINED RECTANGULAR MESHES

A.A. Sukhinov (Soukhinov@gmail.com)

Two-dimensional hierarchical locally-refined rectangular meshes with dynamic adaptation to the solution are considered in this paper. The mesh structure, adaptation criterion and parallel algorithm are described.

Keywords: locally-refined meshes, high-performance computing

INTERPRETATION AS RESEARCH METHOD FOR DYNAMICAL PROPERTIES OF PARALLEL PROGRAM ON INSTRUMENTAL COMPUTER

M.S. Akopyan (manuk@ispras.ru)

Analysis and tuning of parallel program on computational platform is a resource-intensive process. Our approach consists of the model simulation of parallel program using instrumental computer to analyze its dynamic characteristics.

Keywords: developer environment, Java, parallel programming

SIMULATION OF RESILIENT WAVE PROPAGATION THROUGH ENVIRONMENT OF MUD VOLCANO

D.A. Karavaev (dmitry1985@ngs.ru)

Method of solving the problem of numerical modeling of seismic waves propagation in the 3D models of inhomogeneous media is briefly presented in this paper. The method for constructing the 3D model of elastic media is described. The parallel implementation of the program for numerical modeling of elastic waves propagation is described. Some results of numerical modeling that can be used in interpreting geophysical field experiments with complicated media are presented.

Keywords: resilient waves, mud volcano, parallel algorithms

PARALLEL COMPUTING WITH «CELL» MICROPROCESSORS

S.M. Vishnyakov (sergey.vishnyakov@gmail.com), A.S. Mordvintsev (zzznah@gmail.com)

Adaptation of various kinds of computational tasks to CBEA architecture is considered in this work. Ray tracing through 3D-scene that has voxel representation is taken as example – an application that allows user to interact with 3D-scene is developed.

Keywords: Cell microprocessor, voxel graphics

STOCHASTIC SIMULATION OF COMPLEX NETWORKS

I.I. Kolyhmatov (kolyhmatov@rain.ifmo.ru)

Research devoted to design and analysis of parallel algorithms for stochastic simulation of complex networks with respect to heterogeneity and nonstationarity is represented in the article. Proposed algorithms were applied to several problems in mathematical epidemiology.

Keywords: complex networks, parallel algorithms

METROLOGICAL SUPPLY FOR DISTRIBUTED COMPUTATION AND SYSTEMS

A.V. Semyonov (avsemyonov@gmail.com), A.V. Boukhanovsky (boukhanovsky@mail.ifmo.ru)

Modern state of metrological supply for distributed computation and systems technologies problems is considered. Objects, methods and facilities used in the subject field are discussed from measurement unity supply point of view.

Keywords: metrological supply, distributed computing

GRAPHIC ACCELERATORS IN SUPERCOMPUTER MVS-EXPRESS FOR AEROGASDYNAMICS TASKS

A.A. Davydov (alexander.a.davydov@gmail.com)

There is a significant grow in graphic co-processors utilization during last years because of new programming tools emerging. However it's still a rare case when using it for parallel computing. The article discusses perspectives of this new research field on the example of MVS-EXPRESS supercomputer prototype built with graphic accelerators.

Keywords: aerogasdynamics, graphic co-processor

MAIN MEMORY SPEED ANALYSIS AND PC PERFORMANCE MODELING

K.S. Solnushkin (ks@spbstu.ru)

In this work main memory speed has been researched as a function with memory subsystem parameters: bus frequency and behavior mode of memory controller. Performance model has been introduced and approved by experiment. It determines linear dependence between PC performance and main memory speed.

Keywords: performance models, main memory speed

СОДЕРЖАНИЕ