МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ИЗВЕСТИЯ ВЫСШИХ УЧЕБНЫХ ЗАВЕДЕНИЙ

# ПРИБОРОСТРОЕНИЕ

ИЗДАНИЕ ГОСУДАРСТВЕННОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

Журнал издается с января 1958 г.

**ТОМ 52** **ОКТЯБРЬ 2009** **№ 10**

СПЕЦИАЛЬНЫЙ ВЫПУСК

# ТЕХНОЛОГИИ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ И КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ

*Под редакцией доктора технических наук, профессора А. В. Бухановского*

## СОДЕРЖАНИЕ

SPECIAL ISSUE

# TECHNOLOGIES OF HIGH-PERFORMANCE COMPUTING AND COMPUTER SIMULATION

*By Edition A. V. Boukhanovsky, Doctor of Technical Science, Professor*

## CONTENTS



*Editor-in-Chief E. B. Yakovlev*

# ПРЕДИСЛОВИЕ

Выпуск посвящен результатам работы II сессии научной школы-практикума молодых ученых и специалистов „Технологии высокопроизводительных вычислений и компьютерного моделирования“, которая проходила с 14 по 17 апреля 2008 г. в Санкт-Петербурге на базе Института наукоемких компьютерных технологий СПбГУ ИТМО в рамках VI Межвузовской конференции молодых ученых. В работе школы приняли участие более пятидесяти студентов, аспирантов и молодых ученых, активно применяющих высокопроизводительные вычисления при решении широкого круга задач науки, промышленности и бизнеса. „География“ участников охватила 24 научные и образовательные организации из 17 городов России (включая Москву, Новосибирск, Нижний Новгород, Вятку, Оренбург, Челябинск, Саратов, Уфу и др.), а также ближнее зарубежье (Минск, Белоруссия).

В первом и втором разделах выпуска представлены статьи, подготовленные по материалам лекций ведущих ученых и специалистов в области высокопроизводительных вычислений и компьютерного моделирования, представленных на Школе-практикуме. В третьем разделе представлены перспективные научные направления, которые не были в достаточной мере освещены в работе II сессии Школы-практикума, но могут представлять интерес для специалистов данной проблемной области.

Научная школа-практикум „Технологии высокопроизводительных вычислений и компьютерного моделирования“ включена в план регулярных мероприятий ежегодных Межвузовских конференций молодых ученых. Это позволяет нам анонсировать проведение III сессии научной школы-практикума молодых ученых и специалистов „Технологии высокопроизводительных вычислений и компьютерного моделирования“ на базе Института наукоемких компьютерных технологий СПбГУ ИТМО в апреле 2010 г.

*Доктор технических наук, профессор*
*А. В. БУХАНОВСКИЙ*

# INTRODUCTION

The issue covers results of 2nd session of Scientific Workshop for Young Scientists and Specialists "Technologies of High-Performance Computing and Computer Simulation", which was held from 14 till 17 of April, 2008, in Research Institute of Science-Intensive Computer Technologies, Saint-Petersburg State University of Information Technologies, Mechanics and Optics (Russia) during 6th Interuniversity Conference of Young Scientists. More than fifty students, Ph.D. students and young scientists, who actively apply high-performance computing for solving wide range of science, industry and business problems, participated in the event. Participants of the workshop came from 24 science and education organization from 17 cities of Russia (including Moscow, Novosibirsk, Nizhni Novgorod, Vyatka, Orenburg, Chelyabinsk, Saratov, Ufa etc.) and nearby countries (Minsk, Belarus).

Articles based on lectures by leading scientists and specialists in the field of high-performance computing and computer simulation taking place on the Workshop are presented in first and second sections of the issue. Articles on the subjects which were not fully presented on the 2nd session of the Workshop but supposed to be interesting for specialists in the concerned problem field are being announced within the 3rd section of the issue.

Scientific Workshop "Technologies of High-Performance Computing and Computer Simulation" had been included into annual events within the Interuniversity Conference of Young Scientists. According to that we are proud to announce the 3rd session of the Scientific Workshop for Young Scientists and Specialists "Technologies of High-Performance Computing and Computer Simulation" taking place in Research Institute of Science-Intensive Computer Technologies, Saint-Petersburg State University of Information Technologies, Mechanics and Optics on April 2010.

*Doctor of Technical Science, Professor*
*A. V. BOUKHANOVSKY*

# ТЕХНОЛОГИИ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ

УДК 681.3.069, 681.324

А. В. Бухановский, С. В. Ковальчук, С. В. Марьин

## ИНТЕЛЛЕКТУАЛЬНЫЕ ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ПРОГРАММНЫЕ КОМПЛЕКСЫ МОДЕЛИРОВАНИЯ СЛОЖНЫХ СИСТЕМ: КОНЦЕПЦИЯ, АРХИТЕКТУРА И ПРИМЕРЫ РЕАЛИЗАЦИИ

Рассматриваются вопросы применения интеллектуальных технологий в целях создания высокопроизводительного программного обеспечения для компьютерного моделирования, эффективно использующего ресурсы распределенных вычислительных систем различной архитектуры.

***Ключевые слова:*** *высокопроизводительные вычисления, база знаний, моделирование производительности, интеллектуальная система, сервисно-ориентированная архитектура.*

**Введение.** Развитие методов и технологий компьютерного моделирования стимулирует интерес исследователей к изучению так называемых сложных систем (complex systems). Сложной считается система, которая:

1) состоит из большого числа компонентов;

2) допускает „дальние“ связи между компонентами;

3) обладает многомасштабной (в том числе пространственно-временной) изменчивостью [1].

Понятие „сложная“ применительно к системе отражает не объективную сложность реального объекта, а скорее, методологическую сложность и уровень детализации сопоставляемой ему описательной модели. Например, классическая задача прогноза погоды [2] не предполагает описания сложной системы, в то время как климатическая система („ансамбль погод“), обладающая мелкомасштабной, синоптической, сезонной и межгодовой изменчивостью составляющих ее процессов, является сложной. С другой стороны, крайне ресурсоемкие расчеты из первых принципов характеристик атомно-молекулярных систем, состоящих из сотен и тысяч атомов [3], сами по себе к моделированию сложных систем не относятся. Однако их рассмотрение в процессе конструирования макрообъектов (материалов, процессов, устройств) с заданными характеристиками определяет сложную нанотехнологическую систему. Поведение сложных систем в силу перечисленных выше особенностей затруднительно изучать посредством физического эксперимента, поэтому основным способом их исследования в настоящее время является вычислительный эксперимент, выполняемый обычно на суперкомпьютерах.

Компьютерная реализация моделей сложных систем имеет свои специфические особенности. Так, вследствие большого числа компонентов системы увеличивается ресурсоемкость вычислительных процедур и повышаются требования к объему оперативной памяти для

хранения структур данных. Наличие „дальних“ связей критически сказывается на возможностях экстенсивного распараллеливания вычислений формальными методами и требует использования специфических алгоритмов декомпозиции. Наконец, учет дальних связей в системе затрудняет распараллеливание формальными методами. Многомасштабность сложных систем требует использования для их описания комплекса параметрически связанных моделей в смежных диапазонах изменчивости [4]. Как следствие, создание инструментария вычислительного эксперимента со сложными системами требует не только разработки оптимальных (для заданной вычислительной архитектуры) параллельных алгоритмов, но и построения иерархических схем организации вычислений. Такие схемы определяют процесс взаимодействия между одновременно или последовательно исполняемыми вычислительными модулями, каждый из которых, в свою очередь, может работать параллельно на одном или нескольких вычислительных комплексах различной архитектуры.

Таким образом, с точки зрения задачи оптимизации параллельной производительности высокопроизводительные программные комплексы моделирования сложных систем (ВПКМСС) сами могут интерпретироваться как сложные системы. Это естественным образом затрудняет процесс проектирования и разработки такого программного обеспечения. По-видимому, в настоящее время проблема проектирования, разработки и внедрения ВПКМСС не имеет устоявшегося решения; процесс разработки не автоматизируется и жестко ограничен спецификой соответствующей предметной области. В настоящей статье рассматривается общий подход к проектированию и разработке ВПКМСС на основе интеллектуальных технологий, совокупно учитывающих знания предметной области, специфику математических моделей, методов и алгоритмов, а также особенности параллельных архитектур вычислительных комплексов.

**Проблемы разработки композитных приложений для компьютерного моделирования сложных систем.** Сложную систему допустимо рассматривать покомпонентно; поведение каждого компонента детализируется в соответствующем ему диапазоне изменчивости. Поэтому в простейшем случае ВПКМСС представляет собой программную систему, интегрирующую в себе несколько приложений, взаимодействующих посредством потоков входных и выходных данных. При этом сами приложения могут быть разработаны различными авторами, основываться на разных информационных технологиях и предназначаться для моделирования только одного компонента системы под влиянием внешних возмущений. Назовем такие приложения композитными [5, 6], ориентируясь в первую очередь на их реализацию с использованием сервисно-ориентированного подхода [7].

Возрастающий интерес к композитным приложениям обусловлен не только простотой подхода, но и наличием соответствующей „элементной базы“ — достаточно надежных программных продуктов, предназначенных для решения задач компьютерного моделирования различных явлений и процессов. Практически во всех областях знания уже сформировались общепризнанные (эталонные) программные продукты, которые являются результатом интеграции знаний и навыков многих специалистов, работающих в данной области. Обычно такие продукты создаются на основе одного или нескольких успешных прототипов с последующим экстенсивным развитием и доработкой отдельных функциональных модулей. Они отличаются, прежде всего, наличием развитого сообщества пользователей, что порождает обилие публикаций с описанием опыта применения таких пакетов. Наличие такой обратной связи обеспечивает возможность их дальнейшей эволюции как независимых приложений с высокой (и отчасти — гарантированной) надежностью получаемых результатов.

Например, в задаче гидродинамического моделирования морского волнения, несмотря на многообразие существующих подходов, ведущие позиции удерживают только три спектральные модели третьего поколения: WAM, Wave Watch III и SWAN [8]. Несмотря на то что в гидрометеорологической практике успешно используются и другие, альтернативные, моде-

ли, результаты расчетов по ним воспринимаются основным потребителем — специалистом по проектированию и эксплуатации морских объектов — с некоторой настороженностью. В частности, в нормативном документе [9] указано, что расчеты, проводимые по любой из альтернативных моделей, для их практического использования должны быть верифицированы путем сравнения с одним из трех перечисленных выше „эталонов“. Аналогичная ситуация прослеживается и в других предметных областях, например, в области квантово-химических расчетов, традиционно отличающейся многообразием программных решений (около сотни единиц, имеющих сходный функционал). С классическими задачами здесь также устойчиво ассоциирован ряд эталонных пакетов, например, Gaussian, GAMESS, Molpro и Jaguar. Этот перечень более „размыт“ по сравнению с задачей моделирования морского волнения, что обусловлено как широкой областью их применения (от создания лекарств до конструирования полупроводниковых устройств), так и тем, что обычно пользователь на практике использует лишь ограниченное число программных решений. Лицензионная специфика здесь не является определяющей; существенно большая проблема заключается в сопоставлении данных и задач, реализуемых различными пакетами, которое может производить только высококвалифицированный эксперт в данной предметной области, имеющий большой опыт не только (и не столько) выполнения квантово-химических расчетов, но также их физической интерпретации и дальнейшего использования.

Проблема сопоставления существующих программных продуктов на уровне задач усугубляется, когда они рассматриваются как компоненты соответствующего композитного приложения — ВПКМСС, связанные потоками входных и выходных данных. Характерным примером задач, порождающих такие приложения, является моделирование волнения, течений и уровня моря на основе результатов моделирования динамики атмосферы [10], или расчет свойств материалов с использованием первопринципных квантово-химических расчетов их атомно-молекулярной структуры [11].

Сложность построения композитных приложений состоит не столько в обеспечении единого формата данных, сколько в том, что обычно для описания взаимодействия математических моделей на разных уровнях используются специфические замыкающие соотношения, содержащие (полу)эмпирические параметры, которые могут быть определены различными способами, в том числе из справочной литературы, с помощью экспериментов или расчетов в различных постановках. При этом невозможно заранее определить адекватный способ выбора — каким образом выходные данные одного компонента могут быть использованы в качестве входных данных для другого. В работе [12] рассмотрена проблема использования результатов расчета полей ветра (реанализ NCEP/NCAR) для моделирования полей морского волнения по спектральной модели SWAN на акватории Каспийского моря. Показано, что в силу региональной специфики обе модели не могут считаться совместимыми; для их стыковки необходимо введение дополнительной вычислительной процедуры калибровки и усвоения в поле ветра данных попутных судовых наблюдений. Таким образом, создание композитных приложений практически всегда связано с необходимостью разработки специальных компонентов сопряжения, осуществляющих пересчет, реорганизацию и форматирование данных для бесшовного взаимодействия предметно-ориентированных моделей. Эта задача является не столько технологической, сколько содержательной, поскольку метод пересчета должен быть согласован с математическими моделями, реализованными как в пакете-„доноре“, так и в пакете-„реципиенте“.

Задаче интеграции расчетных компонентов в составе композитного приложения присущи и технологические сложности. Это связано с тем, что существующее программное обеспечение компьютерного моделирования в различных предметных областях выполнено коллективами разработчиков разной квалификации с использованием различных технологий программирования; оно рассчитано на работу под управлением различных операционных

систем и имеет различные формы организации ввода—вывода. Совмещение такого программного обеспечения в рамках единого комплекса на основе традиционного компонентного подхода не представляется разумным в силу чрезвычайной разнородности его внутренней архитектуры. При этом следует принимать во внимание, что жизненный цикл программных продуктов компьютерного моделирования может составлять несколько десятков лет, что существенно превышает жизненный цикл информационных технологий, изначально используемых для их разработки. Как следствие, большинство таких продуктов с точки зрения информационных технологий можно назвать морально устаревшими, однако их перевод на современные платформы является столь ресурсоемким (а структура пакета — непрозрачной), что единственный практический способ их использования обеспечивают программные архитектуры с изоляцией на уровне приложений (например, SOA — сервисно-ориентированная архитектура).

**Проблемы применения высокопроизводительных вычислений для моделирования сложных систем.** В отличие от традиционных подходов к созданию композитных бизнес-приложений на основе SOA для программного обеспечения компьютерного моделирования принципиальная проблема состоит в том, что результатом интеграции должно являться не только обеспечение функциональных характеристик (решение вычислительной задачи с заданной точностью), но и достижение при этом максимальной параллельной производительности. Это связано с тем, что расчеты сложных систем являются крайне ресурсоемкими. Однако во многих случаях функциональное (по фрагментам решаемой задачи) распараллеливание, реализуемое в компонентах композитного приложения, даже на кластерах с быстрой коммуникационной сетью эффективно на весьма небольшом количестве вычислителей (16—32). Как следствие, эффективное распараллеливание на значительном количестве вычислителей (и даже на нескольких суперкомпьютерах) может быть реализовано в подавляющем большинстве по данным, например, путем независимых запусков пакета для разных наборов параметров (что, например, характерно для практики квантово-химических расчетов). Именно поэтому проблема создания ВПКМСС, эффективно использующих ресурсы современных суперкомпьютерных систем, является не технологической, а предметно-ориентированной. Механизмы распараллеливания задачи по данным исходя из специфики предметной области (именуемые естественными [13]) могут быть сформулированы только экспертом на уровне сценария запусков компонента. При этом наравне с распараллеливанием по данным может быть реализовано внутреннее функциональное распараллеливание, что позволяет более эффективно использовать вычислительные ресурсы в рамках иерархической схемы. Однако в данном случае принципиальной проблемой является достижение баланса между использованием в программном комплексе внутренних и внешних механизмов распараллеливания. Для наиболее простых приложений эта задача может быть решена на этапе проектирования ВПКМСС путем задания соответствующей статической архитектуры. Например, в статье [14] на основе задачи моделирования экстремальных гидрометеорологических явлений показано, что соотношение между количеством вычислительных процессов и потоков может быть сложной функцией от входных данных и характеристик вычислительного комплекса, неизвестной априори. Как следствие, это порождает программную систему с динамической архитектурой, параметры которой должны определяться непосредственно во время исполнения.

Другой класс программных систем, изначально обладающих динамической архитектурой, представляют собой проблемно-ориентированные оболочки (PSE — Problem Solving Environment). PSE — это программный комплекс, предоставляющий пользователю все необходимые для решения определенного класса задач вычислительные средства: методы решения задач данной предметной области, способы выбора методов, а также способы добавления новых методов решений [15]. Помимо этого, PSE может предоставлять возможность выбора используемых вычислительных устройств, а также вести мониторинг вычислительного процесса.

Концепция PSE активно развивается с середины 1980-х гг. Первые PSE представляли собой монолитные программы, предназначенные для определенных вычислительных задач (ELLPACK, MATLAB), но в дальнейшем их все чаще стали компоновать из унифицированных модулей, решающих различные задачи. Среди PSE можно выделить специфические программные системы распределенных вычислений, такие как квантово-химическая система Ecce [16], система моделирования роста кораллов Morphogenesis PSE [17], система поддержки принятия решений в области эпидемиологии [18], а также система PROTEUS для нужд биоинформатики [19]. Последние две системы имеют в своем составе экспертные подсистемы, которые позволяют пользователю строить описание задач на языке предметной области (например, на основе заданной онтологии понятий), не используя специфических навыков работы с программными или аппаратными средствами PSE.

Предметно-ориентированные модули в составе PSE обычно разрабатываются различными специалистами, реализуют разные принципы распараллеливания, написаны на разных языках программирования и функционируют на удаленных системах принципиально различной архитектуры. Поэтому построение оптимального с точки зрения производительности сценария выполнения вычислений (что эквивалентно динамическому созданию композитного приложения) является нетривиальной задачей. В существующих системах она обычно отдается „на откуп“ пользователю, например, возможен выбор целевой вычислительной системы, мониторинг ее параметров, а также возможно предоставление некоторой элементарной информации, например, оптимального количества вычислителей для каждого из модулей [19]. Следует отметить, что в таких системах эффективность решения задачи конкретного пользователя не является самоцелью. Это обусловлено, по-видимому, объективным фактом их ориентации на исполнение в глобальных Грид-средах: в Грид политика пользователя по обеспечению максимальной производительности собственного приложения является конкурирующей с политикой, продвигаемой системными службами ресурсов Грид, ориентированной на эффективное использование всей среды в целом. Однако положение принципиально меняется в том случае, когда реализуется модель метакомпьютинга: пользователь может сам определять режим запуска вычислительных модулей на различных вычислительных ресурсах исходя из того, чтобы все композитное приложение выполнилось за минимальное время. Как следствие, это требует построения оптимального расписания исполнения задач на иерархической многоуровневой архитектуре (кластер—узел—процессор—ядро) с нестационарным поведением, вызванным ее использованием в немонопольном режиме. Поэтому задача оптимизации загрузки ресурсов уже не может быть решена эффективно исходя только из принципов параллельных вычислений [20] — требуется использовать специфический аппарат на основе априорных знаний предметной области.

**Концепция iPSE.** Учитывая общую сложность задачи проектирования и разработки предметно-ориентированных ВПКМСС, допустимо подходить к ее решению с позиций интеллектуальных технологий, основанных на симбиозе отчуждаемых знаний предметной и проблемной (в данном случае — высокопроизводительные вычисления) областей. Несмотря на новизну такой постановки уже существует успешный опыт применения интеллектуальных технологий в смежных направлениях. В частности, проблема отчуждения „знаний“ о решаемой задаче от архитектуры телекоммуникационной системы породила теорию интеллектуальных сетей электросвязи [21]. В высокопроизводительных вычислениях системы низкоуровневого управления параллельными процессами используются при разработке специфических классов вычислителей [22]. Кроме того, существуют и экспертные (советующие) системы поддержки принятия решений для разработки параллельного кода [23]. Однако все эти решения основаны на знаниях в рамках заданной предметной области (информационно-телекоммуникационные технологии) и не претендуют на проблемно-ориентированные обобщения предметных знаний.

Концепция iPSE (Intelligent Problem Solving Environment) [11] является развитием логики PSE и определяет принципы построения ВПКМСС как интеллектуальной оболочки управления параллельными вычислительными процессами в распределенной иерархической среде, включающей в себя вычислительные системы различной архитектуры. Такой подход обеспечивает эффективное параллельное исполнение композитных приложений в силу того, что использует для управления параллельными вычислениями симбиотические знания об особенностях предметной области и специфике вычислительного процесса. Перечислим достоинства подхода.

— 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 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 выполнением параллельных вычислений является прерогативой iPSE, что позволяет избежать конфликтных ситуаций при разделении ресурсов между различными вычислительными модулями и разными пользователями.

Как программный комплекс оболочка iPSE может быть отнесена к классу интеллектуальных систем, обладающих сложной распределенной структурой (структурная сложность), многоцелевым характером преобразования информации (функциональная сложность), а также ориентированных на учет и формализацию неопределенности (информационная сложность) [24].

Оболочка iPSE функционирует в режиме реального времени, общая схема обработки информации следующая:

— постановка задачи и анализ имеющейся информации (оценка адекватности информации, выбор параметров, типа модели, критериев оценки и стратегии ее построения);

— задание конкретного сценария модельного расчета; анализ информации, получаемой от различных источников с целью идентификации параметров и начальных условий модели;

— выбор метода (алгоритма, модуля) для расчета модели в зависимости от целей и задачи моделирования и особенностей сценария;

— анализ результатов расчета в соответствии с принятыми критериями, выделение множества альтернатив (парето-оптимального множества сценариев);

— выделение оптимальных сценариев на основе проведения многокритериальной оптимизации на полученном множестве альтернатив.

Анализ альтернатив и принятие решений осуществляются на основе предварительной информации, математических моделей и структурированной базы знаний с использованием различных методов обработки результатов мониторинга вычислительной среды с учетом неполноты и неопределенности данных (например, в силу немонопольного использования системы или по причине наличия управляющего фактора высшего уровня — в среде Грид). При этом система совмещает различные подходы к представлению знаний — декларативный и процедурный. Декларативная часть системы обеспечивает описание допустимых возможно-

стей, а процедурная — организацию доступа к данным, реализацию вычислительных алгоритмов, интерфейс пользователя.

**Формальная модель iPSE как интеллектуальной системы.** Одной из центральных проблем разработки интеллектуальных систем, к которым относится iPSE, является формирование инвариантного ядра системы, включающего в себя предметную область, базу знаний и базу данных. Для создания этой совокупности формируется концептуальная модель системы [25]. В функциональном аспекте эта модель включает следующие компоненты:

$$\langle S(F), S(M), S(W), P_F, T_F, A, X, Y \rangle. \quad (1)$$

Первые три компонента кортежа (1) являются структурами, заданными на множестве элементов $S_i \in S$, которые соответствует основным компонентам программной системы. Так, $S(F) = \{S_{F_1}, \ldots, S_{F_N}\}$ — совокупность функциональных подсистем, определяющих основные возможности программной системы; $S(M)$ — структурная схема системы, включающая множество $M \equiv Set(S(M))$ ее компонентов и имеющая собственную организацию $Org(S(M))$; $S(W)$ — условия формирования целостной системы (цели функционирования, принципы и алгоритмы управления, качество результата решения задачи и эффективность). Структурная схема системы $S(M)$ в (1) определяется компонентами:

$$S(M) = \langle M, C, T_M \rangle, \quad (2)$$

где $C$ — множество, определяющее совокупность связей между элементами в соответствии с организацией $Org(S(M))$; $T_M$ — множество моментов времени, определяющих последовательность взаимодействия между компонентами. Таким образом, управляя составом $T_M$, можно строить на основе указанных компонентов различные динамические архитектуры. Далее, в рамках семиотического подхода [26] множество элементов $M$ в структуре $S$ можно описать следующим образом:

$$M = \langle \{m_i\}, I(M), F(M), Q(S) \rangle. \quad (3)$$

Здесь $\{m_i\}$ — множество формальных или логико-лингвистических моделей, реализующих заданные интеллектуальные функции в рамках данного элемента; $I(M)$ — функция выбора необходимой модели (совокупности моделей) в текущей ситуации; $F(M)$ — множество функций модификации моделей $m_i$; $Q(S)$ — функция (множество функций) модификации системой $S$ ее базовых компонентов $I(M), F(M)$. Правила задания (3) определяют характеристики адаптивной составляющей iPSE, отвечающей за вопросы расширения функциональности, получения новых знаний и приобретения знаний на основе накопленных данных.

Условия формирования системы $S(W)$ представляются совокупностью

$$S(W) = \langle G, R, U_R, K_R, E_G \rangle, \quad (4)$$

где $G$ — цели, обеспечивающие реализацию задачи $R$; $U_R$ — принципы и алгоритмы управления объектом разработки; $K_R$ — качество результата решения задачи; $E_G$ — эффективность достижения цели $G$.

В рамках концепции iPSE задача $R$ формулируется непосредственно пользователем на языке предметной области; целью системы, обеспечивающей решение данной задачи, является выполнение вычислений при фиксированном наборе параметров (например, точности расчетов, задаваемых в $K_R$) не более чем за время $T \in T_F$. При этом принципы $U_R$ определяют процесс композитного приложения, которое бы при указанном наборе параметров наиболее эффективно выполнялось на заданной архитектуре, а в качестве показателя эффективности $E_G$ может, в частности, использоваться классическое понятие параллельной эффективности.

В кортеж (1) также входят параметры, определяющие динамические характеристики iPSE: $P_F$ — множество функциональных параметров и $T_F$ — множество моментов времени, инвариантных объектам моделирования, уровню их организации и предметной области. Множество $P_F$ представляет собой наборы функциональных параметров для совокупностей $(X, Y, A)$

$$P_F^X = \left\{\pi_F^X\right\},\ P_F^Y = \left\{\pi_F^Y\right\},\ P_F^A = \left\{\pi_F^A\right\} \tag{5}$$

и подразделяется по области действия на локальные и глобальные параметры. Локальные параметры ($P_F^X, P_F^Y$) являются численными характеристиками отдельного блока (функциональной подсистемы $S_{F_i}$), сфера действия которых ограничена математической моделью этого блока. Глобальные параметры $P_F^A$ представляют собой характеристики всей системы либо ее нижних уровней иерархии (макроблоков). Параметры любого типового блока задаются в виде выражений, содержащих глобальные параметры.

Оператор $A: X \to Y$ определяет процесс интерактивного взаимодействия „пользователь (оператор)—ЭВМ“ при функционировании системы $S(F)$; $\mathbf{X} = X_j \left(j = \overline{1, n}\right)$ и $\mathbf{Y} = Y_i \left(i = \overline{1, m}\right)$ — вектор-множества входных и выходных данных iPSE. Эти данные включают в себя требования к структуре и функционалу разрабатываемого композитного приложения, представляемые на языке предметной области, а также собственно входные и выходные данные — результаты компьютерного моделирования.

Формальная модель (1)—(5) позволяет привести описание iPSE как программной оболочки в соответствие со стандартом IDEFO описания интеллектуальных систем общего плана. При этом концепция iPSE расширяет спецификацию стандарта, поскольку является системой распределенного искусственного интеллекта. Эта система сочетает строгие формальные методы с эвристическими методами и моделями, базирующимися на знаниях экспертов, моделях рассуждений, имитационных моделях, накопленном опыте эксплуатации. Помимо традиционных для систем интеллектуальной поддержки механизмов в состав iPSE входят компоненты имитации, анализа и прогноза проблемной ситуации (моделирования), организации различных видов интерфейса. К интеллектуальным также относятся механизмы поиска решения на базе моделей и методов представления знаний.

**Представление знаний в iPSE.** В состав функциональных параметров (5) входят формализованные данные и знания предметной и проблемной областей. При этом знания проблемной области (высокопроизводительные вычисления) опираются на знания различных предметных областей. Знания в общем случае являются переменной во времени и контексте совокупностью отношений между данными. Непрерывный процесс изменения знаний обеспечивает реализацию контекстной связи данных. База знаний содержит сведения, представленные в виде моделей знаний, которые отражают закономерности предметной области и позволяют прогнозировать и выводить новые факты, не отраженные в базе данных.

Для каждой из предметных областей в рамках iPSE знания определяются как кортеж

$$<Q, R, A, P>, \tag{6}$$

где Q — множество объектов решаемой задачи; R — множество реализаций отношений между объектами множества Q; A — множество действий, которые выполняются с элементами множеств Q и R; P — ресурсы, необходимые для решения задач (логического вывода) на основе знаний. Кортеж (6) несмотря на близость интерпретации с (4) является самостоятельным объектом, поскольку относится к знаниям, в общем случае отчуждаемым от системы (1)—(5). Объекты Q в основном относятся к категории декларативных знаний; для их представления используется комплексная онтология [27]; в общем случае рассматривается иерархическая структура представления знаний. Например, декларативные знания метауровня — это мета-

онтология проблемной области. Метаонтология включает в себя предметные онтологии в виде понятий конкретной предметной области и отношений, семантически значимых для этой области, а также множество интерпретаций этих понятий и отношений (декларативных и процедурных). Онтология в качестве понятий определяет типы решаемых задач, а отношения этой онтологии — декомпозицию задач или подзадач.

Процедурные отношения между знаниями являются основным инструментом для выполнения логического вывода и обоснования принимаемых решений. В частности, решая задачу построения композитного приложения, в заданных условиях обеспечивающего наибольшую параллельную производительность, необходимо принимать решения, основываясь на наборе знаний, характеризующих 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, динамическими изменениями вычислительной среды, а также вариабельностью характеристик приложения в зависимости от вводимых пользователем параметров вычислений.

Основным способом представления знаний о производительности вычислительных модулей в iPSE является параметрическая модель производительности. Эту модель (т.е. время работы параллельного приложения в зависимости от характеристик входных данных и параметров вычислительной системы) можно представить как сумму времени вычислений, времени на обмен данными между узлами вычислительного комплекса (время коммуникаций), а также временных затрат на простои и различные сервисные функции. В общем случае время работы параллельного приложения зависит от особенностей алгоритма, применяемой технологии параллельного программирования и характеристик вычислительной системы.

Для формализации и унификации знаний о параллельной производительности в iPSE использован фреймовый подход [24], в котором знания представляются в виде структуры фреймов. Структура знаний о вычислительных модулях представляется в виде дерева И-ИЛИ фреймов. В процессе работы механизма вывода система, оперируя знаниями экспертов, на основании имеющихся данных производит оценку весов ребер и вершин графа параллельных реализаций.

Таким образом, набор параметров можно представить в виде

$$\Xi=(\Xi_S,\xi_A(\Xi_S),\xi_D(D)), \tag{7}$$

где $\Xi_S$ — параметры системы; $\xi_A$ — функция, определяющая параметры параллельной реализации алгоритма для текущей системы; $\xi_D$ — функция, определяющая параметры конкретного набора данных). Фрейм, описывающий реализацию вычислительного модуля, можно представить в виде тройки, каждый из элементов которой соответствует узлу-потомку дерева:

$$\Phi_M=(I_R(D^*),\Phi_D^*,\phi(\Xi,D^*)), \tag{8}$$

где $I_R$ — императивное описание процесса запуска; $\Phi_D$ — набор ссылок на фреймы, описывающих данные, которыми оперирует вычислительный модуль; $\phi$ — модельная оценка времени работы вычислительного модуля при определенном наборе параметров. Описание данных представляется в виде:

$$\Phi_D=(\mu,I_{dc}(D),I_c(D^*),\phi_{dc}(\Xi,D),\phi_c(\Xi,D^*),\xi_D(D)). \tag{9}$$

Здесь $\mu$ — метаданные, включающие обязательную информацию для идентификации формата данных и их предназначения (смысла); $I_{dc}$ и $I_c$ — описание процесса композиции и декомпозиции данных, представляющих отображения

$$I_{dc}: D \to D^* \text{ и } I_c: D^* \to D; \quad (10)$$

$\phi_{dc}$ и $\phi_c$ — модельная оценка времени композиции и декомпозиции.

Для оценки весов ребер (как взаимосвязей между вычислительными модулями) используется механизм продукции на основании правил. Формально продукцию можно представить в виде

$$\Phi_T = \phi\left(\Phi_M^{(1)}, \Phi_M^{(2)}, D^{(1)}, D^{(2)}\right). \quad (11)$$

Данная функция описывает переход между двумя реализациями, знания о которых хранятся соответственно в фреймах $\Phi_M^{(1)}$ и $\Phi_M^{(2)}$ при работе с выходными данными $D^{(1)}$, полученными от первой из них и требуемыми для работы второй — данными $D^{(2)}$.

На основании предложенной структуры фреймов строится база знаний экспертов предметной области, предназначенная для управления вычислительными модулями, используемыми в процессе работы. Для упрощения структуры базы возможно использование иерархии экземпляров фреймов: такой подход позволит избежать дублирования знаний, одинаковых для всех дочерних фреймов. Так, фрейм, использующий закон Амдала в качестве значения для слота $\phi(\Xi, D^*)$ во фрейме (9), может послужить предком для описания многих экспертных знаний, использующих тот же закон в качестве модели времени параллельного выполнения.

**Архитектура программного комплекса на основе iPSE.** На рис. 1 приведена типовая архитектура iPSE в рамках концепции, описанной выше. Система состоит из шести основных подсистем: логического вывода, управления знаниями, человеко-компьютерного взаимодействия, оболочки параллельного исполнения вычислительных модулей, хранилища данных и информационного портала.

Подсистема человеко-компьютерного взаимодействия предоставляет пользователю когнитивный диалоговый интерфейс, который включает в себя модуль интервьюирования пользователя, конструктор сценариев исполнения (т.е. разрабатываемых пользователем композитных приложений) и интеллектуальный редактор данных, посредством которого пользователь может формировать входную информацию для расчетов, максимально используя понятийную базу и справочную информацию соответствующей предметной области. Дополнительно в состав интерфейса входит так называемый интеллектуальный инструктор, т.е. модуль, отслеживающий и анализирующий действия пользователя с целью возможной коррекции или оптимизации его действий. В подсистему входят модуль научной визуализации, а также модуль валидации и верификации результатов расчетов. Его назначение — используя знания предметной области, формировать соответствующие тестовые задания для оценки работоспособности разрабатываемых пользователем композитных приложений. Таким образом, подсистема позволяет реализовать разнообразные функции ввода и вывода данных в программном комплексе, позволяет пользователю формировать и запускать задания, осуществляет визуализацию результатов расчетов, а также обеспечивает доступ к хранилищам соответствующих данных.

Подсистема управления знаниями включает в себя набор баз знаний предметной и проблемной областей. Взаимоотношения между понятиями из разных баз знаний регламентируются посредством метаописания их структуры и состава в форме комплексной онтологии. В частности, в рамках онтологии определены приемлемые формы представления параметров (9), включая вид и структуру моделей параллельной производительности. Адаптивные функции данной подсистемы реализуются с помощью модуля получения новых знаний. Он объединяет в себе интеллектуальный редактор знаний, позволяющий осуществлять непосредственное взаимодействие с экспертами, компонент приобретения знаний из внешних источни-

ков, например, посредством технологий web-mining, а также компонент ретроспективного анализа результатов предыдущих запусков с целью самообучения системы.

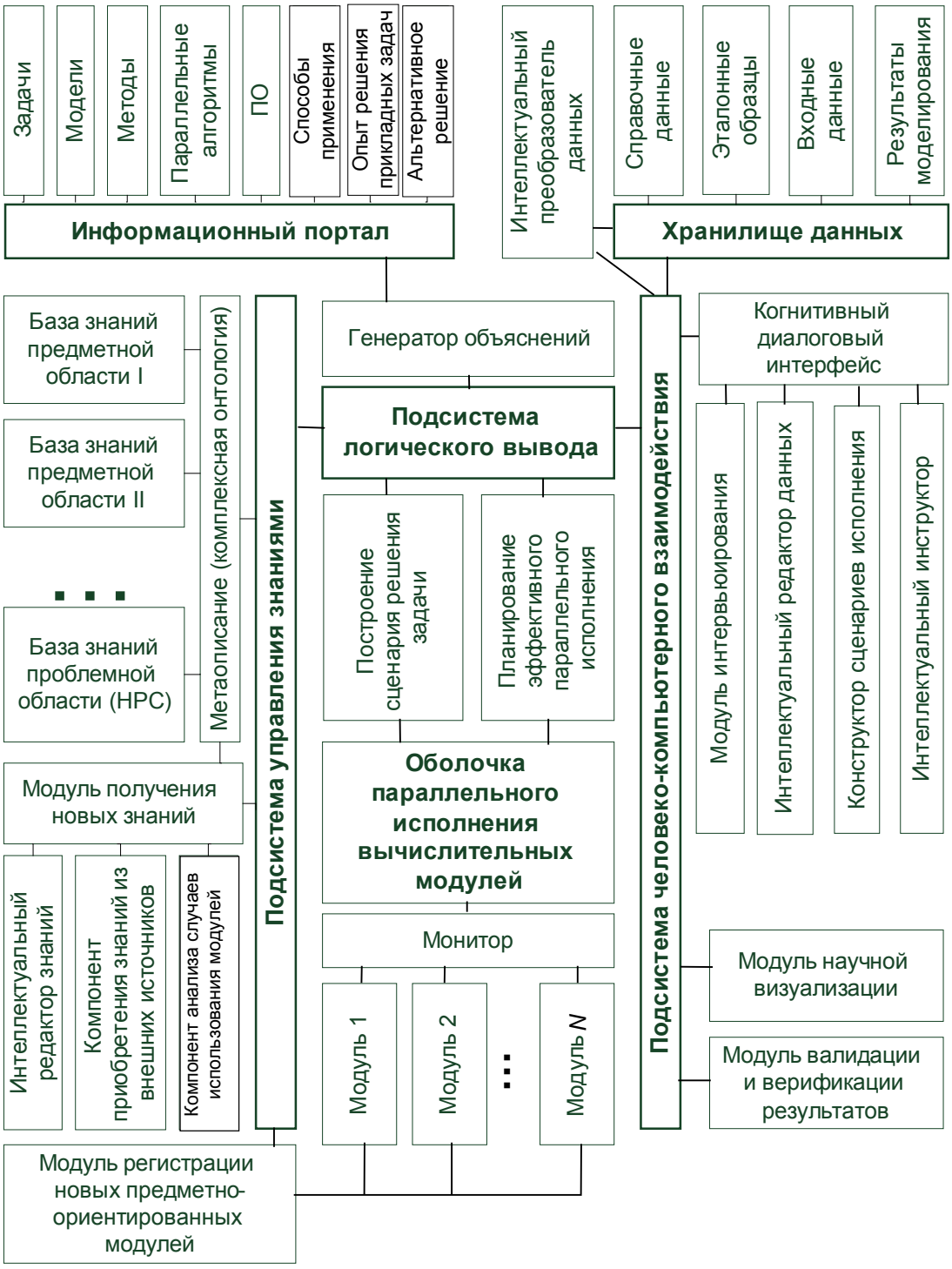


*Рис. 1*

Подсистема логического вывода является по сути основным движущим механизмом интеллектуальной оболочки, реализующей вывод на основе знаний в рамках структуры (1)—(5). Она включает в себя компоненты построения сценариев решения задачи на основе знаний предметной области и планировщик параллельного исполнения, применяющий знания о производительности в форме (6)—(11). На основе этих знаний подсистема интерпретирует задание (метаописание которого предоставляется 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 вычислительного комплекса.

Оболочка параллельного исполнения вычислительных модулей является основным содержательным элементом программного комплекса. Она представляет собой самостоятельную систему метакомпьютинга, которая обеспечивает распределенный запуск заданий и мониторинг их исполнения на наборе высокопроизводительных вычислительных комплексов, объединенных общей высокоскоростной сетью. В ходе работы компонентов блока не формируется сама стратегия управления — блок лишь транслирует управляющие команды, поступающие от интеллектуальной системы конструирования заданий. Оболочка параллельного исполнения обеспечивает унифицированный доступ к вычислительным предметно-ориентированным модулям в составе комплекса. Каждый такой модуль помимо процедурной части (самого программного модуля) имеет декларативную составляющую, содержащую подробную информацию о заложенных в нем математических моделях, методах, алгоритмах и условиях применения, как фрагмент соответствующей базы знаний. И процедурная, и декларативная составляющие компонента могут храниться на разных узлах распределенной вычислительной системы. Вся совокупность зарегистрированных в комплексе вычислительных компонентов с ассоциированными им декларативными описаниями составляет, таким образом, репозиторий прикладных сервисов. Помимо содержательных сервисов в состав блока входят и сервисы-шлюзы для интеграции с независимым программным обеспечением других разработчиков (включая системы с закрытым кодом). Дополнительно в состав оболочки входит модуль регистрации новых предметно-ориентированных сервисов.

Хранилище данных связано с остальными подсистемами посредством интеллектуального преобразователя данных. Интеллектуальная функция этого модуля обусловлена тем, что при преобразовании форматов входных и выходных файлов для различных модулей обеспечивается их эквивалентность на уровне решаемых задач, что требует использования соответствующих декларативных знаний о вычислительных модулях. Хранилище данных содержит собственно входные и выходные данные (результаты моделирования), справочные данные, а также эталонные образцы, необходимые для решения задач верификации и валидации.

Информационный портал является специфической подсистемой iPSE, которая заменяет собой традиционное „Руководство пользователя“. Портал представляет собой динамическую гипертекстовую информационную систему с перекрестной классификацией содержания по следующим категориям понятий: задачи, модели, методы, алгоритмы, программное обеспечение, применение, опыт использования и альтернативы. Пользователь в силу специфики своего восприятия может осуществлять навигацию в различных направлениях, в итоге получая рекомендации по конкретному использованию возможностей комплекса для решения собственной задачи. Дополнительным (и необходимым для интеллектуальной оболочки) компонентом является генератор объяснений, который связывает действия, выполняемые в процессе логического вывода, с содержимым информационного портала, что позволяет предоставить пользователю аргументированную информацию обо всем процессе рассуждений системы.

С учетом масштабов ВПКМСС в рамках iPSE традиционный подход к отчуждению такого программного обеспечения путем тиражирования на электронных носителях видится нецелесообразным; он может использоваться через Интернет в рамках модели ASP (Application Service Provider). Пользователи получают доступ к программному продукту посредством стандартного web-браузера или — для проблемно-ориентированных задач — специального графического клиента. При этом физические особенности исполнения пакетов и хранения данных от пользователя скрыты. В рамках модели ASP все затраты на поддержание работо-

способности комплекса, своевременное обновление и модификацию его компонентов ложатся на организацию-оператор. В рамках рассматриваемой модели метакомпьютинга такая схема, по-видимому, является единственно перспективной.

**Примеры реализации концепции iPSE.** *Высокопроизводительный программный комплекс моделирования экстремальных гидрометеорологических явлений ME²SIM*. В инженерной практике экстремальные гидрометеорологические явления характеризуются расчетными сочетаниями скорости ветра, параметров волнения, скорости течений и уровня моря, возможными один раз в $T$ лет, где $T$ соответствует классу сооружения. Современная концепция получения информации об экстремальных гидрометеорологических явлениях основана на синтетическом подходе: на основе упорядоченных массивов метеорологической информации за несколько десятков лет выполняется гидродинамическое моделирование полей течений, морского волнения и уровня моря. Эти данные используются для идентификации стохастической модели, на основе которой выполняется экстраполяция расчетных характеристик на период повторяемости $T$. Практическая реализация концепции расчета характеристик экстремальных гидрометеорологических явлений требует сочетания в одном программном комплексе набора взаимосвязанных функциональных компонентов, отвечающих за основные этапы гидродинамического и статистического моделирования. 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.

Программный комплекс ME²SIM [28] является не оболочкой iPSE, а скорее, композитным приложением, в состав которого входит модуль, реализующий некоторые интеллектуальные функции. Это обусловлено тем, что в решаемой задаче состав программных компонентов и последовательность их взаимодействия являются строго определенными. Поэтому цели использования iPSE в данном случае сводятся только к интеллектуальному управлению параллельными вычислительными процессами для формирования оптимальной динамической архитектуры. Это позволяет автоматизировать процесс принятия решения, основываясь на наборе знаний, определяющих возможности используемых вычислительных сервисов, а также данных, характеризующих как решаемую в данный момент задачу, так и используемую программно-аппаратную вычислительную среду. Оптимальность решения определяется 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 и пропускной способности каналов и пр.).

На рис. 2 продемонстрирован общий принцип работы интеллектуальной составляющей программного комплекса и представлены ее основные компоненты. В процессе работы механизма вывода система, оперируя знаниями экспертов (включенных в описание конкретных вычислительных сервисов), на основании имеющихся данных производит оценку весов ребер и вершин графа параллельных реализаций в форме (6)—(11), варьируя варианты распараллеливания отдельных вычислительных моделей. Варианты распараллеливания каждой из моделей применяются с использованием технологий параллельного программирования различных уровней (управление потоками и процессами). Каждому из вариантов распараллеливания соответствует своя специфика поведения кривой ускорения на используемой архитектуре. В силу неопределенности во входных данных в результате анализа выявляются несколько конкурирующих, или „допустимых“, путей, которые потом ранжируются.

Применение интеллектуальных технологий для построения оптимальной динамической архитектуры позволило существенно повысить параллельную производительность расчетов. Например, для акватории Каспийского моря на 128 узлах эффективность работы такого композитного приложения составила около 70 %, в то время как без использования интеллектуальной составляющей композитное приложение становится неэффективным уже на 32 вычислителях.

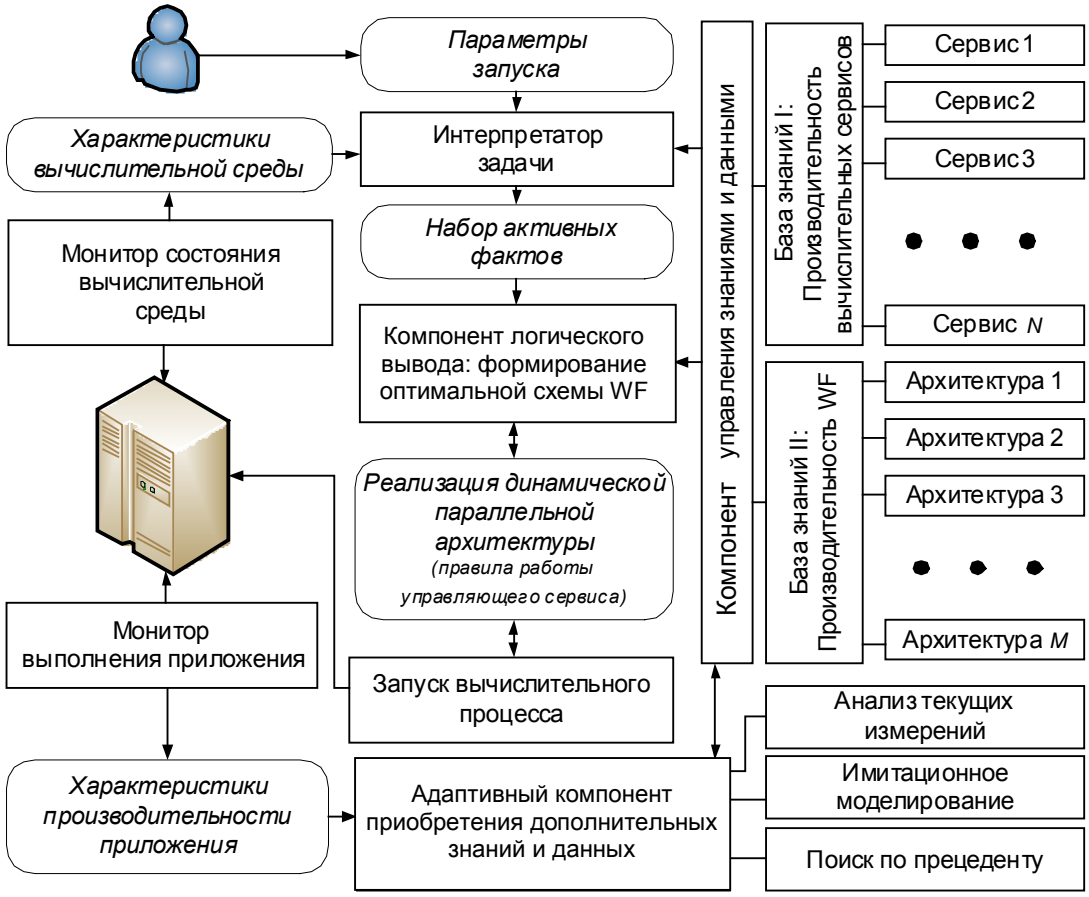


*Рис. 2*

*Инструментальная оболочка проектирования высокопроизводительных приложений в среде Грид iPEG*. Инструментальная оболочка iPEG [29] представляет собой интеллектуальную систему поддержки принятия решений разработчиков приложений в распределенных вычислительных средах. В отличие от традиционных систем визуального прототипирования iPEG обеспечивает 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 (выполнять операции декомпозиции, связывания, агломерации) посредством визуального проектирования с использованием графического языка. Системой выполняется мониторинг вычислительной среды с целью выяснения различных ее характеристик, таких как пропускная способность сети, вычислительная мощность отдельных 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. В соответствии с расписанием производится автоматическая генерация композитного приложения, а также сопутствующих файлов, необходимых для выполнения заданного приложения в Грид.

Поскольку в общем случае преимущества выбора механизмов (или их комбинации) формирования оптимальной архитектуры не очевидны вследствие относительно слабой формализации требований и ограничений, изменчивости внешней среды (в данном случае — инфраструктуры Грид) и неопределенности характеристик задачи, оболочка iPEG решает задачу проектирования средствами искусственного интеллекта. Формализация проекта выполняется посредством нотации потока заданий (workflow, WF) в трех представлениях: мета-WF (MWF), абстрактный WF (AWF) и конкретный WF (CWF), который соответствует композитному приложению.

На рис. 3 приведена концептуальная схема, иллюстрирующая принцип действия интеллектуальной составляющей оболочки iPEG.

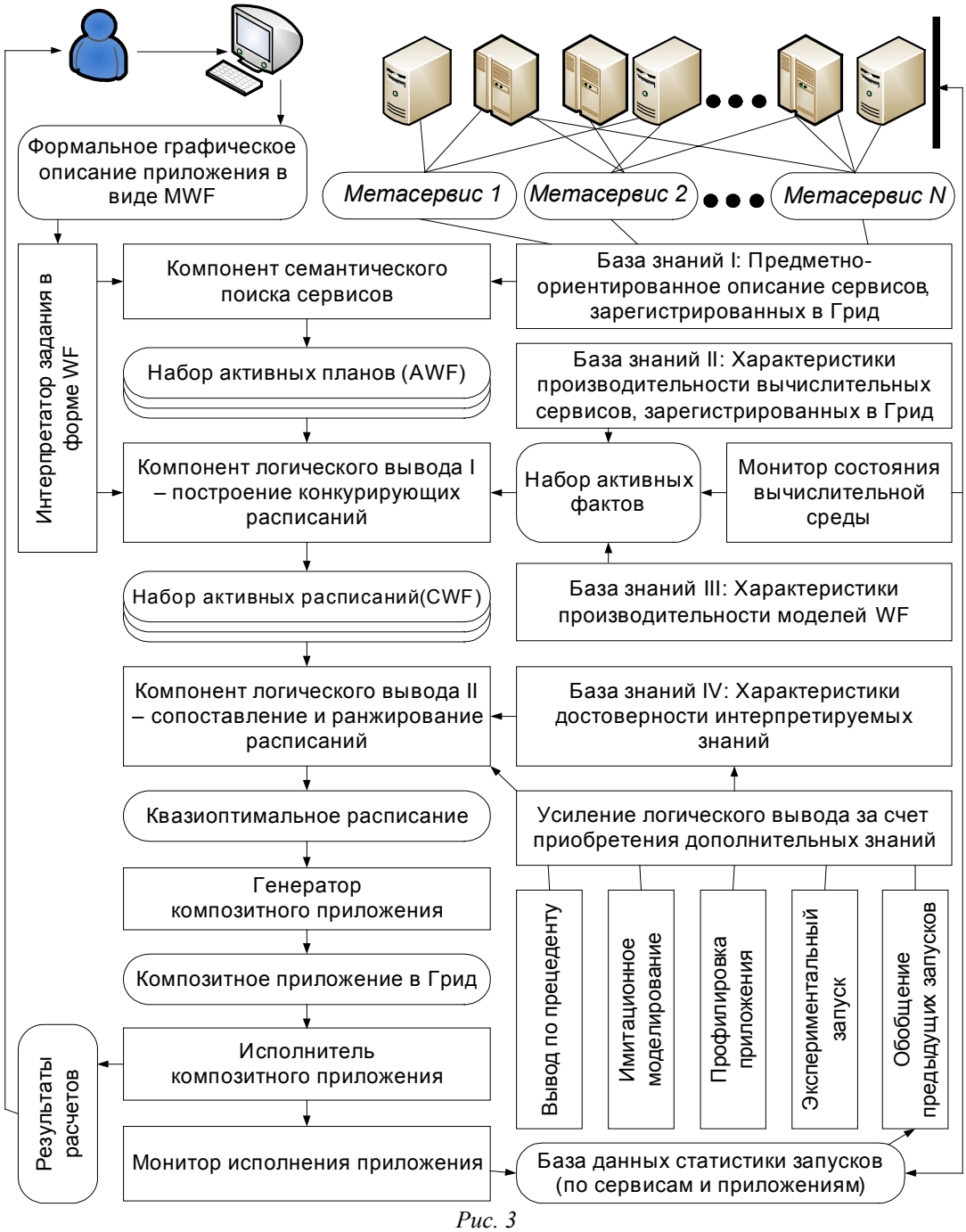


*Рис. 3*

Человеко-компьютерное взаимодействие осуществляется через оболочку визуального проектирования, в которой пользователь описывает композитное приложение в форме MWF,

используя специальную графическую нотацию. На первом этапе выполняется интерпретация MWF, на основании которой осуществляется семантический поиск сервисов, зарегистрированных в Грид, в рамках заданной предметной области. Таким образом, создается набор активных прикладных Грид-сервисов, установленных на конкретных вычислительных системах, входящих в Грид и готовых к использованию. В результате получается набор AWF, построенных с использованием метасервисов, формально подходящих под пользовательские критерии. Второй этап работы связан с осуществлением логического вывода, который представляет собой создание расписания, оптимального с точки зрения критериев пользователя. Для этого используются экспертные знания о производительности прикладных Грид-сервисов, в соответствии с (6)—(11) предоставляемые их разработчиками при регистрации сервиса в Грид, например, в форме параметрической модели времени выполнения в зависимости от параметров задачи и характеристик вычислительной системы. С использованием данных мониторинга текущего состояния Грид и известных характеристик пользовательской задачи оценивается время выполнения конкретных задач, 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 прикладных сервисов 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 отправлять на выполнение в Грид. Результатом выполнения сценария является набор данных, который отправляется в указанное хранилище Грид или на пользовательский терминал.

Таким образом, оболочка iPEG представляет собой узкоспециализированную систему iPSE, ориентированную на вычисления в корпоративных Грид-системах, без специфической предметной ориентации.

*Высокопроизводительный программный комплекс квантово-механических расчетов и моделирования наноразмерных атомно-молекулярных структур HPC-NASIS*. Комплекс HPC-NASIS [11] предназначен для проведения расчетов из первых принципов общего характера, для компьютерного моделирования и расчета наноструктур и наноматериалов с заданными свойствами, а также их поведения в различных условиях эксплуатации, включая основные приоритетные направления развития работ в области нанотехнологий. Комплекс обеспечивает моделирование электронной структуры и расчет ряда физических характеристик исследуемых наносистем и наноустройств, включая энергию возбужденных состояний, силу осцилляторов электронных переходов, плотность фононных состояний, оптические и фотоэлектрические свойства ансамблей наночастиц, степень усиления или подавления комбинационного рассеяния, флуоресценции, переноса возбуждения, транспортные свойства нанотрубок и пр.

На рис. 4 приведена общая архитектура программного комплекса. Взаимодействие его компонентов регламентируется подходом SaaS (Software as a Service), реализуемым посредством SOA. Это позволяет объединять программные компоненты (как отдельные „черные ящики“ — сервисы), разработанные разными авторами, реализованные с помощью различных

технологий, с различной формой распространения и поддержки, интерпретируя их как web-сервисы.

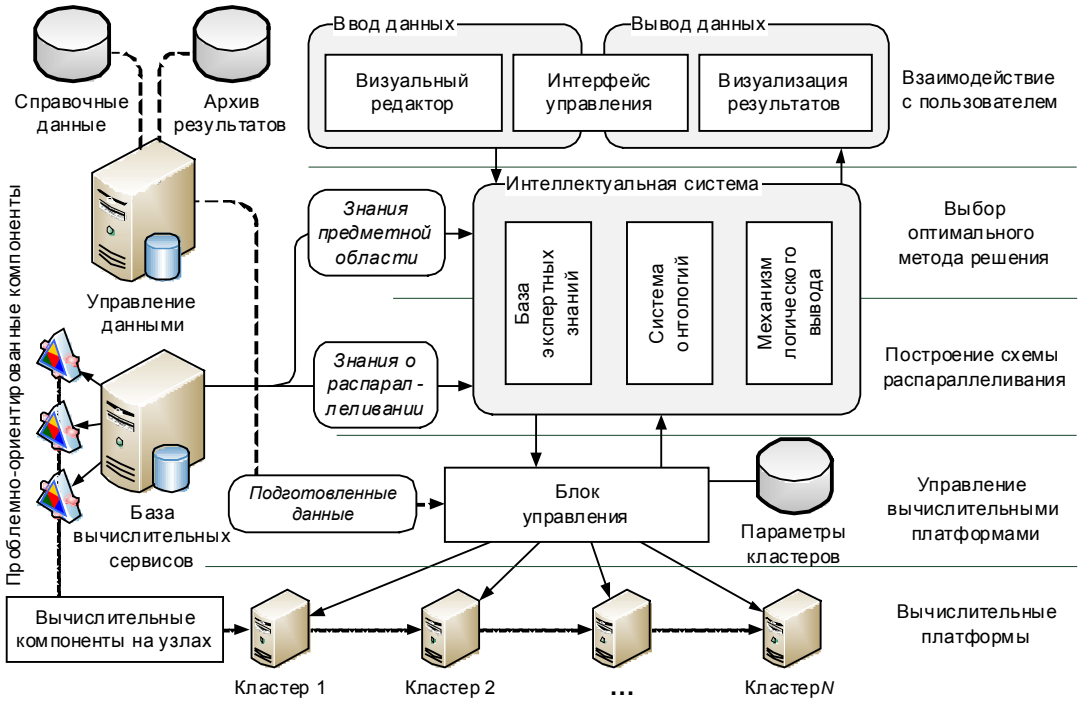


*Рис. 4*

Предметно-ориентированные сервисы квантово-механических расчетов и моделирования наноразмерных систем, материалов и устройств на их основе представлены программными компонентами, которые размещаются на различных суперкомпьютерах; при этом какие-либо связи между компонентами в рамках одного суперкомпьютера не устанавливаются. Каждый из программных компонентов при запуске допускает внутреннее (функциональное) распараллеливание в соответствии с заложенным в нем алгоритмом и доступными ресурсами суперкомпьютера. Подготовка и передача входных данных, запуск пакетов и сбор результатов осуществляются централизованно, на основе команд, поступающих с управляющего ядра комплекса. Следует отметить, что суперкомпьютеры и установленные на них программные компоненты могут функционировать под разными операционными системами и иметь различные системы управления; их объединение выполняется посредством кроссплатформенной управляющей оболочки (платформы исполнения). Она осуществляет управление и мониторинг текущего состояния суперкомпьютеров, исполняемых на них задач, а также общей коммуникационной сети, обеспечивающей функционирование гиперкластера в целом.

Стратегия управления (правила взаимодействия сервисов) формируется управляющим ядром комплекса как набором сервисов, установленных на отдельном управляющем сервере. Основой ядра является главный управляющий сервис (компонент управления параллельным исполнением). Управляющий сервис, используя информацию о текущих заданиях, знания (в форме параметрических моделей) об их производительности и результаты мониторинга состояния целевых систем, осуществляет планировку совместного исполнения цепочки задач наиболее эффективно по пользовательским критериям (времени выполнения, стоимости, надежности и пр.). Этот сервис является по сути генератором команд и централизующим звеном для остальных сервисов, заменяя, таким образом, традиционную сервисную шину (что необходимо для достижения наибольшей производительности комплекса).

Доступ пользователей к возможностям комплекса осуществляется через графический интерфейс, являющийся „толстым клиентом“ для управляющего ядра. Пользователь имеет возможность в привычном оконном интерфейсе формулировать задачу на языке предметной области

(получая при необходимости интерактивную справку), подготавливает и вводит файлы данных, дает команду на выполнение задания, осуществляет мониторинг выполнения. По окончании вычислений пользователь может получить требуемые результаты расчетов, а также визуализировать их посредством инструментария, установленного на его клиентском компьютере. Данные вычислений доступны постфактум; в реальном времени передача данных и визуализация расчетов не производятся вследствие несоответствия физических масштабов реальному времени вычислений и разнородности применяемых пакетов без возможностей их полной унификации.

Пользователь может взаимодействовать с комплексом в трех режимах работы. Ручной режим позволяет пользователю через графический клиентский интерфейс готовить данные и удаленно запускать конкретный программный компонент из доступных. Автоматический режим (режим экспертной системы), напротив, дает возможность пользователю, проходя интервью в терминах предметной области, выбрать наиболее подходящие для его задач программные компоненты, корректно подготовить данные и описать последовательность запусков. Полуавтоматический режим (режим сценариев) позволяет пользователю выбрать уже готовый сценарий сопряжения компонентов для расчета тех или иных физических характеристик. Автоматический и полуавтоматический режимы реализуются посредством компонента интеллектуальной поддержки пользователя, также установленного как сервис на управляющем сервере.

Программный комплекс HPC-NASIS представляет собой, таким образом, полнофункциональную систему iPSE, ориентированную на задачи нанотехнологий. На настоящий момент в состав программного комплекса входят 20 логически взаимосвязанных предметно-ориентированных компонентов, разработанных различными группами специалистов. Их модификация, а также добавление новых компонентов планируются по мере развития комплекса.

**Направления дальнейшего развития iPSE.** Детальная схема общей архитектуры iPSE, приведенная на рис. 1, может интерпретироваться с точки зрения путей дальнейшего развития исследований в этой области, в частности, можно выделить следующие задачи в рамках общего направления.

— Создание формального языка описания знаний о предметно-ориентированных сервисах, отчуждаемых от разработчиков, разработка и продвижение соответствующих стандартов.

— Разработка аппарата метрологического анализа и синтеза для определения параллельной производительности предметно-ориентированных сервисов и построения моделей производительности.

— Разработка предметно-ориентированных технологий web-mining применительно к задаче приобретения новых знаний в iPSE.

— Создание специализированных проблемно-ориентированных баз знаний для организации вычислений на специфических параллельных архитектурах (IBM Cell, FPGA, GPGPU).

— Исследование возможностей развития специализированных средств когнитивной компьютерной графики для интеллектуального пользовательского интерфейса iPSE.

— Разработка унифицированного инструментария, ориентированного на автоматизацию построения ВПКМСС.

Тем не менее определяющим фактором в дальнейшем развитии iPSE является обоснование интерфейсов взаимодействия между уже существующими программными системами в рамках соответствующей области знания. При этом понятие интерфейса охватывает не форму представления входных и выходных данных, а скорее, сущность решаемой задачи и возможность сопряжения математических моделей различных подсистем для исследования поведения сложной системы в целом.

**Заключение.** В работе представлено симбиотическое направление исследований на стыке информационных технологий и предметно-ориентированного компьютерного моделирования. Его квинтэссенция состоит в том, что для достижения эффективности параллельных вычислений недостаточно только рассчитывать на возможности сверхмощных вычислитель-

ных систем и глубинные средства исследования и оптимизации программ. Высокая производительность является следствием совокупного учета специфики предметной области, особенностей применяемых моделей, методов и алгоритмов, а также архитектурных особенностей вычислительных систем и технологий программирования. Проблема сочетания знаний из столь различных областей, по-видимому, на данном этапе может быть решена лишь на основе их отчуждения, обобщения и использования посредством интеллектуальных технологий в рамках концепции iPSE.

В заключение хотелось бы отметить, что на необходимость рационального использования и совершенствования аппарата знаний обращал свое внимание Герберт Уэллс. Еще в 1940 г. он писал: „Огромное и всевозрастающее богатство знаний разбросано сегодня по всему миру. Этих знаний, вероятно, было бы достаточно для решения всего громадного количества трудностей наших дней — но они рассеяны и неорганизованы. Нам необходима чистка мышления в своеобразной мастерской, где можно получать, сортировать, усваивать, разъяснять и сравнивать знания и идеи“.

Авторы выражают искреннюю признательность В.Н. Васильеву, Ю.И. Нечаеву, В.Г. Маслову, С.В. Иванову, А.В. Дунаеву и А.В. Ларченко за возможность плодотворного обсуждения положений и материалов данной статьи.

Работа выполнена при частичной поддержке ФЦП „Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007—2012 гг.“, проект 2008-04-2.4-15-003 „Создание высокопроизводительного программного комплекса для квантово-механических расчетов и моделирования наноразмерных структур и комплексов“.

## СПИСОК ЛИТЕРАТУРЫ

1. *Boccara N.* Modeling Complex Systems. NY: Springer, 2004. 397 p.
2. *Монин А. С.* Гидродинамика атмосферы, океана и земных недр. СПб: Гимиз, 1999. 524 с.
3. *Маслов В. Г.* Расчеты электронных спектров Mg-порфина, Mg-фталоцианина и их ионных форм методом ППДП/С // Теор. и эксп. химия. 1984. Т. 20, № 3. С. 288—298.
4. *Иванов С. В.* Идентификация параметрически связанных моделей сложных систем // Науч.-технич. вестн. СПбГУ ИТМО. Технологии высокопроизводительных вычислений и компьютерного моделирования. 2008. Вып. 54. С. 100—107.
5. *Сипл Р.* Десять правил создания композитных приложений // PC Week/RE. 2006. Vol. (556) 46.
6. *Граничин О. Н., Кияев В. И.* Информационные технологии в управлении. М.: Бином. Лаборатория знаний. Интернет-Университет Информационных Технологий, 2008. 336 с.
7. *Lublinsky B.* Defining SOA as an architectural style. 9 January 2007. [Electronic resource]: <http://www.ibm.com /developerworks/architecture/library/ar–soastyle>.
8. *Бухановский А. В.* и др. Справочные данные по режиму ветра и волнения Балтийского, Северного, Черного, Азовского и Средиземного морей. СПб: Изд-во Российского морского регистра судоходства, 2006. 450 с.
9. Инженерные изыскания на континентальном шельфе для строительства морских нефтегазоносных сооружений. СП11-114-2004. М.: Госстрой России, 2004. 88 с.
10. *Бухановский А. В.* и др. Моделирование экстремальных явлений в атмосфере и океане как задача высокопроизводительных вычислений // Вычислительные методы и программирование. 2008. Т. 9. С. 141—153.
11. *Васильев В. Н.* и др. Высокопроизводительный программный комплекс моделирования атомно-молекулярных наноразмерных систем // Науч.-технич. вестн. СПбГУ ИТМО. Технологии высокопроизводительных вычислений и компьютерного моделирования. 2008. Вып. 54. С. 3—12.
12. *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 // Вестн. СПбГУ. Сер. 7. 2005. Вып. 3. С. 62—74.

13. *Bogdanov A. V., Boukhanovsky A. V.* Advanced high performance algorithms for data processing // Lecture Notes in Computer Science. 2004. Vol. 3036. P. 239—246.

14. *Ковальчук С. В.* и др. Особенности проектирования высокопроизводительных программных комплексов для моделирования сложных систем // Информационно-управляющие системы. 2008. № 3. С. 10—18.

15. *Rice J. R., Boisvert R. F.* From Scientific Software Libraries to Problem-Solving Environments // IEEE Computational Science & Engineering. 1996. Vol. 3, N 3. P. 44—53.

16. *Schuchardt K., Didier B., Black G.* Ecce — a problem-solving environment's evolution toward Grid services and a Web architecture // Concurrency and Computation: Practice and Experience. 2002. Vol. 14. P. 13—15.

17. *Hoekstra A, Kaandorp J., Sloot P. M. A.* A Problem Solving Environment for Modelling Stony Coral Morphogenesis // Proc. of 3rd Int. Conf. on Computational Sciences. 2003. P. 639—649.

18. *Sloot P. M. A., Boukhanovsky A. V., Keulen W., Tirado-Ramos A., Boucher C.* A GRID-based HIV expert system // J. of Clinical Monitoring and Computing. 2005. Vol. 19. P. 263—278.

19. *Cannataro M., Comito C., Lo Schiavo F., Veltri P.* Integrating Ontology and Workflow in PROTEUS, a Grid-Based Problem Solving Environment for Bioinformatics // Proc. of the Int. Conf. on Information Technology: Coding and Computing (ITCC'04). 2004. Vol. 2. P. 90—102.

20. *Blythe J., Jain S., Deelman E., Gil Y., Vahi K., Mandal A., Kennedy K.* Task Scheduling Strategies for Workflow-based Applications in Grids. 2005. P. 1—9.

21. *Гольдштейн Б. С.* и др. Интеллектуальные сети. М.: Радио и связь, 2000. 500 с.

22. *Амамия М., Танака Ю.* Архитектура ЭВМ и искусственный интеллект. М.: Мир, 1993. 400 с.

23. *Жегуло О. А.* Представление знаний о методах распараллеливания в экспертной системе поддержки распараллеливания программ // Искусственный интеллект. 2001. № 3. С. 323—330.

24. *Нечаев Ю. И.* Искусственный интеллект: концепции и приложения. СПб: ГМТУ, 2002. 215 с.

25. Интеллектуальные системы в морских исследованиях и технологиях / Под ред. *Ю. И. Нечаева*. СПб: ГМТУ, 2001. 352 с.

26. *Поспелов Д. А., Эрлих А. И.* Прикладная семиотика – новый подход к построению систем управления и моделирования // Динамические интеллектуальные системы в управлении и моделировании. М.: ЦРДЗ, 1996. С. 30—33.

27. *Gruber T. R.* A translation approach to portable ontologies // Knowledge Acguisition. 1993. N 5(2). P. 199—220.

28. *Бухановский А. В., Ковальчук С. В.* и др. Высокопроизводительный программный комплекс моделирования экстремальных гидрометеорологических явлений. Ч. I. Постановка задачи, модели, методы и параллельные алгоритмы // Науч.-технич. вестн. СПбГУ ИТМО. Технологии высокопроизводительных вычислений и компьютерного моделирования. 2008. Вып. 54. С. 56—63.

29. *Дунаев А. В., Ларченко А. В., Бухановский А. В.* Инструментальная оболочка поддержки принятия решений разработчика высокопроизводительных приложений в Грид // Науч.-технич. ведомости СПбГПУ. 2008. № 5. С. 98—104.

***Сведения об авторах***

***Александр Валерьевич Бухановский*** — д-р техн. наук, профессор; НИИ Наукоемких компьютерных технологий Санкт-Петербургского государственного университета информационных технологий, механики и оптики; директор;
E-mail: avb_mail@mail.ru

***Сергей Валерьевич Ковальчук*** — канд. техн. наук; НИИ Наукоемких компьютерных технологий Санкт-Петербургского государственного университета информационных технологий, механики и оптики; ст. науч. сотр.;
E-mail: sergey.v.kovalchuk@gmail.com

***Сергей Владимирович Марьин*** — аспирант; НИИ Наукоемких компьютерных технологий Санкт-Петербургского государственного университета информационных технологий, механики и оптики; мл. науч. сотр.;
E-mail: sergey.maryin@gmail.com

Рекомендована институтом

Поступила в редакцию
10.03.09 г.

УДК 519.853.4

Р. Г. Стронгин, В. П. Гергель, К. А. Баркалов

# ПАРАЛЛЕЛЬНЫЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ

Рассмотрен параллельный алгоритм решения многоэкстремальных задач с невыпуклыми ограничениями, основанный на приведении исходной многомерной задачи к набору связанных одномерных задач. Предложена новая схема построения множества отображений типа кривой Пеано, сохраняющая часть информации о близости точек в многомерном пространстве. Приведены результаты экспериментов, подтверждающие эффективность использования новой схемы построения множественных отображений.

***Ключевые слова:*** *многоэкстремальная оптимизация, невыпуклые ограничения, кривые Пеано, параллельные алгоритмы.*

**Введение.** В настоящей работе продолжено исследование в области нового подхода к минимизации многоэкстремальных функций при невыпуклых ограничениях, описанного в [1—6] и получившего название *индексного метода* глобальной оптимизации. Метод основан на раздельном учете каждого ограничения задачи, он не требует использования штрафных функций. При этом решение многомерных задач сводится к решению эквивалентных им одномерных задач. Соответствующая редукция основана на использовании разверток единичного отрезка вещественной оси на гиперкуб. Роль таких разверток играют непрерывные однозначные отображения типа кривой Пеано, называемые также кривыми, заполняющими пространство. Предложена новая схема построения множества кривых Пеано („вращаемые развертки“), достоинством которой (в отличие от схемы „сдвиговых разверток“) является простота в построении и использовании. Описан параллельный индексный алгоритм, основанный на одновременном использовании множества вращаемых разверток. Приведены результаты экспериментов, подтверждающие достоинство новой схемы построения множественных отображений.

**Постановка задачи.** Рассмотрим многомерную задачу глобальной оптимизации

$$\varphi(y^*) = \min\{\varphi(y) : y \in \Omega,\ g_j(y) \le 0,\ 1 \le j \le m\}, \qquad (1)$$

где область поиска $\Omega$ задана как единичный гиперкуб $N$-мерного евклидова пространства

$$\Omega = \{y \in R^N : -2^{-1} \le y_i \le 2^{-1},\ 1 \le i \le N\}.$$

Важный в прикладном отношении подкласс задач вида (1) характеризуется тем, что все функции, входящие в определение задачи, заданы некоторыми (программно реализуемыми) алгоритмами вычисления значений $\varphi(y)$, $g_j(y)$ $(1 \le j \le m)$ в точках области поиска $\Omega$. При этом *решение задачи* (1) сводится к построению оценки $y_* \in Q$, отвечающей некоторому понятию близости к точке $y^*$ (например, чтобы $\|y^* - y_*\| \le \varepsilon$ или $|\varphi(y^*) - \varphi(y_*)| \le \varepsilon$, где $\varepsilon > 0$ есть заданная точность) на основе некоторого числа $k$ значений функционалов задачи, вычисленных в точках области $\Omega$.

В задачах многоэкстремальной оптимизации возможность достоверной оценки глобального оптимума обусловлена наличием *априорной информации* о функции, позволяющей связать возможные значения минимизируемой функции с уже известными ее значениями в точках осуществленных поисковых итераций. Весьма часто такая априорная информация о задаче (1) представляется в виде предположения, что целевая функция $\varphi$ (для унификации

обозначаемая также $g_{m+1}$) и левые части ограничений $g_j(y)$ $(1 \le j \le m)$ удовлетворяют *условию Липшица* с соответствующими константами $L_j$ $(1 \le j \le m+1)$, а именно

$$\left\| g_j(y_1) - g_j(y_2) \right| \le L_j \left\| y_1 - y_2 \right\|, \ 1 \le j \le m+1, \ y_1, y_2 \in \Omega. \qquad (2)$$

В общем случае все эти функции могут быть *многоэкстремальными*.

Использование развертки Пеано $y(x)$, однозначно отображающей единичный отрезок вещественной оси на единичный гиперкуб, позволяет свести многомерную задачу условной минимизации в области $\Omega$ к одномерной задаче условной минимизации на отрезке $[0,1]$:

$$\varphi(y(x^*)) = \min\{\varphi(y(x)) : x \in [0,1], g_j(y(x)) \le 0, \quad 1 \le j \le m\}. \qquad (3)$$

Рассматриваемая схема редукции размерности сопоставляет многомерной задаче с липшицевой минимизируемой функцией и липшицевыми ограничениями одномерную задачу, в которой соответствующие функции удовлетворяют равномерному условию Гельдера (см. работу [1]), т.е.

$$\left| g_j(y(x')) - g_j(y(x'')) \right| \le K_j \left| x' - x'' \right|^{1/N}, \ x', x'' \in [0,1], \ 1 \le j \le m+1,$$

где $N$ есть размерность исходной многомерной задачи, а коэффициенты $K_j$ связаны с константами Липшица $L_j$ исходной задачи соотношением $K_j \le 4L_j\sqrt{N}$. Общая теория и вопросы численного построения отображений типа кривой Пеано подробно рассмотрены в работах [1—6]. Отметим, что численно построенная кривая является приближением к теоретической кривой Пеано с точностью не менее $2^{-\kappa}$ по каждой координате (параметр $\kappa$ называется *плотностью развертки*).

Для целей дальнейшего изложения введем классификацию точек $y$ из области поиска $\Omega$ с помощью индекса $v = v(y(x))$, который определяется условиями

$$g_j(y(x)) \le 0, \ 1 \le j \le v-1, \ g_v(y(x)) > 0, \qquad (4)$$

при этом последнее неравенство несущественно, если $v = m+1$, и удовлетворяет неравенствам

$$1 \le v = v(y(x)) \le m+1.$$

Данная классификация порождает функцию

$$f(y(x)) = g_v(y(x)), \ v = v(y(x)), \qquad (5)$$

определенную и вычисляемую всюду в $\Omega$. Ее значение в точке $y$ есть либо значение левой части ограничения, нарушенного в этой точке (случай, когда $v \le m$), либо значение минимизируемой функции ($v = m+1$). Поэтому определение значения $f(y(x))$ сводится к последовательному вычислению величин $g_i(y(x))$, $1 \le i \le v = v(x)$, т.е. последующее значение $g_{i+1}(y(x))$ вычисляется лишь в том случае, когда $g_i(y(x)) \le 0$. Процесс вычислений завершается либо в результате установления неравенства $g_i(y(x)) > 0$, либо в результате достижения значения $v(x) = m+1$.

Описанная процедура, называемая *испытанием* в точке $y$, автоматически приводит к определению индекса $v$ этой точки. Пара значений

$$z = g_v(y), \ v = v(y), \qquad (6)$$

порожденная испытанием в точке $y \in \Omega$, называется *результатом испытания*.

**Использование множественных отображений.** Редукция многомерных задач к одномерным с помощью разверток характеризуется такими важными свойствами, как непрерывность и сохранение равномерной ограниченности разности функций при ограниченности вариации аргумента. Однако при этом происходит потеря части информации о близости точек в

многомерном пространстве, так как точка $x \in [0,1]$ имеет лишь левых и правых соседей, а соответствующая ей точка $y(x) \in R^N$ имеет соседей по $2^N$ направлениям. Как результат, при использовании отображений типа кривой Пеано близким в *N*-мерном пространстве образам $y'$, $y''$ могут соответствовать достаточно далекие прообразы $x'$, $x''$ на отрезке [0,1].

Сохранить часть информации о близости точек позволяет использование множества отображений

$$Y_M(x) = \{y^1(x), ..., y^M(x)\} \qquad (7)$$

вместо применения единственной кривой Пеано $y(x)$ (см. работы [4, 6]). Каждая кривая Пеано $y^i(x)$ из $Y_M(x)$ может быть получена в результате некоторого сдвига вдоль главной диагонали гипериинтервала $\Omega$. Таким образом, сконструированное множество кривых Пеано позволяет получить для любых близких образов $y'$, $y''$, отличающихся только по одной координате, близкие прообразы $x'$, $x''$ для некоторого отображения $y^i(x)$.

**Вращаемые развертки.** К числу недостатков этой ставшей уже классической схемы построения множественных разверток (будем называть их сдвиговыми развертками, или *C*-развертками) можно отнести наличие дополнительного ограничения, порождающего сложную допустимую область на одномерных отрезках.

Преодолеть этот недостаток, сохранив информацию о близости точек в *N*-мерном пространстве, позволяет новая схема построения множественных отображений. Отличительная черта схемы — множество кривых Пеано строится не с помощью сдвига вдоль главной диагонали гиперкуба, а поворотом развертки вокруг начала координат. При этом найдется отображение $y^i(x)$, точкам многомерного пространства $y'$, $y''$ (которым при исходном отображении соответствовали достаточно далекие прообразы на отрезке [0,1]) сопоставляющее более близкие прообразы $x'$, $x''$.

Развертки, порождаемые в соответствии с новой схемой, будем называть вращаемыми развертками, или *B*-развертками.

Для иллюстрации на рис. 1 приведены две кривые, являющиеся приближением к развертке Пеано для случая $N = 2$. Узлы сетки в *N*-мерном пространстве отмечены темными кружками, стрелкой указана одна из пар точек, прообразы которых далеки на одномерной оси при одном отображении и близки при другом.

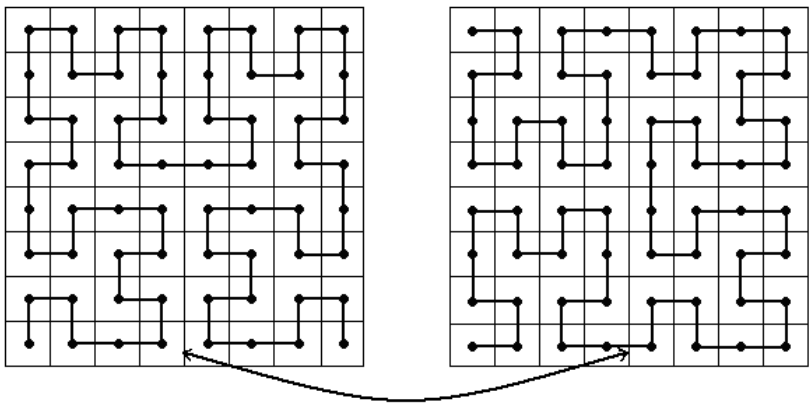

*Рис. 1*

Максимальное число различных поворотов развертки, отображающей *N*-мерный гиперкуб на одномерный отрезок, составляет $2^N$. Использование их всех является избыточным, требуется выбрать лишь часть из всех возможных вариантов.

Предлагается осуществлять преобразование развертки в виде поворота на угол $\pm\pi/2$ в каждой из координатных плоскостей. В качестве примера на рис. 2 приведены матрицы поворота в плоскости ($y_2$, $y_4$) при $N = 5$.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

*Рис.2*

Число подобных пар поворотов определяется числом координатных плоскостей пространства: $C_N^2 = N(N-1)/2$, а общее число преобразований будет равно $N(N-1)$. Учитывая исходное отображение, приходим к заключению, что данный способ позволяет строить до $N(N-1)+1$ разверток для отображения *N*-мерной области на соответствующие одномерные отрезки. При этом дополнительное ограничение, которое возникало при построении сдвиговых разверток [4], отсутствует.

В случае необходимости данный способ построения множества отображений может быть легко „отмасштабирован“ для получения большего (вплоть до $2^N$) числа разверток.

Использование множества отображений $Y_M(x) = \{y^1(x), ..., y^M(x)\}$ приводит к формированию соответствующего множества одномерных многоэкстремальных задач

$$\min\{\varphi(y^l(x)) : x \in [0,1], g_j(y^l(x)) \le 0, 1 \le j \le m\},\ 1 \le l \le M\ . \qquad (8)$$

Каждая задача из данного набора может решаться независимо, при этом любое вычисленное значение $z = g_v(y')$, $y' = y^i(x')$ функции $g_v(y)$ в $i$-й задаче может интерпретироваться как вычисление значения $z = g_v(y')$, $y' = y^s(x'')$ для любой другой $s$-й задачи без повторных трудоемких вычислений функции $g_v(y)$. Подобное информационное единство позволяет решать исходную задачу (3) путем параллельного решения индексным методом *M* задач вида (8) на наборе отрезков [0,1]. Каждая одномерная задача решается на отдельном процессоре. Для организации взаимодействия на каждом процессоре создается *M* очередей, в которые процессоры помещают информацию о выполненных итерациях. Используемая схема не содержит какого-либо единого управляющего процессора, что повышает надежность выполняемых вычислений.

**Организация параллельных вычислений.** Использование множественных отображений позволяет решать исходную задачу (1) путем параллельного решения индексным методом *M* задач вида (8) на наборе отрезков $[0,1]$. Каждая одномерная задача решается на отдельном процессоре. Результаты испытания в точке $x^k$, полученные конкретным процессором для решаемой им задачи, интерпретируются как результаты испытаний во всех остальных задачах (в соответствующих точках $x^{k1}, ..., x^{kM}$). При таком подходе испытание в точке $x^k \in [0,1]$, осуществляемое в *s*-й задаче, состоит в последовательности действий.

1. Определить образ $y^k = y^s(x^k)$ при соответствии $y^s(x)$.

2. Проинформировать остальные процессоры о начале проведения испытания в точке $y^k$ (*блокирование* точки $y^k$).

3. Вычислить величины $g_1(y^k), ..., g_v(y^k)$, где значения индекса $v \le m$ определяются условиями

$$g_i(y^k) \le 0,\ 1 \le j < v,\ g_v(y^k) > 0,\ v \le m\ .$$

Выявление первого нарушенного ограничения прерывает испытание в точке $y^k$. В случае, когда точка $y^k$ допустима, испытание включает вычисление значений всех функционалов задачи, при этом значение индекса принимается равным $v = m+1$. Тройка

$$y^s(x^k),\ v = v(x^k),\ z^k = g_v(y^s(x^k)) \tag{9}$$

является *результатом испытания* в точке $x^k$.

4. Определить прообразы $x^{kl} \in [0,1]$ $(1 \le l \le M)$ точки $y^k$ и интерпретировать испытание, проведенное в точке $y^k \in \Omega$, как проведение испытаний в $M$ точках

$$x^{k1},...,x^{kM} \tag{10}$$

с одинаковыми результатами

$$v(x^{k1}) = ... = v(x^{kM}) = v(x^k),$$
$$g_v(y^1(x^{k1})) = ... = g_v(y^M(x^{kM})) = z^k.$$

5. Проинформировать остальные процессоры о результатах испытания в точке $y^k$.

Каждый процессор имеет свою копию программных средств, реализующих вычисление функций задачи, и решающее правило алгоритма. Для организации взаимодействия на каждом процессоре создается $M$ очередей, в которые процессоры помещают информацию о выполненных итерациях в виде троек: точка очередной итерации, индекс и значение из (9), причем индекс заблокированной точки полагается равным –1, а значение функции в ней не определено.

**Описание алгоритма.** Алгоритм выбора точек итераций для всех процессоров одинаков. Начальная итерация осуществляется в произвольной точке $x^1 \in (0,1)$ (начальные точки для всех процессоров задаются различными). Выбор точки $x^{q+1}$ $(q \ge 1)$ любого последующего испытания определяется следующими правилами.

*Правило 1.* Изъять из всех очередей, закрепленных за данным процессором, записанные для него результаты, включающие множество $Y_q = \{y^{qi} : 1 \le i \le s_q\}$ точек итераций в области $D$ и вычисленные в них значения индекса и величин из (9). Определить множество $X_q = \{x^{qi} : 1 \le i \le s_q\}$ прообразов точек множества $Y_k$ при соответствии $y^l(x)$.

*Правило 2.* Точки множества итераций

$$\{x_1\} \cup X_1 \cup ... \cup X_q$$

перенумеровать нижними индексами в порядке увеличения значений координаты

$$0 = x_0 < x_1 < ... < x_i < ... < x_k < x_k + 1 = 1, \tag{11}$$

где $k = 1 + s_1 + ... + s_q$, и сопоставить им значения $z_i = g_v(x_i)$, $v = v(x_i)$ $(1 \le i \le k)$, вычисленные в этих точках. При этом индекс блокированной точки $x_i$ (т.е. точки, в которой начато проведение испытания другим процессором) полагается равным –1, т.е. $v(x_i) = -1$, значение $z_i$ является неопределенным. Точки $x_0$, $x_{k+1}$ введены дополнительно для удобства последующего изложения, индекс данных точек полагается равным $-2$, т.е. $v(x_0) = v(x_{k+1}) = -2$, а значения $z_0$, $z_{k+1}$ являются неопределенными.

*Правило 3.* Провести классификацию номеров $i$ $(1 \le i \le k)$ точек из ряда (11) по числу ограничений задачи, выполняющихся в этих точках, путем построения множеств

$$I_{-2} = \{0, k+1\},$$
$$I_{-1} = \{i : 1 \le i \le k, v(x_i) = -1\},$$
$$I_v = \{i : 1 \le i \le k, v(x_i) = v\},\ 1 \le v \le m+1,$$

содержащих номера всех точек $x_i$ $(1 \le i \le k)$ с индексами, равными одному и тому же значению $v$. Определить максимальное значение индекса

$$\nu_{\max} = \max\{v = v(x_i),\ 1 \le i \le k\}.$$

*Правило 3.* Вычислить текущие нижние границы

$$\mu_\nu = \max\left\{\frac{|z_i - z_j|}{(x_i - x_j)^{1/N}} : j < i, j \in I_\nu, i \in I_\nu\right\} \tag{12}$$

для относительных разностей функций $g_v$ $(1 \le v \le m+1)$. Если множество $I_v$ содержит менее двух элементов или если $\mu_v$ из (12) оказывается равным нулю, то принять $\mu_v = 1$. Из (12) следует, что оценки $\mu_v$ являются неубывающими начиная с момента, когда (12) порождает первое положительное значение $\mu_v$.

*Правило 4.* Для всех непустых множеств $I_v$ $(1 \le v \le m+1)$ вычислить оценки

$$z_\nu^* = \begin{cases} -\varepsilon_\nu, & \nu < \nu_{\max}, \\ \min\{g_\nu(x_i) : i \in I_\nu\}, & \nu = \nu_{\max}, \end{cases}$$

где вектор

$$\varepsilon_R = (\varepsilon_1, ..., \varepsilon_m), \tag{13}$$

имеющий положительные координаты, называется *вектором резервов*.

*Правило 5.* Для каждого интервала $(x_{i-1}, x_i)$ $(1 \le i \le k+1)$ вычислить характеристику $R(i)$

$$R(i) = \Delta_i + \frac{(z_i - z_{i-1})^2}{r_\nu^2 \mu_\nu^2 \Delta_i} - 2\frac{(z_i + z_{i-1} - 2z_\nu^*)}{r_\nu \mu_\nu},\ \ \nu = \nu(x_{i-1}) = \nu(x_i),$$

$$R(i) = 2\Delta_i - 4\frac{(z_i - z_\nu^*)}{r_\nu \mu_\nu},\ \ \nu(x_{i-1}) < \nu(x_i) = \nu,$$

$$R(i) = 2\Delta_i - 4\frac{(z_{i-1} - z_\nu^*)}{r_\nu \mu_\nu},\ \ \nu = \nu(x_{i-1}) > \nu(x_i),$$

$$\Delta_i = (x_i - x_{i-1})^{1/N}.$$

Величины $r_v > 1$ $(1 \le v \le m+1)$ являются параметрами алгоритма.

*Правило 6.* Определить интервал $(x_{t-1}, x_t)$, которому соответствует максимальная характеристика

$$R(t) = \max\{R(i) : 1 \le i \le k+1\}. \tag{14}$$

*Правило 7.* Провести очередное испытание в серединной точке интервала $(x_{t-1}, x_t)$, если индексы его концевых точек не совпадают, т.е.

$$x^{q+1} = (x_t + x_{t-1})/2,\ v(x_{t-1}) \ne v(x_t).$$

В противном случае провести испытание в точке

$$x^{q+1} = (x_t + x_{t-1})/2 - \operatorname{sign}(z_t - z_{t-1})\frac{1}{2r_\nu}\left[\frac{|z_t - z_{t-1}|}{\mu_\nu}\right]^N,\ v = v(x_{t-1}) = v(x_t).$$

Результаты испытания занести во все очереди, закрепленные за данным процессором. Увеличив $q$ на единицу, перейти к новой итерации.

Описанные правила можно дополнить условием остановки, прекращающим испытания, если $\Delta t \le \varepsilon$, где $t$ (см. выражение (14)), и $\varepsilon > 0$ — желаемая покоординатная точность в задаче.

**Операционные характеристики и сравнение алгоритмов.** Один из известных подходов к оценке эффективности методов безусловной глобальной оптимизации основан на численном решении этими методами всех задач из некоторой случайно генерируемой выборки большого объема (см., например, [7]).

Генераторы таких выборок можно рассматривать как некоторые классы функций $f(y)$ ($y \in \Omega$) с определенной на них вероятностной мерой. Один из таких генераторов $G$, предложенный в работе [8], порождает многоэкстремальные функции в соответствии со схемой

$$\varphi(y_1, y_2) = -\left\{ \left( \sum_{i=1}^{7} \sum_{j=1}^{7} \left[ A_{ij} a_{ij}(y_1, y_2) + B_{ij} b_{ij}(y_1, y_2) \right] \right)^2 + \left( \sum_{i=1}^{7} \sum_{j=1}^{7} \left[ C_{ij} a_{ij}(y_1, y_2) - D_{ij} b_{ij}(y_1, y_2) \right] \right)^2 \right\}^{1/2},$$

где $a_{ij}(y_1, y_2) = \sin(\pi i y_1)\sin(\pi j y_2)$, $b_{ij}(y_1, y_2) = \cos(\pi i y_1)\cos(\pi j y_2)$ определены в области $0 \le y_1, y_2 \le 1$, а параметры $-1 \le A_{ij}, B_{ij}, C_{ij}, D_{ij} \le 1$ являются независимыми случайными величинами, равномерно распределенными в указанном выше интервале. Минимизация подобных функций возникает, например, в задаче оценки максимального напряжения (определяющего прочность) в тонкой пластине при поперечной нагрузке.

Примененная процедура оценки эффективности алгоритмов использует аппарат *операционных характеристик* из работы [8] и состоит в следующем. Пусть некоторая задача вида (1) из рассматриваемой выборки решается с помощью алгоритма $S$. При этом задаче сопоставляется порождаемая алгоритмом последовательность точек испытаний $\{y^k\}$. Указанная последовательность прерывается (т.е. процесс решения прекращается) либо в связи с первым попаданием точки очередного испытания в заданную $\delta$-окрестность решения $x^*$, либо в связи с тем, что в ходе выполнения заданного числа $K$ испытаний такое попадание не имело места. В проведенных численных экспериментах использовалось значение $K = 1000$.

Поскольку задание $\delta$-окрестности предполагает координату $x^*$ известной, ее значение предварительно оценивалось (для каждой функции выборки) путем перебора всех узлов равномерной (в области поиска $\Omega$) сетки с шагом $10^{-3}$.

Результат решения всех задач выборки с помощью алгоритма $S$ представляется функцией $P_s(k)$, характеризующей долю задач, для которых в ходе $k$ шагов поиска имели место попадания точек испытаний в заданную $\delta$-окрестность решения. Такую функцию будем называть *операционной характеристикой* алгоритма $S$.

Эксперименты проводились для описанных выше алгоритмов со сдвиговыми (С) и вращаемыми (В) развертками с плотностью $\kappa = 12$, число разверток $M = 2$. При этом использовались параметры $r_v = 2{,}1$, $\varepsilon = 0{,}01$ и резервы $\varepsilon_v = 0{,}05$ ($0 \le v \le 1$) из (13). Данные параметры являются минимальными для соответствующих методов, при которых достигается решение 100 % задач из выборки.

Операционные характеристики для обоих методов, полученные на выборках, рождаемых генератором $G$, представлены на рис. 3. При этом кривая *1* характеризует метод со сдвиговыми развертками, *2* — метод с единственной разверткой, *3* — метод с вращаемыми развертками. Указанное расположение кривых показывает, что алгоритм с В-развертками

обеспечивает в среднем более быстрое получение оценок, лежащих в заданной окрестности решения, чем алгоритм с С-развертками или алгоритм с единственной разверткой.

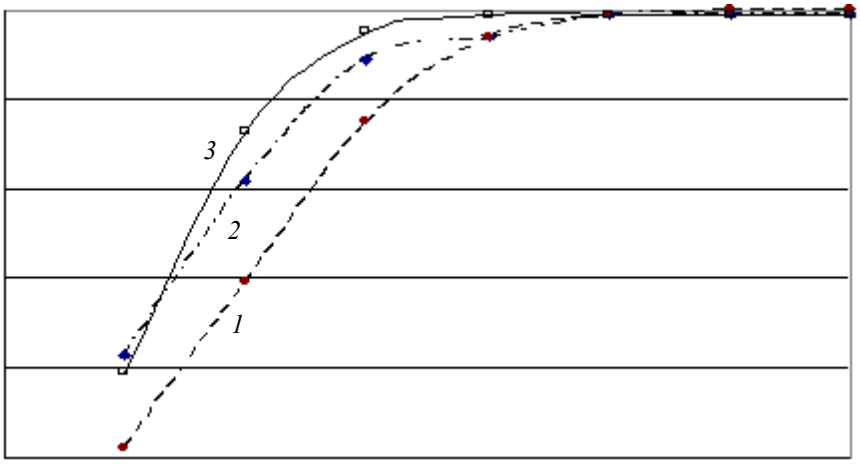


*Рис. 3*

С целью иллюстрации использования параллельного алгоритма с множеством В-разверток рассмотрим известную задачу минимизации функции Растригина

$$\varphi(y) = \sum_{i=1}^{N} (y_i^2 - \cos(18 y_i^2)),\ -1,5 \le y_i \le 1,5,\ 1 \le i \le N,\ N = 6,$$

где минимальное значение $\varphi(y^*) = -N$ достигается в точке $y^* = 0$.

Для решения данной задачи использовались последовательный и параллельный методы с вращаемой разверткой с плотностью $\kappa = 10$, использовался параметр метода $r = 2$, и $\varepsilon = 0,05$ — в критерии остановки. Число разверток $M = 30$ соответствовало числу процессоров, задействованных при решении задачи. И последовательный, и параллельный алгоритмы нашли решение с требуемой точностью, при этом последовательный алгоритм выполнил 173 116 итераций, а параллельный — 8535 (максимальное число итераций на одном процессоре). Ускорение по числу итераций составило 20,3, а ускорение по времени — 7,5.

Работа выполнена при финансовой поддержке РФФИ (грант № 07-01-00467-а) и Совета по грантам Президента Российской Федерации по государственной поддержке ведущих научных школ Российской Федерации (грант № НШ-4694.2008.9).

## СПИСОК ЛИТЕРАТУРЫ

1. *Стронгин Р. Г.* Численные методы в многоэкстремальных задачах. М.: Наука, 1978.
2. *Стронгин Р. Г., Маркин Д. Л.* Минимизация многоэкстремальных функций при невыпуклых ограничениях // Кибернетика. 1986. № 4. С. 63—69.
3. *Стронгин Р. Г.* Поиск глобального оптимума. М.: Знание, 1990.
4. *Стронгин Р. Г.* Параллельная многоэкстремальная оптимизация с использованием множества разверток // Журн. вычислительной математики и математической физики. 1991. Т. 31, № 8. С. 1173—1185.
5. *Стронгин Р. Г., Баркалов К. А.* О сходимости индексного алгоритма в задачах условной оптимизации с ε-резервированными решениями // Математические вопросы кибернетики. М.: Наука, 1999. С. 273—288.
6. *Strongin R. G., Sergeyev Ya. D.* Global optimization with non-convex constraints. Sequential and parallel algorithms. Dordrecht: Kluwer Academic Publishers, 2000.
7. *Гергель В. П., Стронгин Р. Г.* Абсолют. Программная система для исследований и изучения методов глобальной оптимизации. Н. Новгород: Изд-во Нижегородского ун-та, 1998.
8. *Гришагин В. А.* Операционные характеристики некоторых алгоритмов глобального поиска // Проблемы случайного поиска. Рига: Зинатне, 1978. № 7. С. 198—206.

***Сведения об авторах***

***Роман Григорьевич Стронгин*** — д-р физ.-мат. наук, профессор; Нижегородский государственный университет им. Н. И. Лобачевского; кафедра математического обеспечения ЭВМ; президент;
E-mail: strongin@unn.ac.ru

***Виктор Павлович Гергель*** — д-р техн. наук, профессор; Нижегородский государственный университет им. Н. И. Лобачевского; кафедра математического обеспечения ЭВМ; декан;
E-mail: gergel@unn.ru

***Константин Александрович Баркалов*** — канд. физ.-мат. наук; Нижегородский государственный университет им. Н. И. Лобачевского; кафедра математического обеспечения ЭВМ; ст. препод.;
E-mail: barkalov@fup.unn.ru

Рекомендована институтом

Поступила в редакцию
10.03.09 г.

УДК 681.3

Б. Я. Штейнберг

# БЛОЧНО-РЕКУРСИВНОЕ ПАРАЛЛЕЛЬНОЕ ПЕРЕМНОЖЕНИЕ МАТРИЦ

Предложен новый параллельный алгоритм перемножения матриц с количеством пересылок, меньшим, чем у существующих. Для быстрого алгоритма перемножения матриц Штрассена рассмотрена сложность по обращениям к памяти и обоснована его неэффективность для матриц практически значимой размерности.

***Ключевые слова:*** *параллельные алгоритмы, гиперкуб, размещение данных, межпроцессорные пересылки, сложность по обращениям к памяти.*

**Введение.** Новые тенденции в развитии вычислительных архитектур изменяют представления о сложности алгоритмов [1]. Если 40 лет назад время доступа к памяти было не существенно по сравнению со временем выполнения арифметических операций, то сейчас время доступа к памяти на порядок больше, а межпроцессорная пересылка требует существенно больше времени, чем доступ к памяти. Если раньше основной характеристикой сложности вычислительного алгоритма считалось количество умножений, то теперь такой характеристикой должно стать количество обращений к памяти, а для параллельных алгоритмов — количество межпроцессорных пересылок.

В настоящей работе предложен новый параллельный алгоритм перемножения матриц с количеством пересылок, меньшим, чем у существующих. Для быстрого алгоритма перемножения матриц Штрассена рассмотрена сложность по обращениям к памяти и обоснована нецелесообразность его использования для современных компьютеров; предложена процедура блочно-аффинного размещения данных. Следует отметить, что параллельные алгоритмы перемножения матриц на вычислительных системах с распределенной памятью, описанные в работах [2—5], предполагают размещение матриц полосами или блоками — это частные случаи блочно-аффинных размещений.

Приведенные в [2—5] алгоритмы требуют $n^3 / p$ параллельных умножений. Объем пересылаемых данных каждого процессора для алгоритмов Фокса и Кэннона [4, 6] составляет $n^2 / p^{1/2}$. Объем пересылаемых данных каждого процессора в алгоритмах, предложенных в [3], равен $n^2 / p^{2/3}$. Объем пересылаемых данных каждого процессора в предлагаемом нами

блочно-рекурсивном алгоритме равен $\frac{n^2 \log_2 p}{p}$. При этом не используются рассылки данных типа „от одного — всем“. Предполагается использование полнодоступного коммутатора или коммуникационной сети „$n$-мерный куб“. Приводится схема отображения пересылок $n$-мерного куба на кольцевую сеть и кольцевой сети — на решетку, что позволит использовать данный алгоритм на многопроцессорных системах с соответствующей топологией.

**Перемножение матриц при блочно-аффинных размещениях.** Возможно размещать элементы массива в параллельной памяти произвольным образом, но необходимо, чтобы к этим элементам массива было удобно обращаться. Выбор способа размещения данных должен зависеть от исполняемой программы. Различные способы размещения данных вне связи с исполняемым кодом описаны в работе [7].

Будем предполагать, что имеется $p$ модулей памяти. Пусть $X$ — некоторый $m$-мерный массив.

*Блочно-аффинные по модулю $p$ размещения данных.* Пусть натуральные числа $p, d_1, d_2, ..., d_m$ и целые константы $s_0, s_1, s_2, ..., s_m$ зависят только от $m$-мерного массива $X$. Блочно-аффинное по модулю $p$ размещение $m$-мерного массива $X$ — это такое размещение, при котором элемент $X(i_1, i_2, ..., i_m)$ находится в модуле памяти с номером

$$u = \left(\left\lfloor s_1 / d_1 \right\rfloor i_1 + \left\lfloor s_2 / d_2 \right\rfloor i_2 + \ ... + \left\lfloor s_m / d_m \right\rfloor i_m + s_0\right) \bmod p.$$

При описанном блочно-аффинном способе размещения $m$-мерный массив представляется как массив блоков размерностью $d_1, d_2, \ldots, d_m$, в котором у каждого блока все элементы находятся в одном модуле памяти.

Описанные далее способы размещения данных являются частными случаями блочно-аффинного по модулю $p$ размещения.

Аффинные по модулю $p$ размещения данных — такие, при которых элемент массива $X(i_1, i_2, ..., i_m)$ находится в модуле памяти с номером $u = (s_1 i_1 + s_2 i_2 + ... + s_m i_m + s_0) \bmod p$, где $s_0, s_1, s_2, ..., s_m$ — константы, зависящие только от массива $X$. Число $s_0$ будем называть сдвигом массива $X$. Это число соответствует номеру модуля памяти, в котором размещается нулевой элемент $X(0, 0, ..., 0)$.

*{-1, 0, +1}-размещения* — в них константы $s_1, s_2, ..., s_m$ принадлежат множеству {-1, 0, +1}, к ним, в частности, относятся размещение матрицы по строкам или по столбцам, или по диагоналям, или скошенная форма размещения (хранения) матрицы.

*{0, +1}-размещения* — в них константы $s_1, s_2, ..., s_m$ принадлежат множеству {0, +1}.

К такому типу размещений относятся размещения матриц по строкам, по столбцам и в скошенном виде. Размещение по диагоналям к такому типу не относится.

*Покоординатные размещения со сдвигами* — это такие {0, +1}-размещения, в которых лишь одна из величин $s_1, s_2, ..., s_m$ может быть равна единице.

*Покоординатные размещения* — это такие {0, +1}-размещения, в которых лишь одна из величин $s_1, s_2, ..., s_m$ может быть равна единице и $s_0 = 0$.

Размещения матрицы по строкам или по столбцам являются покоординатными.

Рассмотрим фрагмент программы, вычисляющий произведение двух матриц по стандартному алгоритму

```
do i = 1, n
do j = 1, n
do k = 1, n
X(i,j) = X(i,j)+A(i,k)*B(k,j).
```

**Теорема 1.** Не существует таких нетривиальных аффинных размещений массивов $X$, $A$ и $B$ в распределенной памяти, при которых параллельное произведение матриц выполняется без пересылок.

***Доказательство.*** Предположим, что искомые аффинные размещения массивов существуют. Для каждого значения $i$, $j$ и $k$ элементы $X(i,j)$, $A(i,k)$ и $B(k,j)$ должны оказаться в одном и том же модуле памяти. Это возможно тогда и только тогда, когда существуют такие целые числа $s_1, s_2, t_1, t_2, u_1, u_2$, что выполняются равенства

$$s_1 i + s_2 j = t_1 i + t_2 k = u_1 k + u_2 j \mod p.$$

Это возможно для всех значений $i, j, k$ лишь при выполнении равенств $s_1 = s_2 = t_1 = t_2 = u_1 = u_2 = 0 \mod p$, т.е. когда все данные находятся в одном и том же модуле памяти. Итак, задача вычисления произведения матриц не может быть распараллелена на вычислительной системе с распределенной памятью без межпроцессорных пересылок данных.

**Блочно-рекурсивный параллельный алгоритм перемножения матриц.** Будем считать, что количество процессорных элементов $p$ вычислительной системы является степенью числа 4.

Рассмотрим задачу умножения квадратных матриц $X = AB$ размерностью $n$ на вычислительной системе с $n$ процессорными элементами. Матрицы A и B рассмотрим как блочные матрицы 2×2 с квадратными блоками размером $(n/2)\times(n/2)$. Всего в каждой матрице получается по 4 блока. Обозначим $A_{ij}$, $(i, j = 1,2)$, аналогично обозначим блоки матрицы B.

| | |
|---|---|
| Модуль памяти 0<br>$A_{11}, B_{11}, C_{11}$ | Модуль памяти 1<br>$A_{12}, B_{21}, C_{12}$ |
| Модуль памяти 2<br>$A_{21}, B_{12}, C_{21}$ | Модуль памяти 3<br>$A_{22}, B_{22}, C_{22}$ |

*Рис. 1*

Множество всех процессорных элементов разобьем на 4 равных группы с номерами 0, 1, 2, 3 (рис. 1). Разместим элементы обеих матриц по блокам — в каждой группе модулей памяти по одному.

— Блок $A_{ij}$ $(i, j = 1,2)$ матрицы $A$ разместим в группе модулей памяти с номером $(i-1)2 + j - 1$.

— Блок $B_{ij}$ $(i, j = 1,2)$ матрицы $B$ разместим в группе модулей памяти с номером $(i-1)2 + i - 1$.

Для вычисления произведения матриц в этом случае требуется два перемножения, две пересылки блоков (рис. 2, *а* — первое перемножение и первая группа пересылок, *б* — второе перемножение и вторая группа) и два сложения (рис. 3).

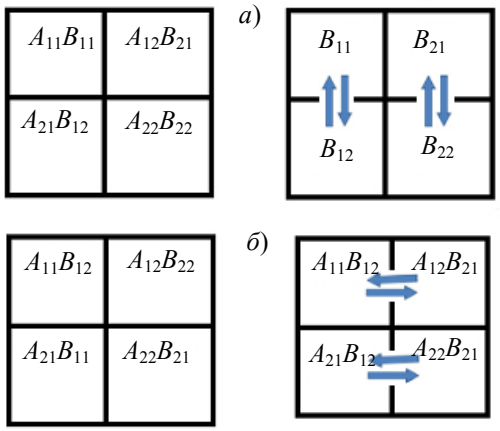


*Рис. 2*

| | |
|---|---|
| Модуль памяти 0<br>$C_{11} = A_{11}B_{11} + A_{12}B_{21}$ | Модуль памяти 1<br>$C_{12} = A_{12}B_{22} + A_{11}B_{12}$ |
| Модуль памяти 2<br>$C_{21} = A_{21}B_{11} + A_{22}B_{21}$ | Модуль памяти 3<br>$C_{22} = A_{21}B_{12} + A_{22}B_{22}$ |

*Рис. 3*

Перемножение блоков в каждой группе процессорных элементов (ПЭ) можно выполнять по такой же схеме рекурсивно. В этом алгоритме возможны $\log_4 p$ шагов. На каждом шаге размерность блоков в два раза уменьшается, но количество блоков в четыре раза возрастает.

На рис. 4, *а* представлен пример разбиения каждой из четырех групп ПЭ на 4 группы второго уровня; рис. 4, *б* — то же разбиение множества процессорных элементов на группы в двоичной кодировке. Нумерация соответствует нумерации узлов четырехмерного куба.

При двойной нумерации цифра перед точкой означает номер группы ПЭ первого уровня, цифра после запятой — номер группы ПЭ второго уровня (см. рис. 4, *а*).

*а*)

| | | | |
|---|---|---|---|
| 0,2 | 0,3 | 1,2 | 1,3 |
| 2,0 | 2,1 | 3,0 | 3,1 |
| 2,2 | 2,3 | 3,2 | 3,3 |
| 0,0 | 0,1 | 1,0 | 1,1 |

*б*)

| | | | |
|---|---|---|---|
| 0000 | 0001 | 0100 | 0101 |
| 0010 | 0011 | 0110 | 0111 |
| 1000 | 1001 | 1100 | 1101 |
| 1010 | 1011 | 1110 | 1111 |

*Рис. 4*

Количество параллельных шагов с умножениями (скалярными) в данном алгоритме, как и в других известных параллельных алгоритмах, равно $n^3 / p$. При наличии $p$ процессорных элементов и при базовом последовательном алгоритме перемножения матриц со сложностью $n^3$ меньшего порядка сложности достичь невозможно.

Для пересылок данных в этом алгоритме удобно использовать полнодоступный коммутатор или ($\log_2 p$)-мерный куб. Всего в алгоритме получается $2\log_4 p = \log_2 p$ блочных пересылок.

На каждом шаге пересылается $n^2$ элементов по $p$ штук одновременно (скалярных пересылочных шагов $n^2 / p$). Общее количество скалярных пересылочных шагов $n^2 \log_2 p / p$. Если $p = n$, то количество одновременных скалярных пересылок равно $n \log_2 n$ — это существенно меньше, чем при размещении матриц по строкам или столбцам, и меньше, чем пересылок в алгоритмах Кэннона и Фокса [4].

Перемножение прямоугольных матриц тоже может быть выполнено с помощью блочно-рекурсивного алгоритма. Действительно, пусть требуется перемножить матрицу A размером $n \times m$ на матрицу B размером $m \times k$. Обозначим $q = p^{1/2}$. Разобьем матрицу A на блоки размером $(n/q) \times (m/q)$, а матрицу B — на $(m/q) \times (k/q)$. Тогда задача перемножения прямоугольных матриц будет сведена к задаче перемножения квадратных блочных матриц размером $q \times q$, где $q = p^{1/2}$.

**Отображение пересылок между сетями различной топологии.** Известна задача переноса программ с одних многопроцессорных вычислительных систем на другие. При этом вычислительные системы могут различаться не только количеством вычислительных устройств, но и коммуникационной системой [8, 9]. Тогда возникает необходимость отображения топологии (моделирования пересылок, ориентированных на одну сеть, на другой сети). Например, в работе [6] рассмотрено моделирование на гиперкубе кольцевой сети с помощью кодов Грея.

Рассмотрим задачу моделирования $n$-мерного куба на кольце с количеством узлов $p = 2^n$. Напомним, что вершины $n$-мерного куба кодируются булевыми векторами $(x_1, x_2, \ldots, x_n)$ длиной $n$.

К сожалению, использовать коды Грея [6] для этой задачи не удобно. Действительно, для $n=3$ расстояние по кольцу между вершинами, соответствующими (0,0,0) и (1,0,0), равно 1, а между (0,0,1) и (1,0,1) — 4, хотя оба этих ребра $n$-куба соответствуют изменению первой координаты. Во многих задачах пересылки данных по таким ребрам предполагается выполнять одновременно. В данном примере на кольце эти пересылки будут выполняться в разное время.

Для случая $n=2$ задача тривиальна, поскольку двумерный куб представляет собой кольцо с 4 вершинами. Вершинам двумерного куба (0,0), (0,1), (1,1) (1,0) сопоставляются узлы кольцевой сети с номерами 0, 1, 2, 3 соответственно, т.е. $\varphi(0,0)=0$, $\varphi(0,1)=1$, $\varphi(1,1)=2$, $\varphi(1,0)=3$.

Определим сопоставление вершин $n$-мерного куба узлам кольцевой сети для $n>2$ по следующей формуле:

$$\varphi\left(x_1,x_2,\ldots,x_n\right)=\left(x_1+2x_2+\ldots+2^{n-3}x_{n-2}\right)+2^{n-2}\varphi\left(x_{n-1},x_n\right).$$

Первое слагаемое в данной формуле — это число, состоящее из первых $n-2$ цифр двоичного кода вершины $n$-мерного куба.

Рассмотрим подробнее указанное сопоставление узлов $n$-мерного куба узлам кольца. Сопоставление узлов $n$-мерного куба узлам кольца будем строить по индукции.

В предположении, что построено сопоставление узлов $n$-мерного куба узлам кольца для $n=k,k>1$, построим его для случая $n=k+1$. Пусть узлы кольца пронумерованы от 0 до $(n-1)$. Рассмотрим подмножество узлов кольца с четными номерами как кольцевую сеть (подкольцо), в которой время пересылки между соседними узлами в 2 раза дольше, чем в исходной кольцевой сети. Это подкольцо будем называть 0-подкольцо. Аналогичным образом узлы с нечетными номерами образуют подкольцо, которое будем называть 1-подкольцом.

Пусть $(x_1,x_2,\ldots,x_{k+1})$ — произвольная вершина $(k+1)$-мерного куба. Рассмотрим подмножество вершин $n$-мерного куба, состоящее из векторов $(0,x_2,\ldots,x_{k+1})$. Это подмножество порождает подграф, изоморфный $k$-мерному кубу. Узлы этого $k$-мерного куба сопоставим узлам 0-подкольца. Аналогично подмножество вершин $(1,x_2,\ldots,x_{k+1})$ также порождает подграф, изоморфный $k$-мерному кубу. Узлы этого $k$-мерного куба сопоставим узлам 1-подкольца.

Рассмотрим в $n$-мерном кубе множество ребер, соединяющих вершины вида $(x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_{k+1})$ и $(x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_{k+1})$. Ребра $n$-мерного куба, принадлежащие одному такому множеству, будем называть однотипными. Всего получается $n$ типов ребер. При реализации принципа сдваивания на $n$-мерном кубе используются одновременные пересылки по однотипным ребрам. Поэтому при моделировании $n$-мерного куба на кольце важно, чтобы такие пересылки эффективно реализовывались.

**Теорема 2.** При описанном выше сопоставлении вершинам $n$-мерного куба узлов кольцевой сети расстояние между концевыми узлами всех однотипных ребер одинаково.

Доказательство может быть проведено методом математической индукции.

***Следствие.*** Пересылки на кольце, соответствующие пересылкам по однотипным ребрам $n$-мерного куба, требуют одинакового времени.

В работе [6] рассмотрен способ представления на гиперкубе двумерной решетки, число узлов на сторонах которой является степенью двойки. Представим кольцо на решетке, что позволит на решетке моделировать пересылки $n$-мерного куба.

Если сеть имеет гамильтонов цикл (цикл, проходящий через каждую вершину строго один раз), то такой цикл и можно рассматривать как кольцевую сеть. Покажем, как построить гамильтонов цикл на прямоугольной решетке типа „тор“.

Допустим, что у прямоугольной решетки хотя бы одна из сторон содержит четное количество узлов. Пусть размерность этой решетки $m(2k)$. Тогда номера узлов, приведенные на рис. 5 (без окаймления), составляют гамильтонов цикл на этой решетке.

Пусть теперь прямоугольная решетка содержит нечетное количество узлов $(2m+1)\times(2k+1)$. В этом случае в решетке выделяется подрешетка размерностью $(2m)\times(2k)$, узлы которой нумеруются в указанной выше последовательности (см. рис. 5, с окаймлением).

На многомерном торе кольцо можно моделировать по правилу индукции, для двумерного тора отображение построено. Предположим, что оно существует для $m$-мерного тора и построим его для $(m+1)$-мерного тора. Пусть $W_1W_2\ldots W_{m+1}$ — $(m+1)$-мерный тор. Тогда, по предположению индукции, на $m$-мерном торе $W_2\ldots W_{m+1}$ можно смоделировать кольцо, которое обозначим $C$. Тогда исходный $(m+1)$-мерный тор можно представить в виде двумерного тора $W_1C$ и на этом двумерном торе смоделировать кольцо.

У многокольцевых соединений [10], как правило, одно из колец является гамильтоновым циклом.

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 2*m* | (2*m*+1) | | | *m*(2*k*) | *m*(2*k*)+1 |
| 2 | 2*m*–1 | (2*m*+2) | | | *m*(2*k*)–1 | *m*(2*k*)+2 |
| … | | | | | | |
| (*m*–1) | (*m*+2) | 3*m*–1 | 3*m*+2 | | | |
| *M* | (*m*+1) | 3*m* | 3*m*+1 | … | *m*(2*k*–1) | *m*(2*k*+1) |
| (*m*+1)(2*k*+1) | … | … | … | … | *m*(2*k*+1)+2 | *m*(2*k*+1)+1 |

*Рис. 5*

**О сложности обращений к памяти алгоритма Штрассена.** Сложность стандартного алгоритма (в котором, согласно определению, строки одной матрицы скалярно умножаются на столбцы другой) равна $n^3$. В данном случае $n^3$ — это количество умножений. У алгоритма Штрассена [5, 11, 12] сложность равна $n^{2,81}$ (более точно, показатель степени равен $\log_2 7$). Этот специфический алгоритм дал импульс развитию теории сложности. Напомним, что сложность алгоритма — это количество операций как функция от количества входных данных, а сложность задачи — это сложность самого быстрого алгоритма. При этом под операциями подразумевались арифметические операции, как правило — умножения, т.е. требующие существенно большего времени, чем сложение. Позднее были найдены алгоритмы перемножения матриц еще менее сложные, чем алгоритм Штрассена. Существуют алгоритмы, представляющие собой комбинацию алгоритма Штрассена и стандартного алгоритма перемножения матриц [13]. Сложность задачи перемножения матриц пока не определена. Есть результаты, показывающие существование таких задач, для которых самого эффективного алгоритма не существует [14] — не исключено, что задача перемножения матриц является такой же.

Рассмотрим реальные преимущества алгоритма Штрассена по сравнению со стандартным. В 1988 г. на кафедре алгебры и дискретной математики механико-математического факультета Ростовского государственного университета (сейчас Южный федеральный университет) был программно реализован алгоритм Штрассена (студенческая работа, руководитель доц. Козак А.В.). Выяснилось, что алгоритм Штрассена начинает давать преимущества по сравнению со стандартным алгоритмом начиная с размерности матриц $n=32$. В 2002 г. опять был реализован алгоритм Штрассена (тоже студенческая работа, руководитель — автор настоящей работы). В этот раз преимущество алгоритма Штрассена начиналось с размерности матриц $n=512$. Попытаемся объяснить этот эффект. Примерно за 15 лет компьютеры стали совсем другими. Изменилось соотношение времени выполнения арифметических опе-

раций и времени обращения к памяти. А соотношение количества этих операций в алгоритме Штрассена и в стандартном алгоритме различно.

Умножение вещественных чисел современным компьютером может быть в 15 раз быстрее, чем считывание этих чисел из оперативной памяти. Попробуем оценить сложность обращений к памяти алгоритма Штрассена.

Предполагается, что размерность матрицы является степенью двойки: $n = 2^k$.

Исходные и результирующая матрицы представляются как блочные $2 \times 2$ матрицы с блоками размерностью $n/2$. Тогда произведение матриц имеет вид

$$\mathrm{C}=\mathrm{AB}$$

$$\mathrm{C} = \{C_{ij}\},\ \mathrm{A} = \{A_{ij}\},\ \mathrm{B} = \{B_{ij}\},\ i, j = 1, 2.$$

Здесь $C_{ij}$, $A_{ij}$, $B_{ij}$ $(i, j = 1,\ 2)$ — блоки матриц C, A и B соответственно.

По алгоритму Штрассена вычисляются семь вспомогательных матриц (размером $n/2$):

$$\begin{aligned}
M_1 &= (A_{12} - A_{22})(B_{21} + B_{22}),\\
M_2 &= (A_{11} + A_{22})(B_{11} + B_{22}),\\
M_3 &= (A_{11} - A_{21})(B_{11} + B_{12}),\\
M_4 &= (A_{11} + A_{12})B_{22},\\
M_5 &= A_{11}(B_{12} - B_{22}),\\
M_6 &= A_{22}(B_{21} - B_{11}),\\
M_7 &= (A_{21} + A_{22})B_{11}.
\end{aligned} \tag{1}$$

После этого элементы результирующей матрицы вычисляются по формулам

$$\begin{aligned}
C_{11} &= M_1 + M_2 - M_4 + M_6,\\
C_{12} &= M_4 + M_5,\\
C_{21} &= M_6 + M_7,\\
C_{22} &= M_2 - M_3 + M_5 - M_7.
\end{aligned} \tag{2}$$

В формулах (1) перемножаются матрицы меньшей размерности, что тоже может выполняться рекурсивно по такому же алгоритму.

Обозначим через $F(n)$ количество операций обращения к памяти (чтение или запись) в алгоритме Штрассена.

В формулах (1), (2) матрицы размером $n/2$ встречаются 37 раз. При этом формулы (1) содержат умножения матриц размером $n/2$, каждое из которых потребует $F(n/2)$ обращений к памяти. Итого можно записать рекуррентную формулу

$$F(n) = 7F(n/2) + 37(n/2)^2.$$

Учитывая, что $F(1) = 3$, из рекуррентной формулы можно получить аналитическую

$$\begin{aligned}
F(n) &= 7F(n/2) + 37(n/2)^2 = 7\{7F(n/4) + 37(n/4)^2\} + 37(n/2)^2 =\\
&= 7^k F(1) + 37\{7^{k-1}(n/2^k)^2 + \ldots + 7(n/4)^2 + (n/2)^2\} =\\
&= 7^k \cdot 3 + 37/7n^2\{(7/4)^k + \ldots + (7/4)^2 + 7/4\} =\\
&= 7^k \cdot 3 + 37/7n^2\, 7/3((7/4)^k - 1) =\\
&= 7^k \cdot 3 + 37/3n^2((7/4)^k - 1).
\end{aligned}$$

Пренебрегая в скобках слагаемым (–1) и учитывая, что $k = \log_2 n$, получим, что количество обращений к памяти в алгоритме Штрассена равно

$$F(n) \approx 7^k 3 + 37/3n^2 \left(7/4\right)^k = 7^k 3 + 37/37^k = 7^k 46/3 \approx n^{2,81} 46/3.$$

Рассмотрим количество обращений к памяти в стандартном алгоритме. Каждый элемент результирующей матрицы вычисляется как скалярное произведение двух векторов (строки первой матрицы на столбец второй). Такое скалярное произведение вычисляется накоплением суммы произведений в регистре и требует $n^2$ чтений и одну запись в память. Итого $2n^3 + n^2 \approx 2n^3$ обращений к памяти.

Асимптотически количество обращений к памяти алгоритма Штрассена меньше, чем у стандартного алгоритма. Проследим, начиная с какой размерности матриц $n$ в алгоритме Штрассена количество обращений к памяти будет меньше, чем в стандартном алгоритме — $2n^3 = n^{2,81} 46/3$, $n \approx 25\,000$.

Численный эксперимент для матриц такой размерности не представляется разумным, поскольку матрицы такой размерности могут быть размещены уже не в оперативной, а во внешней памяти.

Следует отметить, что рекурсивность алгоритма Штрассена тоже может вызвать дополнительные обращения к памяти (обращения к стеку, в котором хранятся параметры вызовов). Глубина рекурсивных вызовов равна $k = \log_2 n$. Если столько параметров вызовов не поместится в регистровой или кэш-памяти, то они будут помещаться в оперативной.

Итак, алгоритм Штрассена асимптотически эффективнее, но преимущество перед стандартным алгоритмом начинается с задач, размерность которых превышает практическую значимость.

## СПИСОК ЛИТЕРАТУРЫ


1. *Пипер Ш., Пол Д., Скотт М.* Новая эра в оценке производительности компьютерных систем // Открытые системы. 2007. № 9. С. 52—58.
2. *Корнеев В. Д.* Параллельное программирование в MPI. М.—Ижевск: Институт компьютерных исследований, 2003. 304 с.
3. *Gupta H., Sadayappan P.* Communication Efficient Matrix-Multiplication on Hypercubes // Parallel Computing. 1996. N 22. P. 75—99.
4. *Гергель В. П.* Введение в методы параллельного программирования. Образовательный комплекс. Н. Новгород: ННГУ, 2005.
5. *Прангишвили И. В., Виленкин С. Я., Медведев И. Л.* Параллельные вычислительные системы с общим управлением. М.: Энергоатомиздат, 1983. 312 с.
6. *Гергель В. П., Стронгин Р. Г.* Основы параллельных вычислений для многопроцессорных вычислительных систем. Н. Новгород: ННГУ, 2003.
7. *Метлицкий Е. А., Каверзнев В. В.* Системы параллельной памяти. Теория, проектирование, применение. Л.: ЛГУ, 1989. 240 с.
8. *Treleaven P.C.* Parallel architecture overview // Parallel Computing. 1988. N 8. P. 59—70.
9. *Корнеев В. В.* Параллельные вычислительные системы. М.: Нолидж, 1999. 311 с.
10. *Подлазов В. С.* Свойства мультикольцевых и гиперкубовых коммутаторов на произвольных перестановках // РАСО'2001 Тр. Междунар. конф. „Параллельные вычисления и задачи управления". М.: ИПУ РАН, 2001. С. 152—164.
11. *Strassen V.* Gaussian Elimination is not Optimal // Numer. Math. 1969. Vol. 13, N 4. P. 354—356.
12. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. М.: Мир, 1979. 536 с.

13. *Елфимова Л. Д.* Быстрый клеточный метод умножения матриц // Кибернетика и системный анализ. 2008. № 3. С. 55—59.

14. *Разборов Л. А.* P=NP? — проблема: взгляд из 90-х // Математические события XX в. М.: ФАЗИС, 2003. С. 399—416.

***Сведения об авторе***

***Борис Яковлевич Штейнберг*** — д-р техн. наук; Южный федеральный университет, Ростов-на-Дону; кафедра алгебры и дискретной математики; зав. кафедрой; E-mail: borsteinb@mail.ru

Рекомендована институтом

Поступила в редакцию 10.03.09 г.

УДК 519.6

Ю. К. Демьянович

# ПРОБЛЕМЫ РАСПАРАЛЛЕЛИВАНИЯ В НЕКОТОРЫХ ЛОКАЛЬНЫХ ЗАДАЧАХ

Дан краткий обзор проблем распараллеливания в локальных задачах, предложены общие принципы архитектурных решений многопроцессорных систем, предназначенных для решения упомянутых задач.

***Ключевые слова:*** *распараллеливание, локальные задачи, архитектура вычислительных систем, кратность накрытия.*

Существует большое количество задач, решение которых требует использования огромных вычислительных мощностей, зачастую недоступных для современных вычислительных систем. К таким задачам прежде всего относятся точные долгосрочные прогнозы климатических изменений и геологических катаклизмов (землетрясений, извержений вулканов, столкновений тектонических плит), прогнозы цунами и разрушительных ураганов, а также экологические прогнозы и т.п. Сюда следует отнести также прогнозирование результатов экспериментов во многих разделах физики, в особенности экспериментов по выявлению основ мироздания (экспериментов на коллайдерах со встречными пучками частиц, экспериментов, направленных на получение антиматерии и так называемой темной материи и т.д.). Важными задачами являются расшифровка генома человека, определение роли каждого гена в организме, влияние генов на здоровье человека и на продолжительность жизни. Не решена задача безопасного хранения вооружений, в особенности ядерного оружия — из-за запрета на ядерные испытания состояние накопленных ядерных зарядов можно определить лишь путем моделирования на компьютере большой мощности.

Постоянно появляются новые задачи подобного рода и возрастают требования к точности и скорости решения уже поставленных задач, поэтому вопросы разработки и использования сверхмощных компьютеров (называемых суперкомпьютерами) актуальны сейчас и в будущем. К сожалению, технологические возможности увеличения быстродействия процессоров ограничены: повышение быстродействия связано с уменьшением размеров процессоров, а при малых размерах появляются трудности из-за квантово-механических эффектов, вносящих элементы недетерминированности; эти трудности пока что не удается преодолеть. Из-за этого приходится идти по пути создания параллельных вычислительных систем, т.е. систем, в которых предусмотрена одновременная реализация ряда вычислительных процессов, связанных с

решением одной задачи. На современном этапе развития вычислительной техники такой способ, по-видимому, является одним из основных способов ускорения вычислений [1—3].

Первоначально идея распараллеливания вычислительного процесса возникла в связи с необходимостью ускорить вычисления для решения сложных задач при использовании имеющейся элементной базы. Предполагалось, что вычислительные модули (процессоры или компьютеры) можно соединить между собой так, чтобы решение задач на полученной вычислительной системе ускорялось во столько раз, сколько использовано в ней вычислительных модулей. Однако скоро выяснилось, что для интересующих сложных задач такого ускорения, как правило, достичь невозможно по двум причинам:

— любая задача распараллеливается лишь частично (всегда имеются фрагменты, которые невозможно распараллелить),

— коммуникационная среда, связывающая отдельные части параллельной системы, функционирует значительно медленнее процессоров, так что передача информации существенно задерживает вычисления.

Из-за этого в согласии с теоретическими выводами на практике удавалось использовать незначительную часть мощности параллельной вычислительной системы: считалось достаточно эффективным использование 20 % мощности системы. Но даже и в этом случае создание параллельных суперкомпьютеров оправдано, поскольку удается добиться значительного прогресса при решении перечисленных суперсложных задач.

В свете сказанного, на первый взгляд, кажется удивительной высокая реальная производительность ряда современных суперкомпьютеров по сравнению с их пиковой производительностью. Так, например, в последней, 32-й, редакции списка TOP500 на первом месте находится суперкомпьютер RoadRunner с пиковой производительностью 1457 Tflops (т.е. $1457 \cdot 10^{12}$ операций с плавающей точкой в секунду — от *англ*. floating point operations per second); при этом реальная производительность компьютера (на тесте LINPACK, содержащем задачи линейной алгебры) составила 1026 Tflops ($1026 \cdot 10^{12}$ операций с плавающей точкой в секунду). На втором месте в этом списке находится суперкомпьютер CrayXT5 с пиковой производительностью 1381 Tflops и реальной производительностью (на тесте LINPACK) 1059 Tflops. В первом случае реальная производительность составляет 70 % от пиковой, а во втором — еще больше — 77 %.

Аналогичная ситуация прослеживается и для других суперкомпьютеров, например, в период с 2003 по 2005 г. на первом месте списке TOP500 находился суперкомпьютер Earth Simulator японской фирмы NEC с пиковой производительностью 43 Tflops и реальной производительностью (на тесте LINPACK) почти 36 Tflops (в последней версии списка TOP500 этот компьютер опустился на 74 место); и в этом случае реальная производительность составила 84 % пиковой производительности.

Объяснение этому феномену, по-видимому, следует искать в согласованности реальных задач, на которых замерялась производительность, с архитектурными решениями, принятыми при создании компьютеров. В связи с этим возникает вопрос, в какой степени обоснованно использование теста LINPACK (и вообще задач линейной алгебры) для характеристики возможностей компьютерных систем при решении суперсложных задач.

Анализ методов решения упомянутых выше суперсложных задач показывает, что при разработке архитектуры суперкомпьютеров очень важно учитывать свойства алгоритмов численного решения систем линейных алгебраических уравнений. Более того, ввиду локального характера всех упомянутых задач матрицы соответствующих им систем уравнений имеют специальную структуру, что позволяет предложить специальную архитектуру суперкомпьютеров, названную нами локально кратной архитектурой.

**О классификации архитектурных решений вычислительных систем.** Остановимся на некоторых общепринятых способах классификации вычислительных систем.

Среди параллельных систем различают конвейерные, векторные, матричные, систолические, спецпроцессоры и т.п. Родоначальниками параллельных систем являются суперкомпьютеры ILLIAC, CRAY и CONVEX. В настоящее время все суперкомпьютеры представляют собой параллельные системы.

Суперкомпьютеры каждого типа создаются в нескольких экземплярах, обычно каждый тип имеет определенные неповторимые архитектурные, технологические и вычислительные характеристики, поэтому сравнение суперкомпьютеров весьма сложная задача, не имеющая однозначного решения. Тем не менее разработаны определенные принципы условного сравнения компьютеров (это важно для их дальнейшего совершенствования и для продвижения на рынке).

Имеется несколько вариантов классификации компьютеров, обычно они носят достаточно формальный характер. Приведем некоторые примеры классификации.

*Классификация Флинна* — определяется единственность или множественность потоков данных и команд (классы SIMD и MIMD).

*Классификация Фенга* — выявляется тип параллелизма: пословный или поразрядный.

*Классификация Джонсона* основана на использовании двух вариантов признаков: компьютеры с общей или распределенной памятью комбинируются с двумя вариантами коммуникаций: с помощью передачи сообщений или с помощью общих переменных.

*Классификация Скилликорна* — компьютер описывается как абстрактная структура, состоящая из компонентов четырех типов: процессора команд, процессора данных, иерархии памяти, коммутатора.

Однако приведенные способы классификации дают мало информации о возможностях системы и о том классе задач, для которого использование системы наиболее эффективно. Отметим, что имеется еще около полутора десятков других вариантов классификации, но они также содержат мало информации.

**Краткая характеристика основных классов параллельных систем.** Одной из важнейших характеристик параллельных вычислительных систем является тип памяти: общая или распределенная.

Общая память представляет собой физически единую структуру, к которой имеют доступ (равный) все процессоры параллельной системы. В этом случае говорят о SMP-системе (симметричной мультипроцессорной). Основные проблемы при такой организации памяти состоят в обеспечении быстрого доступа к памяти большого объема и в быстром разрешении коллизий, возникающих при практически одновременном обращении процессоров к общей памяти. SMP-система состоит из однородных процессоров и массива общей памяти, к любому фрагменту которой все процессоры имеют одинаковый (по скорости) доступ. Когерентность кэш-памяти поддерживается аппаратно. Примерами таких систем являются компьютеры фирм IBM, HP, Fujitsu и др. Из-за того что память общая, число процессоров невелико (не более 32). Система работает под управлением одной операционной системы (ОС), программирование ведется с использованием модели общей памяти (Open MP, POSIX threads).

Распределенная память (MPP — массивно-параллельная архитектура) характеризуется тем, что состоит из отдельных фрагментов — функциональных структур, физически распределенных между вычислительными модулями (процессорами). Основные проблемы, связанные с такой архитектурой, состоят в организации быстрого доступа к памяти как к единой (виртуально) структуре и в поддержании когерентности между ее фрагментами (т.е. в поддержании тождественности „копий“, появляющихся в процессе вычислений). Массивно-параллельная система обычно состоит из однородных вычислительных узлов, которые содержат несколько процессоров, локальную память, сетевой адаптер и, возможно, другие узлы.

Количество узлов может достигать нескольких тысяч, они связываются через коммуникационную среду (коммутатор и т.п.). Примерами таких систем являются IBM RS/6000, CRAY T3E, системы на базе транспьютеров (Parsytec). В программировании используется модель обмена сообщениями; распараллеливание производится с помощью библиотек в соответствии со стандартами MPI, PVM и др.

NUMA-архитектура (Non-Uniform Memory Acess) характеризуется тем, что память физически распределена, но логически общедоступна. Система состоит из однородных базовых модулей, каждый из которых содержит некоторое (небольшое) число процессоров и блока (локальной) памяти. Аппаратно поддерживается доступ к удаленной памяти, причем доступ к локальной памяти значительно быстрее, чем к удаленной. Иногда когерентность кэш-памяти отдельных узлов поддерживается аппаратно. Примерами таких компьютеров являются Sun HPC 10000, NUMA-Q 2000 и др. Число процессоров в NUMA-системах доходит до 256. Система работает под управлением одной ОС, программирование ведется с использованием модели общей памяти (Open MP, потоков POSIX).

Основным свойством параллельных векторных систем (PVP-систем) является наличие векторно-конвейерных процессоров с командами однотипной обработки данных. Несколько векторно-конвейерных процессоров имеют доступ к общей памяти (аналогично симметричной мультипроцессорной системе), составляя единый узел; несколько узлов могут быть объединены с помощью коммутатора (как в массово-параллельных системах). Примерами служат компьютеры фирм CRAY, Fujitsu VPP. При программировании используются векторизация циклов (для одного процессора) и их распараллеливание (для нескольких процессоров).

Наконец, кластерные системы представляют собой совокупность рабочих станций, или персональных компьютеров, которые используются в качестве простого варианта массивно-параллельного компьютера; в качестве коммуникационной среды применяется одна из сетевых технологий (Fast/Gigabit Ethernet, Myrinet и т.п.) с использованием шинной структуры или коммутатора. Кластерная система может иметь гетерогенный характер (т.е. представлять собой объединение компьютеров разной архитектуры). Программирование ведется в рамках модели передачи сообщений (например, с использованием библиотеки, поддерживающей стандарт MPI).

**Пример сложной задачи.** Характерным (и типичным) примером сложной вычислительной задачи является компьютерное моделирование климата. Климатическая система включает в себя атмосферу, океан, сушу, криосферу и биоту (биологическую составляющую). Климатом называется ансамбль состояний, который система проходит за большой промежуток времени. Под климатической моделью подразумевается математическая модель, описывающая климатическую систему с той или иной степенью точности. В ее основе лежат уравнения динамики сплошной среды и равновесной термодинамики. В модели описываются многие физические процессы, связанные с переносом энергии: перенос излучения в атмосфере, фазовые переходы воды, мелкомасштабная турбулентная диффузия тепла, диссипация кинетической энергии, образование облаков, конвекция и др.

Рассматриваемая модель представляет собой систему нелинейных уравнений в частных производных в трехмерном пространстве. Ее решение воспроизводит все главные характеристики ансамбля состояний климатической системы.

Если обозначить

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{v}\nabla = \frac{\partial}{\partial t} + v_x \frac{\partial}{\partial x} + \dots,$$

то простейшая система уравнений, моделирующая динамику атмосферы, решается в приземном сферическом слое (в тропосфере, окружающей Землю); она включает следующие уравнения:

— уравнение количества движения

$$\frac{D\mathbf{v}}{\Delta t} = -\frac{1}{\rho}\nabla p + g - 2\boldsymbol{\Omega}\times\mathbf{v},$$

где $p$ — давление, $\rho$ — плотность, $g$ — ускорение силы тяжести, $\boldsymbol{\Omega}$ — угловой вектор скорости вращения Земли, $\mathbf{v}$ — скорость ветра;

— уравнение сохранения энергии

$$c_p \frac{DT}{Dt} = \frac{1}{\rho}\frac{Dp}{Dt},$$

где $c_p$ — удельная теплоемкость атмосферы, $T$ — температура;

— уравнение неразрывности (сохранения массы)

$$\frac{Dp}{Dt} = -\rho\,\mathrm{div}\mathbf{v},$$

— уравнение состояния

$$p = \rho R T,$$

где $R$ — константа.

Фактически представлена система шести нелинейных скалярных уравнений относительно шести неизвестных функций (зависящих от трех координат $(x, y, z)$ и времени $t$), а именно относительно компонент $v_x, v_y, v_z$ вектора скорости $\mathbf{v}$ и функций $p$, $\rho$, $T$. К этим уравнениям присоединяются начальные и граничные условия; полученная система уравнений представляет собой математическую модель динамики атмосферы. Заметим, что климатическая модель намного сложнее. Работая с ней, приходится принимать во внимание следующее:

— при исследовании климата нельзя поставить глобальный натурный эксперимент;

— проведение численных экспериментов над моделями и сравнение результатов экспериментов с результатами наблюдений — единственная возможность изучения климата;

— сложность моделирования заключается в том, что климатическая модель включает в себя ряд моделей, которые разработаны неодинаково глубоко; при этом лучше всего разработана модель атмосферы, поскольку наблюдения за ее состоянием ведутся давно и, следовательно, имеется много эмпирических данных;

— общая модель климата далека от завершения, поэтому в исследования включают обычно лишь моделирование состояния атмосферы и океана.

Рассмотрим вычислительную сложность обработки модели состояния атмосферы. Предположим, что нас интересует развитие атмосферных процессов на протяжении 100 лет. При построении вычислительных алгоритмов используем принцип дискретизации: вся атмосфера разбивается на отдельные элементы (параллелепипеды) с помощью сетки с шагом 1° по широте и по долготе; по высоте берут 40 слоев. Таким образом, получается $2{,}6\cdot10^6$ элементов. Каждый элемент описывается десятью компонентами. В фиксированный момент времени состояние атмосферы характеризуется ансамблем из $2{,}6\cdot10^7$ чисел. Условия развития процессов в атмосфере требуют каждые 10 минут находить новый ансамбль, так что за 100 лет будем иметь $5{,}3\cdot10^6$ ансамблей. Таким образом, в течение одного численного эксперимента будет получено около $\left(2{,}6\cdot10^7\right)\cdot\left(5{,}3\cdot10^6\right)\approx1{,}4\cdot10^{14}$ числовых результатов. Если учесть, что для получения одного числового результата требуется $10^2$—$10^3$ арифметических операций, то приходим к выводу, что для одного варианта вычисления модели состояния атмосферы на интервале 100 лет требуется затратить $10^{16}$—$10^{17}$ арифметических действий с плавающей точкой. Следовательно, вычислительная система с производительностью $10^{12}$ операций в секунду при полной загрузке и эффективной программе будет работать $10^4$—$10^5$ секунд (иначе говоря, потребуется от 3 до 30 часов вычислений). Ввиду отсутствия точной информации о начальных и краевых условиях требуется просчитать сотни подобных вариантов.

Заметим, что расчет полной климатической модели займет на порядок больше времени.

**Структура и роль памяти в вычислительной системе.** Идеальная память должна обеспечивать процессор командами и данными непрерывно, чтобы не допускать простоев процессора. Кроме того, память должна иметь большую емкость с тем, чтобы обеспечивать обработку необходимых заданий. Это достигается использованием многоуровневой памяти с соответствующей иерархией; иногда говорят об иерархии памятей (во множественном числе).

Время доступа к данным зависит от объема и типа используемой памяти. Малое время доступа характерно для памяти малого объема; поэтому, введя малую быструю память (кэш-память), пересылают данные из основной памяти в буферную, обрабатывают их с использованием этой памяти и результат отправляют обратно в основную память.

Создание иерархической многоуровневой памяти, пересылающей блоки программ и данных между уровнями памяти за то время, пока другие блоки обрабатываются процессором, позволяет существенно снизить простои процессора в ожидании данных.

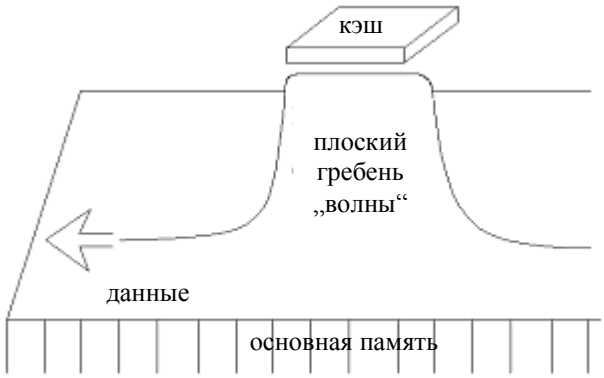


*Рис. 1*

Отметим, что чем более плоский „гребень“ у „волны“ (т.е. чем „шире“ волна), тем эффективнее обработка информации (рис. 1).

Что такое локальность данных? Признаки локальных данных:

1) процессор многократно использует выбранные из основной памяти данные для получения результата до того момента, когда результат будет отправлен в основную память;

2) положение выбираемых данных локализовано в основной памяти (они выбираются последовательно, „подряд“).

Эти признаки характерны для научных и инженерных расчетов (при решении уравнений математической физики), например, когда рассматривается физический процесс с локальными законами. В этом случае при переходе от значения к значению приходится обрабатывать небольшие „порции“ данных с большим количеством вложенных циклов, так что все данные из обрабатываемой „порции“ используются многократно.

В связи с тем что локально обрабатываемые данные могут возникать в динамике вычислений и не быть сконцентрированными в одной области при статическом размещении в основной памяти, буферную память организуют как ассоциативную (в которой данные содержатся в совокупности с их адресом в основной памяти, рис. 2). В результате получаем гибкое согласование структуры данных, требуемых в процессе вычислений, со статическими структурами основной памяти.

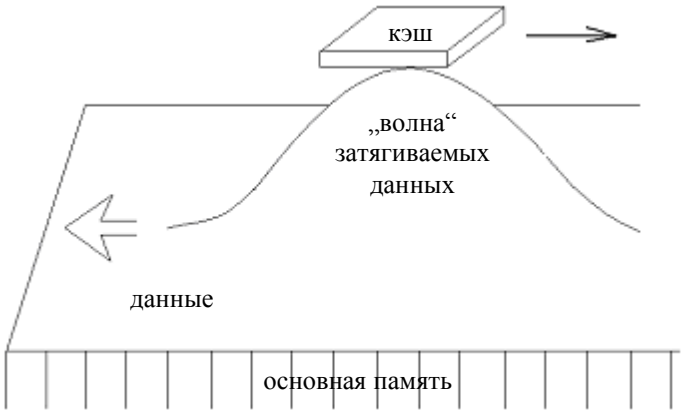


*Рис. 2*

**Локально кратная архитектура параллельной системы.** Наиболее эффективные методы решения упомянутых выше задач сводятся к выбору координатных функций в подходящем линейном пространстве и к построению приближения в его конечномерном подпространстве. Типичным примером является широко распространенный метод конечных элементов (МКЭ), харак-

терная черта которого — применение координатных функций с компактным носителем и с малой кратностью накрытия. Функции с аналогичными свойствами применяются для обработки числовых информационных потоков, высокая плотность которых требует больших вычислительных мощностей (сюда относятся обработка двумерных и трехмерных информационных потоков, связанных с натурными экспериментами, томографией, с обработкой космических потоков, передачей двумерных и трехмерных изображений, распознаванием образов и т.п.).

Для эффективного решения подобных задач необходимы вычислительные системы, архитектура которых согласована с применяемыми методами. Рассмотрим один из вариантов подобной архитектуры.

Пусть $\mathbb{N}$ — множество натуральных чисел, а $\mathbb{N}_0 \overset{\text{def}}{=} \mathbb{N} \cup \{0\}$. На некотором $n$-мерном дифференцируемом многообразии $T$ (для удобства в качестве $T$ можно рассматривать, например, тор или сферу) введем клеточное подразделение $U \overset{\text{def}}{=} \{K_j\}_{j \in J}$, т.е. рассмотрим непересекающиеся множества (клетки) $K_j$, гомеоморфные $n$-мерному открытому шару, объединение замыканий которых совпадает с $T$,

$$T = \bigcup_{j \in J} \bar{K}_j,$$

$J$ — некоторое конечное множество индексов.

Пусть $p \in \mathbb{N}_0$ и $s \in \mathbb{N}$.

**Определение 1.** Клетки называются инцидентными друг другу, если их замыкания пересекаются. Две клетки называются *p-инцидентными*, если пересечение их замыканий имеет размерность $j$, где $j \ge p$, $j \in \mathbb{N}_0$.

Выделим в множестве $U$ некоторое подмножество $U^*$, среди клеток которого нет инцидентных, т.е. $U^* \subset U$, причем для $\forall K', K'' \in U^*$ имеем $\bar{K} \cap \bar{K} = \varnothing$.

**Определение 2.** Клетки множества $U^*$ будем называть *помеченными*. Если известно, что клетка помеченная, то будем обозначать ее символом $K^*$.

**Определение 3.** Две клетки $K'$ и $K''$ называются $(p,s)$-*связанными*, если существует последовательность клеток

$$C^{(p,s)}(K', K'') \overset{\text{def}}{=} \{K' \overset{\text{def}}{=} K^{(0)}, K^{(1)}, K^{(2)}, \ldots, K^{(s)} \overset{\text{def}}{=} K''\}, \qquad (1)$$

каждые две соседних клетки в которой $p$-инцидентны. Последовательность (1) называется $(p,s)$-*связующей цепочкой* (или просто $(p,s)$-*цепочкой*) для клеток $K'$ и $K''$, число $s$ — *длиной цепочки*, а число $p$ называется *гарантированной мощностью цепочки*.

Для заданных клеток $K'$ и $K''$ и фиксированной пары чисел $(p,s)$ может быть несколько цепочек вида (1).

Множество всех цепочек обозначим $C^{(p,s)}(K', K'')$, оно не пусто лишь для конечного количества целочисленных значений $s$ и $p$. Очевидно, что множество

$$C^{(p)}(K', K'') \overset{\text{def}}{=} \bigcup_{s \in \mathbb{N}} C^{(p,s)}(K', K'')$$

представляет собой множество цепочек одинаковой гарантированной мощности, связывающих клетки $K'$ и $K''$, а множество

$$C^{(s)}(K', K'') \overset{\text{def}}{=} \{C^{(p,s)}(K', K'') \mid p \in \mathbb{N}_0\}$$

является множеством цепочек одинаковой длины (возможно, различных гарантированных мощностей), связывающих клетки $K'$ и $K''$.

Введем числа $s^{(p)}_{\min}(K',K'')$ и $s^{(p)}_{\max}(K',K'')$, определяемые формулами

$$s^{(p)}_{\min}(K',K'') \overset{\text{def}}{=} \min_{s\in\mathbb{N}}\{s \mid C^{(p,s)}(K',K'') \neq \varnothing\},$$

$$s^{(p)}_{\max}(K',K'') \overset{\text{def}}{=} \max_{s\in\mathbb{N}}\{s \mid C^{(p,s)}(K',K'') \neq \varnothing\}.$$

Число $s^{(p)}_{\min}(K',K'')$ называется минимальной, а число $s^{(p)}_{\max}(K',K'')$ — максимальной *p-удаленностью* клеток $K'$ и $K''$; полезными оказываются также числа

$$s_{\min}(K',K'') \overset{\text{def}}{=} \min_{p\in\mathbb{N}_0} s^{(p)}_{\min}(K',K''),$$

$$s_{\max}(K',K'') \overset{\text{def}}{=} \max_{p\in\mathbb{N}_0} s^{(p)}_{\max}(K',K''),$$

называемые минимальной и максимальной удаленностью клеток $K'$ и $K''$ соответственно. Отметим некоторые свойства $p$-инцидентности и $(p,s)$-связанности. Пусть $p_1 \le p$, тогда

1) если две клетки $p$-инцидентны, то они и $p_1$-инцидентны,

2) если цепочка (1) — $(p,s)$-связующая для клеток $K'$ и $K''$, то она является и $(p_1,s)$-связующей для этих клеток,

3) если две клетки $(p,s)$-связаны, то они и $(p_1,s)$-связаны,

4) если две клетки $(p,1)$-связаны, то они $p$-инцидентны,

5) справедливы включения

$$C^{(p,s)}(K',K'') \subset C^{(p_1,s)}(K',K''),\ C^{(p)}(K',K'') \subset C^{(p_1)}(K',K'').$$

Архитектуру предлагаемой вычислительной системы согласуем с клеточным подразделением следующим образом.

Будем считать, что каждая клетка представляет собой фрагмент памяти (далее термины „фрагмент памяти“ и „клетка“ отождествляем), а коммуникации между $(p,s)$-связанными фрагментами памяти $K'$ и $K''$ происходят по каналам, ассоциированным с $(p,s)$-связующими цепочками длиной $s$ и мощностью $p$.

Как известно, процессор (вместе с соответствующим оборудованием) можно рассматривать как фрагмент памяти с большим быстродействием. Будем считать, что каждая помеченная клетка представляет собой процессор (дальше термины „процессор“ и „помеченная клетка“ отождествляются). Если помеченная клетка фиксирована, то по отношению к ней любая другая клетка может рассматриваться как кэш-память. Более традиционно каждой помеченной клетке $K^*$ соотнести некоторую группу $G(K^*)$ „близлежащих“ к ней (в смысле упомянутой топологии) непомеченных клеток, например, клеток, связанных с $K^*$ цепочками длиной $s \le s_0(K^*)$, где $s_0(K^*)$ — некоторое фиксированное число. Совокупность клеток $G_s(K^*)$, связанных цепочками длиной $s$ с клеткой $K^*$, $s \le s_0(K^*)$, называется кэшем $s$-го уровня для процессора $K^*$.

**Определение 4.** Объединение $M(K^*) \overset{def}{=} G(K^*) \cap \{K^*\}$ называется вычислительным модулем, а $G(K^*)$ — его кэшем. Совокупность всех вычислительных модулей называется вычислительной системой.

Существенным моментом, отличающим предлагаемую архитектуру от обычно рассматриваемых, является то, что кэши различных вычислительных модулей могут пересекаться; в частности, априори фиксированная клетка $K$ может принадлежать кэшам нескольких вычислительных модулей.

**Определение 5.** Число вычислительных модулей, которым принадлежит данный фрагмент памяти $K \in U$, называется кратностью накрытия этого фрагмента и обозначается $\varkappa(K)$.

Число $\varkappa_0 \overset{\text{def}}{=} \max_{K \in U} \varkappa(K)$ называется кратностью вычислительной системы.

Вычислительная система с рассмотренной архитектурой называется локально кратной вычислительной системой.

Если кратность накрытия можно изменять добавлением или удалением вычислительных модулей, то вычислительную систему можно назвать локально кратной вычислительной системой с изменяемой кратностью накрытия.

Закончим изложение замечанием о том, что для практических целей было бы важно иметь *однородную* локально кратную вычислительную систему, т.е. такую, у которой кратность накрытия для всех фрагментов памяти одинакова: $\varkappa(K) = \varkappa_0 \ \forall K \in U$.

**Заключение.** Анализ сложных задач и методов их численного решения показывает, что при разработке архитектуры суперкомпьютеров очень важно учитывать свойства алгоритмов решения систем линейных алгебраических уравнений. Ввиду локального характера упомянутых задач требования к архитектуре суперкомпьютеров могут быть в значительной степени конкретизированы; они дают архитектурные решения, которые могут привести к созданию суперкомпьютеров, занимающих промежуточное положение между компьютерами с разделяемой памятью и компьютерами с распределенной памятью: память разделена на фрагменты, к каждому фрагменту имеют прямой доступ несколько вычислительных модулей; в свою очередь, каждый вычислительный модуль имеет прямой доступ к нескольким фрагментам памяти. Такая архитектура полностью соответствует локальным аппроксимациям, методам конечных элементов, сплайнам и вейвлетам, применяемым при решении перечисленных выше сложных задач [4, 5]; этот подход приведет к существенному повышению реальной производительности на упомянутых задачах.

Работа частично поддержана грантами РФФИ № 07-01-00451 и 07-01-00269.

## СПИСОК ЛИТЕРАТУРЫ

1. *Воеводин В. В., Воеводин Вл. В.* Параллельные вычисления. СПб: БХВ-Петербург, 2002. 608 с.

2. *Эндрюс Г. Р.* Основы многопоточного, параллельного и распределенного программирования. М.: Вильямс, 2003. 512 с.

3. [Электронный ресурс]: <http://parallel.ru>.

4. *Демьянович Ю. К.* Локальная аппроксимация на многообразии и минимальные сплайны. СПб: Изд-во СПбГУ, 1994. 356 с.

5. *Демьянович Ю. К.* Вэйвлеты на многообразии // Докл. РАН. 2009. Т. 421, № 2. С. 1—5.

***Сведения об авторе***

***Юрий Казимирович Демьянович*** — д-р. физ.-мат. наук, профессор; Санкт-Петербургский государственный университет, кафедра параллельных алгоритмов; зав. кафедрой; E-mail: Yuri.Demjanovich@JD16531.spb.edu

Рекомендована институтом

Поступила в редакцию
10.03.09 г.

# ПРИМЕНЕНИЕ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ В ЗАДАЧАХ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ

УДК 004.272.2

М. В. Якобовский

## ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ НА МНОГОПРОЦЕССОРНЫХ СИСТЕМАХ: АЛГОРИТМЫ И ИНСТРУМЕНТЫ

Обсуждаются проблемы, возникающие при проведении вычислительных экспериментов на многопроцессорных системах. Рассматриваются вопросы рациональной декомпозиции, огрубления триангулированных поверхностей, иерархической обработки и распределенной визуализации больших объемов сеточных данных.

***Ключевые слова:*** *многопроцессорные системы, распределенная визуализация, рациональная декомпозиция, математическое моделирование, вычислительный эксперимент, параллельные алгоритмы, триангуляция, огрубление данных.*

**Введение.** Современные многопроцессорные системы обладают производительностью более $10^{14}$ операций с плавающей точкой в секунду, что потенциально позволяет за короткое время выполнять большие объемы вычислений. С помощью суперкомпьютеров возможно проведение вычислительных экспериментов, направленных на решение фундаментальных и прикладных задач газовой динамики, горения, микро- и наноэлектроники, экологии и многих других. Несмотря на то что потребность в проведении подобных экспериментов велика, большая часть современных суперкомпьютеров востребована далеко не в полной мере. Тому есть весомые причины.

Фундаментальной проблемой создания методов, позволяющих эффективно использовать совокупную мощность множества процессоров, является необходимость разработки алгоритмов, обладающих существенным запасом внутреннего параллелизма на всех этапах расчета. Каждый шаг решения задачи должен содержать достаточное количество взаимно независимых операций, выполнение которых возможно одновременно на всех выделенных для расчета процессорах. В идеале *все* множество операций, необходимых для решения задачи, следует равномерно распределить между *всеми* процессорами на протяжении *всего* времени выполнения расчета. Указанная проблема, безусловно, имеет определяющий характер, но она далеко не единственная.

Рост вычислительной мощности суперкомпьютеров обеспечивается сегодня за счет экстенсивного увеличения числа обрабатывающих элементов — процессоров, процессорных ядер, мультитредовых устройств и им подобных (далее — процессоров). Увеличение вычислительной мощности за счет роста числа поддерживаемых потоков команд, а не за счет скорости обработки одного потока, обусловливает несоответствие между возможностями тради-

ционных средств подготовки и анализа данных и способностью многопроцессорных систем к генерации больших массивов результатов.

Одним из наиболее активно используемых методов изучения процессов, протекающих в сложных многомерных объектах, является метод математического моделирования. Часто этот метод является единственной возможностью изучения сложных нелинейных явлений. В силу естественных временных ограничений невозможен натурный эксперимент при изучении глобальных изменений климата. В соответствии с действующими международными соглашениями невозможен натурный эксперимент в области изучения поведения вещества в экстремальных условиях ядерного взрыва. Дорог и практически невоспроизводим натурный эксперимент, имеющий своей целью определение оптимальных режимов добычи углеводородов на нефтяных месторождениях. Список можно продолжить, но принцип ясен — там, где натурный эксперимент невозможен, необходим эксперимент вычислительный. В его рамках создается математическая модель изучаемого явления. Использование методов математической физики приводит к описанию объекта исследования системой нелинейных многомерных уравнений в частных производных, решение которых определяется с помощью численных методов. Непрерывная среда заменяется дискретным аналогом — конечной сеткой по времени и пространству. Дифференциальные уравнения, действующие в непрерывном пространстве, заменяются алгебраическими, действующими в дискретном пространстве разностной сетки или конечных элементов. Решение алгебраических уравнений возлагается на суперкомпьютер.

Накоплен значительный опыт применения многопроцессорных систем для моделирования физических и технологических процессов, но он ориентирован в первую очередь на параллельные системы средней производительности, содержащие относительно небольшое число процессоров. При переходе к вычислительным системам, количество процессоров в которых исчисляется сотнями и более, требуется создание адекватных средств обработки больших объемов данных.

Увеличение числа процессоров, используемых для решения задачи, приводит к уменьшению объема вычислений, выполняемых на каждом из процессоров. В свою очередь, это ведет к относительному росту доли накладных расходов, обусловленных необходимостью обеспечения взаимодействия процессов передачи данных между процессорами и наличием других операций, обеспечивающих согласованное решение задачи на многопроцессорной системе. С ростом числа процессоров наступает насыщение — момент, после которого увеличение числа процессоров уже не приводит к сокращению времени решения задачи. Таким образом, за счет многопроцессорности сложно сокращать время решения задачи, но с помощью суперкомпьютеров можно решать более сложные задачи, увеличивая степень детализации изучаемых объектов, включая в рассмотрение дополнительные факторы, что влечет за собой увеличение объемов данных, описывающих эти объекты.

Оперирование большими объемами данных предъявляет новые требования и к используемым алгоритмам, и к самим принципам организации работы на суперкомпьютере. Работа становится невозможной без привлечения параллельных алгоритмов и средств, поддерживающих всю цепочку действий, требуемых для численного моделирования на подробных сетках, в том числе методов формирования геометрических моделей высокого качества, генераторов поверхностных и пространственных сеток, средств декомпозиции сеток, библиотек распределенного ввода—вывода, алгоритмов и библиотек балансировки загрузки процессоров, средств визуализации результатов крупномасштабных экспериментов и многих других.

Будем считать объем данных „большим“, если выполняются некоторые из следующих условий:

— вычислительной мощности любого отдельно взятого процессорного узла недостаточно для обработки всего объема данных за приемлемое время;

— оперативной памяти любого отдельно взятого процессорного узла недостаточно для хранения всего объема обрабатываемых данных;

— пропускной способности сетей передачи данных недостаточно для передачи за разумное время всего объема полученной информации от сервера хранения данных до рабочего места пользователя.

Рассмотрим в рамках настоящей статьи проблемы рациональной декомпозиции сеток и визуализации сеточных данных.

**Декомпозиция данных.** Будем руководствоваться широко использующимся на практике методом геометрического параллелизма, в соответствии с которым элементы расчетной сетки и ассоциированные с ними сеточные данные равномерно распределяются по процессорам компактными группами. За счет того что каждый процессор обрабатывает размещенные на нем данные, вычислительная нагрузка так же равномерно распределяется по процессорам.

Можно сформулировать первый критерий разбиения сеток: данные следует распределить так, чтобы объемы вычислительной работы, выполняемой на каждом процессоре, были одинаковыми.

Второй критерий связан с необходимостью минимизации сопровождающих расчет накладных расходов. Распределим сеточные данные так, чтобы сократить объемы передаваемых между процессорами в ходе вычислений данных. В контексте рассматриваемых задач на суперкомпьютер возлагается задача решения систем алгебраических уравнений. Есть два основных метода построения таких систем — на основе явных и неявных схем. Использование явных схем выглядит наиболее привлекательными, поскольку не требует привлечения дорогостоящих в вычислительном плане алгоритмов обращения матриц большого размера. С точки зрения описания нестационарных физических процессов явные схемы также имеют преимущество, поскольку они естественным образом отражают связь между значениями физических переменных, соответствующими последовательно моделируемым моментам времени. Системы алгебраических уравнений, соответствующие явным разностным схемам, обладают важным свойством локальности, позволяющим эффективно использовать многопроцессорные системы с распределенной памятью. Оно заключается в том, что для вычисления новых значений физических величин в некотором узле сетки необходимо знать о значениях сеточных функций только в тех узлах, что находятся на небольшом (по числу ребер) расстоянии от обрабатываемого узла. В первом приближении это означает, что объем данных, передаваемых между процессорами, будет определяться числом узлов, соседствующих с узлами, хранящимися на других процессорах.

Теперь можно сформулировать второй критерий: данные следует распределить так, чтобы минимизировать число ребер, соединяющих узлы, хранящиеся на разных процессорах.

Таким образом, задача рациональной декомпозиции сетки сводится к разбиению вершин графа сетки на заданное число классов эквивалентности — доменов. Требуется найти такой вариант разбиения сетки на домены, содержащие одинаковое количество вершин, при котором число ребер, соединяющих вершины из разных доменов (число „разрезанных“ ребер), минимально. Задача рационального разбиения графов принадлежит классу $NP$-полных, что заставляет использовать для ее решения эвристические алгоритмы.

Существует ряд пакетов декомпозиции графов, среди которых выделяются ParMETIS, CHACO, PARTY, JOSTLE и SCOTCH. Наиболее эффективны иерархические методы рационального разбиения графов (B. Hendrickson, R. Leland, G. Karypis, V. Kumar и др.) [1—3]. В их основе лежит следующая последовательность действий: огрубление графа (построение последовательности уменьшающихся в размере вложенных графов), начальная декомпозиция огрубленного графа на заданное число доменов, восстановление графа и локальное уточнение границ доменов.

Огрубление графа выполняется путем многократного объединения пар соседних вершин в одну, в результате чего формируется граф с небольшим числом агрегированных вершин, каждой из которых соответствует подмножество вершин исходного графа.

Начальная декомпозиция может быть выполнена с помощью практически любого алгоритма, вплоть до алгоритма полного перебора, поскольку число вершин огрубленного графа может быть сокращено до достаточно малого значения. Но, как правило, качество такой декомпозиции будет низким, поскольку оно непосредственно зависит от равномерности процесса огрубления графа. При разбиении огрубленного графа практически неизбежно формируются домены несовпадающих весов, так как веса агрегированных вершин могут иметь значительный разброс. В связи с этим необходимо выполнять локальное уточнение границ доменов, что позволяет уравновесить веса доменов и уменьшить число ребер, пересекающих их границы, например, с помощью алгоритмов Kernighan-Lin (KL) и Fiduccia-Mattheyses (FM) [4], обладающих относительно низкой вычислительной сложностью.

Для ряда задач актуальны два дополнительных критерия декомпозиции:

— минимизация максимальной степени домена (число соседних доменов) с целью снижения числа актов обмена данными на каждом шаге по времени и уменьшения сложности вычислительного алгоритма;

— обеспечение связности каждого из описывающих домен подграфов.

Перечисленные ранее алгоритмы не позволяют контролировать связность и максимальную степень доменов. Одним из методов, обеспечивающих формирование начального приближения высокого качества, является алгоритм спектральной бисекции. Применяя его рекурсивно, можно разбивать граф на произвольное число частей. Декомпозиция графа на две части может быть выполнена с помощью упорядочения вершин графа по значениям компонент вектора Фидлера — собственного вектора, соответствующего наибольшему ненулевому собственному значению спектральной матрицы графа (матрицы Лапласа) [3, 5, 6]. Разбиение графа на большее число частей, согласно компонентам вектора Фидлера, может быть использовано для формирования доменов, имеющих малое число соседей. Существенным недостатком метода является сложность определения компонент вектора Фидлера вырожденной спектральной матрицы графа, что ограничивает число вершин разбиваемого графа.

Для обеспечения связности подграфов каждого из доменов может быть использован инкрементный метод [7]. Определив глубину произвольной вершины как кратчайшее расстояние от нее до множества граничных вершин домена, можно ввести понятие ядра заданного уровня. Ядро уровня $k$ определим как подграф, образованный вершинами глубиной не менее $k$ и ребрами, инцидентными только этим вершинам. Алгоритм инкрементного роста ориентирован на формирование доменов, в каждом из которых ядра заданного уровня связны, что является более существенным условием, чем требование связности каждого из доменов. Тестирование показывает, что использование инкрементного алгоритма обеспечивает и малое число разрезанных ребер, и высокий уровень связности ядер доменов.

**Иерархическая обработка больших сеток.** Моделирование на многопроцессорных вычислительных системах центров коллективного пользования сопряжено с выполнением длительных вычислений с помощью большого количества сравнительно коротких сеансов. Число используемых процессоров может изменяться от одного сеанса к другому, что вынуждает многократно решать задачу балансировки загрузки. Несмотря на высокую эффективность иерархических алгоритмов, разбиение нерегулярных расчетных сеток большого размера занимает значительное время. Для снижения потерь целесообразно использовать иерархический метод хранения и обработки больших сеток. В соответствии с ним расчетная сетка предварительно разбивается на множество блоков небольшого размера — микродоменов. В дальнейшем сетка хранится в виде набора микродоменов и графа, определяющего их взаимосвязи, — макрографа. Каждая из вершин макрографа соответствует микродомену. Перед

очередным сеансом расчета производится разбиение макрографа, и каждому процессору назначается список микродоменов. Поскольку в макрографе значительно меньше вершин, чем в исходной сетке, его разбиение требует меньшего времени и может быть выполнено последовательными алгоритмами. Дополнительный выигрыш по времени достигается за счет возможности распределенного хранения больших графов как совокупности микродоменов, что значительно уменьшает накладные расходы на чтение и запись сеток большим числом процессоров.

**Распределенная визуализация.** С ростом числа процессоров, используемых для проведения вычислительных экспериментов, сложность задачи преобразования больших объемов сеточных данных к виду, пригодному для наглядного отображения, существенно возрастает. Для визуализации уже недостаточно ресурсов персональных компьютеров (ПК) — рабочих мест пользователей [8—10]. Наиболее естественным путем решения проблемы является использование технологии клиент—сервер, в соответствии с которой (рис. 1):

— серверная часть системы визуализации, как правило, выполняемая на многопроцессорной системе, обеспечивает обработку большого объема данных;

— клиентская часть системы визуализации, выполняясь на ПК, обеспечивает интерфейс взаимодействия с пользователем и непосредственное отображение данных, подготовленных сервером, используя при этом аппаратные возможности персонального компьютера как для построения наглядных визуальных образов (с помощью графических ускорителей, стереоустройств), так и для управления ими (с помощью многомерных манипуляторов).

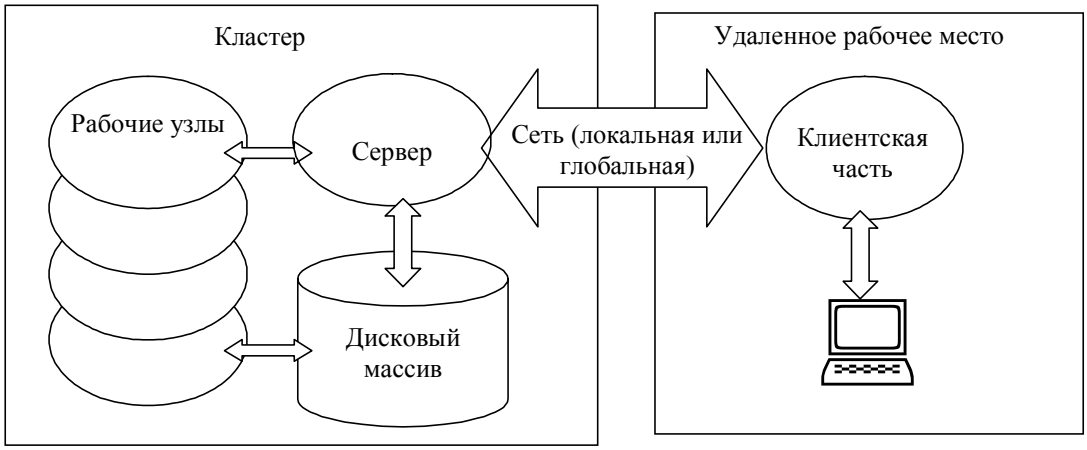


*Рис. 1*

На сервере целесообразно подготавливать данные, обеспечивающие формирование клиентом именно трехмерного образа объекта, манипулирование которым с целью его изучения с различных направлений возможно уже без дополнительных обращений к серверу. В этом заключается одно из существенных отличий обсуждаемого подхода от методов, используемых в большинстве доступных систем визуализации, выполняющих на сервере „рендеринг“ — формирование двумерного растрового образа изучаемого объекта.

Одним из наиболее мощных и наглядных методов визуализации трехмерных скалярных данных является визуализация изоповерхностей, описываемых множеством треугольников. Современные видеоускорители аппаратно поддерживают отображение массивов треугольников, что значительно повышает наглядность визуализации как за счет высокой скорости вывода данных на экран, так и за счет возможности автоматического формирования стереоизображений.

Ядро системы визуализации представлено рядом алгоритмов фильтрации и огрубления первичных данных, позволяющих аппроксимировать результаты вычислений ограниченным объемом данных, достаточным, однако, для восстановления изучаемого образа с заданным уровнем качества. Фактически к алгоритму огрубления предъявляются два требования, вызывающие потребность обеспечить интерактивный режим работы, а следовательно — высокую

скорость предоставления пользователю визуального образа. За короткое время необходимо огрубить данные и передать результат на компьютер пользователя. Следовательно, во-первых, алгоритм огрубления должен работать быстро. Во-вторых, необходимо выполнить огрубление данных *до объема*, заданного временем, отведенным на передачу данных на компьютер пользователя. Число описывающих изоповерхность узлов велико и по порядку величины может совпадать с числом узлов исходной трехмерной сетки, поэтому и необходим этап ее огрубления до размеров, допускающих их передачу через медленные каналы связи за короткое время. Необходимые коэффициенты „сжатия“ — отношения объемов данных, описывающих исходную изоповерхность и ее образ, достигают сотен тысяч и более, поэтому следует использовать методы сжатия с потерей точности. В первую очередь визуально воспринимаются основные контуры и формы трехмерного объекта, спроецированного на двумерный экран, следовательно, при изучении объекта „в целом“ деталями можно пожертвовать. При необходимости фрагменты объекта можно изучить с большим увеличением и меньшей потерей точности.

Простейшие алгоритмы сжатия, основанные на снижении точности представления вещественных чисел, описывающих сетку, и последующей компрессии с помощью стандартных алгоритмов, подобных групповому кодированию (*RLE*) или кодированию строк (*LZW*), значительного выигрыша не дают. Большую часть объема данных о триангуляции составляет целочисленная информация, описывающая ее топологию — связи между узлами. Стандартными алгоритмами сжатия без потерь эта информация практически не обрабатывается. Таким образом, основной интерес представляют алгоритмы, формирующие некоторую новую триангулированную поверхность, аппроксимирующую исходную изоповерхность, но содержащую значительно меньшее количество узлов.

**Огрубление триангулированных поверхностей.** Эффективны алгоритмы огрубления триангулированных поверхностей на основе методов редукции [11, 12]. Их основная идея заключается в итерационном удалении из исходной поверхности некоторого количества узлов таким образом, чтобы триангуляция, определенная на оставшихся точках, аппроксимировала исходную поверхность с требуемой точностью. В качестве иллюстрации на рис. 2, *а* представлена триангулированная сфера (точек 98 880, треугольников 195 884), а на рис 2, *б* — результат огрубления сферы с помощью редукции (точек 215, треугольников 375). Методы редукции допускают огрубление изоповерхностей, обладающих достаточно сложной структурой, например, имеющих самопересечения.

*а*) *б*)

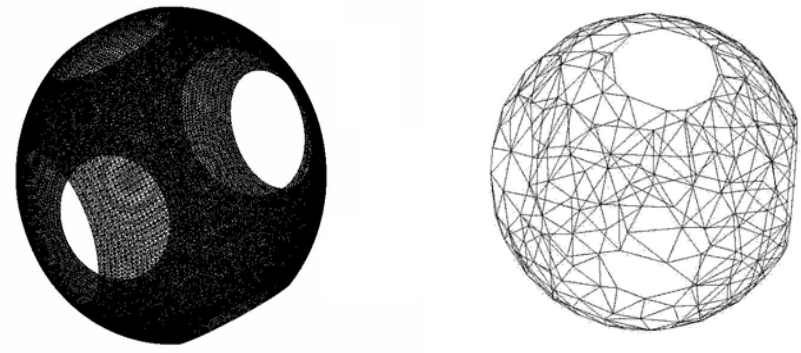

*Рис. 2*

Параллельный алгоритм построения и огрубления изоповерхностей основан на методе геометрического параллелизма. Каждый процессор формирует часть изоповерхности, проходящую через обрабатываемый процессором фрагмент сетки — домен. Результаты, полученные при огрублении фрагментов изоповерхности, передаются на один процессор. Однако с помощью такого алгоритма невозможно обработать произвольно большой объем данных. Суммарный объем данных, описывающих огрубленные на процессорах фрагменты, не может

превышать объема оперативной памяти того единственного процессорного узла, на котором выполняется окончательная обработка изоповерхности. Проблема решается разбиением процессоров на группы и введением дополнительных промежуточных этапов. После первого этапа огрубления результаты, полученные всеми процессорами одной группы, передаются на один из процессоров этой же группы. На каждом из таких процессоров выполняется второй этап огрубления, полученные результаты передаются на один процессор для окончательной обработки. При необходимости можно использовать каскадную схему, увеличив число этапов, что в принципе снимает ограничения на исходный объем обрабатываемых данных.

Каскадная схема позволяет получить дополнительный выигрыш — существенно улучшить качество аппроксимации поверхности. При частичном огрублении на каждом из процессоров присутствуют узлы двух типов: внутренние и граничные. Узел считается граничным, если множество опирающихся на него треугольников распределено по нескольким процессорам. Разрешено удаление только внутренних узлов, граничные не могут быть удалены, поскольку удаление разных узлов одной и той же границы разными процессорами может привести к возникновению разрывов триангулированной поверхности. При сильном огрублении внутренней части поверхности исключается большое число внутренних узлов. Поскольку все граничные узлы сохраняются, на огрубленной таким образом изначально гладкой поверхности возникают артефакты — изломы, описываемые большим числом точек, сглаживание которых на заключительном этапе затруднительно. Каскадная многоэтапная схема позволяет применять на каждом шаге меньшее сжатие внутренней части, что положительно сказывается на картине в целом. Результаты триангуляции передаются на персональный компьютер пользователя, где выполняется отображение поверхности.

На рис. 3, *а* приведена изоповерхность распределения плотности среды, полученная при моделировании обтекания летательного аппарата с использованием тетраэдральной сетки, содержащей более 2 млн узлов и 14 млн тетраэдров (визуализация разработанной системой RemoteViewer).

На рис. 3, *б* приведен результат визуализации изоповерхности плотности среды, полученный с помощью системы Tecplot, широко используемой для визуализации научных данных на персональных компьютерах. Сравнение рис. 3, *а* и *б* показывает хорошее совпадение формы изоповерхностей, полученных с помощью разных инструментов, что подтверждает высокое качество образов, формируемых описанными алгоритмами визуализации.

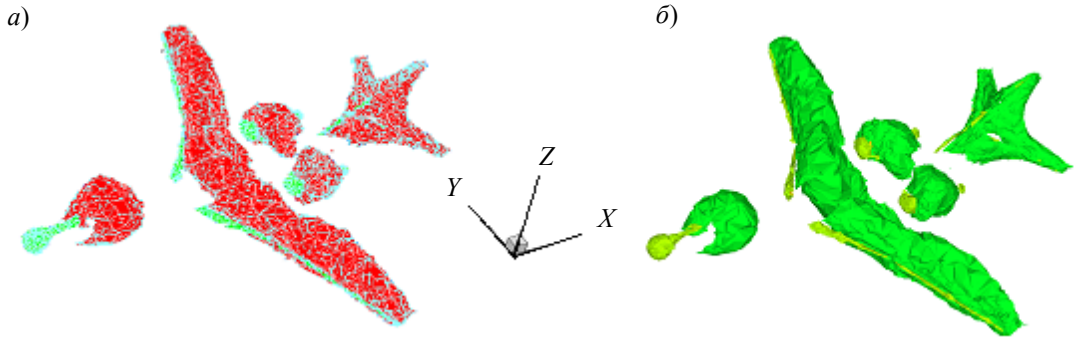


*Рис. 3*

**Заключение.** В заключение следует подчеркнуть, что рассмотренные в статье методы обработки результатов проводимых на суперкомпьютерах вычислительных экспериментов актуальны, если другие методы оказываются неприменимыми. Они актуальны, если используются объемы данных, для обработки которых недостаточно ресурсов ПК и недостаточно возможностей хорошо развитых и обладающих богатой функциональностью последовательных пакетов прикладных программ.

Работа выполнена при поддержке проекта РФФИ № 08-07-00458-а, 09-01-00292-а.

СПИСОК ЛИТЕРАТУРЫ

1. *Hendrickson B. and Leland R. A.* Multi-Level Algorithm for Partitioning Graphs // Tech. Rep. SAND93-1301, Sandia National Laboratories, Albuquerce. October 1993.

2. *Hendrickson B. and Leland R.* An Improved Spectral Graph Partitioning Algorithm for Mapping Parallel Computations // SIAM J. Sci. Comput. 1995. Vol. 16, N 2.

3. *Karypis G., Kumar V.* Multilevel Graph Partitioning Schemes // ICPP (3). 1995. P.113-122.

4. *Fiduccia C. and Mattheyses R.* A linear time heuristic for improving network partitions // Proc. 19th IEEE Design Automation Conf. 1982. P. 175—181.

5. *Fiedler M.* Eigenvectors of aciyclic matrices // Czechoslovak Mathematical J. 1975. Vol. 25(100). P. 607—618.

6. *Fiedler M.* A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory // Czechoslovak Mathematical J. 1975. Vol. 25(100). P. 619—633.

7. *Якобовский М. В.* Инкрементный алгоритм декомпозиции графов // Вестн. Нижегородского Университета им. Н. И. Лобачевского. Сер. Математическое моделирование и оптимальное управление. Вып. 1(28). Н. Новгород: Изд-во ННГУ, 2005. С. 243—250.

8. *Iakobovski M. V., Karasev D. E., Krinov P. S., Polyakov S. V.* Visualisation of grand challenge data on distributed systems // Proc. Symp. Mathematical Models of Non-Linear Excitations, Transfer, Dynamics, and Control in Condensed Systems and Other Media. Moscow—London: Plenum Publishers, 2001. P. 71—78.

9. *Якобовский М. В.* Обработка сеточных данных на распределенных вычислительных системах // Вопр. атомной науки и техники. Сер. Математическое моделирование физических процессов. 2004. Вып. 2. С. 40—53.

10. *Iakobovski M., Nesterov I., Krinov P.* Large distributed datasets visualization software, progress and opportunities // Computer Graphics & Geometry. 2007. Vol. 9, N 2, P. 1—19.

11. *Krinov P.S., Iakobovski M.V., Muravyov S.V.* Large Data Volume Visualization on Distributed Multiprocessor Systems // Parallel Computational Fluid Dynamics: Advanced numerical methods software and applications. Proc. of the Parallel CFD 2003 Conf. Moscow, Russia. Amsterdam: Elsevier, 2004. P. 433—438.

12. *Кринов П.С., Якобовский М.В., Муравьёв С.В.* Визуализация данных большого объема в распределенных многопроцессорных системах // Высокопроизводительные параллельные вычисления на кластерных системах. Мат. 3-го Междунар. науч.-практич. семинара. Н. Новгород: Изд-во ННГУ, 2004. С. 81—88.

***Сведения об авторе***

***Михаил Владимирович Якобовский*** — д-р физ.-мат. наук, профессор; Институт математического моделирования РАН, сектор программного обеспечения вычислительных систем и сетей, Москва; зав. сектором; E-mail: lira@imamod.ru

Рекомендована институтом

Поступила в редакцию
10.03.09 г.

УДК 681.324

Ю. И. НЕЧАЕВ

# НЕЛИНЕЙНЫЕ ЭФФЕКТЫ В СИСТЕМАХ УПРАВЛЕНИЯ СЛОЖНЫМИ ДИНАМИЧЕСКИМИ ОБЪЕКТАМИ

Исследуется проблема управления нелинейными динамическими объектами. Особое внимание уделяется возникновению нелинейных эффектов в условиях непрерывного изменения динамики объекта и характеристик внешней среды. Приведены результаты моделирования поведения нелинейного динамического объекта в нештатных и экстремальных ситуациях.

***Ключевые слова:*** *интеллектуальная система, динамический объект, внешние воздействия, экстремальная ситуация.*

Динамика нелинейных систем — одна из быстро развивающихся областей практических приложений методов и моделей в задачах управления и принятия решений. Математические модели таких систем обладают специфическими свойствами — единой структурой и общим признаком нелинейности. Системы со слабой нелинейностью достаточно хорошо исследованы, для них существуют общие методы анализа и разработаны эффективные алгоритмы практического использования. Методы решения задач с более сильными проявлениями нелинейности основаны на сложных математических построениях и использовании современных высокопроизводительных вычислительных средств [1—7]. Анализ нелинейных эффектов позволяет выявить особенности задач, в которых заложены возможности нетривиальных решений, и построить общую теорию, описывающую поведение нелинейных систем. Возникновение колебательных режимов большой амплитуды активизирует новые виды физических связей, „дремавших“ в случае слабых возмущений. Не случайно эта проблема занимает умы многих исследователей в различных сферах научной деятельности, в том числе и при решении проблем динамики корабля [3, 5].

Целью настоящей работы является рассмотрение некоторых аспектов нелинейной динамики на примере использования моделей, обладающих способностью к самоорганизации и допускающих режимы детерминированного хаотического поведения фазовых траекторий. Рассматриваемые нелинейные модели описывают сложную динамику, включая равновесные режимы (предельные циклы), а также широкий спектр режимов детерминированного хаоса. При практическом использовании таких моделей для обеспечения функционирования бортовых интеллектуальных систем (ИС) важное значение имеют надежная оценка ситуации и прогнозирование ее развития в условиях непрерывного изменения динамики объекта и внешней среды. Особый практический интерес представляет построение нелинейных математических моделей, способных изменять свою структуру при изменении поведения динамического объекта (ДО) на волнении. При синтезе реализующих их алгоритмов используют различные подходы — детерминистский, стохастический и подход на основе принципов самоорганизации [3—5]. Первые два подхода предполагают наличие в исходных данных полного информационного базиса, т.е. всех определяющих параметров и факторов, которые необходимо учитывать при анализе ситуации.

Принцип нелинейной самоорганизации [3] требует минимального объема априорной информации. Его методологической основой является допущение о том, что вся информация о структуре модели содержится в данных измерений и критериальных соотношениях, учитывающих особенности гидроаэродинамического взаимодействия ДО с внешней средой в рассматриваемой ситуации. Реализация принципа нелинейной самоорганизации при разработке

базы знаний ИС требует большого объема вычислительных операций, связанных с предварительной оценкой поведения ДО на основе математического моделирования экстремальных ситуаций с последующей формулировкой соответствующих критериальных оценок [3, 5].

**Принципы управления в системах со сложной динамикой.** Процедуры принятия решений по управлению системами со сложной динамикой в бортовых ИС разрабатываются на основе принципов обработки информации в мультипроцессорной вычислительной среде [3] (рис. 1) в рамках концепции мягких вычислений [7].

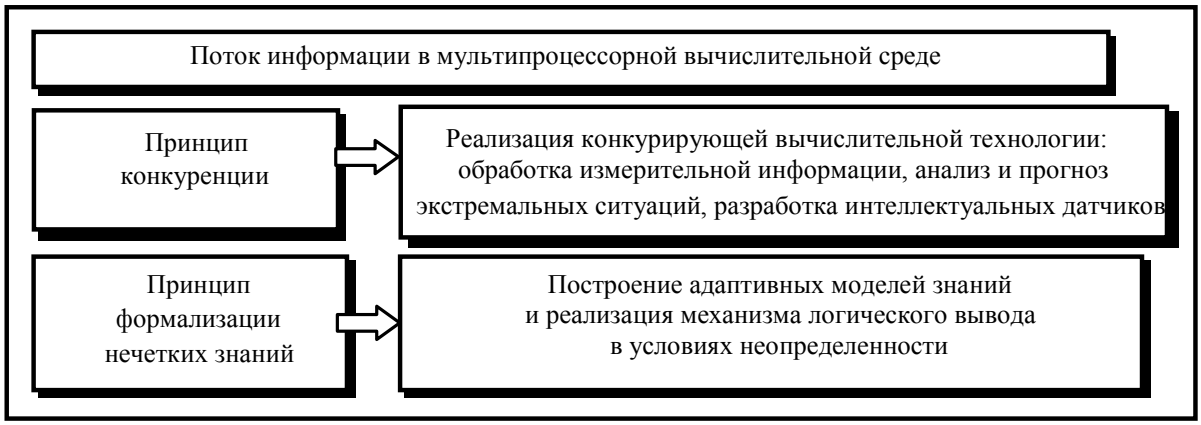


*Рис. 1*

Суть системного управления с использованием этого подхода состоит в способности нелинейной системы перестраивать свою структуру в зависимости от состояния ДО и внешней среды (ВС). В этом случае благодаря бифуркации система может находиться в нескольких устойчивых состояниях. Явление бифуркации характерно для нелинейного поведения системы — появление возможности выбора из нескольких состояний означает, что математическая модель, описывающая эволюцию системы, имеет несколько стационарных состояний. В случае возникновения нестандартных ситуаций в ИС используются конкурирующие вычислительные технологии, основанные на нечеткой логике и искусственных нейронных сетях [3].

Моделирование предполагает анализ поведения существенно нелинейного ДО, обладающего способностями к самоорганизации. В этих условиях контроль поведения ДО осуществляется через распознавание, анализ, прогнозирование и управление процессами самоорганизации [1, 3—5] в рамках системного управления, предполагающего рассмотрение динамической системы как целостной совокупности ее элементов [3]. Главным достижением этого подхода стало понимание того, почему многим сложным системам может быть свойственно достаточно простое поведение. Именно эта простота и позволяет разобраться в сложной эволюции динамических систем — дает возможность строить сравнительно простые модели для сложных явлений, преодолевая тем самым барьер сложности. Таким образом, путь от сложности к простоте лежит через самоорганизацию, а математическая модель, описывающая эволюцию системы, имеет несколько стационарных состояний [1, 5].

При построении системы контроля поведения ДО в штормовых условиях в зависимости от особенностей взаимодействия системы „ВС—ДО“, определяемых выделенными режимами движения, используются различные архитектурные решения. Одно из них реализовано на базе модели с коррекцией правил в виде двухуровневой системы (рис. 2) [3]. Система включает блок анализа ситуации на основе методов классической математики и блок адаптации, содержащий матрицу логических правил и блок нейросетевых моделей. Графическое окно отображает динамику взаимодействия в рассматриваемый момент времени. Принцип работы такой системы заключается в том, что на основании анализа текущего значения вектора ошибки

блок адаптации формирует управляющие воздействия, изменяющие правые части матрицы лингвистических правил (матрицы управления).

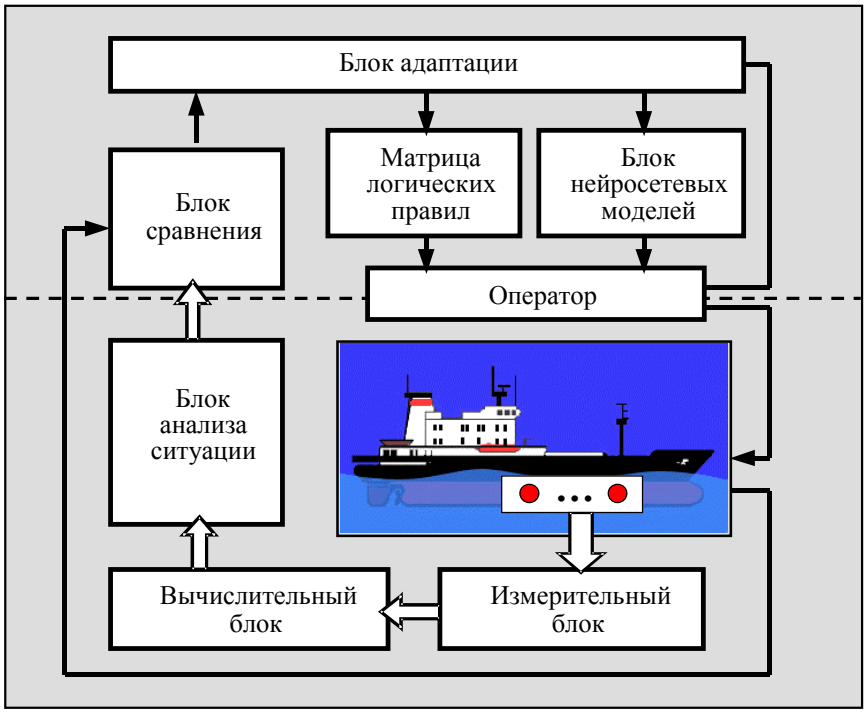


*Рис. 2*

Для обеспечения адаптивности системы в условиях многорежимности используются набор матриц управления и ансамбль нейросетевых моделей. Более сложная структура многорежимной системы принятия решений по управлению ДО осуществляется с использованием методов идентификации, нечеткой адаптивной модели и нейросетевых ансамблей.

**Нелинейные эффекты и потеря устойчивости колебательного движения.** Рассмотрим ДО как систему с шестью степенями свободы. Поведение ДО можно описать системой нелинейных дифференциальных уравнений [3]:

$$F_i\left(\ddot{x}_i, \dot{x}_i, x_i, t, X_{i1},\ldots,X_{im}, Y_{i1},\ldots,Y_{in}\right) = 0\,, \qquad (1)$$

где $F_i(\bullet)$ — нелинейные функции; $x_i$ — линейные и угловые перемещения; $X_{i1},...,X_{im}$ — параметры, характеризующие судно как динамическую систему (инерционные, демпфирующие и восстанавливающие компоненты); $Y_{i1},...,Y_{in}$ — возмущающие силы и моменты; $i = 1, 2, ..., 6$.

Наиболее сложной функцией в системе (1) является восстанавливающий компонент, входящий в дифференциальное уравнение бортовой качки, который отличается существенной нелинейностью, сложностью и многозначностью. Непрерывно изменяясь во времени и пространстве, эта функция в значительной степени определяет особенности взаимодействия ДО с внешней средой (1). Математическое описание нелинейной пространственной функции восстанавливающего момента на волнении представляется формулой [3]

$$M_w = M(\theta,\varphi,t) = D\,[l(\theta,\varphi) + \Delta l(\theta,\varphi)\cos(\sigma_k t - \varepsilon)];$$

$$l(\theta,\varphi) = 0,5\left[\, l(\theta,\varphi)_{\max} + l(\theta,\varphi)_{\min}\,\right], \quad \Delta l(\theta,\varphi) = 0,5\,\left[\Delta l(\theta,\varphi)_{\max} + \Delta l(\theta,\varphi)_{\min}\,\right];$$

$$M_w = \Phi(\theta,\varphi_k,t) = Dl(\theta,\varphi,t)\,, \qquad (2)$$

где $\Delta l(\theta,\varphi)_{\max}$ и $\Delta l(\theta,\varphi)_{\min}$ — экстремальные значения приращений плеч остойчивости, соответствующие положению ДО на подошве и вершине волны при различных курсовых углах $\varphi$; $l(\theta,\varphi,t)$ — плечо восстанавливающего момента, определяемого для различных значений времени, $\varepsilon$ — фаза волны: $\varepsilon = 0$ и $2\pi$ — ДО на подошве волны; $\varepsilon = \pi/2$ — на переднем склоне; $\varepsilon = \pi$ — на вершине волны; $\varepsilon = 3/2\pi$ — на заднем склоне.

Общее выражение для программной реализации функции $\Delta l_w(h_w/\lambda,\theta,\varphi)$ при различных параметрах имеет вид [3]:

$$\Delta l_w(h_w/\lambda,\theta,\varphi) = B\left[\Phi\left(\frac{h_w}{\lambda},\theta,\varphi_k\right) + \sum_{m=1}^{6} A_m f_m(\theta,\varphi_k) + \sum_{n=1}^{8} B_n F_n(\theta,\varphi_k) + \sum_{p=1}^{3} C_p E_p(\theta,\varphi_k)\right], \quad (3)$$

$$\Phi(h_w/\lambda,\theta,\varphi_k),\ \sum_{m=1}^{6} A_m f_m(\theta,\varphi_k),\ \sum_{n=1}^{8} B_n F_n(\theta,\varphi_k),\ \sum_{p=1}^{3} C_p E_p(\theta,\varphi_k).$$

Пример функции $M(\theta,\varphi,t)$ в виде сложной пространственной поверхности приведен на рис. 3 (сплошные кривые — мгновенные диаграммы; пунктир — временные кривые (сечения поверхности при θ = const); штрихпунктир — диаграмма остойчивости на тихой воде).

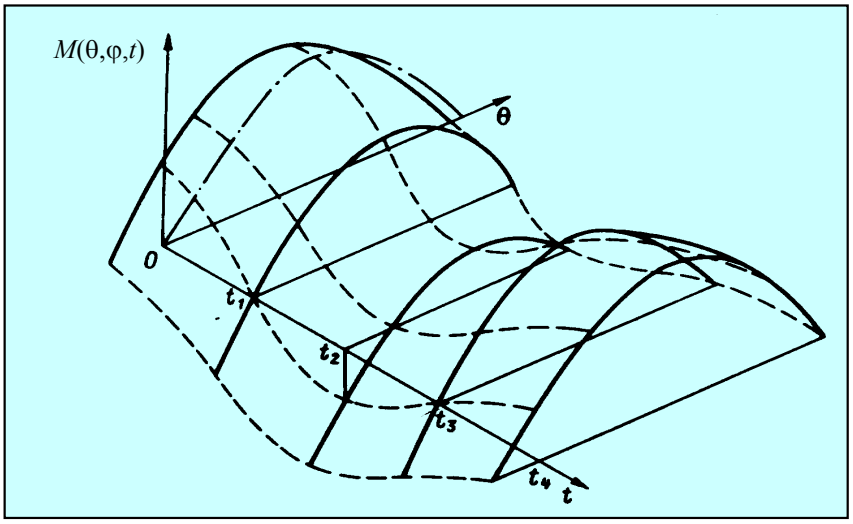


*Рис. 3*

В нелинейных детерминированных системах могут возникать хаотические движения, характеризующиеся тем, что первоначально близкие траектории в ограниченной области фазового пространства могут быстро расходиться. Это свойство нелинейных систем получило название детерминированного хаоса [1, 4]. Нелинейная система может порождать различные виды колебаний: периодические и квазипериодические, субгармонические и хаотические.

В процессе развития динамики нелинейной системы формируются простые (обычные) и странные (стохастические) аттракторы. Простые аттракторы часто встречаются при анализе нелинейных диссипативных систем. Геометрическая интерпретация простого аттрактора диссипативной системы на фазовой плоскости представляет собой либо неподвижную точку (фокус), к которой устремляются фазовые траектории, либо предельный цикл, обладающий таким свойством, что все близкие фазовые траектории представляют собой спирали, неограниченно приближающиеся к замкнутой кривой в фазовом пространстве. Странные аттракторы отличает более сложная структура. Кроме неустойчивых (разбегающихся) траекторий они содержат и устойчивые (притягивающиеся). По существу это седловые

траектории, устойчивые в одних направлениях и неустойчивые в других, и образующие множество сложным образом соединяющихся слоев, не касающихся друг друга [5].

В работе [3] исследованы динамические картины развития параметрических колебаний ДО при прохождении группы волн. Показано, что в этом случае формируются сложные структуры колебательных режимов бортовой качки. Их особенность состоит в том, что аттракторные множества имеют вид неустойчивых предельных циклов. Потеря устойчивости цикла-аттрактора в рассматриваемой однопараметрической системе происходит по различным сценариям. При воздействии пакета волн формируется предельный цикл, характеризующийся стабилизацией амплитуды колебаний вследствие влияния нелинейности. Этот цикл возникает на участке, где последовательность волн в пакете превышает определенное (критическое) значение высоты волны, обеспечивающее колебательный режим с практически постоянной амплитудой. Однако в связи с последующим постепенным уменьшением высоты волны в пакете нарушаются условия устойчивости и цикл исчезает.

Более сложный сценарий — столкновение с неустойчивым циклом. Такая ситуация на практике встречается значительно реже и характеризуется последовательным прохождением пакетов волн, содержащих волны различной интенсивности. Например, первый пакет с небольшой высотой резонансных волн приводит к формированию предельного цикла малой, а второго — большой амплитуды. Возникновение и потеря устойчивости колебательного режима („рождение и смерть цикла“ по терминологии А.А. Андронова [2]) происходят вследствие ограниченности зоны резонансной качки на сравнительно небольшом временном интервале интенсивных колебаний при прохождении волновых пакетов.

Развитие параметрических колебаний при формировании предельного цикла проявляется за счет большей глубины модуляции параметра, стоящего в качестве множителя при периодической функции уравнения Матье [6], в результате чего формируются начальные условия, обеспечивающие преодоление „порога возбуждения“ параметрического резонанса.

Практический интерес к детальному изучению параметрических колебаний вызван серьезной аварией американского контейнеровоза [3], который по своим размерам близок к авианесущим кораблям. Выявленное в результате анализа причин этой аварии явление параметрического резонанса при воздействии пакетов экстремальных волн (о возможности возникновения которого авторы проекта судна даже не предполагали) уже несколько лет является предметом дискуссий на международных конференциях, посвященных безопасности мореплавания.

Математическая модель, характеризующая динамику судна при воздействии пакетов волн в режиме параметрического резонанса, описывается системой дифференциальных уравнений бортовой, вертикальной и килевой качки [3]:

$$(Jx+\mu_{\theta\theta})\theta''+M_R(\theta')+M(\theta,\varphi_k,t)=M_x(t);$$

$$(D/g+\mu_{33})\zeta_{G''}+v_\zeta\zeta_{G'}+\rho gS\zeta_G+\mu_{33}x_1\psi''+(v_{\zeta\psi}-v_0\mu_{33})\psi'+(\rho gSl-v_0v_\zeta)\psi=$$
$$=-r_0(\rho ga_0-\sigma^2a_0''-\sigma b_0')\cos\sigma t-r_0(\rho gb_0-\sigma^2b_0''+\sigma a_0')\sin\sigma t;$$
$$(Jy+\mu_{55})\psi''+[v_\zeta+(v_0{}^2/\sigma^2)v_\zeta]\psi'+(DH\psi-v_0{}^2\mu_{33})\psi+\mu_{33}x_1\zeta_{G''}+$$
$$+(v_{\zeta\psi}+v_0\mu_{33})\zeta_{G'}+(\rho gSl+v_0v_\zeta\zeta_G)=-r_0(\rho ga_1-\sigma^2a_1''-\sigma b_1')\cos\sigma t-$$
$$-r_0(\rho gb_1-\sigma^2b_1''+\sigma a_1')\sin\sigma t, \qquad (4)$$

где $(Jx+\mu_{\theta\theta})\theta''$, $M_R(\theta')$, $M(\theta,\varphi_k,t)$, $M_x(t)$ — инерционный, демпфирующий, восстанавливающий и возмущающий компоненты; остальные обозначения соответствуют принятым в работе [8].

Для иллюстрации на рис. 4 приведены временные кривые и фазовый портрет колебательного движения контейнеровоза на встречном нерегулярном волнении в режиме параметрического резонанса при воздействии крупного пакета волн, структура которого близка к структуре волн зыби.

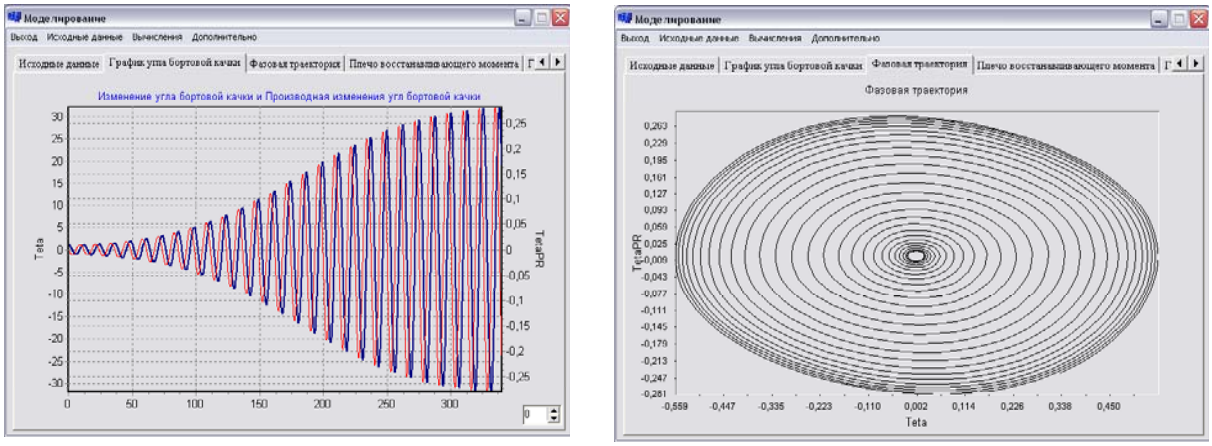


*Рис. 4*

Как видно из рис. 4, амплитуда бортовых параметрических колебаний в этой экстремальной ситуации быстро нарастает и стабилизируется вследствие влияния нелинейных эффектов, достигая около 30°. Интересно отметить, что килевая качка в этих условиях происходит в режиме, близком к основному резонансу. Результаты моделирования колебательного движения ДО в условиях резонанса килевой качки приведены на рис. 5.

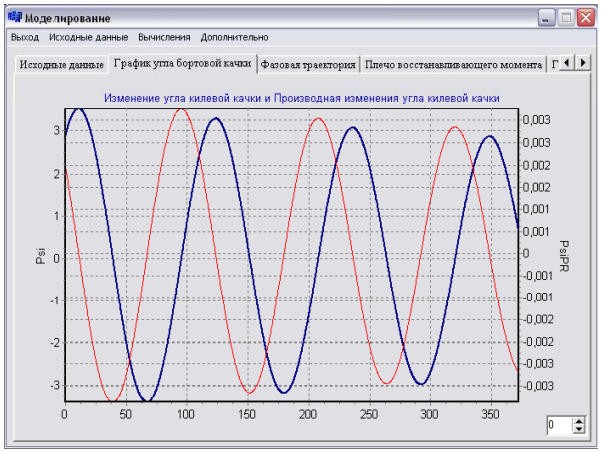


*Рис. 5*

В процессе эксперимента сопоставлялась топология фазового пространства для нелинейной и линеаризованной математических моделей. Полученные данные свидетельствуют о практической невозможности сохранения фазовых потоков, порождающих странные аттракторы и детерминированный хаос, в линеаризованных системах. Закономерности поведения фазовых траекторий в условиях неустойчивости положены в основу разработки динамической базы знаний, позволяющей осуществлять прогноз и интерпретацию экстремальных ситуаций в бортовых ИС реального времени.

**Управление в самоорганизующихся системах.** Охарактеризовать нелинейную динамику объекта можно, построив диаграмму переходов состояний. Такая диаграмма позволяет интерпретировать сложные ситуации, возникающие в различных условиях эксплуатации. Формально диаграмма переходов состояний описывается как структура вида [3]:

$$W = < S, R, A, B, L >, \qquad (5)$$

где $S$ — множество событий (ситуаций); $R$ — множество ребер вида $r_{ij} = (s_i, s_j)$, $i \neq j$, $A$ — множество присоединенных атрибутивных вершин; $B$ — множество присоединенных алгоритмических вершин; $L$ — множество ребер вида $l = (a_i, s_i)$ или $l_{ij} = (b_i, s_j)$, или $l_{ij} = (a_i, a_j)$; $i \neq j$, $a_j \in A$, $b_j \in B$.

Результаты вычислительных экспериментов позволяют построить и проанализировать траектории системы на фазовой плоскости [6] в виде аттракторов [3], описывающих динамику одностабильной, а также бистабильной (рис. 6, *а*) и трехстабильной (рис. 6, *б*) систем, а также характерные временные кривые и одномерные отображения для аттракторных множеств.

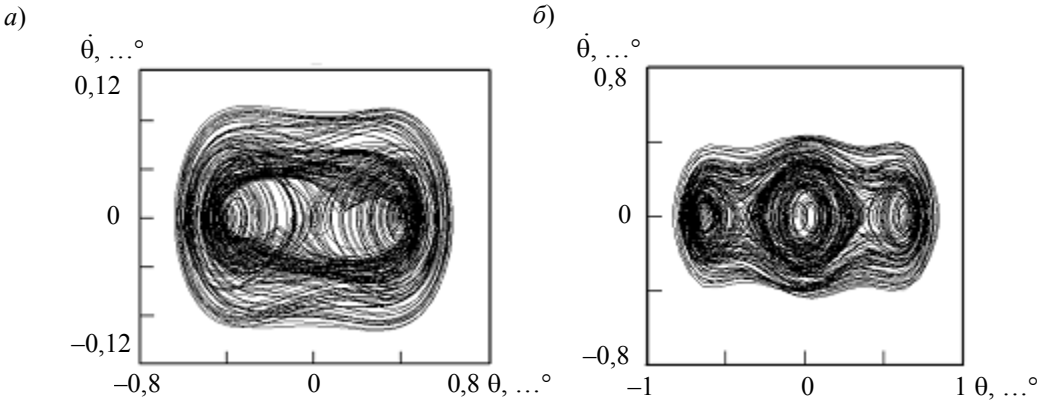


*Рис. 6*

Другая закономерность, иллюстрируемая компьютерной программой, заключается в моделировании пространственно-временного поведения ДО с использованием центральной теоремы синергетики, определяющей динамику сложной системы как взаимосвязь растущих и затухающих ее конфигураций [4, 5].

**Проблемы управления в нестандартных ситуациях.** Одно из интересных приложений нелинейной теории управления связано с задачей о поведении ДО в условиях „захвата“ и разворота на нерегулярном волнении, названной „брочинг“ (от *англ.* broaching — развертка) [3]. *Брочинг* — одна из наиболее сложных и опасных экстремальных ситуаций. Эта ситуация связана с возникновением явления „захвата“ ДО попутной волной, потерей устойчивости движения, ухудшением управляемости и внезапным разворотом ДО на волнении.

Математическая модель поведения ДО в режиме „брочинг“ описывается системой дифференциальных уравнений [3, 5]:

$$\left.\begin{aligned}
&\left[\left(\frac{D}{g}\right)+\mu_{\xi\xi}\right]\left(\dot{\nu}\cos\beta^* - \beta^*\nu\sin\beta^*\right)+\left[\left(\frac{D}{g}\right)+\mu_{\eta\eta}\right]\nu\dot{\chi}\sin\beta^* = F_x = X(t)+P_e-R,\\
&\left[\left(\frac{D}{g}\right)+\mu_{\eta\eta}\right]\left(\dot{\nu}\sin\beta^* - \beta^*\nu\cos\beta^*\right)+\left[\left(\frac{D}{g}\right)+\mu_{\xi\xi}\right]\nu\dot{\chi}\cos\beta^* = F_y = Y(t)+R_{yB}+R_{yR},\\
&\left(J_z+\mu_{\chi\chi}\right)\ddot{\chi} = M_z = M_z(t)+M_{zB}+M_{zP},\\
&\left(J_x+\mu_{\theta\theta}\right)\ddot{\theta}+M_R\left(\dot{\theta}\right)+M(\theta,\phi,t) = M_G+M_A,
\end{aligned}\right\} \qquad (6)$$

где $\nu$ — частота колебаний, $\mu$ — присоединенная масса жидкости, $\beta^*$ — угол дрейфа ДО; $\chi$ — угол рыскания; $X(t)$, $Y(t)$, $M_X(t)$, $M_z(t)$ — возмущающие силы и момент; $M(\theta,\varphi,t)$ — восстанавливающий момент; $R_{yB}$ и $M_{zB}$ — поперечные сила и момент; $M_{zP}$ — момент вихревой природы ДО; $M_G$ и $M_A$ — гидродинамический и ветровой кренящие моменты.

Проведенное исследование явления „брочинг“ позволило сформулировать критериальный базис, упростить интерпретацию экстремальной ситуации и разработать алгоритм управления при принятии решений в процессе функционирования бортовой ИС контроля и прогноза мореходных качеств ДО в условиях эксплуатации.

Особенность задачи состоит в том, что эффективное управление в режиме „брочинг“ возможно только с целью предотвращения возникновения этой опасной ситуации. Однако в случае ее возникновения попытки использовать управление для выхода ДО из этого режима могут приводить только к ухудшению положения, поскольку использование рулевого комплекса неэффективно в условиях полной потери управляемости [3].

**Особенности управления в хаотических системах.** Задачи и методы управления хаосом — область интенсивных исследований последних десятилетий. В рамках концепции детерминированного хаоса система демонстрирует хорошую управляемость и пластичность: чутко реагирует на внешние воздействия, сохраняя при этом тип движения, что характерно для многих динамических систем [1, 4, 5].

С позиций синергетического подхода применение моделей хаотических систем открывает возможность реализации процессов самоорганизации. Одним из наиболее типичных сценариев является переход к хаотическому режиму через последовательность бифуркаций удвоения периода, который наблюдается для систем с вязким трением под действием возмущающих сил и моментов. Проведенное исследование предхаотических и послехаотических изменений динамической системы при вариации ее параметров проводилось на основе нелинейного уравнения бортовой качки с помощью бифуркационных диаграмм [4, 5]. Выборка данных осуществлялась с помощью отображения Пуанкаре, позволяющего выделять удвоение периода и субгармонические бифуркации, которые отчетливо просматриваются при интегрировании нелинейного уравнения Матье [6]. Другим сценарием самоорганизации является возникновение хаотических колебаний, связанное с переходом к хаотическому движению через перемежаемость. При таком движении всплески хаотических колебаний чередуются (перемежаются) с участками, на которых происходят почти периодические движения [4, 5].

Развитие учения о хаотической динамике нелинейных систем выявило целый ряд реальных практических задач, в том числе и в гидродинамике [5], где хаотические режимы действительно могут возникать при сложном взаимодействии ДО с внешней средой. Более того, возникли практически важные классы задач, в которых нелинейной системой необходимо управлять, изменяя степень ее хаотичности [5, 6].

**Заключение.** Рассмотренные задачи нелинейной динамики связаны с применением многопроцессорных вычислительных средств анализа и интерпретации информации при функционировании бортовых ИС реального времени. Вычислительные технологии реализации этих задач отражают только незначительную область научно-технических приложений, в которых находят применение методы анализа существенно нелинейных динамических систем, синергетическая парадигма и теория детерминированного хаоса. При этом в представленной работе сделан акцент именно на те приложения, которые в настоящее время вызывают большой научный и практический интерес при исследовании сложного поведения ДО в различных экстремальных ситуациях.

## СПИСОК ЛИТЕРАТУРЫ

1. *Андриевский Б. Р., Фрадков А. Л.* Управление хаосом: методы и приложения // Автоматика и телемеханика. 2004. № 4. С. 3—34.

2. *Андронов А. А., Витт С., Хайкин С. Э.* Теория колебаний. М.: Наука, 1981.

3. Бортовые интеллектуальные системы. Ч.2. Корабельные системы. М.: Радиотехника, 2006.

4. *Лоскутов А. Ю., Михайлов А. С.* Введение в синергетику. М.: Наука, 1990.

5. *Нечаев Ю. И.* Нелинейная динамика и парадигмы вычислений при анализе экстремальных ситуаций // Мат. Междунар. науч. конф. „Леонард Эйлер и современная наука“. СПб, 2007. С. 385—390.

6. *Хаяси Т.* Нелинейные колебания в физических системах. М.: Мир, 1968.

7. *Zadeh L.* Fuzzy logic, neural networks and soft computing // Commutation on the ASM-1994. Vol. 37, N 3. P. 77—84.

8. Интеллектуальные системы в морских исследованиях / Под ред. *Ю. И. Нечаева.* СПб: Изд-во СПбГМТУ, 2002. 320 с.

***Сведения об авторе***

***Юрий Иванович Нечаев*** — д-р техн. наук, профессор; Санкт-Петербургский морской технический университет, кафедра вычислительной техники; E-mail: petr_oleg@mail.ru

Рекомендована институтом

Поступила в редакцию
10.03.09 г.

УДК 681.3

А. А. Шалыто, Е. А. Мандриков, Ю. К. Чеботарева

# АВТОМАТНОЕ ПРОГРАММИРОВАНИЕ И ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ

Представлены основные положения автоматного программирования, обоснована предпочтительность его использования при разработке программного обеспечения. Описываются возможности применения параллельных вычислительных технологий для генерации автоматов.

***Ключевые слова:*** *автоматное программирование, система управления, генетический алгоритм, фитнес-функция.*

Большинство программистов-практиков считают, что в программировании как таковом нет особых проблем. Видимое отсутствие проблем приводит к тому, что на практике при создании программного обеспечения (ПО) в большинстве случаев используются частные (от *лат.* ad hoc — экспромт, или спонтанное решение) подходы, основанные на персональном опыте программиста. Если трудности при создании программ и возникают, то их смиренно считают „неизбежным злом профессии“ [1]. Тот факт, что при таком подходе достаточно много проектов заканчиваются неудачей, не изменяет точки зрения большинства.

Принципиально другое мнение у теоретиков программирования, которые еще в 1968 г. „открыто признали кризис программного обеспечения“ [1]. В настоящее время возможным вариантом выхода из кризиса ряд теоретиков считают переход от „искусства программирования“ [2] к программной инженерии (Software Engineering) [3, 4]. Однако современные специалисты по программной инженерии почти не используют подходы, разработанные в других инженерных областях.

Альтернативой инженерии является использование междисциплинарных исследований и подходов (Interdisciplinary Software Engineering Network — ISEN). В результате исследований в этом направлении сформировалось мнение, что при разработке ПО, видимо, может быть полезен опыт создания систем автоматического управления, что, в итоге, породило термин „программная кибернетика“ (Software Cybernetics) [5].

Ниже излагаются основы автоматного программирования — междисциплинарного направления, относящегося как к программной инженерии, так и к программной кибернетике, которое базируется на положениях теории автоматов и теории автоматического управления. При этом особо подчеркнем, что под автоматным программированием авторы подразумевают

не программирование с применением автоматов, а соответствующую технологию программирования, направленную на создание систем со сложным поведением [6]. В этом смысле уместно провести параллель между автоматами и автоматным программированием, с одной стороны, и UML (Unified Modeling Language) и RUP (Rational Unified Process) — с другой. Так, автоматы и UML — это нотации, в то время как автоматное программирование и RUP — это процессы, использующие указанные нотации.

**Парадигма автоматного программирования.** Упрощенная трактовка автоматного программирования состоит в том, что это стиль программирования, при его использовании поведение программ предлагается описывать автоматами, которые в дальнейшем преобразуются в код [7]. В частности, в работе [8] такой подход был назван „программирование от состояний“. Однако „программирование с автоматами“ нельзя рассматривать как парадигму программирования, так как при этом остается не ясно, как с использованием автоматов проектировать и реализовывать программы в целом.

Вместе с тем автоматное программирование можно рассматривать как *новую парадигму* программирования. Очень многие системы являются автоматизированными объектами управления, представляющими собой совокупность системы управления (СУ) и объекта управления (ОУ), охваченных обратными связями (рис. 1, *а*).

Задача построения автоматизированных объектов управления рассматривается в любом курсе теории автоматического управления применительно к объектам различных типов. Удивительно, что это почти не коснулось практики программирования, несмотря на то что в теории алгоритмов в качестве одной из основных моделей используется машина Тьюринга (рис. 1, *б*).

Машина Тьюринга по сути является автоматизированным объектом управления, в котором система управления — конечный автомат (КА), а объект управления — лента (ее ячейки памяти) (рис. 1, *в*).

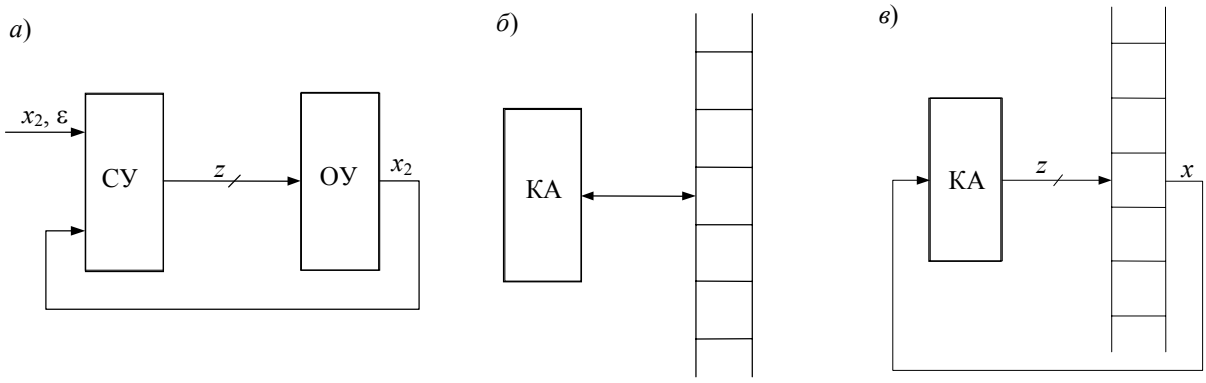


*Рис. 1*

Сложность программирования на машине Тьюринга определяется тем, что в ней используются очень простые объекты управления (ячейки памяти), которые могут выполнять только простейшие действия (операции) по сдвигу головки, записи и стиранию отдельных символов. В этой ситуации „вычисления“ приходится выполнять конечному автомату, который для этой цели не приспособлен, так как его предназначение — управление. Другая особенность машины Тьюринга, которая может резко усложнять программы, — это использование только одного автомата, чего достаточно для проведения теоретических исследований, но бывает недостаточно при практическом применении.

Переход от тьюрингова программирования к практическому (автоматному) осуществляется за счет усложнения объектов управления, которые могут выполнять сколь угодно сложные действия (операции), и применения в качестве системы управления системы взаимодействующих автоматов. Из изложенного следует, что универсальность предлагаемого подхода определяется тем, что он основан на расширении машины Тьюринга, которая позволяет реализовать произвольные алгоритмы.

При этом теоретические положения о том, что автоматы позволяют распознавать регулярные языки, а магазинные автоматы — языки с контекстно-свободными грамматиками, отходят на второй план, так как в рассматриваемом подходе используются не автоматы, а автоматизированные объекты управления, в которых число объектов управления и их сложность не фиксированы, как и число автоматов.

Исходя из этого в автоматном программировании предлагается создавать программы по тем же принципам, как производится автоматизация технологических (и не только) процессов. При этом на основе анализа предметной области выделяются источники входных воздействий и автоматизированные объекты управления, каждый из которых содержит систему управления (систему взаимодействующих конечных автоматов) и объекты управления. Эти объекты реализуют выходные воздействия и формируют значения дополнительной разновидности входных воздействий, которые по обратным связям передаются системе управления.

Все составляющие каждого автоматизированного объекта управления отображаются на схеме связей, которая может совмещаться со схемой взаимодействия автоматов. На схеме связей для каждого входного и выходного воздействия указывается его полное название и краткое символьное обозначение, которое в дальнейшем и используется в качестве пометки в графах переходов автоматов и идентификатора соответствующей переменной в программе. Использование кратких символьных обозначений позволяет даже весьма сложные алгоритмы отражать так компактно, что часто граф переходов удается размещать на одном экране монитора, позволяя разработчику „охватить“ весь граф одним взглядом, что облегчает понимание таких графов.

Использование символьных, а не лингвистических идентификаторов, являющихся обычно сокращенными английскими словами, смысл которых обычно забывается через некоторое время, не ухудшает восприятие автоматных программ, так как в рамках рассматриваемого подхода их построение и внесение изменений в них должно производиться формально (вручную или автоматически) и только по графам переходов. При этом смысл переменных можно понять по схеме связей.

Объекты управления могут быть реальными или виртуальными (реализованными программно). В первом случае их логика изменена быть не может, а во втором — она реализуется в автоматах.

Таким образом, парадигма автоматного программирования состоит в представлении и реализации программ как систем автоматизированных объектов управления [9].

**Основные положения автоматного программирования.** Основным понятием в автоматном программировании является состояние. Предлагается разделять состояния на два класса: управляющие и вычислительные. При этом с помощью небольшого числа управляющих состояний, как и в машине Тьюринга, можно управлять сколь угодно большим числом вычислительных состояний. Во введенной классификации управляющие состояния могут быть названы качественными, а вычислительные — количественными. В рамках автоматного программирования основное внимание уделяется управляющим состояниям, которые, если это не оговаривается особо, и рассматриваются в дальнейшем. При этом справедливо соотношение: „Состояния + входные воздействия = конечный автомат без выхода“. Справедливо также: „Автомат без выхода + выходные воздействия = автомат“.

Автоматы могут быть абстрактными (входные и выходные воздействия формируются последовательно) и структурными (входные и выходные воздействия формируются „параллельно“). В автоматном программировании, в отличие, например, от программирования компиляторов, обычно применяются структурные автоматы.

Понятие времени в автоматах в явном виде не используется. В случае применения элементы задержки рассматриваются как объекты управления. При этом задержки „запускаются“ и

„сбрасываются“ из автоматов, а информация об истечении времени нахождения в состоянии поступает в них в виде входных воздействий.

Автоматы могут задаваться в различном виде, однако при проектировании и использовании они должны обладать когнитивными свойствами, что достигается при задании поведения автоматов в виде графов переходов (диаграмм состояний). В случае, если автоматы генерируются автоматически, они могут задаваться иначе — например, в табличной форме. При этом отметим, что даже при сравнительно небольшом числе состояний и переходов такое задание затрудняет понимание работы автоматов человеком, так как отражение переходов в них ненаглядно.

Понятность графов переходов достигается во многом за счет того, что состояния декомпозируют множество всех входных воздействий соответствующего автомата на группы, каждая из которых определяет переходы из рассматриваемого состояния.

**Достоинства автоматного программирования.** В рамках автоматного программирования предполагается, что собственно написание (генерация) программы начинается только после ее проектирования. При этом в инженерной практике (в отличие от традиционного программирования) проект или его этап обязательно завершается выпуском проектной документации. Поэтому при автоматном программировании, основной областью использования которого являются встроенные системы (чисто инженерная область), должна выпускаться проектная документация, а не только документация пользователя, как это обычно принято в программных проектах. При этом для автоматных программ необходимым компонентом, входящим в состав проектной документации, должны быть графы переходов.

Проектирование автоматов, описывающих логику программ, которая при традиционном программировании неупорядочена и поэтому сложна, а также формальный и изоморфный переход от автоматов к реализующим их программам, приводит к тому, что программы готовы к запуску либо требуют минимальной отладки. Это также связано с тем, что реализация функций входных и выходных воздействий при излагаемом подходе, как отмечалось выше, почти не содержит логики.

При необходимости проведения отладки для автоматных программ могут генерироваться отладочные протоколы, которые отражают поведение программ в терминах автоматов (состояния, переходы, значения входных и выходных воздействий), так как в автоматном программировании автоматы являются не графическими отображениями, а частью программ, представленных в нетрадиционной для программистов визуальной, а не текстовой форме.

При этом отметим, что увеличение времени создания программ при использовании автоматного подхода компенсируется сокращением времени их отладки. Это приводит к тому, что для большинства программ уровень трудоемкости разработки на основе автоматного и традиционного подходов практически одинаков. Однако в первом случае при разработке дополнительно получаются диаграммы, понятные человеку, по которым программа была построена формально, а во втором — только программа, понимание логики которой для представителей заказчика или даже для ее автора через некоторое время часто представляет большую проблему.

Создание программ со сложным поведением без использования диаграмм приводит к трудностям на всех этапах жизненного цикла. Особенно сложно в этом случае реализовывать программы, так как все особенности их поведения приходится „держать в голове“ в течение всего времени их написания, вместо того чтобы отобразить их на диаграмме и на время забыть. Еще одним преимуществом автоматных программ является простота внесения изменений в них специалистами в предметной области, не являющимися профессиональными программистами.

Следующее преимущество автоматного подхода — эффективность верификации автоматных программ на основе метода Model Checking (верификация на модели) [9, 10]. Это объясняется тем, что модель для верификации программ этого класса может строиться автоматически по графу переходов и иметь относительно небольшой размер, так как в графах переходов используются только управляющие состояния.

Для автоматных программ как класса отсутствует семантический разрыв между требованиями к программе и к модели, так как он устраняется в ходе разработки графов переходов на этапе проектирования. Это позволяет считать автоматные программы приспособленными к верификации. Таким образом, процесс верификации программ схож с контролем схем со сложной логикой — эти схемы не удается проверить, если они не спроектированы специальным образом для обеспечения „контролепригодности“.

**Автоматическая генерация автоматных программ.** Основная трудоемкость построения автоматных программ связана с проектированием автоматов. Однако существуют задачи, про которые известно, что они могут быть решены с использованием автоматов, но эвристически построить автоматы в этих задачах крайне трудно или даже невозможно. Поэтому в данном случае допустимо использовать различные подходы эволюционного (динамического) программирования, в частности, генетические алгоритмы. Генетические алгоритмы — это методы оптимизации, базирующиеся на эволюции популяции особей, в процессе которой ведется поиск максимального значения целевой функции. Основной идеей генетических алгоритмов является использование принципа естественного отбора, заключающегося в том, что наиболее приспособленные особи дают потомство, формирующее следующее поколение. В среднем, следующее поколение является более приспособленным к „окружающей среде“, чем предыдущее.

Генерация автоматов посредством генетических алгоритмов позволяет до 80 % кода автоматных программ строить почти автоматически, так как в программах этого класса объем кода, порождаемого автоматами, может достигать указанной величины.

В качестве практической реализации такого подхода рассмотрим человекоподобного робота *Nao* французской компании Aldebaran Robotics. Этот робот является официальной моделью лиги Standard Platform международных соревнований по футболу среди роботов [11]. В рамках международного проекта RoboCup в целях привлечения внимания ученых и разработчиков к проблемам искусственного интеллекта, робототехники и смежных областей ученым в форме соревнования предлагается решать стандартную задачу обучения робота игре в футбол. Участникам соревнования предлагается с помощью симулятора Webots написать программы для управления роботом. Готовые программы можно загружать на сайт, где периодически устраиваются соревнования.

Но даже с использованием парадигмы автоматного программирования создание конкурентоспособной управляющей программы для робота затруднительно без выделения набора базовых действий: шаг вперед, шаг назад, поворот налево, направо, удар по мячу, встать из положения лежа на спине и из положения лежа на животе. В то же время задать эти действия весьма непросто. Каждое действие состоит из некоторого числа фаз (в примерах описаний движений, поставляемых вместе с симулятором, от 10 до 100 и более), каждая из которых определяет значения углов сервомоторов робота. Подобрать требуемые значения вручную практически невозможно. Методы, основанные на знаниях о физике и биомеханике движений, сложны для реализации [12]. Поэтому предлагается использовать генетические алгоритмы для генерации базовых действий, равно как и для генерации автоматных программ, в которых полученные базовые действия будут интерпретированы как управляющие воздействия. В качестве особей генетического алгоритма будем рассматривать конечные автоматы. Представление конечных автоматов в виде хромосом и генетические операторы для работы с данными хромосомами подробно рассматриваются в работах [13—15].

**Применение параллельных вычислений для генерации конечных автоматов.** Применение генетических алгоритмов для автоматической генерации конечных автоматов является крайне ресурсоемкой процедурой. При этом основной проблемой в описанном подходе к решению задачи автоматического построения системы управления является реализация функции оценки особей (функции приспособленности, или фитнес-функции). Для большинства задач не составляет труда произвести имитацию автоматизированного объекта в виртуальной среде (в данном случае — в среде Webots) и по результатам его работы произвести оценку особи. Однако такая операция требует больших затрат вычислительных ресурсов и, как следствие — времени. Поэтому предлагается выполнять вычисление фитнес-функции параллельно в рамках модели распределенной памяти (рис. 2).

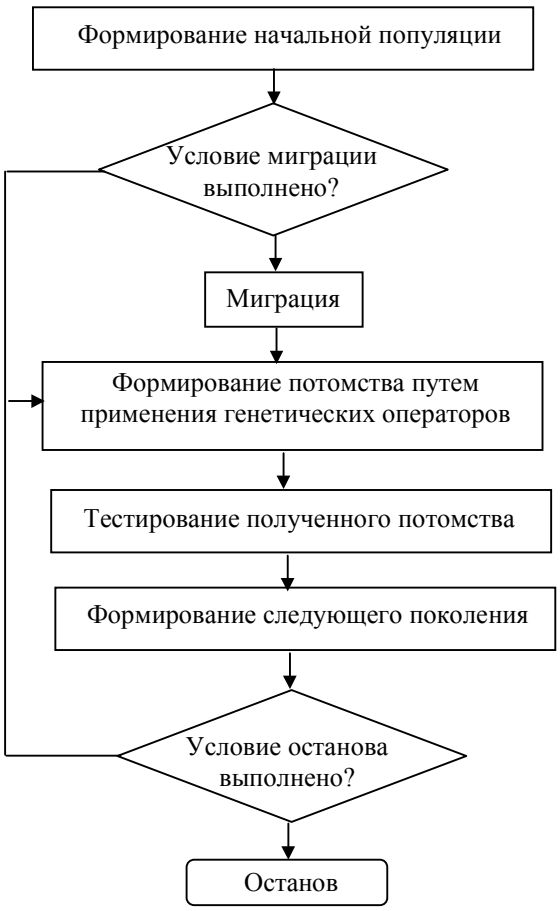


*Рис. 2*

Рассмотрим эффективность реализации параллельных вычислений фитнес-функции на примере задачи построения автомата, управляющего муравьем (задача „Умный муравей“ [13]). Эта задача состоит в определении оптимальной стратегии муравья, передвигающегося по игровому полю. Игровое поле представляет собой двумерный тор, в определенных клетках которого расположена „еда“. Муравей видит перед собой лишь одну клетку. В начале игры муравей находится в клетке с координатами (0, 0). За один ход может быть сделано одно из действий: шаг вперед, поворот направо, поворот налево. Если в клетке, в которую попадает муравей, есть „еда“, то она „съедается“. Цель игры — „съесть“ наибольшее количество еды за ограниченное число ходов. „Еда“ в процессе игры не возобновляется. Муравей „жив“ на протяжении игры — „еда“ не является необходимым ресурсом для его существования. Поведение муравья будем описывать конечным автоматом Мили.

На рис. 3 приведен график зависимости времени работы генетического алгоритма от номера итерации. Первые 128 итераций проводятся на одном компьютере (Intel Celeron M,

1,73 ГГц). После этого к вычислениям подключается второй компьютер (Intel Core 2, 2,50 ГГц), наблюдаемый на графике пик — момент его подключения.

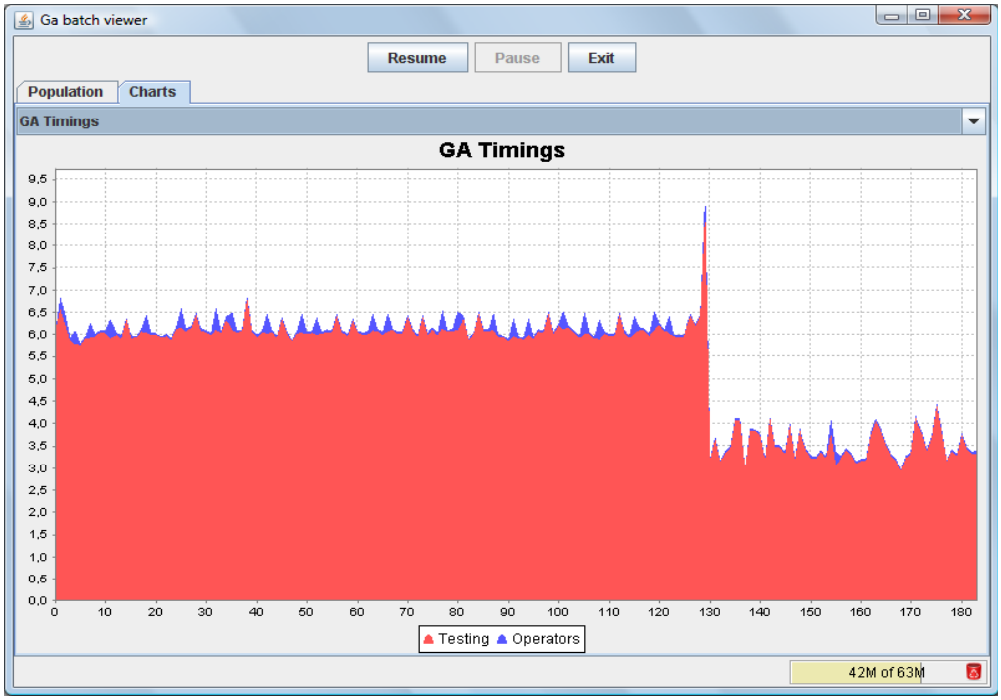


*Рис. 3*

Для оценки эффективности увеличения числа процессоров ($p$), реализующих рассматриваемый генетический алгоритм, были проведены эксперименты, результаты которых приведены в таблице ($N$ — размер популяции, $T$ — время тестирования популяции, $t$ — время тестирования одной особи).

**Анализ производительности параллельного генетического алгоритма**

| $p$ | $N$ | $T$, с | $t$, с | $a$, с | $S_p$ |
|---|---|---|---|---|---|
| 1 | 5000 | 13,00 | 0,0026 | 0,0046 | 1,00 |
| 3 | 2000 | 3,50 | 0,0018 | 0,0018 | 2,50 |
| 5 | 5000 | 7,25 | 0,0015 | 0,0013 | 3,57 |
| 10 | 9000 | 7,50 | 0,0008 | 0,0009 | 5,26 |
| 18 | 9000 | 5,50 | 0,0006 | 0,0007 | 6,67 |

Следует отметить, что компьютеры, используемые в экспериментах, имели различные характеристики и, как следствие — различную производительность. В таблице также приведены теоретические оценки времени тестирования особи ($a$) и ускорения ($S_p$) алгоритма за счет его параллельного исполнения.

В результате анализа полученных данных в соответствии с законом Амдала было установлено, что доля последовательных вычислений составляет всего 10 % от всех, производимых в рассматриваемом генетическом алгоритме. Отметим также, что существенное увеличение числа процессоров в системе увеличивает производительность незначительно. Более того, с определенного числа добавление новых процессоров в систему может увеличивать время тестирования популяции за счет затрат на передачу данных между ними.

Таким образом, совместное использование автоматного программирования [16], генетических алгоритмов и распределенных вычислений, по мнению авторов настоящей работы, является перспективным направлением разработок в области автоматизации процесса построения систем управления, в том числе и человекоподобными роботами.

СПИСОК ЛИТЕРАТУРЫ

1. *Дейкстра Э.* Смиренный программист // Лекции лауреатов премии Тьюринга за первые двадцать лет. 1966—1985. М.: Мир, 1993.

2. *Кнут Д.* Искусство программирования. Т. 1. Основные алгоритмы. М.: Вильямс, 2000.

3. Software Engineering. Germany: NATO Science Committee, 1968. [Electronic resource]: <http://www.europrog.ru/book/nato1968e.pdf>.

4. Software Engineering Techniques. Italy: NATO Science Committee, 1969. [Электронный ресурс]: <http://www.europrog.ru/book/nato1969e.pdf>.

5. *Cai K.-Yu., Chen T. Y., Tse T. H.* Towards Research on Software Cybernetics // Proc. of 7th IEEE Int. on High-assurance Systems Engineering (HASE 2002). Los Alamitos: IEEE Computer Society Press, 2002.

6. *Шалыто А. А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб: Наука, 1998.

7. *Непейвода Н. Н.* Стили и методы программирования. М.: Интернет-Университет Информационных Технологий, 2005.

8. *Непейвода Н. Н., Скопин И. Н.* Основания программирования. М.—Ижевск: Институт компьютерных исследований, 2003.

9. *Шалыто А. А.* Автоматное программирование // Тез. докл. Междунар. науч. конф. памяти проф. А. М. Богомолова „Компьютерные науки и информационные технологии“. Саратов: СГУ, 2007.

10. *Кузьмин Е. В., Соколов В. А.* Моделирование, спецификация и верификация „автоматных“ программ // Программирование. 2008. № 1. С. 38—60.

11. *Michel O.* Professional Mobile Robot Simulation // Int. J. of Advanced Robotic Systems. 2004. Vol. 1, N 1. P. 39—42.

12. *Zhou C., Hu L., Acosta C., Yue P.* Humanoid Soccer Gait Generation and Optimization Using Probability Distribution Models, 2006.

13. *Данилов В. Р., Шалыто А. А.* Метод генетического программирования для генерации автоматов, представленных деревьями решений // Сб. докл. XI Междунар. конф. по мягким вычислениям и измерениям. СПб: СПбГЭТУ „ЛЭТИ“, 2008. С. 248—251.

14. *Поликарпова Н. И., Точилин В. Н., Шалыто А. А.* Применение генетического программирования для реализации систем со сложным поведением // Сб. тр. IV Междунар. конф. „Интегрированные модели и мягкие вычисления в искусственном интеллекте“. Т. 2. М.: Физматлит, 2007. С. 598—604.

15. *Бедный Ю. Д., Шалыто А. А.* Применение генетических алгоритмов для создания системы управления танком в игре Robocode // Сб. докл. XI Междунар. конф. по мягким вычислениям и измерениям. СПб: СПбГЭТУ „ЛЭТИ“, 2008. С. 261—265.

16. *Поликарпова Н. И., Шалыто А. А.* Автоматное программирование. СПб: Питер, 2009.

***Сведения об авторах***

***Анатолий Абрамович Шалыто*** — д-р техн. наук, профессор; Санкт-Петербургский государственный университет информационных технологий, механики и оптики, кафедра технологии программирования; зав. кафедрой; E-mail: shalyto@mail.ifmo.ru

***Евгений Андреевич Мандриков*** — студент; Санкт-Петербургский государственный университет информационных технологий, механики и оптики, кафедра компьютерных технологий; E-mail: mandrikov@rain.ifmo.ru

***Юлия Константиновна Чеботарева*** — студентка; Санкт-Петербургский государственный университет информационных технологий, механики и оптики, кафедра компьютерных технологий; E-mail: chebj@rain.ifmo.ru

Рекомендована институтом

Поступила в редакцию
10.03.09 г.

# ПЕРСПЕКТИВНЫЕ ИССЛЕДОВАНИЯ

УДК 004.021

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

## ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ И РАЗРАБОТКИ МАССОВЫХ МОБИЛЬНЫХ СЕРВИСОВ

Рассматриваются вопросы применения интеллектуальных технологий для проектирования, разработки и организации эффективного взаимодействия мобильных сервисов в распределенных средах сотовой связи.

***Ключевые слова:*** *мобильный сервис, мобильная коммерция, база знаний, локальное взаимодействие, инструментальная среда.*

Современный мобильный телефон представляет собой многоцелевой инструмент, обеспечивающий широкие возможности взаимодействия пользователя с реальным миром посредством разнообразных интерфейсов (голосовая связь, обмен данными через Интернет, технологии локального взаимодействия). Как следствие, сообщество пользователей мобильных телефонов в процессе повседневной деятельности формирует распределенную динамическую информационную среду со сложным коллективным поведением и неоднородной структурой, инвариантной относительно мобильных сервисов. Под мобильным сервисом понимается механизм, позволяющий пользователю получать (а провайдеру — предоставлять) товары или услуги посредством мобильной связи на возмездной основе. Система мобильных сервисов лежит в основе инфраструктуры мобильной коммерции (mCommerce), которая является разновидностью электронной коммерции — широкого спектра форм коммерческой деятельности, основанной на предоставлении услуг посредством телекоммуникационных технологий [1].

Архитектура мобильных сервисов принципиально обусловлена не столько технологическими особенностями сетей связи и инструментов разработки, сколько нормативно-правовой базой в области электронных платежей, принятой в той или иной стране. Например, в России на настоящий момент наиболее перспективной является схема платежей на основе предоплаченного финансового продукта (ПФП) по технологии ПэйКэш (мобильный кошелек Вымпелком, ПФП ПэйКэш) [2, 3]. В рамках данной схемы мобильный сервис представляет собой распределенное приложение, обеспечивающее взаимодействие между пользователем, провайдером услуги и процессинговым центром. Провайдер обеспечивает доступ как к собственным сервисам, так и к внешним сервисам других провайдеров через стандартную коммуникационную среду. Пользователь взаимодействует с провайдером, 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, WiMAX и пр.). Платежи за оказанные услуги выполняются в режиме реального времени через процессинговый центр платежной системы, взаимодействующий с клиентом независимо от провайдера. Это снимает с операторов сотовой

связи ответственность за качество предоставляемых провайдерами услуг. К мобильному сервису также относится 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, IrDa, Bluetooth или NFC. Пример мобильного сервиса продажи электронных билетов на развлекательные мероприятия был рассмотрен в статье [3].

Несмотря на кажущуюся простоту процесс разработки массовых мобильных сервисов затруднен потребностью в применении специальных навыков создания распределенных мобильных приложений и их сопряжения с платежной системой на основе технологии ПэйКэш. Вследствие отсутствия устоявшихся подходов к описанию требований и архитектуры таких сервисов процесс их проектирования не может быть полностью реализован на основе формальных методов и требует применения интеллектуальных технологий, базирующихся на экспертных знаниях. Примером такого решения является инструментальная технологическая среда (ИТС) [4], важнейшую часть которой составляет интеллектуальный спецификатор требований. Он 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 их из репозитория, или использует типовые сценарии (эталонные сервисы), представленные в базе знаний. Дополнительно в состав ИТС включены мастера генерации кода, которые на основе диалога с пользователем формируют архитектуру распределенного приложения, а также генерируют сами программные коды. В отличие от спецификатора требований, взаимодействующего с пользователем в терминах бизнес-процессов, мастера генерации кода отображают, в первую очередь, технологические аспекты реализации сервиса. ИСТ реализована как надстройка над средой программирования Netbeans; создаваемое в ИСТ клиентское приложение использует платформу Java ME CLDC 1.0/MIDP 2.0, что позволяет запускать приложения на мобильных телефонах в операционных системах Windows Mobile или Symbian с использованием соответствующих виртуальных машин.

ИСТ может использоваться как для разработки сервисов, так и для экспериментального прототипирования с целью исследования эффективности их функционирования (как систем массового обслуживания) в различных условиях. Для этого применяется подход на основе имитационного моделирования; для его валидации рассматриваются различные модификации метода очередей [5]. Таким образом, это обеспечивает адаптивность ИСТ и делает целесообразным ее использование на всех этапах жизненного цикла мобильных сервисов.

## СПИСОК ЛИТЕРАТУРЫ

1. *Новомлинский Л.* Электронная коммерция. Тенденции развития в мире и в России. [Электронный ресурс]: <www.tops.ru/publishing/pub_007.html>.

2. Описание технологии PayCash и подключения к платежной системе. [Электронный ресурс]: <www.paycash.ru>.

3. *Гринина Е. А.* и др. Инструментальная технологическая среда для создания массовых мобильных он-лайн-сервисов нового поколения. Ч. I: Принцип действия и программная архитектура // Науч.–технич. вестн. СПбГУ ИТМО. 2008. Вып. 54. С. 80—85.

4. *Золотарев О. А., Гринина Е. А., Бухановский А. В.* Инструментальная технологическая среда для создания массовых мобильных он-лайн-сервисов нового поколения. Ч.II: Технологии локального взаимодействия // Там же. С. 86—91.

5. *Lazowska E. D.* et al. Quantitative system performance: computer system Analysis using queueing network models. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1984. 420 p.

***Сведения об авторах***

***Екатерина Александровна Гринина*** — канд. физ.-мат. наук; ООО „Технологии процессинга“, Санкт-Петербург, руководитель научного отдела; E-mail: grinina@tprs.ru

***Олег Анатольевич Золотарев*** — ООО „Технологии процессинга“, Санкт-Петербург, генеральный директор; E-mail: zolotarev@tprs.ru

***Илья Андреевич Пименов*** — студент; Санкт-Петербургский государственный университет информационных технологий, механики и оптики, кафедра компьютерных технологий; E-mail: zolotarev@tprs.ru

***Александр Валерьевич Бухановский*** — д-р техн. наук, профессор; НИИ Наукоемких компьютерных технологий Санкт-Петербургского государственного университета информационных технологий, механики и оптики; директор; E-mail: avb_mail@mail.ru

Рекомендована институтом

Поступила в редакцию
10.03.09 г.

УДК 519.85

А. В. ГЕРГЕЛЬ

# АДАПТИВНЫЕ ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ ДЛЯ МНОГОМЕРНОЙ МНОГОЭКСТРЕМАЛЬНОЙ ОПТИМИЗАЦИИ

Рассматриваются методы решения многомерных многоэкстремальных задач, использующие многошаговую схему редукции размерности. Предложена новая схема параллельного глобального поиска на основе адаптивной многошаговой схемы редукции размерности.

***Ключевые слова:*** *многомерная многоэкстремальная оптимизация, алгоритмы параллельного глобального поиска, адаптивная многошаговая схема редукции размерности.*

Задачи многомерной многоэкстремальной оптимизации обладают вычислительной сложностью, что определяется, прежде всего, экспоненциальным ростом объема вычислений при увеличении размерности (числа варьируемых параметров). Кроме того, во многих случаях расчет значений функционалов оптимизационной задачи (целевой функции и ограничений) требует существенного объема вычислений, поскольку зачастую связан с проведением вычислительного эксперимента с той или иной математической моделью исследуемого объекта, системы или явления. Именно на такие вычислительно-трудоемкие задачи ориентирована разработка параллельных методов многомерной многоэкстремальной оптимизации в настоящей работе.

**Задачи многомерной многоэкстремальной оптимизации и многошаговая схема редукции размерности.** Задача многомерной многоэкстремальной оптимизации может быть представлена как проблема поиска наименьшего значения действительной функции $\varphi(y)$

$$\varphi(y^*) = \min\{\varphi(y): y \in D\}, \qquad (1)$$

где $D$ есть область поиска, представляющая собой некоторый гиперпараллелепипед $N$-мерного евклидова пространства.

Многие методы решения многомерных многоэкстремальных оптимизационных задач используют многошаговую схему редукции размерности, согласно которой решение задачи (1) может быть получено посредством решения последовательности „вложенных“ одномерных задач (см., например, [1—3]):

$$\min_{y \in D} \varphi(y) = \min_{y_1 \in [a_1, b_1]} \ldots \min_{y_N \in [a_N, b_N]} \varphi(y_1, \ldots, y_N). \qquad (2)$$

Согласно выражению (2), решение многомерной многоэкстремальной задачи оптимизации сводится к решению одномерной задачи:

$$\varphi^* = \min_{y \in D} \varphi(y) = \min_{y_1 \in [a_1, b_1]} \tilde{\varphi}_1(y_1), \tag{3}$$

где

$$\tilde{\varphi}_i(y_i) = \varphi_i(y_1, ..., y_i) = \min_{y_{i+1} \in [a_{i+1}, b_{i+1}]} \varphi_{i+1}(y_1, ..., y_i, y_{i+1}), \quad 1 \le i < N, \tag{4}$$

$$\varphi_N(y_1, ..., y_N) = \varphi(y_1, ..., y_N). \tag{5}$$

Правила (3)—(5) определяют множество задач

$$F_l = \{\tilde{\varphi}_i(y_i), 1 \le i \le l\}, \tag{6}$$

порождаемых в соответствии с многошаговой схемой редукции. Количество задач в множестве $F_l$ в процессе поиска может изменяться: увеличиваться при переходе к следующей переменной и уменьшаться по завершении решения какой-либо из задач (можно отметить, что при этом количество задач не превышает размерности решаемой задачи $N$). При этом активной — решаемой — в множестве $F_l$ является только одна задача — с максимальным номером варьируемой переменной.

Многошаговой схеме редукции размерности присущи определенные недостатки — так, например, вычисления являются избыточными, поскольку решение исходной задачи оптимизации сводится к минимизации одномерных функций в раздельных подобластях области поиска.

Для повышения эффективности глобального поиска может быть предложена обобщенная (адаптивная) многошаговая схема редукции размерности, в которой осуществляется одновременное решение всех задач множества $F_l$ из (6). В этом случае при решении каждой задачи множества $F_l$ могут быть использованы результаты решения всех других задач семейства и, кроме того, решение задач может осуществляться параллельно с использованием многопроцессорных многоядерных вычислительных систем [4]. Применение высокопроизводительных компьютеров позволит существенно повысить сложность решаемых задач глобальной оптимизации.

Выполнение итерации глобального поиска для любой задачи множества $F_l$ с номером переменной, меньшим, чем $N$ ($N$ есть размерность решаемой задачи), будет порождать новую одномерную задачу вида (4), и, тем самым, количество задач в семействе $F_l$ может оказаться значительным (десятки и сотни тысяч).

**Вычислительная схема параллельного глобального поиска для адаптивной многошаговой схемы редукции размерности.** Введем множество номеров задач семейства $F_l$ из множества (6):

$$L = \{1, 2, \dots, l\}$$

и пусть для проведения вычислений имеется $p>1$ процессоров. Распределим имеющиеся задачи между процессорами — данное распределение можно зафиксировать при помощи соответствующего разделения множества $L$ на подмножества:

$$\Pi = \{\pi_1, \pi_2, \dots, \pi_p\}, \tag{7}$$
$$\pi_i = \{j_s : j_s \in L, 1 \le s \le l_i\}, \ 1 \le i \le p,$$
$$\forall\, i \in L \ \exists\, j : i \in \pi_j, \ \forall\, i,j \Rightarrow \pi_i \cap \pi_j = \{\varnothing\},$$

где $\pi_i$ ($1 \le i \le p$) есть множество задач, распределенных для решения на процессоре $i$.

Построенная схема является децентрализованной — все процессоры работают параллельно и самостоятельно генерируют точки проведения испытаний. С другой стороны, между процессорами существует информационное взаимодействие — при получении в какой-то задаче новой улучшенной оценки минимального значения оптимизируемой функции эта оценка должна передаваться задаче-родителю. Поскольку решаемые задачи теперь распределены

между процессорами, то передача получаемых оценок может приводить к сложным информационным зависимостям между процессорами. Для минимизации числа таких зависимостей распределение решаемых оптимизационных задач между процессорами должно быть согласовано со структурой задач.

Для решения проблемы распределения задач между процессорами может быть предложена простая и эффективная децентрализованная схема вычислений, в которой в достаточной степени учитывается иерархическая структура зависимости решаемых задач оптимизации.

Пусть номера терминальных задач (задач, в которых номер варьируемой переменной равен $n$) образуют множество

$$L^n = \{i_j : i_j \in L,\ 1 \le j \le l^n\}.$$

Распределим терминальные задачи между процессорами и представим применяемое распределение по аналогии с (7) при помощи множества:

$$\Pi^n = \{\pi_0, \pi_1, \pi_2, \dots, \pi_p\}, \qquad (8)$$
$$\pi_0 = L \setminus L^n,\ \pi_i = \{j_s : j_s \in L^n,\ 1 \le s \le l_i\},\ 1 \le i \le p,$$
$$\forall\, i \in L\ \exists\, j : i \in \pi_j,\ \forall\, i,j \Rightarrow \pi_i \cap \pi_j = \{\varnothing\}.$$

В (8) каждое подмножество $\pi_i$ определяет список терминальных задач, распределенных для решения на процессоре $i$. Подмножество $\pi_0$ содержит номера всех структурных задач (задач с уровнем, меньшим $n$). Вопрос распределения задач подмножества $\pi_0$ по процессорам необходимо разрешить дополнительно; в наиболее простом случае для задач данного подмножества можно выделить дополнительный процессор. Новая схема (8) отличается систематическим характером информационных взаимодействий между процессорами. Процессоры с терминальными задачи пересылают получаемые оценки минимальных значений исходной оптимизируемой задачи только управляющему процессору. Управляющий процессор занимается обработкой только структурных задач без выполнения трудоемких вычислений значений функционалов исходной решаемой задачи. При этом управляющий процессор может порождать терминальные задачи и в этом случае задачи должны переправляться для выполнения на те или иные процессоры с терминальными задачами.

Работа выполнена при поддержке РФФИ (грант № 07-01-00467-а) и Совета по грантам Президента Российской Федерации по государственной поддержке ведущих научных школ Российской Федерации (грант № НШ-4694.2008.9).

## СПИСОК ЛИТЕРАТУРЫ

1. *Стронгин Р. Г.* Численные методы в многоэкстремальных задачах. М.: Наука, 1978.
2. *Strongin R. G., Sergeyev Ya. D.* Global Optimization with non-convex constraints: Sequential and parallel algorithms. Dordrecht: Kluwer Academic Publishers, 2000.
3. *Городецкий С. Ю., Гришагин В. А.* Нелинейное программирование и многоэкстремальная оптимизация. Н. Новгород: Изд-во ННГУ, 2007.
4. *Гергель В. П.* Теория и практика параллельных вычислений. М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2007.

***Сведения об авторе***

***Александр Викторович Гергель*** — Нижегородский государственный университет им Н. И. Лобачевского, кафедра математического обеспечения ЭВМ; программист 1 категории; E-mail: gergelm@unn.ru

Рекомендована институтом

Поступила в редакцию
10.03.09 г.

УДК 621.382

В. Ю. Холкин

# МОДЕЛЬ ВОЗНИКНОВЕНИЯ 1/*f*-ШУМА КАК РЕЗУЛЬТАТ ПУАССОНОВСКОГО ПРОЦЕССА

Возникновение 1/*f*-шума интерпретируется как результат эволюции пуассоновского процесса. Сопоставляются представления спектральной плотности пуассоновского процесса и эквивалентного ему 1/*f*-шума.

***Ключевые слова:*** *1/f-шум, 1/f-зависимость, низкочастотный шум.*

Изучение случайных сигналов с распределением спектральной плотности мощности, обратно пропорциональной частоте (так называемого 1/*f*-шума), было инициировано развитием теории твердотельных полупроводниковых устройств. Несмотря на то что 1/*f*-шум, как показано экспериментально, можно наблюдать в различных системах (что обусловливает неоднозначность его названия: фликкер-шум, токовый, контактный, избыточный и т.д.), до настоящего времени нет единого мнения о природе этого явления. Как следствие, это не позволяет построить адекватную математическую модель, необходимую, в частности, для поверочных расчетов и оценки характеристик собственных шумов полупроводниковых приборов на этапе их проектирования.

Предлагаемая в статье [1] барьерная модель возникновения 1/*f*-шума хорошо согласуется с экспериментальными данными [2] и объясняет возникновение 1/*f*-участков наличием потенциальных барьеров. Однако эта модель является достаточно грубой в силу того, что она объясняет лишь асимптотическое поведение функции спектральной плотности такого процесса [3]. В настоящей статье рассматривается детализированная модель, которая определяет спектральную плотность зависимостью $1/f^{y}$, где пределы степени $y$ изменяются от 0,8 до 1,2.

В качестве механизма генерации 1/*f*-шума в рамках данной модели используется пуассоновский процесс, который описывается случайной последовательностью точек $\{t_k\}$ вдоль временной оси

$$N(t)=\sum \omega(t-t_k); \quad 0\le t_1 \le t_2 \le \dots , \tag{1}$$

где $\omega(t)$ — единичная функция включения $\omega(t)=\left(1+\mathrm{sign}t\right)/2$. Таким образом, случайное значение $N(t)$ отражает число точек включения между 0 и $t$. Спектральная плотность такого процесса задается выражением [4]

$$K_{xx}\left(f\right)=\frac{2\lambda P\left(1-P\right)}{\left(2\pi f\right)^2+\lambda^2}+P^2\delta\left(f\right), \tag{2}$$

где $P^2\delta\left(f\right)$ — постоянная составляющая при $f = 0$; $P$ — вероятность появления события (включений или появления импульса) в единичный интервал времени. Величина $P$ соответствует коэффициенту заполнения процесса, т.е. отношению средней длительности импульса к среднему периоду процесса. Величина $\lambda$ характеризует среднюю частоту (интенсивность) процесса, а $f$ — текущую частоту.

Уравнение (2) является обобщением выражения $c/f^{y}$ на весь диапазон частот. При достаточно большом значении $\lambda$ кривая спектральной плотности не может быть выражена в форме 1/*f*. Напротив, при малой частоте (интенсивности) процесса $\lambda<<f$ для сверхнизких частот спектральная плотность начинает вести себя как $c/f^{y}$.

В результате проведенных исследований пределов изменчивости параметров модели (1)—(2) получено, что для модели приближения 1/*f*-шума пуассоновским процессом существует нижняя граница параметра $y = 0,8$ в эмпирической формуле $c/f^y$. Аналогично определена верхняя граница параметра $y = 1,5$. Обе оценки подтверждаются экспериментальными данными [3].

Таким образом, 1/*f*-шум в ряде случаев может интерпретироваться как результат эволюции некоторого пуассоновского процесса, в явном виде не выделяемого в системе. Это ограничивает применение эмпирической формулы $c/f^y$ для спектральной плотности диапазоном сверхнизких частот, что связано с введением пределов изменения величины *y*.

## СПИСОК ЛИТЕРАТУРЫ

1. *Холкин В. Ю.* Модель барьерного механизма возникновения 1/*f*-шума в полупроводниковых устройствах // Изв. вузов. Приборостроение. 2008. Т. 51, № 1. С. 54—57.

2. *Beutler F. J., Leneman O. A. Z.* The spectral analysis of impulse processes // Information and Control. 1968. Vol. 12. P. 236—258.

3. *Кешнер М. С.* Шум типа 1/*f* // Академия Тринитаризма. Эл № 77—6567. Публ. 10993, 10.02.2004.

4. *Френкс Л.* Теория сигналов / Пер. с англ; под ред. *Д. Е. Вакман.* М.: Сов. радио, 1974.

***Сведения об авторе***

***Владимир Юрьевич Холкин*** — канд. техн. наук, доцент; Северо-Западный государственный заочный технический университет, кафедра технологии и дизайна радиоэлектронной техники, Санкт-Петербург; E-mail: vkholkin@mail.ru

Рекомендована институтом

Поступила в редакцию
10.03.09 г.

УДК 004.021

П. Д. РАБИНОВИЧ

# ИНТЕЛЛЕКТУАЛЬНОЕ ВЗАИМОДЕЙСТВИЕ В РАСПРЕДЕЛЕННОЙ ИНФОРМАЦИОННОЙ СРЕДЕ

Вводится понятие „интеллектуальное взаимодействие“ как обобщение различных форм обмена между участниками содержательной информацией. Представлены подходы к информатизации интеллектуального взаимодействия.

***Ключевые слова:*** *интеллектуальное взаимодействие, структурированная информация, информатизация, обучение.*

Передача знаний, умений и навыков (ЗУН) может осуществляться в различных формах: профессиональное обучение, повышение квалификации специалистов, стажировка и т.п. Поэтому важно не только обеспечить передачу имеющихся ЗУН, но и постоянно формировать новые.

С позиций системного подхода процессы формирования и передачи знаний тождественны обмену информацией между людьми, между человеком и компьютером, а также между компьютерами, что может быть обобщено понятием „интеллектуальный обмен“, или „интеллектуальное взаимодействие“.

Как известно, „взаимодействие“ — философская категория, отражающая процессы воздействия объектов друг на друга, их взаимную обусловленность и порождение одним объектом другого. Взаимодействие — объективная и универсальная форма движения, развития, которая определяет существование и структурную организацию любой материальной системы как следствие.

*Интеллектуальное взаимодействие* — это обмен информацией между участниками с обязательным подтверждением ее принятия и последующей обработкой (включая анализ и синтез). Интеллектуальное взаимодействие может существовать в следующих видах (формах): обучение, конференция, семинар, стратегическая сессия, консилиум, круглый стол, переговоры, диспут, творческая встреча и т.д. Указанные формы интеллектуального взаимодействия имеют ряд общих черт. В частности, им свойственно наличие содержательного наполнения (контента), специфического тезауруса, системы управления контентом. Различия форм интеллектуального взаимодействия наблюдаются непосредственно в самом содержании и в подходах (способах) к его созданию и передаче. Интеллектуальное взаимодействие, как и обучение, может интерпретироваться как динамическая система с множеством траекторий поведения участников. *Управление* интеллектуальным взаимодействием можно определить как поиск (выделение) среди всего множества возможных траекторий подмножества „допустимых“, т.е. тех траекторий движения участников, которые гарантированно приводят к требуемому результату интеллектуального взаимодействия. Конструкция такого подмножества определяется путем задания принципов (правил) формирования подмножества допустимых состояний, установления правил построения допустимых траекторий с последующей оптимизацией выявленных траекторий по заданным критериям.

Основу интеллектуального взаимодействия составляет его содержание, или контент (от *англ.* content). Контент может быть определен как уже имеющиеся, переработанные, усвоенные, накопленные и(или) новые знания, подвергающиеся последующей верификации. Отличительной особенностью контента от информации (сведений) является его структурированность и пригодность для стандартизации. Таким образом, основная задача участника интеллектуального взаимодействия — структурировать предоставляемую информацию в форме контента [1]. Работа с контентом включает его создание, актуализацию, структуризацию и оптимизацию. В ряде случаев возможно динамически формировать контент параллельно с движением по нему, в частности, если интеллектуальное взаимодействие происходит в виде диспута, семинара, мозгового штурма и т.п.

Вопросы информатизации интеллектуального взаимодействия неотделимы от проблемы выбора форм и методов его организации. Взаимодействие может быть организовано как в традиционной (очное обучение, конференции, консилиумы и т.д.), так и в электронной форме (распределенное обучение, электронные форумы, чаты, видеоконференции и т.д.).

Необходимо помнить, что информатизация интеллектуального взаимодействия, равно как и обучения, является комплексной сложной задачей. Основными составляющими качественного решения являются: методологические и технологические аспекты (методички, регламенты, стандарты), технические (аппаратные) средства, программные средства (информационные системы), информационная среда (распределенная физически и информационно) [2].

Среди технических аппаратных средств интеллектуального взаимодействия необходимо выделить интерактивные (интерактивные доски, планшеты, панели, комплексы оперативного контроля знаний). Они позволяют обеспечить качественное визуальное представление контента, интерактивное взаимодействие с ним, а также оперативную обратную связь.

В качестве программного (информационного) обеспечения можно использовать системы управления базами данных, электронного документооборота, управления контентом, поддержки принятия решений, управления знаниями и другие. Данные системы успешно применяются для решения конкретных задач в области создания, накопления, хранения и обработки знаний.

Использование подходов распределенного образования обеспечивает эффективное интеллектуальное взаимодействие экспертов и рабочих групп. Системы дистанционного обучения позволяют организовать распределенный доступ к учебным материалам, проводить тестирование. Грид-технологии позволяют проводить вычислительные эксперименты.

## СПИСОК ЛИТЕРАТУРЫ

1. *Рабинович П. Д.* Исследование и разработка моделей, алгоритмов и программного обеспечения в компьютерных обучающих системах. Дис. ... канд. техн. наук: 05.13.18. М., 2005. 150 с.

2. *Рабинович П. Д., Баграмян Т. Э.* К вопросу об инфраструктуре распределенного обучения // Тр. Института системного анализа РАН. Проблемы вычислений в распределенной среде. М.: Издательство ЛКИ, 2008. Т. 32. С. 205—228.

***Сведения об авторе***

***Павел Давидович Рабинович*** — канд. техн. наук; Педагогическая академия, Москва; проректор по развитию; E-mail: pavel@rabinovitch.ru

Рекомендована институтом

Поступила в редакцию
10.03.09 г.

# SUMMARY

**P. 5—24.**

## INTELLIGENT SOFTWARE PLATFORM FOR COMPLEX SYSTEM COMPUTER SIMULATION: CONCEPTION, ARCHITECTURE AND IMPLEMENTATION

Issues of intelligent technologies application to development process of high-performance software for complex system simulation are considered. Making the best use of distributed calculation resources of different architectural types assumed as a goal of the research.

**Keywords:** high-performance compuring, knowledge base, performance modeling, intelligent system, service-oriented architecture.

***Data on authors***

***Alexander V. Boukhanovsky*** — Dr. Techn. Sci., Professor; Research Institute of Science-Intensive Computer Technologies, St. Petersburg State University of Information Technologies, Mechanics and Optics; Head of Research Institute; E-mail: avb_mail@mail.ru

***Sergey V. Kovalchuk*** — Cand. Techn. Sci., Research Institute of Science-Intensive Computer Technologies, St. Petersburg State University of Information Technologies, Mechanics and Optics, Senior Researcher; E-mail: sergey.v.kovalchuk@gmail.com

***Sergey V. Maryin*** — Ph.D Student, Research Institute of Science-Intensive Computer Technologies, St. Petersburg State University of Information Technologies, Mechanics and Optics, Junior Researcher; E-mail: sergey.maryin@gmail.com

**P. 25—33.**

## PARALLEL METHODS FOR GLOBAL OPTIMIZATION PROBLEM SOLVING

Parallel algorithm for solving multiextremal optimization problems with nonconvex constraints is considered. It is based on the reduction of multidimensional problems to the set of joint one-dimensional ones. New scheme of construction of the set of Peano-type space-filling curves is proposed. Suggested scheme preserves a part of information about closeness of points in multidimensional space. The results of numerical experiments showing acceleration of algorithm convergence with use of the new scheme of multiply curves construction are presented.

**Keywords:** multiextremal optimization, nonconvex constraints, Peano curves, parallel algorithms.

***Data on authors***

***Roman G. Strongin*** — Dr. Phys.-Math. Sci., Professor; N. I. Lobachevsky State University, Software Department, Nizhni Novgorod; President; E-mail: strongin@unn.ac.ru

***Victor P. Gergel*** — Dr. Techn. Sci., Professor; N. I. Lobachevsky State University, Software Department, Nizhni Novgorod; Dean; E-mail: gergel@unn.ru

***Konstantin A. Barkalov*** — Cand. Phys.-Math. Sci.; N. I. Lobachevsky State University, Software Department, Nizhni Novgorod; Senior Teacher; E-mail: barkalov@fup.unn.ru

**P. 33—41.**

## BLOCK-RECURSIVE PARALLEL MULTIPLICATION OF MATRIXES

The new algorithm of a parallel matrix production is introduced in this paper. This algorithm uses less data transferes then others. A memory accesses algorithm complexity for Strassen algorithm of matrix production is considered.

**Keywords:** parallel algorithms, hypercube, data placement, data transferes, memory accesses algorithm complexity.

***Data on author***

***Boris Ya. Steinberg*** — Dr. Techn. Sci.; Southern Federal University, Rostov-on-Don; Department of Algebra and Descrete Mathematics; Head of Department;
E-mail: borsteinb@mail.ru

**P. 41—49.**

## PARALLELIZATION PROBLEMS IN SOME LOCAL TASKS

A short review of parallelization in the local problems is presented in the paper. Some principles for architecture solutions of parallel computer systems are discussed.

**Keywords:** parallelization, local problems, architecture of computer systems, multiplicity of covering.

***Data on author***

***Yuri K. Dem'yanovich*** — Dr. Phys.-Math. Sci., Professor; St. Petersburg State University, Parallel Algorithm Department; Head of Department;
E-mail: Yuri.Demjanovich@JD16531.spb.edu

**P. 50—57.**

## COMPUTING EXPERIMENT ON MULTIPROCESSOR SYSTEMS: ALGORITHMS AND TOOLS

The paper is devoted to the problems that occur during numerical simulation on parallel systems. We considered the problems of rational decomposition, coarsening of triangulated surfaces, hierarchical processing and distributed visualization of large grid datasets.

**Keywords:** multiprocessor systems, distributed visualization, rational decomposition, mathematical modeling, computing experiment, parallel algorithms, triangulation, data coarsen.

***Data on author***

***Mikhail V. Iakobovsky*** — Dr. Phys.-Math. Sci.; Sector of Mathematical Modeling Institute of Russian Academy of Science, Moskow; Head of Sector; E-mail: lira@imamod.ru

**P. 58—66.**

## NONLINEAR EFFECTS IN CONTROL SYSTEMS OF COMPLEX DYNAMIC OBJECTS

Issue of nonlinear dynamic objects control is examined. Special attention is directed to nonlinear effects appeared on conditions of persistent changing of object's dynamic and environment's characteristics. Results of nonlinear dynamic object's behavior in case of emergency or extreme event simulation are presented in the article.

**Keywords:** intelligent system, dynamic object, external disturbance, extreme event.

***Data on author***

***Yuri I. Nechaev*** — Dr. Techn. Sci., Professor; State Marine Technical University, St. Petersburg;
E-mail: petr_oleg@mail.ru

**P. 66—73.**

## AUTOMATIC PROGRAMMING AND PARALLEL CALCULATIONS

This article describes the main provisions of automatic programming and justify the advantages of its use in software development. Also described usage of parallel computing technology for constructing finite state machines.

**Keywords:** automata-based programming, control system, genetic algorithm, fitness-function.

***Data on authors***

***Anatoly A. Shalyto*** — Dr. Techn. Sci., Professor; St. Petersburg State University of Information Technologies, Mechanics and Optics, Department of Programming Technologies; Head of Department; E-mail: shalyto@mail.ifmo.ru

***Evgeny A. Mandrikov*** — Student; St. Petersburg State University of Information Technologies, Mechanics and Optics, Department of Computer Technologies; E-mail: mandrikov@rain.ifmo.ru

***Yulia K. Chebotareva*** — Student; St. Petersburg State University of Information Technologies, Mechanics and Optics, Department of Computer Technologies; E-mail: chebj@rain.ifmo.ru

**P. 74—76.**

## INTELLIGENT TECHNOLOGIES FOR MASS MOBILE SERVICES DESIGN AND DEVELOPMENT

Issues of intelligent technologies applying to design, development and effective interaction organization of mobile services in distributed cellular communication environment are considered.

**Keywords:** mobile service, mobile commerce, knowledge base, local interaction, instrumental environment.

***Data on authors***

***Ekaterina A. Grinina*** — Cand. Phys.-Math. Sci.; „Processing Technologies“,St. Petersburg; Head of R&D Department; E-mail: grinina@tprs.ru

***Oleg A. Zolotarev*** — „Processing Technologies“, St. Petersburg; General Manager; E-mail: zolotarev@tprs.ru

***Ilya A. Pimenov*** — Student, St. Petersburg State University of Information Technologies, Mechanics and Optics, Department of Information Systems; E-mail: ilya.pimenov@gmail.com

***Alexander V. Boukhanovsky*** — Dr. Techn. Sci., Professor; Research Institute of Science-Intensive Computer Technologies, St. Petersburg State University of Information Technologies, Mechanics and Optics, Head of Research Institute; E-mail: avb_mail@mail.ru

**P. 76—78.**

## ADAPTIVE PARALLEL COMPUTATIONS FOR MULTIDIMENSIONAL MULTIEXTREME OPTIMIZATION

Methods for multidimensional multiextreme tasks solving using multistep scheme for dimension reduction are examined. The new scheme of parallel global search based on adaptive multistep scheme for dimension reduction are proposed.

**Keywords:** multidimensional multiextreme optimization, parallel global search algorithms, adaptive multistep scheme for dimension reduction.

***Data on authors***

***Alexander V. Gergel*** — N. I. Lobachevsky State University, Nizhni Novgorod; 1st Category Programmer; E-mail: gergelm@unn.ru

**P. 79—80.**

## MODEL OF 1/*f*-NOISE APPEARANCE AS A RESULT OF POISSON PROCESS

Appearance of 1/*f*-noise is interpreted as a result of Poisson process. Spectral density of Poisson process and equivalent 1/*f*-noise are compared.

**Keywords:** 1/*f*-noise, 1/*f*-dependence, low frequency noise.

***Data on author***

***Vladimir Yu. Kholkin*** — Cand. Techn. Sci.; North-West State Technical University of Distant Education, St. Petersburg; E-mail: vkholkin@mail.ru

**P. 80—82.**

## INTELLIGENT INTERACTION IN DISTRIBUTED INFORMATION ENVIRONMENT

The term “intelligent interaction” is proposed as a generalization of different forms of substantive information exchange between participants. Approaches to informatization of intelligent interaction are presented.

**Keywords:** intelligent interaction, structured information, informatization, education.

***Data on author***

***Pavel D. Rabinovich*** — Cand. Techn. Sci.; Pedagogical academy, Moskow; vice rector; E-mail: pavel@rabinovitch.ru