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 Ξ_S — параметры системы; ξ_A — функция, определяющая параметры параллельной реализации алгоритма для текущей системы; ξ_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; Φ_D — набор ссылок на фреймы, описывающих данные, которыми оперирует вычислительный модуль; ϕ — 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 μ — 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

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

ϕ_{dc} и ϕ_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.

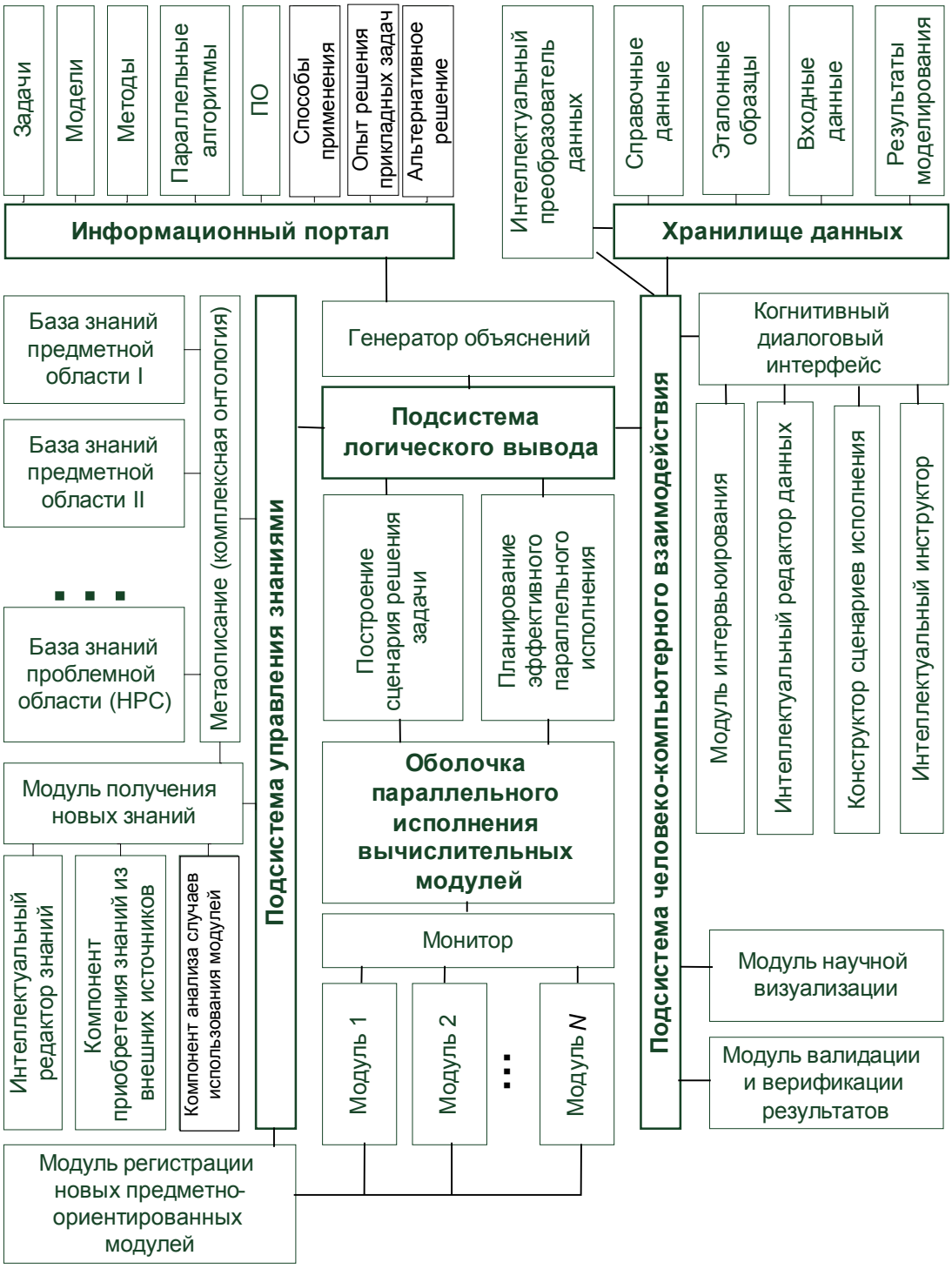

Рис. 1

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.

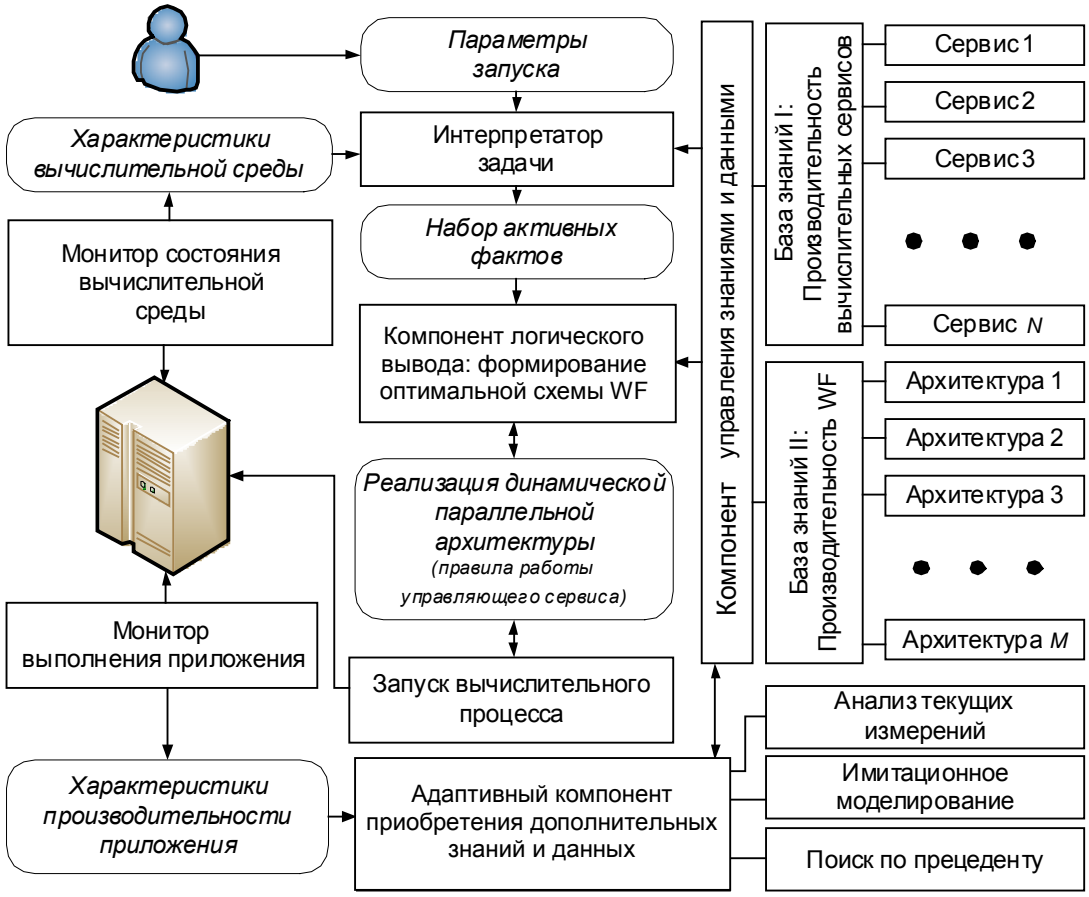

Рис. 2

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.

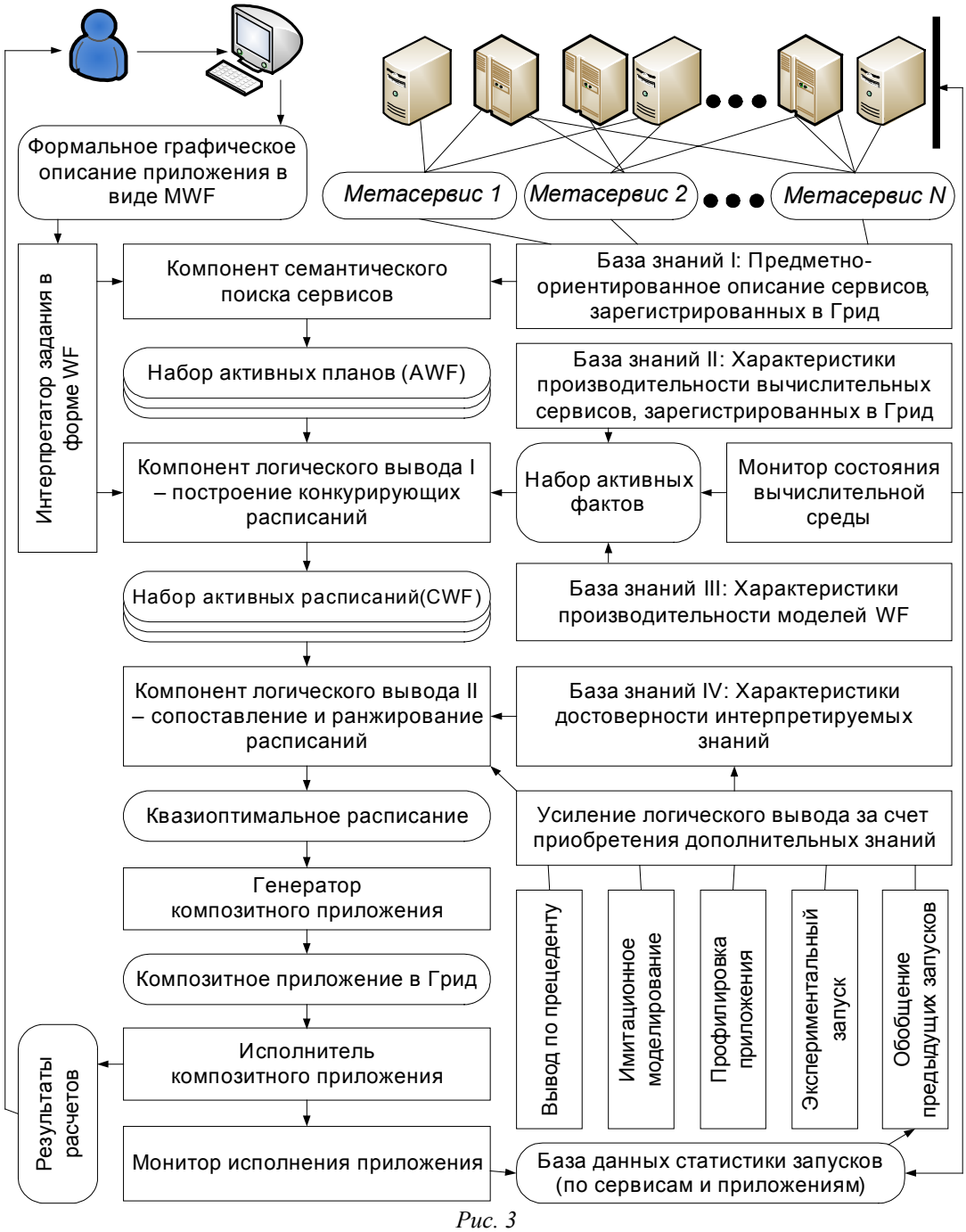

Рис. 3

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.

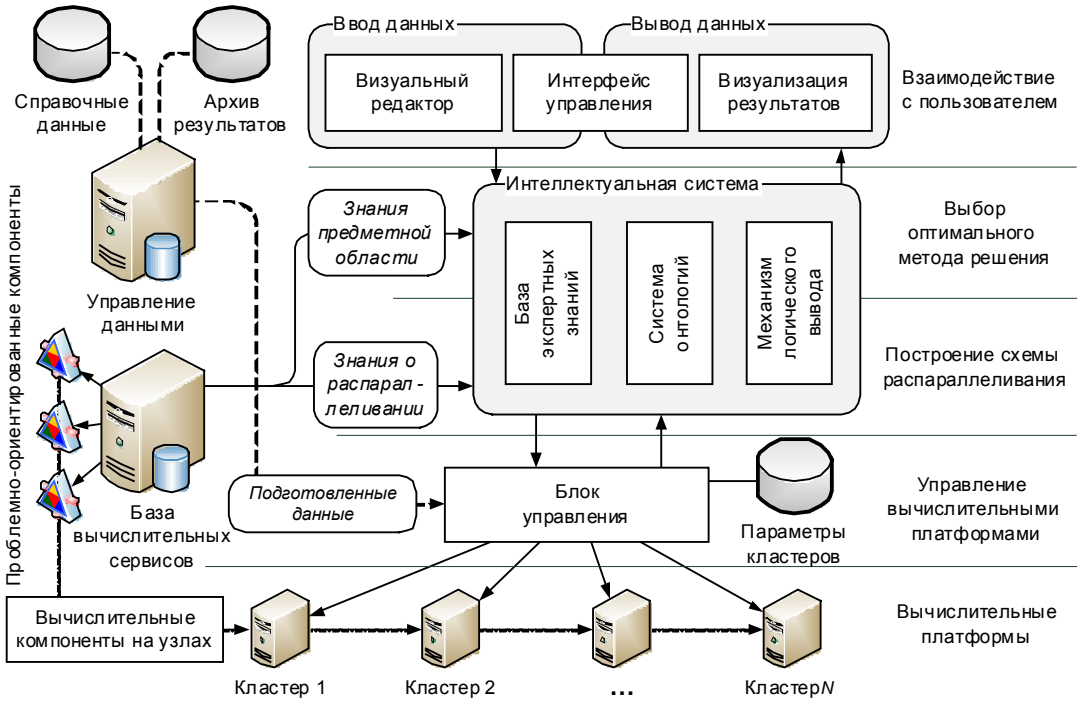

Рис. 4

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.