ВЕСТНИК РОССИЙСКОЙ АКАДЕМИИ НАУК, 2016, том 86, № 4, с. 316–326

ОБОЗРЕНИЕ

ФУНДАМЕНТАЛЬНЫЕ ВОПРОСЫ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

© 2016 г. В.П. Ильин

Институт вычислительной математики и математической геофизики СО РАН, Новосибирск, Россия
Новосибирский государственный университет, Новосибирск, Россия

e-mail: ilin@sscc.ru

Поступила в редакцию 14.09.2015

Сегодня математическое моделирование процессов и явлений занимает всё большее место в разных сферах человеческой деятельности, включая научно-технические, производственные и социальные, а проблемы его развития обретают и фундаментальные, и прикладные, и технологические аспекты, разграничение и взаимосвязи между которыми требуют философско-методологического осмысления. В публикуемой статье освещаются три проблемы: автоматизация вычислений, их суперкомпьютерное представление и создание масштабных систем моделирования с интегрированным окружением открытого типа.

Ключевые слова: математическое моделирование, вычислительные методы и технологии, автоматизация построения алгоритмов, интегрированное программное окружение, междисциплинарные прямые и обратные задачи.

DOI: 10.7868/S086958731604006X

Математическое моделирование – чрезвычайно многогранное понятие, поэтому его рассмотрение может осуществляться в различных плоскостях. Одно из возможных направлений классификации – с точки зрения типов решаемых задач, которые относятся к электромагнетизму, теплофизике, упругопластичности, гидрогазодинамике, многофазной фильтрации, химической кинетике, квантовым явлениям, динамическим системам и т.д. Важно отметить, что особую актуальность приобретают междисциплинарные прямые и обратные задачи. При этом математические формулировки могут быть представлены по-разному – как дифференциальные и/или интегральные, классические или обобщённые, вариационные и смешанные уравнения.

ИЛЬИН Валерий Павлович – доктор физико-математических наук, главный научный сотрудник ИВМиМГ СО РАН, профессор НГУ.

Другой возможный взгляд на это разнообразие – через призму отраслевых приложений: машиностроение, металлургия, энергетика, геофизика, погода и климат, экология и катастрофы, биомедицина, материаловедение, экономика и социум. Главным образом по этому принципу построена доступная в Интернете программа фундаментальных исследований Президиума РАН № 1 по стратегическим направлениям науки (конкурс 2014 г.) “Фундаментальные проблемы математического моделирования”.

Следующим критерием типологизации выступают вычислительные методы и технологии, применяемые на различных стадиях компьютерного эксперимента: геометрическое и функциональное моделирование, генерация сеток, аппроксимация исходных уравнений, решение получаемых алгебраических систем, методы оптимизации для обратных задач, обработка и визуализация численных результатов и т.д.

Распространению моделирования способствуют такие внешние факторы, как фантастический рост производительности многопроцессорных вычислительных систем (МВС), бурное развитие методов теоретической и прикладной математики, а также практические потребности эпохи реиндустриализации и прорывных наукоёмких технологий. Всё это повышает практический потенциал моделирования, превращая его, с одной

стороны, в третий путь познания наряду с теоретическими и эмпирическими (натурными) исследованиями, а с другой – в одно из определяющих условий увеличения производительности труда и внутреннего валового продукта.

Узким местом является производительность программистского труда, рост которой катастрофически отстаёт от темпов увеличения мощностей суперкомпьютеров. Преодоление назревшего кризиса прикладного программирования требует создания новой парадигмы его развития. Сложившаяся многолетняя практика заключается в реализации пакетов прикладных программ (ППП), коммерческих или общедоступных, для конкретных классов задач. Примерами таких продуктов являются ANSYS [1] и Nastran [2]. Разработками другого вида являются библиотеки программ, реализующие совокупность алгоритмов для какого-то типа вычислительных задач. Так, Netgen [3] отвечает за генерацию сеток, PETSc [4] – алгебраические решатели и т.д. Ещё один вариант, набирающий в последнее время популярность, – инструментальные вычислительные системы OpenFOAM [5], DUNE (Distributed Unified Numerical Environment) [6], базовая система моделирования (БСМ) [7].

В случае инструментальных вычислительных систем речь идёт о формировании интегрированного открытого (open source) окружения, разрабатываемого широким сообществом. Оно поддерживает все основные алгоритмические стадии вычислительного эксперимента и позволяет оперативно разрабатывать ППП, в том числе закрытые, для конкретных и разнообразных приложений. Исходным в данном случае оказывается не проблемная, а методологическая и инструментальная ориентация. Этот подход позволяет преодолеть принципиальное противоречие между универсальностью и эффективностью – при избыточности состава алгоритмов для решения конкретной задачи (из допустимого широкого класса) можно выбрать достаточно экономичную реализацию.

Основанием концепции открытых вычислительных систем является представление, согласно которому при практически бесконечном многообразии задач, различающихся индивидуальными особенностями, все они описываются конечным набором математических моделей, методы решения которых могут быть “разложены по полочкам”, а организация работ с такими интеллектуальными объектами должна иметь адекватную степень автоматизации. Создание подобного крупного программного комплекса требует длительного жизненного цикла, расширяемости состава моделей и алгоритмов, адаптации к эволюции компьютерных платформ, взаимодействия с внешними продуктами, наличия гибких интерфейсов, рассчитанных на пользователей различной профессиональной подготовки, а также компонентной архитектуры, поддерживающей принципы модульного (или сборочного) программирования. Фактически речь идёт о создании глобальной высокопроизводительной системы наукоёмких технологий нового поколения, призванной кардинально изменить востребованность математического моделирования, ориентированной на обеспечение импортозамещающих программных разработок, актуальных для национальной безопасности, а также обладающей высокой конкурентоспособностью на внешнем рынке.

Настоящая статья посвящена рассмотрению перечисленных крупномасштабных и неразрывно связанных один с другим вопросов: во-первых, автоматизации построения моделей и алгоритмов, во-вторых, отображения структуры алгоритмов на архитектуру современных и будущих гетерогенных МВС, в-третьих, разработки компонентных архитектур и технологий создания сверхбольших унифицированных комплексов, рассчитанных на то, чтобы прикладное программное обеспечение (ППО) имело не только высокую рентабельность и эффективность, но и гибкие возможности широкого применения, сравнимые с эксплуатационными и интерфейсными качествами операционных систем или компиляторов.

АВТОМАТИЗАЦИЯ ПОСТРОЕНИЯ МОДЕЛЕЙ И АЛГОРИТМОВ – ОСНОВА ИНТЕЛЛЕКТУАЛИЗАЦИИ ППО

Вне зависимости от предметной ориентации соответствующего ППО вычислительный эксперимент проходит по однотипным технологическим стадиям.

Геометрическое и функциональное моделирование. На первом этапе пользователь должен сформулировать задание на расчёт, которое может включать описание сложной геометрической конфигурации какой-то области, состоящей из подобластей с различными материальными свойствами. Если геометрические объекты и операции давно освоены в многочисленных САПРовских продуктах (CAD, CAE, CAM, PLM) и графических системах, то функциональное моделирование требует оперирования с такими формализмами, как уравнения в подобластях, краевые условия на граничных сегментах, различные коэффициенты и т.д. Кроме исходных данных, необходимо указать, что и в какой форме требуется получить. Могут быть также предписаны и применяемые методы или даже детальные вычислительные схемы, определяющие однозначно процесс математического моделирования в конкретной операционной обстановке. Делая акцент на перечисленных аспектах, мы фактически приходим к автоматизации построения моделей и алгоритмов.

Проблемой остаётся создание гибкого входного и выходного интерфейса для конечного пользователя. В идеальном случае такой интерфейс представлен языком программирования естественного типа с возможным активным использованием как текстовых, так и графических средств. Поскольку пользователями одного и того же разрабатываемого обеспечения могут быть специалисты самого разного профиля – математики, программисты, инженеры, техники и т.д., становится очевидной необходимость формирования богатой гаммы прикладных когнитивных средств. Это, в свою очередь, обусловливает задачу организации "фабрики" многочисленных специализированных под приложения языков, за которыми в литературе закрепилось название Domestic Specific Language (DSL) [8].

Важно отметить, что уже первая стадия компьютерного эксперимента свидетельствует: широкая востребованность суперкомпьютерных технологий требует не только высокопроизводительных вычислений, но и искусственного интеллекта высокого уровня. В работе [8] этот момент ассоциируется с переходом от "палеоинформатики" к "неоинформатике".

Дискретизация задачи. Практически всегда решение нетривиальных математических уравнений начинается с построения сетки. Чтобы представить многообразие возникающих здесь вопросов, достаточно перечислить наиболее часто применяемые типы сеток, среди них: адаптивные, структурированные, неструктурированные и квазиструктурированные, согласованные, несогласованные и мортарные, регулярные и нерегулярные, статические и динамические и т.д.

Наиболее эффективные подходы к дискретизации связаны с достаточно сложными дискретными объектами и их преобразованиями, включая последовательности иерархических сеток и их локальные сгущения (zoom), декомпозиции сеточных областей на подобласти, динамическую перестройку сеток, апостериорный и/или априорный учёт свойств искомых решений. Существует значительное количество показателей качества сеток, однако определение оптимальной сетки остаётся слишком сложной проблемой и в практических разработках даже не формулируется. Наиболее распространённый принцип выбора сводится к эмпирическому подходу – использованию плотностей распределения сеточных узлов исходя из общих качественных соображений. Отдельные методические рекомендации относятся к частным случаям и являются только исключениями, подтверждающими общее правило. Одновременно на мировом рынке ППО представлено много как очень дорогих, так и бесплатных сеточных генераторов, в которых используется некоторое количество признанных вычислительным сообществом сеточных структур данных (ССД). Эффективное использование этого огромного овеществлённого интеллектуального потенциала представляется весьма актуальной задачей.

Фундаментальные и прикладные проблемы построения сеток давно стали предметом активных исследований, в которых используются самые различные вариационные и инженерные подходы. Однако проблема оптимальности сеток и их декомпозиции чрезвычайно многогранна и потому по-прежнему не сформулирована достаточно строго. К этому вопросу мы ещё вернёмся, здесь же отмечу, что сегодня можно говорить лишь о построении "хороших" сеток, то есть удовлетворяющих апробированным, но весьма многочисленным качественным или количественным критериям: в треугольниках не должно быть слишком малых углов, объёмы или площади ячеек не должны вырождаться и т.д. Эти критерии либо получаются из теоретических оценок, либо выводятся чисто эмпирически, в силу чего даже сложилось мнение, что нужно строить "красивые" сетки, а потому для контроля сеточных генераторов совершенно естественно прибегать к средствам визуализации.

На описательном уровне можно дать определение качества адаптивности сетки, являющегося если фактически не обязательным, то, по крайней мере, весьма желательным: узлы и рёбра кусочно-гладкой границы расчётной области должны совпадать с узлами и рёбрами сетки, а шаги сетки должны быть мельче там, где производные искомого решения относительно велики. Последнее высчитывается или по априорным теоретическим оценкам, или по апостериорному анализу промежуточных численных результатов. Подчеркнём, что соблюдение этих требований может, как правило, во много раз сэкономить общие вычислительные ресурсы для получения решения с требуемой точностью.

Дискретизация – технологическая стадия, важная с точки зрения как ресурсоёмкости вычислений в целом, так и разрешающей способности расчётов, от которой в большой степени зависит успех моделирования. Особенно это справедливо в отношении задач со сложным пространственно-временным поведением решения, включая актуальные ситуации с сильно разномасштабными (multi-scale) характеристиками. Таким образом, генерация сеток – высокоинтеллектуальная методология, а отсутствие существенных теоретических и алгоритмических результатов делает наиболее реалистичной ориентацию не на автоматическое, а на автоматизированное построение сеток, сопровождаемое их оперативной визуализацией и участием эксперта в контроле вычислительного процесса.

Аппроксимация уравнений. Когда после выполнения предыдущих этапов сформирована ССД, в совокупности с геометрической и функциональ-

ной структурами данных (ГСД и ФСД) отображающая на дискретном уровне всю информацию об исходной задаче, наступает черёд её аппроксимации. Результатом оказывается система конечномерных алгебраических соотношений – алгебраическая структура данных (АСД), в которой могут эффективно использоваться широко распространённые матричные представления для разреженных алгебраических систем. В качестве примера укажем сжатый строчный формат CSR (Compressed Sparse Row) [9].

Исполняемые в данном случае операции оказываются самыми наукоёмкими и представлены разнообразными теоретическими подходами: методами конечных разностей (МКР), конечных объёмов (МКО) и конечных элементов (МКЭ) [10, 11], различными спектральными алгоритмами, методами интегральных уравнений и т.д. Логическая сложность "аппроксиматоров" особенно усиливается, когда применяются методы повышенного порядка точности, особенно на неструктурированных сетках, формулы для которых занимают по несколько страниц. Именно это препятствует их широкому распространению, несмотря на большие преимущества.

Кардинальным решением в данной ситуации является использование возможностей искусственного интеллекта, а именно средств автоматизации аналитических символьных преобразований. В принципе такие инструменты присутствуют в больших специализированных системах типа Reduce или Maple и успешно применяются, например, в пакетах FEniCS и Helmholtz [12]. В названных случаях задача облегчается благодаря разработанной в рамках МКЭ и МКО уникальной поэлементной технологии независимого и легко распараллеливаемого вычисления локальных матриц с последующей сборкой (assembling) глобальной матрицы. Более того, на основе аддитивной методологии удаётся охватить вопросы учёта краевых условий различных типов, автономных аппроксимаций типовых дифференциальных операторов Лапласа, дивергенции, ротора, градиента и т.д., а также их интегральных аналогов в обобщённых вариационных постановках.

Решение алгебраических задач. На данной стадии выполняются различные матрично-векторные операции, требующие наибольших компьютерных ресурсов, поскольку объём арифметических операций и необходимой памяти здесь зачастую растёт нелинейно с увеличением числа степеней свободы задачи. Проводимые вычисления могут заключаться в осуществлении рекуррентных последовательностей, решении систем алгебраических уравнений (линейных – СЛАУ, и нелинейных – СНАУ), решении проблем собственных значений, реализации оптимизационных алгоритмов для задач математического программирования. Названные задачи составляют обширнейшие направления вычислительной алгебры, отличающиеся огромным разнообразием концептуальных подходов, конкретных версий методов и особенностями их применения. Именно здесь актуализируются вопросы распараллеливания алгоритмов и их отображения на архитектуры МВС, в частности, на кластерные системы, содержащие гетерогенные узлы с классическими и специализированными процессорами.

На международном рынке имеется огромное количество алгебраического программного обеспечения, которое непрерывно обновляется и расширяется за счёт адаптивных модификаций для вновь появляющихся компьютерных платформ и архитектур, а также быстрого развития новых алгоритмов. Бурный рост и постоянное обновление создают проблему согласованного переиспользования уже имеющихся продуктов. Надо сказать, что в данной области есть серьёзные достижения: разработаны типовые универсальные структуры данных и библиотеки с основным набором матрично-векторных операций (BLAS, SPARSE BLAS) [9].

Многообразие алгебраических методов вызывается, в первую очередь, отличием типов участвующих матриц – эрмитовых и неэрмитовых, вещественных и комплексных, симметричных и несимметричных, вырожденных и невырожденных, положительно определённых и знаконеопределённых и др. Все типы матриц делятся на плотные и разреженные, и подходы к их обработке существенно различаются. Кроме того, для выбора оптимальных алгоритмов большое значение имеют структурные свойства матриц (ленточные, треугольные и пр.), а также их размерности (понятие "больших" матриц постоянно меняется в зависимости от мощности текущего поколения компьютеров и в постпетафлопсную эру связывается с порядками 10^9–10^{12}). Особую сложность представляют плохо обусловленные задачи с сильной неустойчивостью численных решений относительно ошибок исходных данных или машинных округлений.

Наиболее эффективные современные алгоритмы отличаются высокой логической сложностью: многосеточные подходы, методы декомпозиции областей, приёмы оптимизации последовательностей переменных или масштабирования матриц и т.д. Можно констатировать, что наиболее ресурсоёмкие алгебраические методы также требуют активного использования искусственного интеллекта.

Оптимизационные подходы к решению обратных задач. Решения прямых задач математического моделирования, в которых требуется найти искомые функции при всех заданных коэффициентах уравнений и начальных и граничных условий, могут отличаться большой вычислительной сложностью, но обычно они составляют только

часть проблемы, заключающейся в решении обратной задачи. Последняя характеризуется тем, что какая-то часть её исходных данных зависит от неизвестных параметров, которые требуется найти по условию минимизации описываемого целевого функционала при некоторых дополнительных ограничениях на свойства задачи. Так, когда рассчитываются технические устройства или приборы, конечной целью инженера, как правило, является не просто изучение их свойств, а автоматизированное проектирование оптимальных конфигураций, обеспечивающих требуемые характеристики. При этом практически всегда имеются дополнительные ограничения, связанные с размером, весом или иными функциональными условиями. Другим характерным примером обратной задачи является проблема идентификации параметров математической модели на основе сравнения расчётных результатов с данными натурных (косвенных) измерений.

Основные универсальные подходы к решению обратных задач опираются на применение методов условной минимизации, заключающихся в направленном последовательном поиске локального или глобального минимума, причём на каждом шаге вычисляются промежуточные значения целевого функционала, а это представляет собой не что иное, как решение прямой задачи. Следовательно, решение обратной задачи в общем случае требует многократного решения прямых задач.

Методы оптимизации в последние десятилетия активно развиваются, благодаря чему появились такие новые направления, как алгоритмы внутренних точек, последовательного квадратичного программирования, доверительных интервалов [13]. Однако нужно отметить, что минимизация функционалов со сложными геометрическими характеристиками, особенно овражного типа, – это занятие на грани науки и искусства, ввиду чего полностью автоматизированный вычислительный процесс возможен только в простейших ситуациях. Фактически и здесь требуются высокоинтеллектуальные технологии с поэтапной реализацией всей задачи в диалоговом режиме с пользователем, который на основе своего опыта должен контролировать поведение последовательных приближений и управлять параметрами алгоритмов для быстрейшего достижения конечной цели.

Постобработка и визуализация результатов. Управление вычислительным процессом и средства принятия решения. Результаты алгебраических расчётов лишены какого-либо физического смысла и наглядности, не в последнюю очередь в силу их больших объёмов. Например, МКЭ позволяет получить коэффициенты разложения искомых решений по использованным базисным функциям в ячейках сетки, тогда как пользователю требуется компактное и наглядное изображение многомерных векторных полей. Поэтому ППО должно иметь развитый набор инструментов для формирования таких типичных представлений, как изоповерхности, силовые линии, сечения, всевозможные графики и т.д. Это первое требование. Второе связано с тем, что всего заранее предусмотреть нельзя, и в интеллектуальной системе моделирования должны содержаться средства автоматизации программирования различных возможных характеристик итоговых данных. И наконец, третий фактор – возможное разнообразие профессий конечных пользователей, желающих получать комфортное представление итогов использования компьютера, что и определяет его производственный эффект.

Важно подчеркнуть, что даже при наличии идеального прикладного программного обеспечения компьютерное моделирование сложных процессов или явлений – многогранная творческая деятельность. Так, для систематического изучения каких-то приложений сначала надо убедиться, что применяемые модели и методы соответствуют техническим требованиям, для чего предварительно проводятся пробные расчёты с анализом адекватности получаемых данных. Затем настаёт очередь самих исследований, которые могут представлять собой крупномасштабный машинный эксперимент, который должен предваряться процедурами планирования и подбора методики. Последнее недостижимо без обеспечения гибких возможностей составления расчётных схем, что предполагает создание соответствующих языков (декларативного или императивного типов) для управления вычислительными процессами. Наконец, поскольку моделирование является не самоцелью, а орудием познавательной или производственной деятельности, то для принятия решения по результатам расчётов в состав ППО должны быть заложены или какие-то когнитивные принципы, или средства подключения к САПРовским инфраструктурам, или технологии поддержки и оптимизации эксплуатационных режимов конкретных процессов. Однако эти вопросы уже выходят за рамки самого математического моделирования.

РАСПАРАЛЛЕЛИВАНИЕ АЛГОРИТМОВ И ИХ ОТОБРАЖЕНИЕ НА АРХИТЕКТУРУ МВС

Одной из главных тенденций развития вычислительных наук и технологий является конвергенция алгоритмических структур и компьютерных архитектур. Поскольку вторая из названных сторон процесса оказывается за пределами рассматриваемой в настоящей статье тематики, ограничимся анализом кластерных систем с гетерогенными узлами, содержащими классические (центральные) и графические процессоры с мно-

гочисленными вычислительными ядрами, на которых реализация алгоритмов осуществляется средствами гибридного программирования с организацией MPI-процессов (Message Passage Interface) на распределённой памяти и многопотоковых операций над общей памятью.

Некоторые общие вопросы распараллеливания. Универсальное требование к ППО заключается в отсутствии программных ограничений на число степеней свободы решаемой задачи (ч.с.с., или d.o.f. – degrees of freedom) и количество используемых процессоров и/или ядер. Следует отметить также такие важные характеристики распараллеливания алгоритмов, как слабое и сильное масштабирование. Слабое обозначает примерное сохранение времени расчёта при одновременном увеличении ч.с.с. и количества вычислительных устройств, а сильное – пропорциональное уменьшение времени решения фиксированной задачи с ростом числа вычислителей.

В идеале решение задачи автоматизации и оптимизации распараллеливания алгоритмов хотелось бы искать путём имитационного моделирования компьютерной системы в целом, однако это слишком сложно, поэтому приходится привлекать полуэмпирические приёмы или простейшие модели машинных вычислений. Примерами характеристик распараллеливания могут служить две величины – коэффициенты ускорения вычислений и эффективности использования процессоров:

$$S_p = T_1/T_p, \quad E_p = S_p/P,$$

где T_p – время выполнения задачи или алгоритма на p процессорах. Идеальной ситуацией является такая, когда значение S_p прямо пропорционально, а $E_p = 1$, но на практике зачастую приходится довольствоваться коэффициентами эффективности в несколько процентов.

Развитие суперкомпьютерных технологий идёт в двух основных направлениях: высокопроизводительные вычисления (HPC – High Performance Computing) и работа с большими данными (Big Data), причём в последние годы наблюдается конвергенция этих двух тенденций (Intensive Data Computing). В целом эволюция поколений МВС и экстремальных задач моделирования сопровождается примерно одинаковым ростом быстродействия и объёма оперативной памяти (количество тера- или петафлопсов не сильно отличается от числа тера- или петабайтов компьютера).

Главная цель при программировании параллельных алгоритмов заключается в минимизации информационных обменов, поскольку общее время решения задачи T равно сумме двух слагаемых:

$$T_a = N_a\tau_a, \quad T_c = M(\tau_0 + N_c\tau_c),$$

где τ_a и τ_c – усреднённые времена выполнения одной арифметической операции и передачи одного числа, N_a – количество арифметических действий, M – число обращений к памяти, τ_0 – время задержки (настройки) обменной операции, а N_c – средний объём одного передаваемого массива данных. При этом надо иметь в виду характерное соотношение $\tau_0 \gg \tau_c \gg \tau_a$. Требование к уменьшению передачи данных вызвано не только необходимостью повышения быстродействия, но и энергозатратностью коммуникационных операций.

Из сказанного становится ясно, что для больших задач меняется понятие качества алгоритмов: из двух сравниваемых методов лучше не тот, который имеет меньший объём вычислений, а тот, который реализуется быстрее на МВС рассматриваемого типа. Другими словами, появляется новая концепция оптимизации вычислений, основанная на поиске подходов, пусть даже увеличивающих количество арифметических действий, но существенно снижающих объём передачи данных между процессами.

Термин “большая задача” нуждается в формальном уточнении. В данном случае подразумевается проблема, требующая для своего решения достаточно длительного машинного времени. В определённом смысле это понятие является инвариантом по отношению к поколению компьютера. Также нужно конкретизировать такую употребляемую категорию, как “экстремальное моделирование”, то есть решение больших (или сверхбольших) задач на суперкомпьютерах, к которым можно отнести, например, те МВС, которые по основным мощностным показателям в 10 раз уступают последнему компьютеру из текущего мирового списка TOP-500.

Говоря об интенсивных вычислениях, нельзя не коснуться фундаментальной проблемы устойчивости к машинным округлениям и к ошибкам исходных данных. Ограничусь констатацией факта, что для решения очень “плохих” или плохо обусловленных (так называемых жёстких, некорректных) задач требуются особые условия контроля точности результатов, а зачастую и поиск специальных алгоритмов. За вычетом подобных исключительных ситуаций получаемые расчётные погрешности определяются длиной мантиссы в представлении чисел с плавающей запятой. Сейчас для задач средней и повышенной сложности общепринятым считается использование арифметических операций “с двойной точностью” (double precision), что соответствует машинному представлению вещественных чисел с помощью 64 битов. Это, с одной стороны, значительно упрощает оптимизацию алгоритмов, с другой – вопросы, связанные с оптимизацией, в действительности просто выносятся за скобки.

Идеальным решением стало бы использование переменной длины машинного слова, но такая стратегия рассматривается сегодня как относящаяся к отдалённой перспективе.

Последнее, на что следует обратить внимание, – необходимость преодоления психологического комплекса при отсутствии "под рукой" суперкомпьютера. Современные облачные технологии делают достаточным условием только наличие интернет-доступа к вычислительному центру коллективного пользования (ВЦКП, или Data Center). Разумеется, для интеллектуализации пользовательского интерфейса желательно иметь специализированные средства автоматизированного рабочего места, но это уже тема отдельного обсуждения.

Особенности распараллеливания технологических стадий. Тактика распараллеливания на каждом вычислительном этапе обусловливается объёмом обрабатываемых данных и количеством операций. Стадия геометрического и функционального моделирования, несущая существенную интеллектуальную нагрузку и определяющая входной интерфейс пользователя, имеет дело с макрообъектами, которых не должно быть слишком много (десятки, сотни, в крайнем случае тысячи). Поэтому представляется целесообразным обойтись совсем без обменов, копируя вычисления во всех MPI-процессах и сохраняя в них же получаемые геометрические и функциональные структуры данных.

Генерацию сетки формально допустимо представить как преобразование данных: ГСД + ФСД $\rightarrow$ ССД, причём сеточная структура данных для всей расчётной области может занимать большой объём. В силу этого естественно создавать каждым MPI-процессом ССД для "своей" сеточной подобласти (с некоторым перекрытием). Образование распределённых данных на начальном этапе тем более оправданно, что декомпозиция областей является основным средством распараллеливания. Однако, поскольку расчётная сеточная область должна быть также определена как целостный объект, все её узлы и другие элементарные объекты (рёбра, грани, ячейки) должны быть дважды пронумерованы – локально по подобластям и глобально. В задачах декомпозиции возможны две тактики: построение подобластей, предваряющее генерацию сеток (естественно, например, разделять среды с контрастно отличающимися материальными свойствами), или непосредственное формирование сеточных подобластей. Необходимо также иметь в виду, что многие эффективные алгоритмы основаны на специальных переупорядочениях компонент (про эту задачу можно говорить и в терминах графов), и все соответствующие процедуры должны быть доступны всем MPI-процессам, или подобластям, что в целом существенно сократит информационные обмены. В качестве эффективного инструмента переупорядочивания назовём популярные пакеты программ METIS и PARMETIS.

Наиболее сложными для расчётов являются задачи с движущимися границами, поскольку при этом адаптивные сетки являются ещё и динамически перестраиваемыми. Многие экономичные методы основаны на локальных сгущениях и многосеточных подходах, инструментальная поддержка которых также должна осуществляться распределённым образом.

После получения распределённых информационных массивов, отображаемых в ССД, ГСД и ФСД, параллельным образом может быть реализована аппроксимация исходной задачи. В МКЭ и МКО для этого существует уникальная технология вычисления локальных матриц и сборки (assembling) глобальной матрицы. Поскольку "аппроксиматор" работает параллельно по подобластям, или MPI-процессам, с уже распределёнными необходимыми данными, получаемые в результате матрично-векторные структуры должны находиться в своей подобласти, поэтому данная стадия идеально реализуется при отсутствии обменов. А выполнение главных операций по ячейкам сетки независимо друг от друга позволяет эффективно распараллеливать их с помощью многопотоковых вычислений. В нестационарных задачах, а также в нелинейных или оптимизационных расчётах аппроксимации приходится делать многократно, но с точки зрения адаптации к вычислительным устройствам это ничего, как правило, не меняет.

Важнейшим промежуточным элементом при решении алгебраических задач являются линейные системы, поэтому на них остановимся подробнее. Особое внимание стоит уделить очень большим СЛАУ с разреженными матрицами, возникающими после аппроксимации с помощью МКЭ или МКО дифференциальных или соответствующих вариационных многомерных задач на неструктурированных сетках. С точки зрения классификации алгоритмов решаемые СЛАУ можно разбить на два основных класса – специальные и общего вида. Для первого, к которому относятся системы, возникающие в краевых задачах с разделяющимися переменными, существуют сверхбыстрые прямые и/или итерационные решатели типа быстрого преобразования Фурье или неявные методы переменных направлений с оптимальными наборами итерационных параметров [10], которые в последнее десятилетие востребованы благодаря актуальным матричным уравнениям Ляпунова и Сильвестра.

Прямые методы для больших разреженных СЛАУ общего вида активно совершенствуются, однако даже в наиболее продвинутых версиях широко распространённых программ Pardiso [9] и MUMPS они имеют ограниченную примени-

мость, главным образом из-за требований к объёму оперативной памяти. Основным орудием высокопроизводительного параллельного решения СЛАУ этого типа являются итерационные аддитивные методы декомпозиции областей (МДО, или DDM – Domain Decomposition Methods), по которым имеется огромная специальная литература (см., например, обзор в [14]) и уже прошли 23 крупные международные конференции. Суть этих методов декомпозиции заключается в разбиении сеточной расчётной области на подобласти с параметризованными пересечениями (в частном случае – без пересечений), на внутренних границах которых ставятся какие-то интерфейсные краевые условия, определяющие информационные взаимосвязи между смежными подобластями. В простейшем случае итерации формируются по блочному методу Якоби, заключающемуся в одновременном решении вспомогательных СЛАУ в подобластях и с организацией обмена данными между ними. Для ускорения процесса в первую очередь применяются оптимальные алгоритмы в подпространствах Крылова. В целях дальнейшего повышения быстродействия используются различные двухуровневые или многоуровневые подходы: дефляция, агрегация, грубосеточная коррекция, малоранговая матричная аппроксимация и др.

Для достижения масштабируемого распараллеливания реализуются технологии гибридного программирования: формируются MPI-процессы над распределённой по вычислительным узлам памятью – по одному на каждую из подобластей, внутри которых осуществляются многопотоковые вычисления средствами OpenMP на общей памяти. При этом существенного ускорения можно добиться, если совместить межпроцессорные обмены с синхронным выполнением арифметических операций в подобластях. Отдельную проблему представляет эффективное использование универсальных графических ускорителей с огромным количеством вычислительных ядер, но относительно медленными коммуникациями (GPGPU – General Purpose Graphic Processor Units), а также перспективных программируемых логических интегральных схем (ПЛИС, или FPGA – Free Programmable Gate Array).

Адаптация современных методов декомпозиции к имеющимся компьютерным платформам составляет, прибегая к философско-методологической терминологии, задачу отображения алгоритмов на архитектуру МВС. Это фундаментальное по своей значимости научное направление в значительной степени является экспериментальным, и только многочисленные сопоставления измерений реальной производительности могут послужить основой для выработки практических рекомендаций по решению классов задач.

Постпроцессинг и визуализация результатов расчёта – самое благодатное для распараллеливания поле деятельности. При кажущейся математической простоте данная технологическая стадия является ключевой для успешности больших проектов по моделированию. Качественная цветная графика, особенно с динамическими сценариями и многократным контролем промежуточных данных, требует значительных компьютерных ресурсов и в крупномасштабном вычислительном эксперименте может занимать львиную долю машинного времени. А поскольку одним из главных требований к качеству визуализации является быстрота генерации изображений, то естественным техническим решением становится использование высокоскоростных графических процессоров GPU (Graphics Processing Unit). Важная особенность визуализации заключается в том, что итоговые многомерные векторные поля, которые надо наглядно представить пользователю, распределены по блокам иерархической памяти различных процессоров. Другое обстоятельство связано с наличием большого количества профессиональных продуктов графического назначения (Visual Studio, Open GL и т.д.), и одна из главных проблем разработчиков системы моделирования заключается в их эффективном переиспользовании.

Оптимизационные методы и управление вычислительным экспериментом с точки зрения крупномасштабного распараллеливания представляют собой надстройку над ресурсоёмкими расчётными этапами, и здесь особых проблем не предвидится, хотя принимаемые решения на верхнем блочном уровне играют существенную роль в достижении итоговой высокой производительности.

КОМПОНЕНТНАЯ АРХИТЕКТУРА ИНТЕГРИРОВАННОГО ОКРУЖЕНИЯ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

Как подчёркивается в дорожной карте Международного комитета Джека Донгарры (IESP – International Exascale Software Project), появление к 2020 г. экзафлопсных суперкомпьютеров с сотнями миллионов вычислительных ядер требует создания новой парадигмы программирования и масштабных разработок, возможных только при широкой и хорошо скоординированной интернациональной кооперации [15]. В области математического моделирования этот вызов требует формирования и реализации принципов построения ППО нового поколения, а именно – создания интегрированного инструментального окружения, поддерживающего все стадии наукоёмкого эксперимента и служащего основой оперативных разработок пакетов программ для конкретных приложений и пользователей. Такой методо-проблемно-ориентированный комплекс должен составить но-

вое поколение ППО, создаваемого многочисленными группами разработчиков на основе согласованных технологий и структур данных. Крупноблочная конфигурация этого глобализированного проекта, который можно назвать базовой системой моделирования (БСМ), определяется вычислительными стадиями, рассмотренными выше с функциональной точки зрения.

Концепция БСМ, а также принципы построения основных инструментальных блоков рассмотрены в работах [7, 16–20]. Данный проект основан на следующих положениях:

• БСМ создаётся как богатый и даже избыточный, но никак не минимальный набор вычислительных методов для решения широкого круга прикладных проблем, в том числе обратных и междисциплинарных, с возможностями гибкого выбора и оптимизации алгоритмов на основе анализа свойств конкретной решаемой задачи;

• состав алгоритмов и математических моделей является расширяемым и оперативно пополняемым, в том числе за счёт использования внешних программных продуктов;

• программные реализации модулей адаптируются к эволюции компьютерных архитектур и платформ, открывая перспективы длительного жизненного цикла БСМ;

• обеспечение внешних и внутренних программных интерфейсов, а также взаимодействия с пользователями осуществляется на основе множественного представления структур данных при активном использовании их конвертации и специализированных языков;

• алгоритмы реализуются с ориентацией на массивное масштабируемое распараллеливание, без программных ограничений на число степеней свободы задачи и количество используемых процессоров и/или ядер, на типичные технологии решения больших задач – облачные вычисления на суперкомпьютерных ВЦКП.

Что касается архитектуры БСМ, то каждый из её структурных, или этапных, блоков фактически представляет собой расширяемую библиотеку алгоритмов и программ, другими словами, интегрированную и достаточно автономную среду для соответствующей методо-экологической ниши. Подчеркну, что блоки конструируются на принципах множественности как структур данных, так и алгоритмов для решения частных подзадач. Например, такое многообразие может включать многоверсионность реализации на разных языках программирования или для различных вычислительных устройств (центральных или графических процессоров). Рассматриваемые функционально-ориентированные блоки могут разрабатываться практически независимо друг от друга (и даже эксплуатироваться как самостоятельный продукт в других производственных условиях), а взаимодействие между ними определяется только интерфейсными структурами данных (ГСД, ССД, АСД и т.д.).

Широко используемые уже несколько десятилетий компонентные технологии, такие как CORBA (Common Object Request Broker Architecture) [21] и COM/DCOM (Distributed Component Object Management) [22], в определённом смысле представляют собой развитие объектно-ориентированного программирования, но в них не получает должной реализации обеспечение высокой производительности наукоёмких вычислений с масштабируемым параллелизмом, сложными типами данных и разноязычными программными модулями. С целью преодоления этих ограничений в 1997 г. был организован Форум по проблемам общей компонентной архитектуры (Common Component Architecture (CCA) Forum) [23], поставивший задачей определение основных стандартов, инструментов и технологий. Департамент энергетики США создал Центр компонентных технологий для программного обеспечения терамасштабного моделирования (CCTTSS – Center for Component Technology for Terascale Simulation Software), который совместно с ведущими национальными лабораториями создал ряд инструментальных средств в рамках спецификаций CCA [24]. Среди таких разработок можно выделить язык описания компонентных интерфейсов SIDL (Scientific Interface Definition Language) [25], являющийся обобщением языка IDL из проекта CORBA и послуживший основой поддержки многоязыковости, включая F77, F90/95, C, C++, Python, реализованный в группе Babel [26], а также инструментальный комплекс CCAFFEINE [26] для осуществления технологических операций в иерархической распределённой памяти. Сегодня уже имеется ряд публикаций по результатам успешного использования компонентных технологий для серьёзных приложений. Примерами могут служить модернизация в методологии CCA пакета программ SPARSKIT [27] для решения задач линейной алгебры и разработанные в национальной лаборатории SANDIA (США) интерфейсные стандарты для решения уравнений [28].

В целом идеи компонентных архитектур перекликаются с другими технологиями программирования – модульного, сборочного, фрагментарного. В последние годы появилось направление сервис-ориентированных архитектур (SOA – Service Oriented Architecture), также предполагающее кардинальное повышение уровня автоматизации (а следовательно, и производительности) труда программистов. В значительной степени это связано с сокращением сроков трудоёмкого процесса отладки программ, включающего многократные пропуски тестовых задач и поиск неизбежных технических ошибок. Разработчику не-

обходимо кодировать алгоритмический модуль только по заданным требованиям к представлению входной и выходной информации, а система программирования должна обеспечить его корректное исполнение в составе всего комплекса, независимо от того, в какой операционной системе и на каком вычислительном устройстве будет выполняться сам запуск (свойство кросс-платформенности).

Свойство многоязыковости здесь не менее важно, и этому есть две причины: первая – обеспечение возможности переиспользования в БСМ сторонних модулей, вторая обусловлена тенденцией создания специализированных языков для быстрого написания алгоритмов определённых классов. Такие естественные или функциональные языки с широким использованием не только императивных (командных) стилей, но и декларативных средств значительно повышают интеллектуальность взаимодействия человека с компьютером и имеют конечной целью рост эффективности моделирования. Заманчивой остаётся перспектива автоматизации распараллеливания алгоритмов на языковом уровне, однако многочисленные попытки пока не привели к разработке производственного продукта.

* * *

В силу возрастания сложности задач в области фундаментальных исследований, а также прорывных производственных и социальных технологий глобализация математического моделирования на суперкомпьютерах становится одним из главных требований научно-технического прогресса. Однако следует, во-первых, помнить, что его широкое внедрение зависит от решения большого числа производственных, организационных и кадровых вопросов [29], а во-вторых, чётко различать, какие проблемы относятся собственно к моделированию, а какие – к дисциплинам или отраслям, где оно применяется. Можно провести следующую аналогию: математик изучает математические объекты, а физик-теоретик активно использует различные формулы и методы для достижения своих целей, но не занимается доказательством теорем. На современном этапе вместо качественного анализа свойств дифференциальных или интегральных уравнений, чему зачастую посвящает своё время физик (хотя реально это возможно только в простейших случаях), он может и должен брать “умный” прикладной программный продукт и быстро получать в наглядной форме необходимые результаты, оперативно исследуя всевозможные варианты. То же самое относится и к инженеру, занимающемуся оптимизацией какого-то проектируемого устройства. Однако даже при наличии совершенного инструмента моделирования искусство его эффективного применения – это уже проблема пользователей, представляющих конкретные предметные области, которая также нуждается в пристальном внимании исследователей.

Работа выполнена при поддержке гранта Российского научного фонда № 15-11-10024.

ЛИТЕРАТУРА

1. ANSYS – Simulation Driven Product Development // www.ansys.com (дата обращения 10.01.2015).
2. MSC Nastran – Industry Leading Multidisciplinary FEA // http://www.mscsoftware.com/product/msc-nastran (дата обращения 11.02.2015).
3. *Schoberl J.* Netgen – an advancing front 2D/3D-mesh generator based on abstract rules // Computing and Visualization in Science. 1997. V. 1. P. 41–52.
4. PETSc // http://www.mcs.anl.gov/petsc/ (дата обращения 5.02.2015).
5. OpenFOAM – The Open Source Computational Fluid Dynamics (CFD) Toolbox // http://www.openfoam.com (дата обращения 01.04.2015).
6. DUNE Numerics. Distributed Unified Numerical Envirenment // http://www.dune-project.org (дата обращения 15.01.2015).
7. *Ильин В.П., Скопин И.Н.* Технологии вычислительного программирования // Программирование. 2011. № 4. С. 53–72.
8. *Kleppe A.* Software Language Engineering: Creating Domain-Specific Language Using Metamodels. N.Y.: Addison-Wesley, 2008.
9. Intel ® Mathematical Kernel Library from Intel // http://software.intel.com/en-us/intel-mkl (дата обращения 2.03.2015).
10. *Ильин В.П.* Методы конечных разностей и конечных объёмов для эллиптических уравнений. Новосибирск: Изд-во ИВМиМГ СО РАН, 2001.
11. *Ильин В.П.* Методы и технологии конечных элементов. Новосибирск: Изд-во ИВМиМГ СО РАН, 2007.
12. *Butyugin D.S., Il'in V.P.* Solution of problems of harmonic electromagnetic field simulation in regularized and mixed formulations // RJNAMM. 2014. V. 29. P. 1–12.
13. *Ильин В.П.* О численном решении прямых и обратных задач электромагнитной электроразведки // Сибирский журнал вычислительной математики. 2003. Т. 6. С. 381–394.
14. *Ильин В.П.* Параллельные процессы на этапах петафлопного моделирования // Вычислительные методы и программирование. 2011. Т. 12. С. 93–99.
15. IESP: International Exascale Software Project // http://www.exascale.org/iesp (дата обращения 2.03.2015).
16. *Голубева Л.А., Ильин В.П., Козырев А.Н.* О программных технологиях в геометрических аспектах математического моделирования // Вестник НГУ. Серия “Информационные технологии”. 2012. Т. 10. С. 25–33.

17. *Ильин В.П.* DELAUNAY: технологическая среда генерации сеток // СибЖИМ. 2013. Т. 16. С. 83–97.
18. *Бутюгин Д.С., Ильин В.П.* Chebyshev: принципы автоматизации построения алгоритмов в интегрированной среде для сеточных аппроксимаций начально-краевых задач // Труды Международной конференции ПАВТ-2014. Челябинск: Изд-во ЮУрГУ, 2014. С. 42–50.
19. *Бутюгин Д.С., Гурьева Я.Л., Ильин В.П. и др.* Функциональность и технологии алгебраических решателей в библиотеке Krylov // Вестник ЮУрГУ. Серия "Вычислительная математика и информатика". 2013. Т. 2. С. 92–105.
20. *Ильин В.П.* О компонентных технологиях высокопроизводительного математического моделирования // Труды Международной конференции ПАВТ-2015. Екатеринбург: УФУ, ИММ УрО РАН, 2015. С. 166–171.
21. Object Management Group. "CORBA Components" // http:// www.omg.org (дата обращения 10.02.2015).
22. *Maloney J.* Distributed COM Application Development Using Visual C++. N.Y.: Prentice Hall, 1999.
23. CCA-Forum. The DOE common component architecture project // http://www.CCA-forum.org (дата обращения 10.02.2015).
24. CCTTSS. DOE SciDAC Center for Component Technology for Terascale Simulation Software // http:// www.cca-forum.org/ccttss (дата обращения 10.03.2015).
25. *Kohn S., Kumfert G., Painter J., Ribben C.* Divorcing language dependencies from a scientific software library // http://computation.llnl. gov/casc/components/docs/2001-stat-pp.pdf (дата обращения 15.10.2014).
26. Babel Team. The DOE Babel Project // http://www.llnl.gov/case/components/babel (дата обращения 10.04.2015).
27. *Jones J., Sosonkina M., Saad Y.* Component-based iterative methods for sparse linear systems // Concurrency and Computation. Practice and Experience. 2007. V. 19. P. 625–635.
28. *Allan B., Armstrong R., Wolfe A. et al.* The CCA core specification in a distributed memory SPMD framework // Concurrency Computation. Practice and Experience. 2002. V. 14. P. 323–345.
29. *Il'in V.P.* Computational Mathematics and Informatics: Global Challenges and Russia's Roadmap // Herald of the Russian Academy of Sciences. 2015. V. 85. № 1. P. 8–14; *Ильин В.П.* Вычислительная математика и информатика: мировые вызовы и российская "дорожная карта" // Вестник РАН. 2015. № 2. С. 107–114.