О компонентных технологиях высокопроизводительного математического моделирования *

В.П.Ильин[1,2]

Институт вычислительной математики и математической геофизики СО РАН[1],
Новосибирский государственный университет[2]

Последние полтора десятилетия характеризуются значительным повышениям внимания к общим компонентным архитектурам наукоемкого программного обеспечения для высокопроизводительного моделирования на многопроцессорных вычислительных системах. Появление прямых и обратных междисциплинарных суперзадач неизбежно ведет к драматическому росту объемов неуправляемых программных кодов, которые становятся трудно поддерживаемыми и несовместимыми. Мы рассматриваем подходы к решению возникающих проблем в применении к базовой системе моделирования, ориентированной на интегрированную поддержку всех основных этапов крупномасштабного вычислительного эксперимента.

1. Введение

Как отмечалось в [1], [2] и цитируемых там работах, характерная тенденция развития современного прикладного и программного обеспечения заключается в его "глобализации", т.е. в переходе от разработок пакетов прикладных программ (ППП, например, ANSYS [3]) или библиотек, ориентированных на конкретные типы задач или представляющих собой наборы вспомогательных алгоритмических инструментов, к интегрированным программным окружениям для решения широкого класса прикладных проблем, поддерживающим все основные стадии математического моделирования и рассчитанным на длительный жизненный цикл с непрерывным пополнением состава моделей, методов и технологий, на адаптацию к эволюции многопроцессорных вычислительных систем (МВС), а также на скоординированную реализацию различными группами разработчиков. Примерами таких проектов являются FOAM [4], DUNE [5] и базовая система моделирования БСМ [1].

С прикладной, или предметной, точки зрения, особенностью проблемы является, с одной стороны, нацеленность на очень широкий класс прямых и обратных междисциплинарных задач, а с другой – избыточное многообразие включаемых в функциональное наполнение вычислительных методов и технологий, что позволяет выбрать в конкретном приложении наиболее подходящий или даже оптимальный алгоритм, устраняя или хотя бы ослабляя неизбежный конфликт между универсальностью и эффективностью.

Особенностью таких разработок является очень большой объем программных кодов и их сложная внутренняя структура, многоверсионность, а также необходимость многократных реконфигураций, гибкого управления вычислительным процессом и поддержки как межмодульных интерфейсов, так и взаимодействия с внешними продуктами. Очевидно, что такие обстоятельства требуют неизбежных временных издержек, но при этом нужно сохранить достаточно высокую производительность алгоритмов и программ на гетерогенных МВС, со сложной иерархией общей и распределенной памяти.

Существующие уже несколько десятилетий широко используемые компонентные технологии, такие как CORBA (Common Object Request Broker Architecture, [6]) и COM/DCOM [7] (Distributed Component Object Management), которые в определенном смысле представляют развитие объектно-ориентированного программирования, но в силу своих историче-

*Работа поддержана грантом Российского научного фонда № 14-11-00485.

ских рамок, недостаточно акцентированы на обеспечение высокой производительности наукоемких вычислений с масштабируемым параллелизмом, сложными типами данных и разноязычными программными модулями. С целью преодоления этих ограничений в 1997 г. был организован Common Component Architecture (CCA) Forum [8], поставивший задачей определение основных стандартов, инструментов и технологий. Департамент энергетики (DOE) в США создал Центр компонетных технологий для программного обеспечения терамасштабного моделирования (Center for Component Technology for Terascale Simulation Software – CCTTSS, [9]), который совместно с ведущими Национальными лабораториями создали ряд инструментальных средств в рамках спецификаций CCA. Среди таких разработок можно отметить язык описания компонентных интерфейсов SIDL (Scientific Interface Definition Language, [10]), являющейся обобщением языка IDL из проекта CORBA и послуживший основой поддержки многоязыковости, включая F77, F90/95, C, C++, Python, реализованный в группе Babel [11], а также инструментальный комплекс CCAFFEINE [12] для осуществления технологических операций в иерархической распределенной памяти. Имееется уже ряд публикаций по результатам успешного использования компонентных технологий для серьезных приложений. Таким примером может служить модернизация в методологии CCA пакета программ SPARSKIT [13] для решения задач линейной алгебры, а также разработанные в национальной лаборатории SANDIA (США) интерфейсные стандарты для проблем решения уравнений [14].

Целью данной работы является рассмотрение подходов к применению компонентных технологий для построения базовой системы моделирования [1], ориентированной на методологическую поддержку различного типа прямых и обратных междисциплинарных задач. В п. 2 мы анализируем структурные свойства ядра БСМ и ее основных функциональных блоков, а в п. 3 формулируем характерные методологические требования к системному наполнению с точки зрения конечной эффективности и производительности создаваемого интегрируемого программного окружения.

2. Структурные особенности ядра БСМ

Несмотря на бесчисленное разнообразие возможных вычислительных проблем и алгоритмов, в целом функциональное наполнение математического и программного обеспечения задач наукоемкого экстремального моделирования формально можно представить следующим единообразным, неформальным определением.

Данная категория включает пространственно-временные процессы и явления, описываемые начально-краевыми задачами для систем дифференциальных, интегральных и/или дискретных уравнений, а также соответствующими обобщенными или вариационными постановками. Если требуется найти решение при полностью заданных граничных и начальных условиях в расчетной области с известными материальными свойствами сред, то такая задача относится к классу прямых задач. Более сложными являются обратные задачи, в которых любые из исходных данных могут быть определены с помощью неизвестных параметров, оптимальные значения которых надо найти по условиям минимизации задаваемого целевого функционала от решения, при дополнительных линейных или нелинейных ограничениях, которые также могут описываться какими-то функционалами. Решение обратных задач подразумевает в общем случае многократное нахождение решений прямых задач, при направленном поиске необходимых значений параметров с помощью методов оптимизации. Здесь, в свою очередь, требуется разделить задачи локальной минимизации (если соответствующий изолированный в некоторой окрестности параметров минимум существует) и проблемы глобальной минимизации, т.е. нахождение всех экстремальных точек.

Под наукоемкими и экстремальными можно понимать задачи, требующие для своего понимания большого объема математических и других знаний, решение которых описывается громоздкими формулами высокой логической сложности и требующими для своей

реализации сверхбольших вычислительных ресурсов (в отношении существующего на текущий момент парка компьютеров). К задачам повышенной вычислительной сложности относятся междисциплинарные и разномасштабные проблемы с контрастными материальными свойствами, в которых достижение гарантированной точности расчетов требует огромного числа степеней свободы неизвестных (порядка $10^8 - 10^{10}$).

Функциональное наполнение вычислительного ядра БСМ, в соответствии с основными стадиями математического моделирования, составляется из достаточно автономных блоков, каждый из которых представляет собой расширяемую подсистему, взаимодействующую с остальными через гибкие согласованные, допускающие множественные представления со взаимной конвертацией, структуры данных. В целом рассматриваемые этапы и архитектура ядра могут быть описаны следующим образом.

- Геометрическое и функциональное моделирование, определяющее уровень автоматизации построения математических моделей и интеллектуальности пользовательского интерфейса, а также взаимодействие с внешними, графическими и САПРовскими продуктами. Итоговая информация обо всех математических объектах, а также об их взаимосвязях, однозначно определяющая постановку исходной задачи, представляется геометрической и функциональной структурами данных (ГСД и ФСД), обеспечивающими внутренние интерфейсы с остальными компонентами ядра БСМ и взаимную конвертацию форматов, используемых внешними программными разработками.

- Генерация адаптивных неструктурированных многомерных сеток в областях со сложной конфигурацией кусочно-гладких границ, поддерживающая разнообразные типы конечных элементов, контроль качества, современные многосеточные и декомпозиционные подходы и являющаяся ключевой технологией для успешного вычислительного эксперимента в реальных задачах со сложными конфигурациями областей и контрастными свойствами материальных сред. Подчеркнем, что на данной стадии допускается применение как собственных, так и внешних генераторов сеток, а результирующая информация представляется сеточной структурой данных, которая в совокупности с ГСД и ФСД отображает свойства решаемой задачи уже на дискретном уровне.

- Аппроксимация решаемых систем дифференциальных и/или интегральных уравнений – наиболее теоретизированная стадия, отвечающая за точность численных результатов и представляющая бурно развивающуюся область математики, кардинально программистское решение которой заключается в реализации машинных символьных вычислений, что значительно повышает уровень автоматизации построения алгоритмов и имеет большое значение для повышения производительности программного труда, особенно при реализации актуальных методов высокой точности. Конечным результатом данной стадии является алгебраическая структура данных.

- Численное решение сверхбольших систем получаемых линейных и нелинейных алгебраических уравнений – наиболее ресурсоемкий этап, поскольку здесь трудоемкость алгоритмов сильно увеличивается с ростом порядка. В данном случае для обеспечения масштабируемого распараллеливания расчетов требуется эффективное отображение структуры алгоритмов на архитектуру МВС, предполагающее тонкое знание особенностей работы арифметических и коммуникационных устройств.

- Оптимизационные алгоритмы условной минимизации функционалов для решения обратных задач. Эта область вычислительной математики претерпела революционные изменения в последние десятилетия, и инновационные программистские решения здесь особенно важны, поскольку каждый эксперимент связан с решениями множества прямых задач.

- Постобработка и визуализация результатов отвечает за обратную связь компьютера с человеком, важность которой трудно переоценить. При необходимости качественного графического анализа данный этап является экстремально ресурсоемким и требует массивного параллелизма. К счастью, здесь успеха решения зачастую можно достичь путем грамотного использования существующих профессиональных графических систем.

  Перечисленные пункты можно дополнить подсистемой принятия решений по результатам вычислительного эксперимента, что является венцом математического моделирования и должно определять его практическую востребованность.

Важно отметить, что каждый из рассмотренных блоков ядра является формально методо-ориентированным и проблемно-независимым в том смысле, что он может применяться для различных типов приложений, т.е. имеет междисциплинарный характер. В силу этого он должен представлять собой мета-алгоритм, или библиотеку программ, для решения однотипных подзадач определенного класса (например, генерации каких-либо сеток, аппроксимации различных видов и порядков для дифференциальных уравнений и т.д.), которые могут применяться к самым разным приложениям, классифицируемым или по физико-математическим признакам (задачи электромагнетизма, упруго-пластичности, гидро-газодинамики и т.д.), или по техническому назначению моделируемых устройств (электроника, машиностроение, гео-электроразведка и др.).

3. Основные требования и принципы компонентной архитектуры БСМ

Вычислительное ядро базовой системы моделирования представляет собой инструментальное окружение, с помощью которого средствами внутреннего конфигурационного управления для конкретного класса задач (и пользователя) формируется приложение, или пакет прикладных программ. Конструируемый ППП может или отторгаться, или эксплуатироваться в составе БСМ. Эти операции требуют соответствующей системной поддержки, и здесь необходимо выделить такую важную методологическую проблему, как автоматизация тестирования задач и алгоритмов, требующая создания презентативного банка пробных примеров.

Другая важная функция системного наполнения — обеспечение внутренних межмодульных интерфейсов, включая проблему поддержки многоязыковости. Связанная с этим, но совершенно независимая проблема — внешние интерфейсы вычислительных инструментариев БСМ с окружающим миром, что подразумевает как кросс-платформенность проекта, так и взаимодействие со сторонними прикладными продуктами, среди которых, в первую очередь, надо иметь разнообразные САПРовские разработки, имеющие многолетний опыт использования в промышленности. Интерфейсные аспекты имеют и третью важную сторону — обеспечение комфортных контактов с пользователями, под которыми надо подразумевать как разработчиков модулей, так и лиц отраслевых профессий, эксплуатирующие ППП, и от успеха которых зависит достижение конечной цели математиками и программистами. Очевидно, что качество решения всех интерфейсных проблем в значительной степени определяет уровень интеллектуальности разработки, а главное направление деятельности в этом направлении — это множественность структур данных и создание достаточно содержательного набора проблемно-ориентированных языков (Domestic Specific Language) для обеспечения взаимной конвертации информационных форматов.

Еще одна функция системного наполнения заключается в расширении и модернизации состава моделей и алгоритмов в ядре БСМ. Здесь существенно разнятся ситуации, когда новый включаемый модуль создан в “родной” инструментальной обстановке или привносится извне. Пополнение и обновление функционального наполнения ядра БСМ – это залог длительного жизненного цикла проекта, необходимого для его внедренческого и практического

успеха. В определенном смысле данная характеристика означает “инновации в квадрате”, поскольку само применение БСМ к решению конкретных практических проблем подразумевает существенный экономический эффект в соответствующей отрасли.

К рассмотренной только что проблематике тесно примыкает другая технологическая задача – адаптация к перманентной эволюции архитектуры компьютерных систем. Последняя может означать как количественное наращивание производственных параметров МВС (количество вычислительных узлов, процессоров и ядер или объемы памяти на разных ступенях иерархии), так и кардинальное изменение структурных элементов компьютера, что происходит, вообще говоря, достаточно редко.

Большой философский смысл всегда имеет проблема антагонизма между универсальностью и экономичностью большой программной системы. В отношении БСМ выбрана единственно правильная, по-видимому, стратегия множественности состава алгоритмов для решения каждой достаточно ресурсоемкой задачи или подзадачи. Именно она позволяет выбрать на каждом конкретном этапе оптимальный или почти оптимальный алгоритм из уже имеющегося состава библиотек БСМ. Конечно, такая концепция существенно увеличивает объем работ по созданию вычислительных модулей, но в стратегическом плане такая игра стоит свеч.

В данной проблеме имеется один очень важный момент. Чем больше решаемая задача и чем сложнее ее структура со множеством подзадач, и чем богаче множество алгоритмов для их решения, тем труднее и дороже становится сама проблема поиска оптимального метода. А поскольку зачастую процесс оптимизации приходится вести почти перебором, то такая процедура может оказаться более трудоемкой, чем само решение исходной задачи каким-то подходящим из общих естественных соображений методом. Истина находится обычно где-то посередине, и палиативным, или соломоновым, решением чаще всего является “квазиоптимизация” метода.

Разработка сверхбольших программных комплексов, каковым планируется быть БСМ, естественным образом предполагает уменьшение общих трудозатрат при выработке общего стиля и технологий работ. Однако, с другой стороны, необходимость широкой кооперации различных групп разработчиков со своими производственными условиями делает обязательным создание развитой системы поддержки компонентных технологий, обеспечивающих многоверсионность и пополняемость состава вычислительных модулей, а также закрывающих проблемы разноязычности, кросс-платформенности и другую техническую “кухню” от математиков-программистов.

Аналогичные вопросы очень актуальны и при формировании интеллектуальных интерфейсов для генерируемых в рамках БСМ пакетов прикладных программ, ориентированных на конечных пользователей с разной профессиональной подготовкой. Здесь особо следует отметить учебные и демонстрационные версии ППП, в том числе для студентов и преподавателей, а также для треннинга и переподготовки специалистов.

Важно еще выделить такую компопненту БСМ, актуальную и для разработчиков алгоритмов, и для конечных пользователей, как систему автоматического тестирования вычислительных методов и технологий, включающую презентативные каталоги пробных задач для каждой из стадий моделирования.

В заключение можно сказать, что рассмотренные вопросы относятся к проблеме реиндустриализации прикладного программного обеспечения, с целью доведения его в идеале до уровня технологичности использования компиляторов и операционных систем. В определенном смысле мы подтверждаем ту истину, что “новое – это хорошо забытое старое”, поскольку данные аспекты активно обсуждались еще в 80-е годы (естественно, на своем уровне развития методологии, см., например, [15]).

Литература

1. Ильин В.П., Скопин И.Н. Технологии вычислительного программирования.//Программирование, 2011. №4. С. 53-72.
2. Ильин В.П. Стратегии и тактики “заоблачного” математического моделирования.// Труды Международной конференции ПАВТ’2014. Челябинск, изд. ЮУрГУ, 2014. С. 99–107.
3. ANSYS – Simulation Driven Product Development: URL: http:// www.ansys.com (дата обращения: 27.10.2014).
4. OpenFOAM – The Open Source computational Fluid Dynamics (CFD) Toolbox: URL: http:// www.openfoam.com (дата обращения: 30.10.2014).
5. DUNE Numerics: URL: http:// www.dune-project.org (дата обращения: 15.11.2014).
6. Object Management Group. “CORBA Components”:URL: http:// www.omg.org (дата обращения: 10.10.2014).
7. Maloney J. Distributed COM Application Development Using Visual C++. Prentice Hall, 1999.
8. CCA–Forum. The DOE common component architecture project:URL: http:// www.CCA-forum.org (дата обращения: 5.09.2014).
9. CCTTSS. DOE SciDAC Center for Component Technology for Terascale Simulation Software: URL: http:// www.cca-forum.org/ccttss (дата обращения: 10.10.2014).
10. Kohn S., Kumfert G., Painter J., Ribben C. Divorcing language dependencies from a scientific software library: URL: http:// computation.llnl.gov/casc/components/docs/2001-stat-pp.pdf (дата обращения: 15.10.2014).
11. Babel Team. The DOE Babel Project: URL: http:// www.llnl.gov/case/components/babel.html (дата обращения: 20.10.2014).
12. Allan B., Armstrong R., Wolfe A., Ray J., Bernholdt D., Kohl J. The CCA core specification in a distributed memory SPMD framework. Concurrency Computat. Practice and Experience, 2002. Vol. 14. P. 323-345.
13. Jones J., Sosonkina M., Saad Y. Component–based iterative methods for sparse linear systems.//Concurency Computat. Pract. Exper, 2007. Vol. 19. P.625-635.
14. The Equation Solver Interface Standards Forum: URL: http:// z.ca.sandia.gov/esi (дата обращения: 31.07.2006).
15. Ершов А.П., Ильин В.П. Пакеты программ – технология решения прикладных задач.–Новосибирск, 1978 (Препринт ВЦ СО АН СССР; № 121).