К 110-летию СПбГУ ИТМО

УДК 681.3.069, 681.324

А. В. Бухановский, В. Н. Васильев

СОВРЕМЕННЫЕ ПРОГРАММНЫЕ КОМПЛЕКСЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ E-SCIENCE

Рассматриваются основные особенности современных программных комплексов компьютерного моделирования в рамках парадигмы e-Science — их архитектура, специфика развертывания на вычислительных платформах, характеристики информационного обеспечения, внедрения и поддержки жизненного цикла.

Ключевые слова: *проблемно-ориентированные оболочки, сервисы, суперкомпьютерные технологии, интеллектуальные технологии, виртуальное сообщество.*

Введение. Современный этап эволюции представлений о программных комплексах компьютерного моделирования и обработки результатов научных экспериментов тесно связан с продвижением парадигмы „электронной науки“ — e-Science [1]. Понятие e-Science ассоциируется с проведением разнородными группами специалистов совместных научных исследований, требующих консолидации вычислительных и программных ресурсов для решения сложных междисциплинарных задач на основе технологий распределенных вычислений и систем. Устоявшийся облик инструментария компьютерного моделирования, ориентированного на персональные компьютеры и вычислительные кластеры монопольного применения, в настоящее время претерпевает существенные изменения, связанные, в первую очередь, с глобализацией сферы его применения, ограниченной, пожалуй, лишь охватом сети Интернет. Это касается особенностей архитектуры программных комплексов, специфики их развертывания на вычислительных платформах, характеристик информационного обеспечения, поддержки жизненного цикла и продвижения на рынке. Данные вопросы рассматриваются ниже.

От интегрированных комплексов — к проблемно-ориентированным средам. Традиционное представление о предметно-ориентированном комплексе компьютерного моделирования предполагает наличие в его составе как минимум трех функциональных составляющих: препроцессора данных, „решателя“ (solver) и постпроцессора данных (часто совмещенного с системой визуализации). Численная реализация заложенных в „решателе“ моделей возможна одним или несколькими методами, выбор которых предопределен разработчиком и тем самым навязан пользователю; способы подготовки данных и обработки результатов расчетов ориентированы на особенности „решателя“.

Альтернативой такому подходу является концепция проблемно-ориентированных сред (Problem Solving Environment, PSE) [2]. Она подразумевает организацию модульной архитектуры программного комплекса, в которой различные предметно-ориентированные модули (аналоги традиционных „решателей“) функционируют в рамках управляющей среды. Задачей среды является обеспечение единого интерфейса взаимодействия модулей, централизованный контроль их исполнения на вычислительных ресурсах, а также управление потоками данных (включая внешние средства их обработки). По сравнению с традиционными ком-

плексами это обеспечивает существенную гибкость, что позволяет пользователю с помощью средств компьютерного моделирования самостоятельно строить и выполнять различные сценарии исходя из собственных представлений о логике организации исследований.

От компонентов — к сервисам. Описание архитектуры проблемно-ориентированной среды требует определения способа интеграции ее модулей, что тесно связано с общемировой тенденцией снижения сложности процессов разработки, тестирования и поддержки программных продуктов на фоне увеличения общей сложности решаемых задач. В частности, это привело к переходу от структурного, объектно-ориентированного и компонентно-ориентированного подходов в инженерии программного обеспечения к специфическим концепциям, таким как CCA (Common Component Architecture), AOA (Aspect Oriented Architecture) и (наиболее общая) — SOA (Service Oriented Architecture). Эти концепции ориентированы на достижение высокого уровня функциональной изоляции компонентов программного комплекса, что позволяет интерпретировать отдельные программные модули в рамках PSE как сервисы (SaaS, Software as a Service) [3].

Сервисы в общем случае могут быть разработаны различными группами авторов, реализованы с использованием разных технологий программирования и функционировать, в общем случае, на различных вычислительных системах. Как следствие, такой подход существенно упрощает внедрение в PSE ранее разработанного инструментария компьютерного моделирования на основе морально устаревших информационных технологий. Объединяя отдельные сервисы в вычислительную цепочку (или более сложный поток заданий), пользователь может строить различные композитные приложения для решения сложных междисциплинарных задач, не сосредоточиваясь на низкоуровневой реализации отдельных модулей.

От суперкомпьютеров — к распределенным вычислительным комплексам. Моделирование сложных явлений и систем в настоящее время тесно связано с использованием суперкомпьютерных технологий. Несмотря на появление относительно недорогих кластерных архитектур и стандартизацию соответствующих интерфейсов параллельного программирования для них, на практике они доступны далеко не каждому пользователю. Развитие концепции Грид первого поколения лишь отчасти способствовало улучшению ситуации [4]. Это связано с тем, что Грид первого поколения ориентирован, в первую очередь, на обеспечение пользовательским приложениям доступа к распределенным вычислительным ресурсам. Вопросы подготовки и настройки программного обеспечения для исполнения в Грид остаются за пользователем; при этом эффективная работа параллельной программы при запуске на целевой системе в Грид также не гарантируется. Кардинальное решение данной проблемы стало возможным лишь с появлением концепции Грид второго поколения. Грид второго поколения ориентирован на консолидацию не столько распределенных вычислительных ресурсов, сколько сервисов — прикладных программ, установленных на вычислительных системах в рамках распределенной среды и поддерживающих единый интерфейс взаимодействия [5]. В силу того что пользователь получает удаленный доступ к сервису как к программно-аппаратному решению, не возникает проблем совместимости внутренних интерфейсов и эффективности параллельного исполнения.

От формальных инструкций — к интеллектуальному взаимодействию сервисов. Интерпретация прикладной программы как сервиса подразумевает описание формальных правил ее использования (например, форматы входных и выходных данных) и знаний, касающихся логики ее применения, исходя из специфики предметной области [6]. К таким знаниям, например, относятся правила, регламентирующие область применения, используемые методы, а также методические (не программные) ограничения.

Необходимость использования технологий, основанных на знаниях, связана с тем, что разработчик сервиса, предоставляемого в распределенной среде, является, по сути, экспертом в вопросах его практического использования. Как следствие, совокупность сервисов и

ассоциированных с ними априорных знаний предметной области, объединенных в рамках PSE, образует распределенную базу знаний. Используя базу знаний (например, заданную в форме продукций), пользователь может получить рекомендации по выбору конкретного сервиса для решения поставленной задачи. При этом акценты интеллектуальной поддержки могут быть смещены как в сторону поиска подходящих сервисов в открытых распределенных системах [7], так и оптимальной настройки сервиса для решения конкретной прикладной задачи [8]. Возможность использования интеллектуальных технологий взаимодействия сервисов является принципиальным преимуществом e-Science, в частности, в силу существенных различий в интерпретации одних и тех же подходов, методов и моделей в рамках различных научных школ.

От профилей приложений — к параметрическим моделям параллельной производительности. Для эффективного использования вычислительных сервисов требуются знания об их параллельной производительности на заданной вычислительной платформе применительно к характеристикам решаемой задачи. Традиционно такие знания основываются на результатах определения профилей вычислительного сервиса, представленных в табличной форме или в виде некоторой аппроксимации. Этот подход правомерен для обоснования качества распараллеливания, однако не является достаточным для отчуждения знаний о производительности, в силу того что в процессе эксплуатации пользовательские задачи могут не соответствовать условиям, в рамках которых строился профиль. Поэтому для представления знаний необходимо использовать параметрические модели параллельной производительности.

Параметрическая модель связывает характеристику производительности (время выполнения, параллельное ускорение, эффективность, реактивность) с характеристиками задачи и параметрами вычислительной системы (количеством и производительностью процессоров и пропускной способностью коммуникационной сети). Несмотря на формальный способ записи параметрическая модель может интерпретироваться как представление экспертного знания, поскольку она представляется непосредственно экспертом — разработчиком сервиса. При этом структура модели отражает компетенции разработчика в области параллельных вычислений и может варьироваться в широких пределах — от модификаций закона Амдала до специфических конструкций, учитывающих стохастические эффекты исполнения программы [9]. При создании композитного приложения из нескольких сервисов его параллельная архитектура может формироваться динамически таким образом, чтобы минимизировать время исполнения на основе знаний о параллельной производительности каждого из сервисов [10].

От „коробочного“ программного обеспечения — к ASP-приложениям. Характерной особенностью традиционных программных комплексов компьютерного моделирования являются высокие трудозатраты на их установку и настройку. Это связано с тем, что большинство отчуждаемых программных решений такого рода сформировались на основе исследовательского инструментария с невысокими эргономическими характеристиками. Проблема полностью устраняется, когда пользователь ориентируется на распределенные композитные приложения, создаваемые на основе отдельных предметно-ориентированных сервисов, которые устанавливаются непосредственно самими разработчиками на выделенных целевых системах и могут даже не существовать в форме „коробочных“ решений. Для распространения таких приложений эффективно использовать бизнес-модель ASP (Application Service Provider), в рамках которой провайдер через Интернет предоставляет пользователю в аренду приложение, функционирующее на его технологической площадке. При этом в качестве таких приложений могут рассматриваться как отдельные сервисы, так и объединяющая их проблемно-ориентированная оболочка в целом.

Использование модели ASP изменяет требования и к самим вычислительным сервисам. Так, для многих задач использование суперкомпьютерных систем традиционной архитектуры далеко не всегда является оправданным в силу весьма низкой параллельной эффективности. Если пренебречь переносимостью программы (тем самым принципиально отказываясь от

традиционного „коробочного“ способа ее распространения), в ряде случаев можно эффективно реализовать ее сервисную версию для исполнения на вычислительных системах реконфигурируемой архитектуры (FPGA) [11]. Стоимость разработки в данном случае будет выше в несколько раз, но этот недостаток окупается за счет последующего владения комплексом в целом.

От сайтов информационной поддержки — к профессиональным виртуальным сообществам в сети Интернет. Модель распространения программного комплекса в сообществе пользователей определяет форму информационного сопровождения его жизненного цикла. Программные комплексы компьютерного моделирования помимо справочной системы-руководства для пользователя необходимо дополнять документацией, связанной с особенностями его применения к задачам предметной области (например, в форме печатного документа или интерактивного справочного портала). Однако ориентация на композитные приложения, созданные на основе сервисов, разработанных разными группами специалистов, делает такой путь неэффективным. Альтернативой ему является организация профессиональных виртуальных сообществ в Интернете, при этом во главу угла ставится не программный продукт или сервис, а ассоциированные с ним специалисты — эксперты, разработчики, пользователи. В отличие от традиционных социальных сетей, эволюция которых происходит за счет процессов самоорганизации и „самомодерации“, в данном случае устойчивое развитие сообщества обеспечивается за счет мотивации пользователей на получение необходимых им сервисов (включая справочную информацию) и установление соответствующих профессиональных связей. Иными словами, возникновение такого сообщества стимулируется доступностью проблемно-ориентированных сред на основе сервисов. Образуемая при этом обратная связь „провоцирует“ дальнейшее развитие инструментария проблемно-ориентированной среды, одновременно формируя рыночную конъюнктуру на предоставляемые услуги и сервисы. Этот аспект является принципиальным с точки зрения вывода на рынок наукоемких инновационных программных продуктов, для которых традиционные способы внедрения являются малоэффективными вследствие кажущегося отсутствия потенциальных потребителей.

Заключение. Характерным примером, в полной мере учитывающим рассмотренные выше особенности, является высокопроизводительный программный комплекс HPC-NASIS для расчета и моделирования свойств наноразмерных структур и наноматериалов [12], опытный образец которого разработан в СПбГУ ИТМО в 2008—2009 гг. Программный комплекс HPC-NASIS предназначен для проведения квантово-механических расчетов *ab initio*, компьютерного моделирования и расчета наноструктур и наноматериалов с заданными свойствами, в различных условиях эксплуатации. Он обеспечивает моделирование электронной структуры и расчет различных характеристик 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.

Архитектурно комплекс HPC-NASIS реализован на основе перспективной концепции iPSE [13] как открытая интеллектуальная проблемно-ориентированная среда, объединяющая распределенные вычислительные сервисы различных разработчиков. В рамках концепции осуществляется интеллектуальная поддержка пользователя на всех технологических этапах выполнения расчетов — от подготовки исходных данных до анализа и обобщения полученных результатов с использованием разнообразных средств человеко-компьютерного взаимодействия. Заложенные в ходе разработки комплекса решения позволяют гибко переносить и адаптировать созданные технологии в другие предметные области.

Работа выполнена при частичной поддержке ФЦП „Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007—2012 гг.“,

проект „Создание высокопроизводительного программного комплекса для квантово-механических расчетов и моделирования наноразмерных структур и комплексов“ и ФЦП „Научные и научно-педагогические кадры инновационной России на 2007—2013 гг.“, проект „Интеллектуальные технологии распределенных вычислений для моделирования сложных систем“.

СПИСОК ЛИТЕРАТУРЫ

1. *Sloot P. M. A., Frenkel D., Vorst H. A. Van der* et al. Computational e-Science: Studying complex systems in silico. A National Coordinated Initiative. White Paper, February 2007. [Electronic resource]: <http://www.science.uva.nl/research/scs/papers/archive/Sloot2007a.pdf>.
2. *Gallopoulos S., Houstis E., Rice J.* Computer as Thinker/Doer: Problem–Solving Environments for Computational Science // IEEE Computational Science and Engineering. 1994.
3. *Cohen S.* Ontology and Taxonomy of Services in a Service-Oriented Architecture // The Architecture Journal. 2007. N 11. P. 30—35.
4. *Foster I.* What is the Grid. A three point checklist. GridToday / July 22, 2002. [Electronic resource]: <http://www.gridtoday.com/02/0722/100136.html>.
5. *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.
6. *Parastadis S.* A Platform for All That We Know: Creating a Knowledge-Driven Research Infrastructure // The Fourth Paradigm. Data-Intensive Scientific Discovery. 2009. P. 165—172.
7. *Дунаев А. В., Ларченко А. В., Бухановский А. В.* Инструментальная оболочка проектирования высокопроизводительных приложений в Грид. Ч. III. Приобретение и формализация знаний // Науч.-технич. вестн. СПбГУ ИТМО. 2008. Вып. 54. С. 46—55.
8. *Васильев В. Н.* и др. Высокопроизводительный программный комплекс моделирования атомно-молекулярных наноразмерных систем // Там же. С. 3—12.
9. *Ковальчук С. В., Бухановский А. В.* Параллельная производительность стохастических алгоритмов // Изв. вузов. Приборостроение. 2008. Т. 51, № 12. С. 7—14.
10. *Ковальчук С. В.* и др. Особенности проектирования высокопроизводительных программных комплексов для моделирования сложных систем // Информационно-управляющие системы. 2008. № 3. С. 10—18.
11. *Каляев И. А., Левин И. И., Семерников Е. А., Шмойлов В. И.* Реконфигурируемые мультиконвейерные вычислительные структуры. Ростов-на-Дону: Изд-во ЮНЦ РАН, 2008. 393 с.
12. *Васильев В. Н.* и др. Ядро высокопроизводительного программного комплекса для квантово-механических расчетов и моделирования наноразмерных атомно-молекулярных систем и комплексов “HPC-NASIS”. Свид-во о гос. регистрации программы для ЭВМ № 20010610161 от 11.01.2010 г.
13. *Бухановский А. В., Ковальчук С. В., Марьин С. В.* Интеллектуальные высокопроизводительные программные комплексы моделирования сложных систем: концепция, архитектура и примеры реализации // Изв. вузов. Приборостроение. 2009. Т. 52, № 10. С. 5—24.

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

Владимир Николаевич Васильев — д-р техн. наук, профессор; ректор Санкт-Петербургского государственного университета информационных технологий, механики и оптики, кафедра компьютерных технологий; зав. кафедрой

Рекомендована кафедрой компьютерных технологий

Поступила в редакцию 15.01.10 г.