Стратегии и тактики "заоблачного" математического моделирования*

В.П.Ильин[1,2]

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

Рассматривается проблема обеспечения суммарной высокой производительности массовых крупномасштабных наукоемких численных экспериментов, выполняемых в рамках коллективной базовой системы моделирования (БСМ) на многопроцессорной вычислительной системе (МВС) или гетерогенной компьютерной сети петафлопного уровня в соответствии с технологиями "облачных" вычислений. Фактически задача сводится к оптимизации расписания потока расчетных заданий (РЗ), каждое из которых характеризуется объемом требуемых арифметических операций, а также информационных обменов между разнородными сегментами иерархической памяти. Структура КБСМ представляется совокупностью программных текстов и информационных массивов, из которых для отдельного РЗ формируется минимально необходимая конфигурация, так что все задания являются независимыми по данным и не требуют информационных обменов между собой. Различные вычислительные процессы классифицируются по типам задач: линейные и нелинейные, стационарные и нестационарные, прямые и обратные, — для которых отображение алгоритмов на архитектуру МВС осуществляется по своим соответствующим принципам и технологиям. Формирование расписаний определяется как проблема условной минимизации общих объемов используемых ресурсов и коммуникационных потерь МВС при наличии ограничений на выполнение отдельных приоритетных задач.

1. Введение

С точки зрения математического моделирования, требующего выполнения крупномасштабных вычислительных экспериментов при решении прямых и обратных междисциплинарных задач, ориентированных на достижение качественно нового уровня получения принципиальных фундаментальных знаний и производственных решений, в суперкомпьютерном развитии следует отметить две главные тенденции. С одной стороны — это концентрация экстремальных ресурсов в рамках центров коллективного пользования (Data Centers, [1]) пост–петафлопных масштабов с миллионами разнородных компонент, включая многоядерные центральные и графические процессорные устройства (CPU, GPU), программируемые логические интегральные схемы (ПЛИС), а также вычислительные сети (GRID), объединяющие удаленные расчетные мощности.

Другая тенденция — это формирование математического и программного обеспечения в рамках многопользовательской вычислительной интегрированной среды, поддерживающей основные технологические этапы реализации наукоемких проблем: геометрическое и функциональное моделирование, генерация сеток, аппроксимация и дискретизация исходных уравнений, решение алгебраических задач, оптимизационные подходы к решению обратных задач, пост–обработка и визуализация результатов, средства принятия решений по итогам расчетов, см. [2–4]. В некотором смысле примерами программных прототипов могут служить такие комплексы, как OpenFOAM [5] и DUNE (Distributed Unified Numerical Envirenment [6]), или более узко направленные OpenCascade [7] и Salome [8] .

Мотивации такого пути развития заключаются в следующем:

- наличие общих эволюционных направлений создания открытых программных систем

*Работа поддержана грантом РФФИ № 14-07-00128, а также грантами Президиума РАН № 15.9-4 и ОМН РАН № 1.3.3-4.

(Open Source), в соответствии с разработанной международным комитетом под руководством Дж.Донгарры “дорожной карты” экзавычислений [9], а также принципов открытых инноваций, см. [10, 11];

- несоответствие современным условиям суперкомпьютерного развития широко распространенных традиционных проблемно–ориентированных пакетов прикладных программ (ППП) типа ANSYS [12], NASTRAN [13] и многих других, имеющих историю развития еще с прошлого века, а также узкая область применимости различных алгоритмических библиотек (алгебраических, графических и т.д.);

- насущная потребность создания вычислительных интегрируемых программных окружений с длительным жизненным циклом, адаптируемых к эволюции компьютерных платформ, легко развиваемым для новых математических моделей и алгоритмов, а также скрывающим от конечных пользователей излишние технические детали путем предоставления дружественных интеллектуальных интерфейсов; фактически здесь идет речь о резком повышении интеллектуальности прикладного программного обеспечения (ППО), связанной с автоматизацией построения математических моделей и алгоритмов или, образно говоря, о переходе от палео-информатики к нео-информатике [14].

Целью разработки таких инструментальных сред является создание предпосылок для внедрения массового применения математического моделирования путем оперативной генерации приложений нового поколения для фундаментальных исследований или анализа и оптимизации производственных процессов. Необходимо иметь в виду, что темпы развития современных и будущих высокотехнологичных производств (новые материалы, био– и нанотехнологии, и пр.) напрямую зависят от уровня математизации и компьютеризации данных сфер деятельности, а главным орудием познания здесь становится моделирование. Другая важная диалектическая тенденция заключается в том, что наряду с активным развитием различных научных дисциплин, в последние десятилетия идет бурное накопление результатов в теоретической, прикладной и вычислительной математике, которые ждут своего компьютерного воплощения. А это означает, что новое ППО должно быть “живым организмом”, комфортная перманентная эволюция которого требует самой серьезной поддержки.

В данной работе излагается концепция коллективной (многопользовательской) базовой системы моделирования (БСМ), включая архитектуру построения разработки и особенностей ее функционирования в рамках облачных вычислений и технологий программно–вычислительных услуг (SaaS – Software as a Service). В п. 2 рассматриваются вопросы стратегического и тактического планирования параллельных вычислений с точки зрения различных целевых функций в оценках эффективности использования технического оборудования вычислительного центра коллективного пользования (ВЦКП). Надо сказать, что новое — это хорошо забытое старое, и в определенной степени аналогичные проблемы обсуждались в работе [15] 1988 года. Пункт 3 посвящен некоторым системным аспектам, возникающим при реализации предлагаемых приложений, а также их возможным обобщениям и дальнейшим путям развития.

2. Критерии эффективности коллективного распараллеливания

Для некоторого абстрактного рассмотрения многоцелевого использования коллективных компьютерных ресурсов необходимо в первую очередь выбрать формальную модель многопользовательской вычислительной системы, а также и самого вычислительного процесса.

В достаточно общем виде мы можем рассмотреть неоднородную, или гетерогенную, многопроцессорную – многоядерную вычислительную систему (или сеть), которую будем сокращенно называть МВС, как совокупность соединенных между собой различных кла-

стеров и серверов. Кластер, в свою очередь, представляется набором многопроцессорных узлов, в которых процессоры могут быть или “стандартные” (центральные, CPU), содержащие относительно небольшое количество вычислительных ядер (например, 8 или 12) с общей памятью, или специальные (наиболее распространенные — это графические процессоры общего назначения, GPGPU), насчитывающие несколько сот ядер (например, 512). Представляемая для расчетов память имеет, как правило, иерархическую структуру и включает оперативную память процессоров и три уровня кэша, отличающиеся между собой объемом и быстродействием. Отдельный сервер формально можно считать кластером, состоящим из одного “большого” узла, содержащего достаточно емкую общую оперативную память и значительное количество многоядерных процессоров различных типов.

Вычислительный сеанс на такой МВС заключается в выполнении арифметических действий на соответствующих устройствах, а также в информационных обменах между ними. Идеальным является выполнение расчетов непосредственно в процессоре с использованием только сверхбыстрых регистров памяти, однако их количество ограничивается несколькими десятками.

Ограничиваясь тривиальной моделью вычислений, время $t(A)$ реализации фрагмента алгоритма (или задачи) A на каком-то одном процессоре компьютера можно выразить формулой

$$t(A) = t_a + t_c + t_w, \;\; t_a = \tau_a N_a, \;\; t_c = (\tau_0 + \tau_c N_c) m_c, \tag{1}$$

где τ_a и N_a – среднее время выполнения одного действия и общее количество арифметических операций, t_c – время реализации межпроцессорных информационных обменов, τ_c – время межпроцессорной передачи одного числа, N_c – количество передаваемых чисел, τ_0 – время задержки (настройки) одной транзакции, m_c – число обменов, а t_w – время простоев (ожидания) процессора, которые можно рассматривать как неизбежную плату за рассогласованность структуры конкретного алгоритма и архитектуры МВС, на которой он выполняется. Для реальных вычислительных систем имеют место соотношения $\tau_a \ll \tau_c \ll \tau_0$, в силу чего повышение производительности расчетов в значительной степени связано с минимизацией коммуникаций. Здесь следует отметить и другой немаловажный фактор: операции передачи данных являются намного более энергозатратными, в сравнении с арифметическими действиями, что сильно влияет на стоимость эксплуатации МВС (потребление электроэнергии суперкомпьютера постпетафлопсной производительности достигает нескольких мегаватт).

Несовершенство формул в (1) с введением величины τ_a сродни понятию “средней температуры по больнице”, поскольку реальная длительность различных арифметических действий сильно отличается друг от друга и, более того, на самом деле необходимо учитывать совокупную скорость выполнения арифметических выражений на процессорном конвейере.

Второе грубое приближение заключается в том, что τ_0 и τ_c тоже представляют собой усредненные характеристики скоростей обменов между различными устройствами. В пределах одного CPU — это связи умножителей и сумматоров с процессорными регистрами, обмены между регистрами и кэшем, а также между различными уровнями оперативной общей памяти. Намного медленнее идет передача данных между CPU и GPU, между вычислительными узлами одного кластера и тем более — между различными кластерами, соединяемыми в рамках ВЦКП с помощью Grid-технологий.

С другой стороны, значительного уменьшения расчетного времени $t(A)$ можно добиться путем специальной организации вычислительного процесса: совмещение обменов и арифметических операций во времени, организация информационных буферов, различные приемы оптимизации программного кода типа “развертка циклов”, и т.д.

Качество распараллеливания задачи A на P процессорах в первом приближении можно охарактеризовать двумя параметрами — коэффициентом ускорения S_P и эффективностью

использования процессоров:

$$S_P(A) = \frac{t_1(A)}{t_P(A)}, \quad E_P(A) = S_P(A)/P, \tag{2}$$

где $t_P(A)$ – время выполнения задачи на P процессорах.

В определенном смысле идеальным считается линейное распараллеливание, когда ускорение S_P пропорционально числу процессоров P, а эффективность E_P близка к единице. На практике зачастую считаются приемлемыми значения $E_P = 0.1 \div 0.5$. Однако иногда случаются и чудеса со значениями $E_P > 1$ (сверхлинейное ускорение), когда, например, с удвоением P алгоритм, изначально выполняемый в медленной памяти, начинает "умещаться" в кэше.

Здесь и ниже под числом понимается вещественная переменная со стандартной двойной точностью в 64-битовом представлении с плавающей запятой, что является необходимым (и зачастую достаточным) при решении больших задач, например — при решении СЛАУ с размерностью порядка миллиона. Отметим, что в памяти объемом один гигабайт размещается $128 \cdot 10^6$ таких чисел. Отсюда, например, следует, что для хранения элементов разреженной квадратной матрицы с размерностью $N = 10^9$ и со средним числом ненулевых элементов в строке, равным 100, требуется около одного терабайта памяти. Это означает, во-первых, что размещение такого объема памяти требует большого числа процессоров, а во-вторых, что соответствующие межпроцессорные пересылки представляют весьма дорогую операцию.

Парадоксальным, на первый взгляд, является тот факт, что аналитические оценки компьютерной производительности являются "терра инкогнито" вычислительной информатики. Реальные операции при наличии конвейеров, многоуровневой памяти и внутренних особенностях компиляторов настолько сложны. что даже смоделировать выполнение простейших векторно-матричных выражений не представляется возможным. Поэтому проблема оптимизации программного кода приоткрыта только немногим посвященным, а единственный реальный путь ее исследования и понимания — это систематический эксперимент с квалифицированными измерениями, в которых должны помогать инструментальные профилировщики вычислительных процессов.

Отметим, что попытки построения систем автоматического распараллеливания алгоритмов (типа долго развиваемого, но затем закрытого, проекта HPF — High Performance Fortran) пока не завершились созданием промышленного компилятора. Существующие же средства распараллеливания — различные варианты системы передачи сообщений MPI для распределенной по кластерным узлам памяти и инструменты OpenMP для общей памяти многоядерных CPU имеют "полуручной" принцип управления. Вопросы же высокопроизводительных вычислений в "облаках" и в сетевых технологиях в большой степени относятся не непосредственно к эффективности распараллеливания алгоритмов, а к управлению расчетными заданиями и к интегрированию компьютерных ресурсов, см. [16].

В заключение данного пункта укажем, что в разные годы изобреталось большое количество специализированных вычислительных устройств, которые для определенного класса задач или алгоритмов достигали рекордных скоростей. Однако эти спецпроцессоры не выдержали рыночной конкуренции с универсальными МВС. Возможным исключением являются реконфигурируемые ПЛИС [17], однако вопросы высокопроизводительных вычислений с их использованием представляют собой самостоятельную проблему, выходящую за рамки данной статьи.

3. О концепции "заоблачного" моделирования

Для качественного ППО, которое могло бы способствовать массовой востребованности и практической эффективности математического моделирования на современных МВС, можно сформулировать следующие технические требования:

- применимость к решению широкого класса проблем: прямые и обратные междисциплинарные задачи, описываемые системами дифференциальных и интегральных уравнений в многомерных расчетных областях со сложной геометрией и контрастными материальными свойствами, включая стационарные и динамические, линейные и нелинейные постановки;

- высокое разрешение численных решений, определяемое адекватностью математических моделей и аппроксимационными качествами методов, адаптируемых под свойства задачи, а также робастностью (надежностью и безотказностью) алгоритмов;

- высокая производительность расчетов, включая масштабируемую параллельность с отображением алгоритмов на архитектуру гетерогенных многопроцессорных и многоядерных вычислительных систем;

- интеллектуальность программных реализаций, обеспечивающая автоматическое формирование моделей и построение алгоритмов, управление вычислительным процессом и создание дружественных профессиональных интерфейсов для различных категорий пользователей;

- гибкие конфигурационные свойства, способствующие длительному жизненному циклу ППО с непрерывным пополнением состава применяемых моделей и алгоритмов, а также адаптации к изменениям компьютерных платформ;

- открытость к внешним программным разработкам и возможность переиспользования сторонних продуктов.

Перечисленные экстремальные условия ориентированы главным образом на решение суперзадач на суперкомпьютерах постпетафлопсного уровня, а их воплощение предполагает создание проекта не отдельной группы разработчиков, а сообщества разнопрофильных специалистов с широкой координацией и кооперацией многолетнего фронта работ.

Реализация данного амбициозного круга проблем рассматривается в рамках базовой системы моделирования, структура которой включает следующие основные части:

- интегрированное инструментальное ядро, состав которого определяет класс решаемых задач и применяемых алгоритмов;

- конфигурационные средства, ответственные за уровень интеллектуальности и пользовательской доступности разрабатываемого ППО;

- семейство ППП и алгоритмических библиотек, характеризующих конечную практическую направленность и эффективность создаваемых приложений.

Ядро БСМ представляет собой методо-ориентированное окружение, предназначенное для поддержки всех основных технологических этапов крупномасштабного вычислительного эксперимента, а его функциональное наполнение определяется следующим набором компонент, или подсистем:

- геометрическое и функциональное моделирование, на основе которого реализуется автоматизация когнитивных методик, формируются профессиональные пользовательские интерфейсы для конкретных приложений, а также импорт-экспорт с САПРами и внешними графическими пакетами;

- генерацию адаптивных неструктурированных сеток, с поддержкой параллельных методов декомпозиции сеточных подобластей, многосеточных подходов и локальных сгущений дискретизации в окрестности сингулярностей решений, на основе априорного и/или апостериорного анализа решаемой задачи;

- сеточные и спектральные алгоритмы различных порядков аппроксимации исходных задач, обеспечивающие высокое разрешение численных решений и апостериорные оценки точности (методы конечных объемов, конечных элементов, методы граничных интегральных уравнений, разрывные схемы Галеркина и др.);

- параллельные алгебраические решатели для возникающих в практических задачах больших плохо обусловленных систем сеточных линейных и нелинейных уравнений, а также для решения проблем собственных значений соответствующих матриц;

- методы оптимизации для решения проблем условной минимизации, локальной или глобальной, как основной подход к реализации обратных задач (идентификация параметров модели, автоматизация проектирования, оптимизация эксплуатационных режимов и т.д.);

- обработка и визуализация результатов численных расчетов;

- управление вычислительным процессом, формирование и реализация сценариев моделирования;

- анализ результатов моделирования и средства принятия решений.

Важно подчеркнуть, что каждая из подсистем ядра является открытой и легко пополняемой инструментальной средой для соответствующей технологической стадии, которая практически не зависит от других компонент, а взаимодействие между ними происходит через унифицированные структуры данных: геометрические (ГСД), функциональные (ФСД), сеточные (ССД), алгебраические (АСД) и т.д., – со множественными представлениями и возможностями конвертизации, для обеспечения гибких внутренних интерфейсов между различными вариантами методов или моделей и внешних контактов со сторонними программными продуктами. Подчеркнем, что многоверсионность данных и алгоритмов требует специальных средств поддержки, но зато позволяет достичь такого уникального качества, как совмещение универсальности и эффективности разработки, за счет ее адаптивности к специфическим свойствам задачи.

Помимо перечисленных алгоритмических компонент, функциональное наполнение должно включать средства методической поддержки математического и программного обеспечения: верификации, тестирования, сравнительного анализа, т.е. экспериментального исследования. Это подразумевает создание каталогизированной библиотеки тестовых примеров и типовых задач с архивами расчетных результатов и соответствующими описаниями.

Средства конфигурационного управления составляют системное наполнение БСМ и обеспечивают автоматизированное пополнение состава математических моделей решаемых задач и применяемых алгоритмов, конструирование вычислительных схем в составе ППП для классов приложений, диалоговых графических интерфейсов для профессионалов в прикладных областях, а также адаптацию на конкретные компьютерные платформы и операционные обстановки. Фактически набор системных средств определяет интеллектуальный уровень БСМ, а их реализация предполагает активное использование специализированных языков программирования [14]. Главное назначение системного блока — это поддержка длительного жизненного цикла всего проекта, который должен обеспечиваться пользователями высокого уровня — разработчиками новых алгоритмов и приложений в составе БСМ.

Семейство ППП, образуемое конфигуратором из инструментальных модулей ядра, представляет собой непосредственное орудие математического моделирования и определяет производственные сферы его реального эффективного применения.

В целом конгломерат предлагаемых в составе БСМ приложений — это и есть непосредственный пакет решений для конечного пользователя, которому неинтересны (а в определенном смысле — даже вредны) знания о “внутренней кухне” информационно-вычислительных технологий, а нужны реальные возможности продукта для исследования его конкретной

проблемы. В некотором смысле идеальным является представление ППП в виде “черного ящика” с минимальным набором кнопок и инструкций. Однако на практике неизбежны и компромиссные варианты в форме “серых ящиков” с возможностями модификаций режимов моделирования для заинтересованных продвинутых конечных пользователей.

Типовая вычислительная блок-схема для решения стационарной линейной краевой задачи представляется следующим образом: формирование с помощью графического редактора исходных данных о расчетной области и решаемых уравнениях (в виде ГСД и ФСД), построение сетки (ССД), реализация аппроксимаций и формирование АСД, решение СЛАУ, пост-обработка, визуализация и анализ полученных результатов. Если поставленная проблема является нелинейной, то необходимые этапы (а возможно, и все) повторяются с проведением итераций по нелинейным членам. В случае нестационарности моделируемого процесса рассмотренные этапы выполняются последовательно для различных шагов по времени. И наконец, для обратных задач с формулируемым целевым функционалом, зависящим от некоторых параметров в исходных данных, все предыдущие стадии, представляющие собой решение одной прямой задачи, циклически повторяются в соответствии с оптимизационными алгоритмами условной минимизации функционала.

Таким образом, формально вычислительная схема решения одной абстрактной большой задачи в общем случае представляется вложенными циклами, в каждом из которых распараллеливание алгоритмов может осуществляться в соответствии со своими тактиками и стратегиями.

Начать рассмотрение следует с нижнего уровня — этапа решения СЛАУ, который играет ключевую роль в силу того, что его ресурсоемкость нелинейным образом растет с увеличением порядка системы, или числа степеней свободы. Главным орудием распараллеливания здесь является метод декомпозиции расчетной области на подобласти (с пересечениями или без), в каждой из которых независимо и синхронно (в идеальном случае) решаются соответствующие подсистемы, а с их помощью устраивается итерационный процесс по подобластям.

Одно из важных условий распараллеливания методов решения современных сверхбольших СЛАУ на МВС — это отсутствие программных ограничений на порядки решаемых систем и на количество используемых процессоров и/или ядер. По этой причине реализацию алгоритмов надо делать в строго распределенном варианте, без чрезмерной концентрации ресурсов на корневом процессе системы MPI.

Отметим еще один серьезный момент: во избежание простоя вычислительного оборудования при одновременном решении вспомогательных задач в подобластях разными MPI-процессами декомпозиция расчетной области должна быть сделана сбалансированной. Последнее понятие, строго говоря, означает, что алгебраические подсистемы для подобластей должны решаться за одинаковое время, которое в первом приближении определяется порядком, т.е. количеством строк в соответствующем блоке глобальной матрицы.

Разбиение краевой задачи на подзадачи может рассматриваться в трех ипостасях: геометрическом, сеточном и алгебраическом. Наиболее близки между собой две последние интерпретации, поскольку каждому узлу сетки соответствует своя строка матрицы, а ребру сеточного графа соответствуют симметрично расположенные ненулевые внедиагональные элементы матрицы.

Алгебраическая декомпозиция предполагает, что дана только информация о портрете большой разреженной матрицы порядка N, представленной в одном из сжатых форматов, например — CSR (Compressed Sparse Row), когда хранятся только ненулевые элементы матрицы и информация об их расположении. Задача в данном случае сводится к формированию CSR-форматов для заданного числа блочных строк $P \ll N$, причем в одном блоке должны быть собраны наиболее тесно связанные между собой матричные строки. Проблема эта решается методами преобразования графов и является довольно ресурсоемкой.

Более перспективным является сеточная декомпозиция на этапе дискретизации расчетной области. Сеточная структура данных является намного более экономичной, чем АСД, поэтому достаточно большую сеточную расчетную область можно создать в корневом MPI-процессе, а затем из нее сформулировать сеточные подобласти в распределенном по процессорам варианте. Более того, после генерации исходных сеток в подобластях к ним можно применить несколько раз процедуру сгущения и получить семейства иерархических вложенных сеток. После окончания этапа дискретизации выполнение этапа аппроксимации и формирования АСД для подобластей проводится сразу в параллельной (распределенной) версии, без чрезмерной перегрузки корневого MPI-процесса.

Отметим, что проблема декомпозиции на самом верхнем — геометрическом — уровне с информационной точки зрения вообще не представляет трудностей, поскольку объемы данных в ГСД и ФСД малы, так что они легко могут быть растиражированы по всем процессам.

Рассмотренная выше алгоритмическая структура может составлять часть схемы расчетов серии однотипных задач, которые должны выполняться или одновременно, или только последовательно. Такие вычислительные сеансы с решениями однотипных прикладных проблем могут быть как эпизодические, так и составлять регулярную область деятельности некоторой группы пользователей. В таких случаях целесообразно формирование некоторого "частного облака" с выделением виртуального кластера проблемно-ориентированной конфигурации, производительность которого может быть оптимизирована на основе анализа соответствующего вычислительного процесса.

Следующий уровень организации вычислений в абстрактном плане — это решение не связанных между собою задач независимыми пользователями, объединяемых только общностью используемого прикладного программного обеспечения, т.е. БСМ. Здесь проблемы управления интегрированными ресурсами ВЦКП и планирования компьютерных экспериментов требуют создания некоторой инфраструктуры с использованием не только облачных, но и сетевых технологий. Соответствующие решения требуют еще самих определений и оптимизационных постановок, понимание и актуальность которых неизбежно приходит с погружением в практику массового математического моделирования.

Литература

1. DATA CENTER: URL: www.datacenterknowledge.com (дата обращения: 01.12.2013).
2. Ильин В.П., Скопин И.Н. Технологии вычислительного программирования// Программирование. – 2011. – N 4. С. 53-72.
3. 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. – N 2. – С. 25-33.
4. Ильин В.П. DELAUNAY: технологическая среда генерации сеток//СибЖИМ. – 2013. – Т.16, № 2(54). – С. 83-97.
5. OpenFOAM® — The Open Source Computational Fluid Dynamics (CFD) Toolbox: URL: www.openfoam.com (дата обращения: 01.12.2013).
6. DUNE Numerics: URL: www.dune-project.org (дата обращения: 01.12.2013).
7. OPEN CASCADE: URL: www.opencascade.org (дата обращения: 01.12.2013).
8. SALOME: URL: www.salome-platform.org (дата обращения: 01.12.2013).
9. IESP: URL: www.exascale.org/iesp (дата обращения: 01.12.2013).

10. Чесбро Г. Открытые инновации. – М., изд. “Поколение”. – 2007. – 333с.

11. OPEN NOVATION: URL: www.opennovation.org (дата обращения: 01.12.2013).

12. ANSYS — Simulation Driven Product Development: URL: www.ansys.com (дата обращения: 01.12.2013).

13. MSC Nastran — Industry Leading Multidisciplinary FEA: URL: www.mscsoftware.com/product/msc-nastran (дата обращения: 01.12.2013).

14. Kleppe A. Software Language Engineering: Creating Domain–Specific Language Using Metamodels//N.Y. Addison–Wesley. – 2008. – 207 p.

15. Алексеев А.С., Гололобов В.И., Ильин В.П., Карначук В.И. Комплексный центр математического моделирования: концепция программного обеспечения//Новосибирск, препринт N 821; ВЦ СО АН СССР. – 1988. – 43с.

16. Городничев М.А., Малышкин В.Э., Медведев Ю.Г. HPC Community cloud: эффективная организация работы нучно-образовательных суперкомпьютерных центроов//Научный вестник НГТУ. – 2013. – N 3(52). – P. 92-96.

17. Каляев И.А., Левин И.И. Семейство реконфигурируемых вычислительных систем в высокой производительностью//Вычислительные методы и программирование. – 2009. – Т. 10. – N 1. – С.207-214.