ПРОБЛЕМЫ ПАРАЛЛЕЛЬНОЙ РЕАЛИЗАЦИИ КРУПНОМАСШТАБНЫХ ЧИСЛЕННЫХ МОДЕЛЕЙ НА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ ЭКЗАФЛОПСНОЙ ПРОИЗВОДИТЕЛЬНОСТИ

В. Э. Малышкин

Институт вычислительной математики и математической геофизики СО РАН,
630090, Новосибирск, Россия
Новосибирский национальный исследовательский университет,
630090, Новосибирск, Россия
Новосибирский государственный технический университет,
630073, Новосибирск, Россия

УДК 519.685.1

Рассматриваются и анализируются проблемы параллельной реализации крупномасштабных численных моделей с большим объемом используемых при моделировании данных на вычислительных системах с большим числом процессорных элементов (ПЭ). Рассмотрение ведется на конкретном примере параллельной реализации метода частиц-в-ячейках в его приложении к моделированию природных явлений в астрофизике и физике плазмы. Сформулирована задача разработки системы параллельного программирования крупномасштабных численных моделей, определены первоочередные задачи и предложены решения, которые в совокупности позволят исключить параллельное программирование из процесса создания численной модели. В первую очередь необходимо разработать новые распределенные системные технологические алгоритмы с локальными взаимодействиями, без чего решение задачи невозможно.

Ключевые слова: крупномасштабные численные модели, параллельное программирование крупномасштабных численных моделей, метод частиц-в-ячейках, технология фрагментированного программирования, распределенные системные алгоритмы с локальными взаимодействиями.

The problems of parallel implementation of the large-scale numerical models with the use of large volume of data on multicomputers with the huge number of nodes is considered and analyzed. Consideration is done on the basis of the analysis of parallel implementation of the well-known Particle-In-Cell (PIC) method application to modeling of natural phenomena in astrophysics and plasma physics.

The problems of the development of a parallel programming system supporting large-scale numerical models programming are formulated. Top-priority problems are defined and their acceptable solutions are suggested that provide elimination of the stage of parallel programming from the process of such models development.

Consideration demonstrate that implementation of parallel programming systems demands first of all the development of new technological distributed system algorithms with local interactions (DSALI) for dynamic construction as control and as data distribution.

Key words: large-scale numerical modeling, parallel programming of large-scale numerical models, particle-in-cell method, fragmented programming technology, distributed system algorithms with local interactions, distributed system algorithms with local interactions.

Введение. Современные суперкомпьютеры (в основном мультикомпьютеры) по большей части заняты крупномасштабным численным моделированием в физике, химии, биологии и т. д., которое вовлекает в обработку большие данные (big data). Пета- и ожидаемые экзафлопсные вычислительные системы[1] (ЭВС) тоже, видимо, будут использованы прежде всего для численного моделирования.

Параллельная реализация крупномасштабных численных моделей на ЭВС и реалистическое моделирование[2] природных явлений вовлекут в обработку еще большие объемы данных. Это, в свою очередь, повлияет на

— выбор/разработку новых системных алгоритмов обработки этих данных, таких, например, как алгоритмы статического и динамического распределения ресурсов и вычислений;

— создание новых алгоритмов организации системных и прикладных вычислений и, более того,

— создание новых принципов проектирования и новой техники разработки приложений;

— создание принципиально новых языков и систем параллельного программирования.

Данные в численных моделях часто носят регулярный характер. Но в крупномасштабных численных моделях при моделировании сложных явлений на ЭВС нередко возникают нерегулярные структуры данных и даже структуры данных с динамически меняющейся нерегулярностью (адаптивные сетки, переменный временной шаг, нерегулярные скопления частиц и т. д.). По этой причине сложность параллельного программирования таких моделей в современных технологиях программирования оказалась высокой, близкой к сложности системного программирования.

Изложение материала ведется в основном на примерах. Чтобы стало возможным формулировать и анализировать проблемы параллельной реализации крупномасштабных численных моделей и моделирования природных явлений на ЭВС, подробно рассматривается параллельная реализация сложного, широко используемого метода частиц-в-ячейках (Particle-In-Cell, PIC) в приложении к задачам физики плазмы [1–3] и астрофизики [4]. Выделяются и описываются особенности такой реализации, предлагаются решения в форме технологии фрагментированного программирования (ТФП) [5, 8]. Подход основан на разбиении алгоритма на части (фрагменты) и сборке целой программы из заранее заготовленных модулей, каждый из которых реализует фрагмент вычислений. В отличие от технологии модульного программирования, в ТФП фрагментированная структура программы сохраняется в ходе исполнения, а каждый фрагмент вычислений определяет независимо исполняющийся процесс, который в состоянии мигрировать с одного ПЭ мультикомпьютера на другой и взаимодействовать с другими процессами.

Эта идея эксплуатируется в программировании уже долгое время [4–17]. Различные системы программирования [8–17] в той или иной мере использовали этот подход. Для организации вычислений в большинстве систем [9–12] использовались run-time системы. В системе Charm [10–12] программа собирается из модулей, но представление алгоритмов содержит частичное распределение ресурсов, что уменьшает переносимость прикладных

[1]Вычислительные системы экзафлопсного диапазона производительности с большим числом процессоров.

[2]Модели (моделирование) настолько полные и настолько большого размера, что в процессе моделирования они ведут себя близко (насколько это требуется в вычислительном эксперименте) к поведению реального явления.

Charm-программ. В них также нет средств динамического управления распределением ресурсов и выбора управления в программе.

В PaRSEC [14] данные назначаются на определенные узлы, что ведет к уменьшению переносимости и качества настройки на конкретные условия конкретного вычислителя. Замена интерпретации операций также ограничена, что, конечно, является наследованным свойством библиотеки D-PLASMA [15].

В [18] вместо обычно используемой run-time системы для исполнения собранной из модулей программы были разработаны специальные мультикомпьютер и операционная система. Решение хорошее, но дорогое, годится лишь для решения особо важных задач.

1. PIC метод. Рассматриваемые ниже алгоритмы параллельной реализации приложений PIC метода [1, 2] к разработке крупномасштабной модели процессов обмена энергией в облаке плазмы и моделированию этих процессов показывают сложность программирования таких моделей, близкую к сложности системного программирования, и демонстрируют необходимость использования новых системных и прикладных алгоритмов в параллельной реализации реалистических численных моделей. Рассматривается лишь вычислительная схема приложения PIC, которая необходима для понимания проблем параллельной реализации таких моделей. Содержательная и математическая сущности модели не обсуждаются. Предполагается исполнение модели на мультикомпьютерах с различной структурой коммуникационной сети: дерево (кластеры), 2D- и 3D-решетки, тор.

1.1. *Идея PIC метода.* На первом шаге применения PIC описывается пространство моделирования (ПМ). ПМ представляется прямоугольной сеткой, на которой дискретизируется описание электромагнитного поля. В это пространство вводится огромное число модельных заряженных частиц плазмы (чем больше частиц, тем точнее решение), которые движутся под воздействием электромагнитного поля. Каждая заряженная частица описывается ее массой, координатами, скоростями и зарядом. Рассматривается бесстолкновительная плазма, частицы влияют друг на друга опосредовано, через изменения электромагнитного поля.

Поведение частиц облака плазмы описывается системой дифференциальных уравнений в 6-мерном пространстве (3 координаты частиц и 3 скорости на каждом временном шаге), включающей уравнение Власова [1, 2]. Вместо интегрирования уравнения Власова, вычисляются траектории движения частиц (отсюда потребность в больших вычислительных ресурсах) в электромагнитном поле, которые и являются желаемым решением системы дифференциальных уравнений.

В [5] PIC метод применялся к решению астрофизической задачи моделирования поведения облака пыли в пространстве. Суть метода и его программная реализация принципиально те же, лишь вместо электромагнитного поля задавалось гравитационное поле.

1.2. *Представление данных в суперкомпьютере.* Реальное физическое пространство представляется прямоугольной 3D-сеткой (ПМ), на которой дискретизируются описание электрического E и магнитного B полей. Они задаются векторами напряженности (сеточные значения) в определенных, специальным образом выбранных точках пространства (рис. 1 и 2).

Для разработки методов параллельной реализации PIC существенно то обстоятельство, что, в отличие от других численных методов, в PIC присутствуют две разные структуры данных – частицы и сетки. Обработка частиц в подобных задачах может занимать до 90 % времени счета.

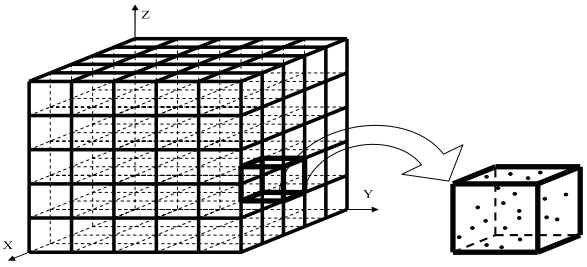

Рис. 1. ПМ, собранное из ячеек

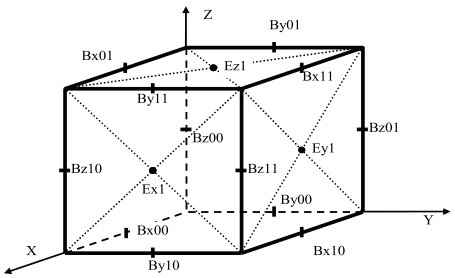

Рис. 2. Ячейка ПМ. Показана дискретизация электрического E и магнитного B полей

Большая сетка разбивается на прямоугольные фрагменты, например, параллелепипеды ячеек (фрагменты данных – ФД). По всем трем направлениям размеры ФД – константы, которые выбираются таким образом, чтобы ФД мог быть размещен в любом ПЭ. Все ФД распределяются по ПЭ таким образом, чтобы все ПЭ были нагружены примерно одинаково. В каждый момент времени каждая частица принадлежит некоторой ячейке сетки и, следовательно, некоторому ФД, и размещается в том же ПЭ, где находится эта ячейка.

1. 3. *Теоретический алгоритм моделирования.* Теоретический алгоритм не учитывает влияния технологических факторов (производительность ПЭ и коммуникационной сети, архитектура сети, количество и качество распределения ресурсов, поведение моделируемого явления и многие другие). Он описывает содержательные шаги моделирования безотносительно к возможности их реализации на мультикомпьютерах.

В начальный момент моделирования известны сеточные значения и значения всех атрибутов <координаты, скорости, масса, заряд> всех частиц. На каждом временном шаге t_{k+1}:=t_k+Δt, k=0, 1, ..., n моделирования выполняется следующая обработка:

1) для каждой частицы из значений электромагнитного поля в ближайших точках сетки вычисляется действующая на нее Лоренцева сила (*gathering phase*);

2) для каждой частицы вычисляются новые координаты и скорости. Частицы под действием этой силы двигаются. За время Δt они могут сдвинуться и остаться в ячейке, либо могут передвинуться в другую ячейку, но не далее соседней (*moving phase*);

3) из новых координат и скоростей частиц вычисляются значения средней плотности тока и заряда в узлах сетки (*scattering phase*);

4) из полученных значений средней плотности тока и заряда в узлах сетки пересчитываются значения (векторы напряженности) электрического и магнитного полей (*mesh phase*).

Алгоритм не учитывает особенностей его всевозможных реализаций (программ) исполнения на различном оборудовании. Чтобы обеспечить его эффективное параллельное исполнение на мультикомпьютерах, необходимо преобразовать его в другой, технологический алгоритм[3], который допускает эффективное параллельное исполнение в классе мультикомпьютеров.

[3]Технологический алгоритм должен быть хорошо реализуемым на классе мультикомпьютеров, а именно: иметь приемлемую сложность, например линейную, учитывать свойства прикладной задачи и быть реализованным с обеспечением всех необходимых практически важных свойств, как то: надежность, точность решения, высокая производительность и т. п.

1.4. *Идеи технологического алгоритма (Эйлерова декомпозиция).* Все ПМ делится на прямоугольные подобласти (неделимые в дальнейшем фрагменты данных – ФД). Это могут быть слои, блоки колонок, параллелепипеды ячеек. Здесь и далее будет рассматриваться самый общий случай – деление ПМ на параллелепипеды ячеек. Каждая ячейка сетки включается в некоторый ФД со всеми своими сеточными значениями и частицами. Обработка данных такого ФД определяется в программе как независимый процесс (фрагмент вычислений – ФВ), который может обмениваться данными с другими аналогичными процессами. На множестве ФД определяется бинарное отношение соседства, при котором прилегающие ФД (с общей вершиной, либо ребром, либо гранью) являются соседними. Отношение соседства может быть определено и так, что соседями являются ФД, составляющие множество входных переменных некоторого ФВ. ФД назначаются на обработку в процессорные элементы (ПЭ) мультикомпьютера таким образом, чтобы

— соседние ФД были бы размещены в как можно более близких ПЭ для минимизации расходов на коммуникации. Лучше всего, если они размещены в одном и том же ПЭ, либо в соседних (связанных физическим линком);

— в начальный момент моделирования должна обеспечиваться примерно равная нагрузка всех ПЭ.

Если в ходе моделирования частица покидает ФД и перелетает (migrate) в другой, то частица в программе должна перелететь на тот ПЭ мультикомпьютера, где хранится этот последний ФД. Таким образом, через несколько шагов моделирования нагрузка ПЭ может стать несбалансированной, т. е. разница нагрузок соседних ПЭ превышает порог допустимого дисбаланса. В этом случае необходимо сбалансировать нагрузку за счет миграции некоторого количества ФД с перегруженного ПЭ в соседние, менее нагруженные. Так как траектории частиц определяются не только уравнениями, но и данными (сеточными значениями, координатами, скоростями, зарядами и массами частиц), то это исключает возможность статического планирования нагрузки ПЭ мультикомпьютера. Следовательно, балансировка нагрузки ПЭ мультикомпьютера должна делаться *динамически* по мере возникновения дисбаланса в той или иной части процессорного поля.

Основная проблема программирования приложений PIC заключается в том, что программа, в первую очередь управление [19] и распределение ресурсов вычислителя, зависит не только от объема данных, но и от свойств этих данных, от распределения частиц в ПМ, от поведения модели. И управление, и распределение ресурсов в такой программе должны конструироваться динамически.

1.5. *Типичные задачи моделирования.* Ниже даны примеры задач моделирования с разными распределениями частиц.

— *Однородное* распределение частиц в ПМ.

— *Поток.* Множество частиц делится на два подмножества: частицы с нулевой начальной скоростью и активные частицы с ненулевой скоростью. Активные частицы образуют поток, пересекающий ПМ в некотором направлении.

— *Взрыв.* Заданы два подмножества частиц. Фоновые частицы с нулевой начальной скоростью в начальный момент равномерно распределены в ПМ. Все активные частицы образуют симметричное облако, $r \ll h$, где r – радиус облака активных частиц, а h – шаг сетки, т. е. все активные частицы находятся в одной ячейке.

Если активных частиц достаточно много, то и ячейка, и содержащий ее ФД обычно не могут быть размещены в памяти одного ПЭ. В этом случае ФД представляется несколькими *виртуальными* ФД. Все виртуальные ФД содержат одни и те же сеточные значения,

а частицы поровну делятся между всеми виртуальными ФД. При этом все виртуальные ФД рассматриваются как соседние и размещаются в мультикомпьютере компактно, максимально близко друг от друга. Если через некоторое время частицы начинают покидать виртуальные ФД, они по мере возможности и необходимости попарно сливаются. Использование виртуальных ФД должно, конечно, поддерживаться в прикладной программе системным ПО.

2. Проблемы параллельной реализации приложений PIC на ЭВС. На этом этапе накоплена вся необходимая информация для формулировки главных проблем параллельной реализации крупномасштабных численных моделей на ЭВС.

1) ЭВС могут содержать сотни тысяч и даже миллионы ПЭ. По этой причине параллельные программы крупномасштабного моделирования, предназначенные для исполнения на ЭВС, не должны содержать центрального управления, чтобы избежать медленно реализуемых пересылок между удаленными ПЭ. Всякие решения по конструированию управления и распределению ресурсов в ходе моделирования должны вырабатываться и приниматься локально в каждом ПЭ с использованием, как правило, данных, размещенных не далее, чем в соседних ПЭ[4]. Таким образом, для реализации системных функций необходимо разработать и использовать распределенные системные алгоритмы с локальными информационными связями, локальными взаимодействиями (РСАЛВ).

2) В крупномасштабных моделях размещение данных в ресурсах ЭВС меняется в ходе моделирования вследствие миграции объектов моделирования (частиц и ФД). Следовательно, программа моделирования не должна содержать имен (номеров) конкретных физических ресурсов, не должна содержать никаких фиксированных решений о распределении ресурсов (программа тоже должна реализовать некоторый РСАЛВ). В общем случае, все ресурсы, используемые в процессе моделирования, должны назначаться и переназначаться динамически с учетом складывающейся в вычислителе и в модели ситуации.

3) В любой момент периода моделирования распределение данных и назначение ресурсов, включая начальное распределение ресурсов, должны в максимально возможной степени сохранять отношения соседства на множествах ФД и ФВ.

4) Большинство коммуникаций, исключая, может быть, небольшое число особых случаев, должны выполняться между близкими, например, соседними ПЭ.

5) Дисбаланс нагрузки ПЭ, который может возникнуть в ходе моделирования, должен выравниваться процедурой динамической балансировки нагрузки, которая работает в каждом ПЭ, вырабатывает и реализует распределенный план балансировки на основе данных соседних процессов и процессоров. Выполнение динамической балансировки делается локально в перегруженном/недогруженном ПЭ с вовлечением соседних ПЭ. Принципиально алгоритм динамической балансировки похож на алгоритм моделирования диффузии в жидкости или газе [20–22], но технологический алгоритм балансировки нагрузки, конечно, сильно отличается от него, все-таки ФД – это не молекулы. Его главное необходимое свойство – локальность и распределенность решения задачи динамической балансировки нагрузки ПЭ. Алгоритм загружается во все ПЭ и начинает исполняться лишь в тех ПЭ, где возникает дисбаланс нагрузки, т. е. динамическая балансировка должна реализоваться некоторым РСАЛВ. Понятно, что чисто диффузионный алгоритм динамической балансиро-

[4]ПЭ соседние, если соединены физическими линками (1-соседство). Могут рассматриваться и 2-соседство и вообще k-соседство, когда соседние ПЭ связаны последовательностью ПЭ и линков длины не более k.

ровки работает очень медленно. В экспериментах – более чем стократно медленнее, чем технологические РСАЛВ, построенные на идее диффузии.

6) Алгоритмы распределения и перераспределения ресурсов в ходе моделирования должны учитывать поведение модели. Например, в ходе моделирования облака пыли [6] образуются солитоны, плотность частиц в которых значительно выше, чем в окружающем пространстве, возможно, и в миллионы раз. Если такой солитон прилетает в ПЭ, то его нагрузка станет много больше нагрузки соседних ПЭ. Эффективный алгоритм динамической балансировки должен не просто балансировать нагрузку соседних ПЭ, но обязательно должен учесть траекторию движения солитона и в текущей балансировке постараться с упреждением уменьшить нагрузку тех многих ПЭ, что лежат на траектории движения солитона и вблизи нее, чтобы они смогли легче поглотить новую нагрузку, которая вскоре в них прилетит. Неучет поведения модели в алгоритме балансировки может привести к многократному падению производительности моделирования.

7) Должны обеспечиваться масштабируемость и переносимость прикладной программы в широком классе мультикомпьютеров.

В последние годы нередко можно слышать и встречать в журнальных статьях термин *экзафлопсные вычисления*, который не определяется точно, но предполагает наличие вычислителей с сотнями тысяч и миллионами вычислительных узлов и с экзафлопсной производительностью. Однако большое количество узлов и экзафлопсная производительность вычислителя здесь явно несущественны. В параллельном программировании экзафлопсные вычисления начинаются не тогда, когда программа исполняется на вычислителях экзафлопсной производительности, а тогда, когда известные в последовательном программировании системные алгоритмы организации исполнения программ приходится заменять на РСАЛВ. Использование РСАЛВ может понадобиться и на мультикомпьютерах со сравнительно небольшим числом ПЭ и небольшой производительностью, если реализация алгоритма решения прикладной задачи потребует использования программ с обеспечением высокой динамики моделируемых процессов.

РСАЛВ решает многопараметрическую задачу распределения ресурсов и выбора управления, в которой необходимо динамически найти приемлемый компромисс в достижении высокой производительности минимумом требуемых ресурсов, обеспечении практически неограниченной масштабируемости программ.

В большей или меньшей степени эти сложные проблемы встречаются при реализации большого числа крупномасштабных численных моделей. Сейчас их должен преодолевать разработчик модели при ее программировании, из чего ничего хорошего происходить не может. Следовательно, разработка параллельных программ численного моделирования должна быть поддержана подходящим системным программным обеспечением [8–18], которое в том числе должно автоматически обеспечивать все практически необходимые свойства прикладной программы. Один из возможных подходов к решению этих проблем предлагает технология фрагментированного программирования [4], к рассмотрению которой и переходим.

3. Параллельное программирование крупномасштабных численных моделей. К этому моменту у нас накоплено достаточно данных, чтобы сформулировать требования к системе программирования, поддерживающей параллельную реализацию крупномасштабных численных моделей, а также предложить некоторые решения. Идеи системной поддержки параллельной реализации крупномасштабных численных моделей демонстрируются на примере подробно рассмотренного выше распараллеливания прило-

жения PIC метода и параллельной реализации приложения в системе фрагментированного программирования LuNA [4, 8, 23], которая разрабатывается в ИВМиМГ СО РАН. LuNA ориентирована на параллельную реализацию больших численных моделей. Внешне, в синтаксисе языка, это почти не проявляется, проблемная ориентация LuNA отражена в алгоритмах ее реализации. Ключевые алгоритмы реализации системы LuNA – РСАЛВ распределения ресурсов и управления.

Теоретическую основу проекта LuNA составляет теория структурного синтеза параллельных программ [24]. Система LuNA работает в экспериментальном режиме и используется в НГУ и НГТУ в лабораторных работах к курсам „Основы параллельного программирования“ и „Технология фрагментированного программирования“.

Теперь нетрудно сформулировать основные прагматические принципы проектирования и реализации системы для разработки параллельных программ крупномасштабного численного моделирования.

1. В соответствии с [25–27], прикладной алгоритм в LuNA представляется рекурсивно перечислимым множеством функциональных термов. Таким образом, все переменные (ФД) должны быть единственного присваивания, а все ФВ – единственного срабатывания, управление – это отношение частичного порядка на множестве ФВ. Такое исходное представление алгоритмов необходимо для того, чтобы можно было автоматизировать конструирование параллельной программы, реализующей алгоритм.

2. Для автоматического динамического конструирования распределения и перераспределения ресурсов мультикомпьютера, миграции процессов и балансировки нагрузки ПЭ мультикомпьютера программа моделирования должна представляться в ходе выполнения рекурсивно перечислимым множеством независимо исполняющихся и взаимодействующих последовательных (это позволит наследовать все накопленное последовательное прикладное ПО) процессов. Процессы должны быть в состоянии мигрировать из перегруженных ПЭ в недогруженные, выравнивая тем самым нагрузку во всех ПЭ.

3. Система фрагментированного программирования (СФП) должна в два шага конструировать прикладную программу моделирования, должна быть двухуровневой. На нижнем уровне разрабатывается структура вычислений с определением всех функциональных фрагментов вычислений (вычисления внутри параллелепипедов ячеек сетки в примере распараллеливания PIC) и фрагментов данных (ФД) (сеточные данные и частицы внутри параллелепипедов). Таким образом, алгоритмы решения задачи на этом этапе должны быть *фрагментированы* (разбиты на фрагменты вычислений и данных) [3, 28, 29]. Каждый ФВ определяется как независимо выполняющаяся единица программы (например, процедура) со своими входными и выходными переменными. Далее программируются коды фрагментов вычислений, определяется представление фрагментов данных и все то, что позволяет разработать последовательную программу обработки данных внутри каждого параллелепипеда.

4. Конечно, каждый алгоритм может быть так или иначе фрагментирован. Однако для получения хорошо работающей программы можно сформулировать несколько технологических требований к представлению численных алгоритмов:

— прежде всего, алгоритм должен быть достаточно мелко фрагментирован, чтобы в каждом ПЭ могли разместиться несколько ФВ со своими входными и выходными ФД;

— операции алгоритма, собственно ФВ, должны иметь как одинаковый объем вычислений, так и одинаковый запрос ресурсов;

— незначительная информационная зависимость между ФВ, при которой в ходе вычислений всегда существует достаточно много готовых (в смысле определения асинхронной модели вычислений [19]) к выполнению ФВ;

— межпроцессные взаимодействия должны быть локальными;

— объемы передачи данных между ФВ должны быть минимальными.

Численные алгоритмы в силу их массовости и регулярности в большинстве своем могут быть фрагментированы так, чтобы удовлетворять вышеперечисленным условиям. Однако эта работа может оказаться непростой и потребует значительных усилий и затрат времени. Например, фрагментация прогонки заняла в сумме более 2-х лет [28].

5. На втором уровне выполняется сборка необходимой программы из фрагментов вычислений (как это описано в структурном синтезе параллельных программ [24]), т. е. компилятор, например, в системе LuNA, принимает на вход фрагменты вычислений и данных, списки входных и выходных переменных алгоритма и генерирует автоматически (одним из многих возможных способов) параллельную прикладную программу требуемого качества. Понятно, что алгоритмы решения прикладной задачи должны быть изначально фрагментированы.

6. Технологически необходимо представлять прикладной алгоритм двумя рекурсивно перечислимыми множествами: множеством ФВ и множеством ФД, а также множествами входных и выходных ФД. Множество функциональных термов, представляющих прикладной алгоритм, выводится в ходе компиляции, либо динамически в процессе вычислений.

7. Моделируемые процессы нередко характеризуются высокой динамикой поведения (как данных, так и вычислений). По этой причине распределение ресурсов и управление в программе моделирования должны в основном конструироваться динамически в ходе исполнения, учитывая, в частности, свойства входных данных и поведение моделируемого явления в текущий момент и предсказанное поведение моделируемого явления в последующие моменты времени [6].

8. Входной язык СФП на базе принципа фрагментации алгоритмов и вычислений будет простым, но его хорошая реализация окажется весьма сложной из-за необходимости реализовать СФП с обеспечением распределенного исполнения ФП и с необходимостью распределенными системными алгоритмами с локальными взаимодействиями добиться глобальной оптимизации исполнения вычислений на мультикомпьютерах с сотнями тысяч и даже миллионами ПЭ.

Конструирование РСАЛВ плохо поддается изучению и анализу ввиду высокой сложности многопараметрической задачи, высокой динамики поведения модели, сложности конструирования управления в программе, сравнимой со сложностью вывода в аксиоматических теориях, да еще и зависящее от входных данных и распределения ресурсов. По этой причине целесообразно разрабатывать технологические РСАЛВ, основанные на алгоритмах моделирования таких природных явлений, как растекание жидкости в системе сообщающихся сосудов, диффузия, тяготение. Алгоритм может строить начальную загрузку мультикомпьютера, организуя такое распространение – „растекание“ – ФВ и ФД фрагментированной программы по всем ПЭ вычислителя, при котором выравнивается нагрузка во всех ПЭ.

Имитация движения жидкости в системе сообщающихся сосудов годится для построения начального распределения ресурсов, но этого не хватит, например, для учета поведения моделируемого явления. Сценарий начального этапа исполнения ФП может выглядеть так: ФД вводятся в один ПЭ, в несколько или во все ПЭ с некоторым произвольным рас-

пределением. Затем, если качество соблюдения отношений соседства неудовлетворительное, ФД мигрируют („растекаются“ по ПЭ мультикомпьютера), выравнивая нагрузки всех ПЭ и улучшая реализацию отношения соседства с целью минимизации коммуникационных затрат при исполнении ФП.

Таким образом, первые необходимые РСАЛВ должны решать задачу построения начального распределения ресурсов и миграции ФД и ФВ в ходе моделирования.

Другая важнейшая задача – определение тех ФВ из множества готовых к исполнению (см. асинхронную модель исполнения в [19]), которые должны исполняться раньше, а какие позже. От этого выбора существенно зависит загрузка мультикомпьютера. Принятие неудачного решения может, например, привести к тому, что в мультикомпьютере долгое время может просто не быть ФВ, готовых к исполнению, и, скажем, половина ПЭ может долго простаивать.

Реализация в СФП вышеперечисленных идей имеет множество различных воплощений и, в частности, может привести к созданию СФП типа системы фрагментированного программирования LuNA. В СФП LuNA для организации гибкого исполнения ФП и обеспечения динамических свойств ФП используются run-time система и множество высокоуровневых средств выражения знаний пользователя о модели и ее поведении, которые используются для оптимизации моделирования.

Заключение. СФП LuNA на настоящий момент генерирует не очень эффективную программу на основе РСАЛВ первого поколения, и, конечно, используемые РСАЛВ сейчас активно модернизируются. Тем не менее, LuNA уже сейчас на некоторых классах численных алгоритмов позволяет исключить параллельное программирование из процесса разработки крупномасштабных численных моделей на прямоугольных сетках. Она автоматически генерирует программы численного моделирования со всеми необходимыми свойствами: практически неограниченная масштабируемость, настройка программы на все доступные ресурсы, наличие динамической балансировки, программировать все эти свойства пользователю не надо.

Невысокое качество либо отсутствие подходящих для использования РСАЛВ сужают область применения систем параллельного программирования типа LuNA. Например, в СФП LuNA нет (еще не разработан) РСАЛВ динамической балансировки нагрузки мультикомпьютера для случая модели с высокой динамикой поведения (взрыв, коллапс). Его необходимо разработать и включить в СФП.

Такого сорта недостатки не стоит драматизировать. У СФП LuNA есть свои важные достоинства. Она автоматически обеспечивает многие полезные свойства прикладной программы, например, неизменное представление прикладного алгоритма, миграцию процессов, автоматические распределение ресурсов и балансировку нагрузки, учет поведения модели при исполнении фрагментированной программы и другие свойства, сложности реализации которых сконцентрированы в СФП, не в прикладной программе. По мере совершенствования реализации СФП LuNA будет улучшаться и качество исполнения прикладных программ без изменения текста самой программы. Уже сейчас СФП LuNA обеспечивает исключение параллельного программирования из процесса разработки некоторых крупномасштабных численных моделей, построенных на базе PIC метода.

В каждой предметной области для достижения высокого качества прикладных программ, при генерации и исполнении программ моделирования должны учитываться свойства предметной области, а потому для каждой предметной области, в общем случае, для реализации СФП, должны разрабатываться свои РСАЛВ. При этом прикладные

фрагментированные алгоритмы не изменяются. Эти обстоятельства сильно увеличивают шансы на создание библиотек параллельных модулей, для которых обеспечены все необходимые динамические свойства.

Список литературы

1. Березин Ю. А. Метод частиц в динамике разреженной плазмы. Новосибирск: Наука, 1980.

2. Григорьев Ю. Н., Вшивков В. А., Федорук М. П. Численное моделирование методами частиц в ячейках. Новосибирск: Изд-во СО РАН, 2004.

3. Kraeva M. A., Malyshkin V. E. Assembly Technology for Parallel Realization of Numerical Models on MIMD-Multicomputers // Int. J. on Future Generation Computer Systems, Elsevier Science. 2001. V. 17. N 6. P. 755–765.

4. Kireev S. E. A Parallel 3D Code for Simulation of Self-gravitating Gas-Dust Systems / PaCT-2009 Proc., Springer, 2009. LNCS 5698. P. 406–413.

5. Малышкин В. Э. Технология фрагментированного программирования // Вестник ЮУрГУ. Серия „Вычислительная математика и информатика“. 2012. № 46 (305). С. 45–55.

6. Киреев С. Е. [Электрон. рес.]. http://ssd.sscc.ru/en/dlb.

7. Walker D. W. Characterising the parallel performance of a large-scale, particle-in-cell plasma simulation code // Concurrency: Practice and Experience. 1990. V. 2 (4). P. 257–288.

8. Malyshkin V. E., Perepelkin V. A. The PIC implementation in LuNA system of fragmented programming // The Journal of Supercomputing, Springer. 2014. V. 69. I. 1. P. 89–97. [Electron. res.]. http://link.springer.com/journal/11227/69/1/page/1/

9. Malyshkin V. E. Assembling of Parallel Programs for Large Scale Numerical Modeling // Handbook of Research on Scalable Computing Technologies. IGI Global, USA, 2010. Chapter 13. P. 295–311.

10. Kale L. V. et al. Programming Petascale Applications with Charm++ and AMPI. Petascale Computing: Algorithms and Applications, Chapman & Hall: CRC Press, USA, 2008. P. 421–441.

11. Charm++ [Electron. res.]. http://charm.cs.uiuc.edu.

12. Shu W., Kale L. V. Chare Kernel – a Runtime Support System for Parallel Computations // J. Parallel. Distrib. Comput. 1991. V. 11 (3). P. 198–211.

13. Kireev S. E., Kuksheva E. A., Snytnikov A. V., Snytnikov N. V., Vshivkov V. A. Strategies for Development of a Parallel Program for Protoplanetary Disc Simulation / PaCT-2007 Proc., Springer, 2007. LNCS 4671. P. 128–139.

14. PaRSEC. [Electron. res.]. http://icl.cs.utk.edu/parsec.

15. DPLASMA. [Electron. res.]. http://icl.utk.edu/dplasma/

16. SMP Superscalar. [Electron. res.]. http://www.bsc.es/computer-sciences/programming-models/smp-superscalar.

17. OpenTS. [Electron. res.]. http://www.opents.net/index.php/ru.

18. Торгашев В. А., Царев И. В. Средства организации параллельных вычислений и программирования в мультипроцессорах с динамической архитектурой // Программирование. 2001. № 4. С. 53–68.

19. Малышкин В. Э., Корнеев В. Д. Параллельное программирование мультикомпьютеров. Новосибирск: Изд-во НГТУ, сер. „Учебники НГТУ“, 2006.

20. Краева М., Малышкин В. Динамические балансировки нагрузки в реализации PIC метода на MIMD мультикомпьютерах // Программирование. 1999. № 1. С. 47–53.

21. Hu Y. F., Blake R. J. An improved diffusion algorithm for dynamic load balancing // Parallel Computing, Elsevier Science. 1999. I. 4. P. 417–444.

22. Corradi A., Leonardi L., Zambonelli F. Performance Comparison of Load Balancing Policies Based on a Diffusion Scheme / Proc. of the Euro-Par'97 LNCS. V. 1300.

23. Malyshkin V. E. and Perepelkin V. A. Optimization methods of parallel execution of numerical programs in the LuNA fragmented programming system // The Journal of Supercomputing, Springer. 2012. V. 61. N 1. P. 235–248, DOI: 10.1007/s11227-011-0649-6

24. Вальковский В. А., Малышкин В. Э. Синтез параллельных программ на вычислительных моделях. Новосибирск: Наука, 1988.

25. Клини С. К. Введение в метаматематику. Изд-во Математика, 2009.

26. Мальцев А. И. Алгоритмы и рекурсивные функции. М.: Наука, 1965.

27. Rogers H. Theory of Recursive Functions and Effective Computability. N. Y.: McGraw-Hill, 1967.

28. Terekhov A. V. Parallel dichotomy algorithm for solving tridiagonal system of linear equations with multiple right-hand sides // Parallel Computing, Elsevier Science. 2010. V. 36 (8). P. 423–438.

29. Kireev S. and Malyshkin V. Fragmentation of numerical algorithms for parallel subroutines library // The J. of Supercomputing, Springer. 2011. V. 57. N 2. P. 161–171.

Малышкин Виктор Эммануилович – д-р технич. наук, профессор, зав. лабораторией Института вычислительной математики и математической геофизики СО РАН; e-mail: malysh@ssd.sscc.ru

Дата поступления — 07.07.2015