СТРУКТУРЫ СЕТОЧНЫХ АЛГОРИТМОВ И ИХ ОТОБРАЖЕНИЕ НА АРХИТЕКТУРУ МВС

В.П. Ильин

1. Введение

Последние десятилетия ознаменовались значительным прогрессом в быстродействии многопроцессорных вычислительных систем (МВС), характеризуемым массовым появлением компьютеров терафлопной производительности. Более того, уже с помощью скоростных телекоммуникационных сетей стали широко доступны МВС с сотнями вычислительных ядер, обеспечивающих производительность в десятки терафлопс. Нет сомнения в том, что эта тенденция будет и дальше развиваться по закону Мура, и такой бурный количественный рост приводит к проблеме неизбежных качественных изменений в параллельных технологиях крупномасштабного вычислительного эксперимента.

Дело в том, что развивающиеся МВС остаются на текущий момент в рамках типовой кластерной архитектуры: соединенные общей шиной вычислительные узлы, состоящие из определенного количества процессоров и/или ядер с общей многоуровневой памятью. Основные программные средства распараллеливания алгоритмов – это системы MPI и OpenMP для работы с распределенной и общей памятью соответственно. Создаваемые так же интенсивно GRID-системы предназначены в основном для интеграции разнородных вычислительных ресурсов и не должны, по-видимому, рассматриваться как орудие эффективного распараллеливания алгоритмов.

Лет 20 и более назад в тематике компьютерных архитектур наблюдалось заметное разнообразие: разрабатывались транспьютеры, обсуждались клеточные автоматы, нейросети и нейрочипы, исследовались специализированные процессоры. Но все они не выдержали конкуренции с универсальными компьютерами общего назначения.

Однако в последние годы ситуация начинает серьезно меняться. Появились многоядерные видеокарты, серьезно повышающие быстродействие формирование многоцветных и многоплановых графических изображений. А главное – благодаря современным кремниевым технологиям активно разрабатываются программируемые логические интегральные схемы (ПЛИС), которые делают реальной давнюю мечту математиков и программистов о создании компьютеров, ориентированных на экономичную реализацию конкретного класса задач или алгоритмов.

В этой связи становится особенно актуальной проблема углубленного структурного анализа сеточных алгоритмов (методов конечных элементов или конечных объемов – МКЭ, МКО), являющихся главным средством решения наукоемких задач математического моделирования. В этой области можно отметить следующие характерные тенденции:

- переход к междисциплинарным исследованиям взаимосвязанных процессов и явлений различной природы, приводящим к системам дифференциальных уравнений или соответствующим вариационным постановкам со многими неизвестными функциями;
- активизация обратных задач, связанных или с идентификацией параметров модели по имеющимся натурным измерениям, или с оптимизацией параметров какого-либо изучаемого или проектируемого научно-технического устройства; в общем случае такие проблемы связаны с условной минимизацией формулируемых целевых функционалов и многократным решением прямых задач с направленно изменяемыми данными;
- повышение “интеллектуальности” и логической сложности алгоритмов, за счет чего возрастает их конечная эффективность, но усложняется ситуация с непосредственным распараллеливанием на традиционных вычислительных платформах; наглядным примером является разнообразный набор многосеточных методов, обеспечивающий во многих случаях теоретически оптимальный порядок сходимости, но не ставший до сих пор основным практическим инструментом в промышленных программных продуктах.

Данная работа построена следующим образом. В п. 2 мы приводим графовое представление сеточных задач и алгоритмов с акцентами на функциональное разделение арифметических и обменных операций. Пункт 3 посвящен анализу примеров и общих проблем эффективности распараллеливания на типовых конфигурациях МВС. В заключение мы рассматриваем возможные пути конвергенции алгоритмических структур и компьютерных архитектур.

2. Сеточные задачи и алгоритмы: классификация и графовая интерпретация

Рассмотрим достаточно простую сеточную задачу, которую можно интерпретировать как полученную после неявной аппроксимации многомерной начально-краевой задачи для диффузионно-конвективного нелинейного уравнения на неструктурированной сетке, а позже отметим ее возможные обобщения и усложнения:

$$a_{i,i}^{m,n} u_i^n - \sum_{k \in \omega_i} a_{i,k}^{m,n} u_k^n = f_{i,k}^{m,n}, \quad i \in \Omega^h . \tag{1}$$

Здесь u_i^n – значение искомого решения в i-м узле пространственной сетки Ω^h на n-м шаге по времени, ω_i – сеточный шаблон i-го узла, т.е. совокупность соседних узлов, входящих в i-е сеточное уравнение (1), а $f_i^{m,n}$ и $a_{i,k}^{m,n}$ – правая часть и коэффициенты данного уравнения, которые зависят от пространственного и временного индексов i,n (более конкретно, естественно считать, что эти величины определяются только значениями u_i^n, u_i^{n-1} и $u_k^n, u_k^{n-1}, k \in \omega_i$). При этом предполагается, что исходная нелинейная задача на каждом n-м шаге как-то последовательно линеаризуется, а m означает номер итерации соответствующего уточняющего процесса. Обозначим через N_h количество узлов сетки Ω^h, N_ω – среднее число "соседей" для каждого i-го узла, N_n – количество временных шагов, которые необходимо рассчитать и N_m – среднее число "нелинейных" итераций на n-м шаге. Отметим, что, вообще говоря, система (1) несимметрична, т.е. $a_{i,k}^{m,n} \neq a_{k,i}^{m,n}$. Для наглядности на рис. 1 представлен плоский фрагмент сетки, где на каждом ребре изображен разделенный пополам прямоугольник, символизирующий прямые и обратные связи между узлами-соседями.

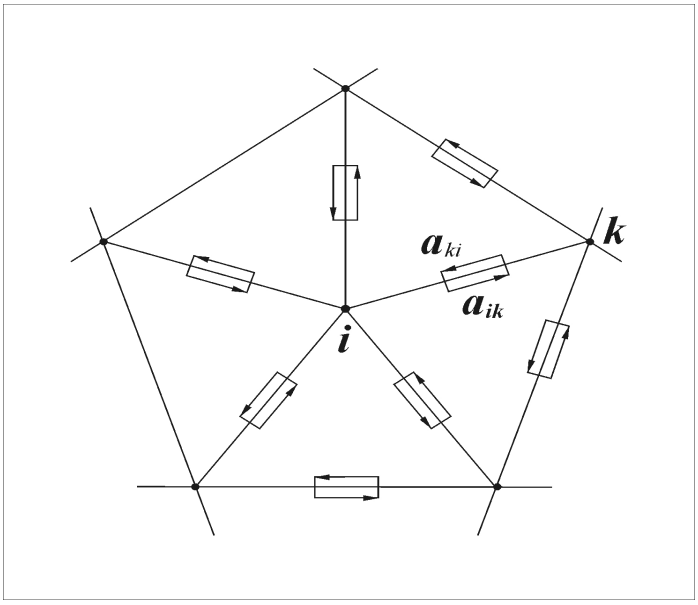

Рис. 1. Фрагмент пространственной сетки

Очевидно, что сетка представляет собой двунаправленный граф, в котором вершинам соответствуют величины $a_{i,i}, u_i$ и f_i, а ребрам – коэффициенты $a_{i,k}$ и $a_{k,i}$.

Укажем на некоторые типичные характеристики такой задачи. Размер сетки N_h может достигать $10^8, 10^9$ и более, количество временных шагов – до миллиона ($N_n \le 10^6$), а величины N_ω и N_m можно оценивать в несколько десятков.

Для каждых значений индексов m,n (которые далее опускаем) необходимо решать систему линейных алгебраических уравнений (СЛАУ)

$$Au = f, \quad u = \{u_i\}, \quad f = \{f_i\}, \quad A = \{a_{i,k}\}, \tag{2}$$

которую считаем квадратной размерности N_h. Число итераций N_{it}, необходимое для решения "плохих", но достаточно часто встречаемых на практике, СЛАУ можно оценить в 10^2 (вообще говоря, величина N_{it} с ростом N_h возрастает нелинейным образом, и поэтому данный этап является "узким горлышком" в решении больших задач). Таким образом, общий объем вычислений для решения задачи (1), (2), качественно

выражаемый как $N = N_h \cdot N_\omega \cdot N_m \cdot N_n \cdot N_{it}$, достигает величины 10^{20} и более, что составляет уже заслуживающую внимания задачу даже для МВС петафлопной производительности.

Основным подходом к распараллеливанию рассматриваемых задач является метод декомпозиции, заключающийся в разбиении расчетной (и сеточной) области на подобласти, одновременном решении задач в подобластях (на разных процессорах), последующем обмене необходимой информации между соседними подобластями и проведении дальнейших итерационных уточнений. Иллюстрация простой двумерной декомпозиции приведена на рис. 2.

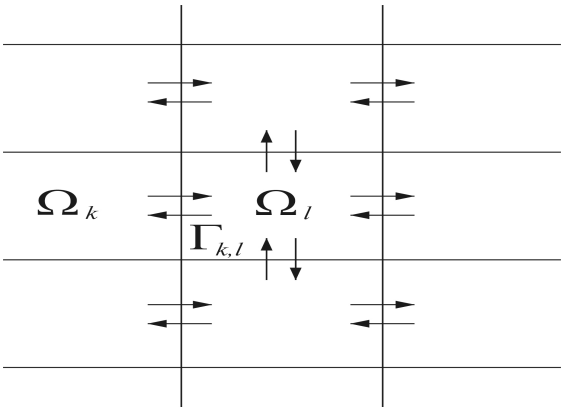

Рис 2. Иллюстрация двумерной декомпозиции

Любая декомпозиция образует некий макрограф, вершинами которого являются подобласти, а ребрами – информационные связи между ними. В соответствии с декомпозицией можно переписать в блочном виде СЛАУ (2), после чего u_i и f_i будут являться подвекторами, а $a_{i,k}$ – матричными блоками. Время работы каждого процессора на одной итерации можно считать примерно пропорциональным объему (числу узлов) соответствующей подобласти, а длительность межпроцессорных обменов – пропорциональным поверхности общей части границы соседних подобластей.

Переходя к классификации сеточных задач, следует выделить три основных класса постановок и соответствующих методов их решения:

- краевые задачи с разделяющимися переменными, аппроксимируемые на регулярных многомерных сетках, представляющих собой тензорное произведение одномерных сеток; возникающие при этом СЛАУ с “хорошими” структурными и спектральными свойствами допускают применение сверхбыстрых методов быстрого преобразования Фурье, циклической редукции и некоторых других специальных алгоритмов;
- более общего вида задачи с переменными коэффициентами функциональных уравнений и сложными конфигурациями расчетных областей, приводящие к алгебраическим системам с разреженными матрицами, имеющими нерегулярные портреты (расположение ненулевых элементов) и хранящиеся с помощью универсальных сжатых форматов; это дает значительную экономию памяти, но приводит к существенному снижению производительности алгоритмов из-за усложненного доступа к значениям матричных элементов;
- к третьей группе мы относим различного типа задачи с дополнительными факторами, значительно усложняющими их вычислительную сложность; это могут быть междисциплинарные проблемы, в которых, например, величины u_i и f_i из (1) суть векторы, что многократно увеличивает общую размерность СЛАУ; другой пример – обратные задачи, в которых система (1) есть только часть общего оптимизационного процесса; наконец, нетривиальными являются нестационарные задачи с динамическими сетками, в которых меняются топологические связи между узлами и вся структура сеточной задачи.

Мы не останавливаемся на таких этапах математического моделирования, как дискретизация расчетной области и аппроксимация исходных функциональных уравнений на построенной сетке. Отметим, что универсальные конечно-элементные технологии с формированием локальных и сборки глобальных матриц в МКО и МКЭ позволяют реализовывать хорошо масштабируемое распараллеливание. Наличие временных и/или нелинейных зависимостей при этом, как правило, только увеличивает вычислительную сложность задачи, связанную с многократным пересчетом коэффициентов в уравнениях вида (1), но не приводит к ухудшению производительности МВС.

В заключение данного пункта можно сказать, что несмотря на многообразие рассматриваемых вычислительных проблем, на повестке дня стоит выработка унифицированной технологии для распараллеливания совокупности типовых подзадач, на основе их структурного или модульного анализа. Конечной целью таких исследований является формирование “доктрины” по расширяемому математическому

инструментарию, ориентированному на эффективную и автоматизированную поддержку отображения сеточных алгоритмов не только на существующие, но и на будущие компьютерные архитектуры. Несомненно, такой систематический анализ должен дать и обратный эффект, в смысле влияния на пути развития реконфигурируемых МВС.

3. Эффективность распараллеливания: от частных примеров к общим проблемам

Эффективность распараллеливания оценивается двумя главными критериями – ускорением и коэффициентом использования процессоров:

$$S_p = T_1 / T_p, \quad E_p = S_p / p, \tag{3}$$

где T_p – время решения задачи (или реализации алгоритма) на p процессорах.

Если $R_p = p$ и $E_p = 1$, то ускорение называется линейным, и такой случай можно считать идеальным. На практике зачастую приходится мириться с ситуацией $E_p \le 0.5$, хотя случай $E_p < 0.1$ уже обычно рассматривается как "криминальный". Иногда удается достигать и сверхлинейного ускорения $S_p > p,\ E_p > 1,$ вследствие неоднородности памяти МВС (например, данные о задаче могут не умещаться в кэше одного процессора, но они могут разместиться в кэшах 20 процессоров, что приведет к резкому сокращению времени расчета.

Время реализации задачи T_p складывается (в первом приближении) из времени выполнения арифметических действий T_p^a и времени межпроцессорных обменов T_p^c, которые рассматриваются как коммуникационные потери. При синхронной работе процессоров эти времена оцениваются (очень грубо) с помощью выражений

$$T_p^a \approx N_a \tau_a, \quad T_c = N_0 \tau_0 + N_c \tau_c, \tag{4}$$

где N_a – общее число арифметических операций, выполняемых одним процессором, N_0 – количество информационных обменов, N_c – общий объем передаваемых данных (в нашем случае это число пересылаемых вещественных чисел с двойной стандартной точностью), а τ_a, τ_0, τ_c – среднее время выполнения одной арифметической операции, время задержки (настройки) при реализации одной передачи и время пересылки одного числа соответственно.

Как правило, наиболее быстрые в традиционном (однопроцессорном) смысле алгоритмы наихудшим образом распараллеливаются, так что вопросы наилучшего выбора алгоритма решения задачи и его отображения на МВС заданной архитектуры составляют слабоформализованную постановку и требуют в значительной степени профессиональной интуиции математика и программиста. Тем более, что реальные модели функционирования многоядерных вычислительных систем с неоднородной памятью фактически отсутствуют, а методические исследования производительности различных вариантов распараллеливания – это сугубо экспериментальная работа методом проб и ошибок.

В качестве простого и достаточно актуального примера рассмотрим задачу решения большого количества $M >> 1$ трехдиагональных СЛАУ одинакового порядка $N >> 1$:

$$-a_k u_{k-1} + b_k u_k - c_k u_{k+1} = f_k, \quad k = 1, \ldots, N, \quad a_1 = c_N = 0, \tag{5}$$

которая возникает во многих приложениях. Здесь из одним наиболее хорошо распараллеливаемых является метод редукции без обратного хода [11], [12].

Мы рассмотрим блочный двухуровневый алгоритм редукции, являющийся обобщением известных подходов и ориентированный на эффективную реализацию на кластере, состоящем из большого числа $(P >> 1)$ многоядерных вычислительных узлов.

Пусть множество индексов $\Omega = \{k\}$, каждый из которых ассоциируем с узлом одномерной сетки, разбито на P сеточных подмножеств (подобластей) $\Omega_l = \{k_{l-1} < k \le k_l,\ l = 1, \ldots, P\}$. Схема декомпозиции сеточной области, представленная для $P = 4$ на рис. 3 верхней линией, может интерпретироваться также как

разбиение матричного графа A из (5). Для простоты будем считать, что число узлов (и неизвестных) во всех подобластях одинаково и равно $m = N/P$ (на рис. 3 $m = 4$).

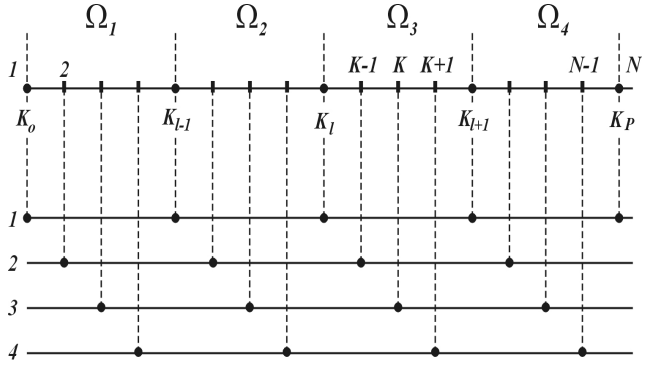

Рис. 3. Схема декомпозиции для блочного метода редукции

В каждой из подобластей любая неизвестная с помощью исключения может быть выражена через значения u_{k_l} в граничных точках:

$$u_k = \alpha_k^{(l)} u_{k_{l-1}} + \beta_k^{(l)} u_{k_l} + \gamma_k^{(l)}, \quad k \in \Omega_l, \tag{6}$$

где коэффициенты $\alpha_k^{(l)}, \beta_k^{(l)}, \gamma_k^{(l)}$ выражаются с помощью простых формул через a_k, b_k, c_k, f_k.

В свою очередь, с помощью такого типа соотношений можно исключить значения u_{k_l} в узлах-разделителях и получить m несвязанных трехдиагональных СЛАУ P-го порядка следующего вида:

$$-A_{j_l} u_{j_l - m} + B_{j_l} u_{j_l} - C_{j_l} u_{j_l + m} = g_{j_l}, \quad j_l = k_{l-1} + 1, \ldots, k_l, \quad l = 1, \ldots, P, \tag{7}$$

где значения $A_{j_l}, B_{j_l}, C_{j_l}, g_{j_l}$ вычисляются через $\alpha_k^{(l)}, \beta_k^{(l)}, \gamma_k^{(l)}$.

Каждая из систем (8) может быть решена независимо на "своем" процессоре с помощью экономичного метода прогонки. В итоге мы получим искомое решение, разнесенное по m процессорам.

Остановимся теперь на вопросах производительности описанного алгоритма с учетом коммуникационных потерь, в предположении использования арифметики со стандартной двойной точностью. Будем считать, что СЛАУ (5) получены после применения двойного быстрого преобразования Фурье (БПФ, см. [12]) к решению трехмерного уравнения Пуассона на сетке с числом узлов $N \times M$, где M – число узлов в плоскости, перпендикулярной оси, вдоль которой определены значения u_k в (5). Для простоты можно допустить, что трехмерная сетка имеет $N \times N \times N$ узлов к $M = N^2$. Тогда общее число действий при решении N^2 систем на одном процессоре методом прогонки составляет $c_1 N^3$ ($c_1 \approx 8$, здесь и далее величины $c_q, q = 1,2,\ldots$). При $N = 10^3$, например, такой объем вычислений сравнительно невелик, т.е. их реализация на одном процессоре (при наличии достаточной оперативной памяти) с быстродействием 1 гигафлопс (гф) заняла бы время $T_a = c_1$ сек. Однако типичный объем памяти такого процессора составляет 1 гигабайт (гб), что соответствует примерно 10^8 вещественным числам с двойной точностью, в то время как хранение только самих СЛАУ требует $4 \cdot 10^9$ чисел.

Сделаем теперь естественное предположение, что исходные данные по уравнениям (5) расположены в памяти P процессоров так, что в l-м процессоре хранятся значения $a_k, b_k, c_k, f_k, k \in \Omega_l \, (k_{l-1} < k \le k_l)$ для всех N^2 СЛАУ. В этом случае при формировании исключающих соотношений (6) никаких межпроцессорных обменов не нужно, но построение отдельной системы вида (7) требует, во-первых, пересылки информации объемом $c_2 m N^2$ между соседними процессорами для вычисления значений $A_{j_l}, B_{j_l}, C_{j_l}, g_{j_l}$ в сеточных узлах "своей" l-й подобласти, а во-вторых, обмены объема $c_3 P N^2$ для сбора

исходных данных о каждой из mN^2 СЛАУ (7) в каком-то одном процессоре. Только после этого все редуцированные СЛАУ P-го порядка могут одновременно решаться методом прогонки.

Отметим теперь основные структурные особенности алгоритма, существенные с точки зрения эффективности распараллеливания:

- вычисление параметров соотношений (6) в одной подобласти фактически требует решения $M = N^2$ трехдиагональных подсистем m-го порядка, т.е. проведения $c_4 mN^2$ действий; соответствующие арифметические операции распараллеливаются идеально, т.е. с линейным ускорением, если они выполняются с помощью OpenMP на m-ядерном узле с общей памятью; с учетом независимости этих действий во всех P подобластях Ω_l данный этап реализуется с коэффициентом эффективности $E = 1$ на P вычислительных узлах с общим количеством ядер mP, а время его исполнения составляет $T_1 = \tau_a c_4 M$;
- объем арифметических действий на этапе расчета коэффициентов СЛАУ (7) для каждой Ω_l равен $c_5 mN^2$, а их параллельная реализация идеально масштабируется также на mP-ядерной МВС (при этом время исполнения равно $T_2 = \tau_a c_5 M$ и не зависит от m); критичными в данном случае являются два указанных выше типа обменов (с соседними процессорами и "каждый с каждым"), схематически изображенных на рис. 4.

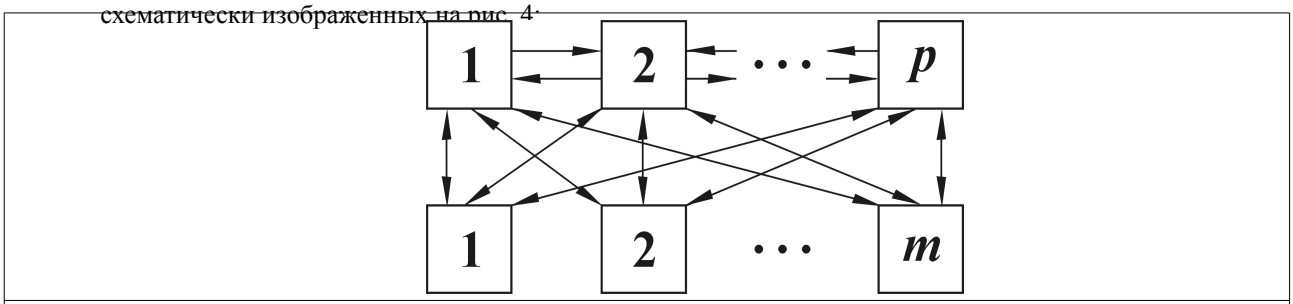

Рис.4. Схема коммуникаций для двухуровневого метода редукции

- если все mM системы вида (7) и порядка P сформированы на отдельных процессорах, то для решения каждой из них необходимо всего лишь $c_6 P$ операций, а параллельная реализация данного этапа на mP вычислительных ядрах требует времени $T_3 = \tau_a c_6 M$; однако для приведения полученного сеточного решения в исходный вид "по подобластям" надо из узлов нижнего ряда на рис. 4 сделать "обратную" пересылку значений u_{l_j} в соответствующие процессоры Ω_l.

На основе проведенного анализа можно сделать следующие выводы. Во-первых, при $N >> P >> m >> 1$ все непосредственно вычислительные этапы на кластере, состоящем из P m-ядерных узлов, идеально масштабируются при различных значениях m и P, однако значительная потеря производительности может произойти из-за коммуникационных потерь, вследствие необходимости поддержки не только близких, но и дальних межпроцессорных обменов. Во-вторых, данное утверждение остается в силе при пропорциональном увеличении (или уменьшении) значений N, P и m. В третьих, при фиксированных значениях размеров сетки M и N вопрос о повышении производительности МВС при увеличении числа узлов P и ядер m очевидным образом требует специальных исследований с учетом конкретных особенностях архитектуры. При этом стратегии и тактики распараллеливания могут быть различными в зависимости от того, ставится ли целью минимизация общего календарного времени решения задачи или повышение коэффициента эффективности использования МВС.

В качестве примера можно рассмотреть два альтернативных варианта из описанной выше схемы. Первый соответствует случаю $m = P$, когда соответствующие вычислительные узлы из верхнего и нижнего рядов на рис. 4 можно фактически совместить. Во второй версии полагается $m = 2$, что приводит к четно-нечетной редукции, в результате которой получаем две независимые подсистемы порядка $P/2$. Из них, в свою очередь, с помощью аналогичной процедуры можно получить четыре подсистемы порядка $P/4$. А продолжая

такой процесс до логического конца, придем за $\log_2 P$ шагов к P уравнениям первого порядка, т.е. фактически получим искомое решение. Такая последовательность требует на каждом q-м этапе редукции пересылок информации от l-го процессора к $\left(l \pm 2^q\right)$-м процессорам. Эти коммуникационные потери грозят стать значительными, но их можно было бы избежать путем организации конвейерной реализации уровней редукции (что на кластерной архитектуре является проблематичным).

Перейдем теперь к рассмотрению аналогичных проблем при решении общего вида многомерных задач методом декомпозиции, ограничиваясь ссылками к иллюстрации двумерного случая на рис. 2. Мы остановимся на наиболее содержательных алгебраических аспектах, связанных с решением сеточных СЛАУ вида (1) или (2), в которых матрицы являются неструктурированными и имеют переменные коэффициенты, что исключает возможность применения быстрых прямых алгоритмов. Обозначая через $\bar{u}_l$, $\bar{f}_l$ подвекторы решения и правой части, ассоциированные с подобластью Ω_l, решаемую систему запишем в блочной форме

$$A_{l,l}\bar{u}_l - \sum_{k \in \omega_l} A_{l,k}\bar{u}_k = \bar{f}_l, \quad l = 1,\ldots,L, \tag{8}$$

где L есть число подобластей, а ω_l – множество подобластей, соседних к Ω_l.

Общая схема параллельных алгоритмов итерационного решения задачи (8), при всем многообразии их модификации, единообразна и проста: формируются независимые подзадачи в подобластях, реализуется их параллельное решение на соответствующих процессорах, производятся межпроцессорные обмены, соответствующие моделируемым условиям сопряжения на внутренних границах, осуществляется циклическое повторение вышеуказанных этапов и управление итерационным процессом, который в общей форме можно записать как

$$\bar{u}^{n+1} = T\bar{u}^n + \bar{g}, \quad \bar{u}^n = \left\{ \bar{u}_l^n \right\}, \tag{9}$$

где матрица перехода T и вектор правой части $\bar{g}$ в явном виде реально не фигурируют. Отметим, что если $\bar{u}^n$ сходятся к искомому вектору $\bar{u}$, то итерации (9) фактически реализуют решение предобусловленной системы

$$\bar{A}\bar{u} \equiv \left(I - T\right)\bar{u} = \bar{g} \tag{10}$$

и могут быть легко ускорены каким-то из методов в подпространствах Крылова, где реализация каждого шага сводится к независимым решениям задач в подобластях Ω_l. Если последние, в свою очередь, рассчитываются итерационным алгоритмом, что наиболее реально для больших проблем, то в итоге получаем двухуровневый итерационный процесс. Очевидно, что векторные операции в методах сопряженных направлений (для симметричных СЛАУ), как и алгоритмы полу- или бисопряженных направлений распараллеливаются естественным образом практически с линейным ускорением.

Не останавливаясь на методических аспектах, которые могут повлиять на количество внешних и внутренних итераций, обратим главное внимание на возможные коммуникационные потери. При этом будем предполагать, что исходные данные по СЛАУ (8) уже получены и располагаются в "своих" процессорах $\Omega_l, l = 1,\ldots,L$. Здесь существенны следующие моменты общего характера для многомерной декомпозиции, в которой подобласти, или диагональные матричные блоки, являются вершинами, а информационные связи между ними – ребрами графа, отображающего топологическую структуру сеточной расчетной области.

а) Реализация метода декомпозиции требует обменов только между ближними (соседними в геометрическом смысле) подобластями, что можно объяснить простейшим итерационным характером используемого подхода.

б) Естественное отображение алгоритма на архитектуру МВС заключается в том, что каждой подобласти сопоставляется вычислительный m-ядерный узел, что позволяет распараллеливать расчет каждой подзадачи в Ω_l без обменов.

в) Межпроцессорные обмены на каждой внешней итерации могут и должны быть совмещены по времени с вычислениями для внутренних узлов подобластей.

4. Заключение. Предпосылки к конвергенции алгоритмических структур и компьютерных архитектур

Традиционно проблема создания хорошо распараллеливаемых численных методов и программ формулируется как отображение алгоритмов на архитектуру МВС, причем последняя рассматривается как заданная "свыше". Такой подход сложился исторически с тех времен, когда software рассматривался как почти бесплатное приложение к дорогостоящему "хардверу". Однако современные технологические возможности

быстро и достаточно дешево создавать реконфигурируемые и программируемые компоненты интегральных схем позволяют обсуждать вопросы встречного движения вычислителей и компьютерных конструкторов. Здесь важно сформулировать алгоритмические "технические задания", выполнение которых действительно технически реализуемо, позволяет значительно ускорить ресурсоемкое моделирование и, что не менее важно, найдет широкое применение. Очевидно, что серийное производство "коммуникационных монстров" из сотен или тысяч соединенных каждый с каждым процессоров экономически не оправдано, да и не будет востребовано.

В целом развитую МВС можно представить в виде графа, определяемого характеристиками своих вершин, ребер и топологией. Например, изображенный на рис. 1 фрагмент математической сетки определяется только сеточными значениями функций u_k и f_k, коэффициентами $a_{i,k}$ и $a_{k,i}$, и арифметической формулой (1). Для полноты сюда надо добавить нелинейные зависимости между указанными локальными величинами и, возможно, дополнительные дальние взаимосвязи.

С точки зрения сеточных задач и алгоритмов, топология коммуникаций МВС, в дополнение к существующим общим шинам, должна обеспечивать быстрые ближние связи в одномерных, двумерных и трехмерных сетях, а также отдельные дальние связи. В идеальном варианте конфигурация графа должна быть настраиваемой (программируемой) и даже динамической, т.е. изменяемой во время счета.

Что касается арифметических действий и их операндов, то сейчас общепринятым и оправданным при решении больших задач является использование стандартной двойной точности с 64-битовым представлением вещественного числа с плавающей точкой. Однако зачастую такая точность излишня, а иногда – недостаточна, и в последнем случае сложным программным образом реализуется арифметика с большим числом знаков. Проблема выбора арифметики с переменной длиной машинного слова, несомненно, актуальна, но требует еще серьезного изучения, в первую очередь теоретического. Определенно можно сказать, что демонстрируемые в некоторых публикациях результаты нейросетевого моделирования физических процессов с использованием "короткой" арифметики и нейрочипов без их обоснования вызывают большое сомнение.

Большим резервом повышения производительности МВС является углубленная конвейеризация вычислений, буферизация и синхронизация обменов данными, создание спецустройств для быстрой реализации различных нелинейных функций и многое другое. Можно только констатировать, что технологии ПЛИС и их дальнейшее развитие открывают новое широкое поле научной деятельности.

Работа поддержана грантом РФФИ № 08-01-00526 и грантом Президиума РАН № 2.5.

ЛИТЕРАТУРА:

1. Ильин В.П. Методы конечных разностей и конечных объемов для эллиптических уравнений.–Новосибирск: Изд. ИВМиМГ СО РАН, 2001.
2. Ильин В.П. Методы и технологии конечных элементов.–Новосибирск: Изд. ИВМиМГ СО РАН, 2004.
3. Saad Y. Iterative methods for sparse linear systems.–PWS Publishing: New York, 1996.
4. Воеводин В.В. Отображение проблем вычислительной математики на архитектуру вычислительных систем.–Вычислительные методы и программирование, т. 1, 2000, 105-112.
5. Воеводин Вл.В. Курс лекций "Параллельная обработка данных".–http://parallel.ru/vvv/
6. Ильин В.П. Проблемы высокопроизводительных технологий решения больших разреженных СЛАУ. Вычислительные методы и программирование.–М., изд. МГУ, т. 10, N 1, 2009, 130-136.
7. Ильин В.П. Параллельные алгоритмы для больших прикладных задач: проблемы и технологии.// Автометрия, т.43, N 2, 2007, 3-21.
8. Богачев К.Ю. Основы параллельного программирования.–М.: БИНОМ. Лаборатория знаний, 2003.
9. IntelRMath Kernel Library 10.0–Overview.http://www.intel.com/cd/software/products/asmona/eng/307757.htm.
10. Каляев И.А., Левин И.И. Семейство реконфигурируемых вычислительных систем с высокой реальной производительностью. Вычислительные методы и программи- рование.–М., изд. МГУ, т. 10, N 1, 2009, 207-214.
11. Ильин В.П., Кузнецов Ю.И. Трехдиагональные матрицы и их приложения. – М.: Наука, 1985.
12. Ильин В.П., Кныш Д.В. Параллельные алгоритмы решения разделяющихся краевых задач.–Санкт-Петербург, Труды Конф. ПАВТ-2008, изд. Политехн. Ун-та (СПбПГУ), 2008, 107-118.