РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ

А В Т О М Е Т Р И Я

2002, том 38, № 4

МОДЕЛИРОВАНИЕ В ФИЗИКО-ТЕХНИЧЕСКИХ ИССЛЕДОВАНИЯХ

УДК 681.3

В. А. Вшивков, Г. А. Тарнавский, Е. В. Неупокоев

(Новосибирск)

РАСПАРАЛЛЕЛИВАНИЕ АЛГОРИТМОВ ПРОГОНКИ: МНОГОЦЕЛЕВЫЕ ВЫЧИСЛИТЕЛЬНЫЕ ЭКСПЕРИМЕНТЫ*

Рассматриваются способы сегментирования расчетной области на ряд подобластей при распараллеливании алгоритмов прогонки и методы организации параллельных вычислений. Проводятся многоцелевые вычислительные эксперименты по определению эффективных способов параллелизации и особенностей их реализации на различных мультипроцессорных системах.

Введение. Различные процедуры прогонки для решения систем алгебраических уравнений – дискретных представлений дифференциальных уравнений – являются часто используемым сегментом в алгоритмах задач механики сплошной среды, в частности, в области аэродинамики и физической газовой динамики. Операции выполнения этих процедур в ряде методов (см., например, [1]) интегрирования сложных нелинейных систем дифференциальных уравнений Эйлера, Навье – Стокса, Барнетта зачастую являются доминирующими (или, по крайней мере, существенными) с точки зрения затрат вычислительных ресурсов. Поэтому такие процедуры требуют применения специальных методов реализации, сокращающих эти затраты. Здесь отметим два возможных подхода. Первый является традиционным и связан с улучшением характеристик собственно алгоритмов (см., например, [2]). Второй позволяет кардинально снизить требуемое для вычислений астрономическое время и связан с применением новых компьютерных технологий – параллелизацией операций прогонки, что делает возможным решение задач нового уровня, ранее недоступных по чисто техническим причинам. Однако это направление приводит к достаточно сложной аналитике организации прогонки в случаях, когда расчетная область «разрезана» на сегменты, решение в каждом из которых обеспечивает отдельный процессор, с необходимо-

* Работа выполнена при поддержке Российского фонда фундаментальных исследований (гранты № 02-01-00864, № 00-07-90297, № 01-01-00781).

стью «сшивки» решений на границах сегментов [3]. Практическая реализация (см., например, [4]) методов распараллеливания прогонки приводит к многочисленным собственным весьма существенным проблемам с необходимостью их всестороннего анализа. В связи с этим следует согласиться, что «гигантская производительность параллельных компьютеров с лихвой компенсируется сложностью их использования» [5]. Анализу этих проблем посвящена данная работа.

Постановка задачи. В двумерной области $R = X \otimes Y$, $X \in [0,1]$, $Y \in [0,1]$, которая покрыта равномерной дискретной сеткой x_i ($i \in [1, ia]$) $\otimes$ $\otimes\, y_j$ ($j \in [1, ja]$), проводится численное решение уравнения Пуассона

$$\Delta\Phi(x,y) = \rho(x,y), \tag{1}$$

где $\Phi(x,y)$ – неизвестная функция с заданной правой частью $\rho(x,y)$ и некоторыми граничными условиями (задача Дирихле или Неймана). Конкретный вид операторной записи (1) уравнения Пуассона для дальнейших рассуждений значения не имеет и здесь не приводится. Метод расщепления типа [1] приводит решение (1) к алгоритму дробных шагов (детали опускаются):

$$\frac{\partial^2 G}{\partial x^2} = \rho_x(x,y), \tag{2}$$

$$\frac{\partial^2 H}{\partial y^2} = \rho_y(x,y). \tag{3}$$

В дискретном (сеточном) пространстве решение (2), (3) сводится к решению системы алгебраических уравнений:

$$A^1 G_{i-1,j} + B^1 G_{i,j} + C^1 G_{i+1,j} + D^1 = 0, \tag{4}$$

$$A^2 H_{i,j-1} + B^2 H_{i,j} + C^2 H_{i,j+1} + D^2 = 0. \tag{5}$$

Здесь $A^k = \left\| A_{ij}^k \right\|$, $B^k = \left\| B_{ij}^k \right\|$, $C^k = \left\| C_{ij}^k \right\|$, $D^k = \left\| D_{ij}^k \right\|$, $k = 1,2,$ – известные матричные коэффициенты.

Решение (4) проводится прогонками по X-направлению, решение (5) – по Y-направлению. Количество операций в (4) и (5) одинаково, если одинаковы размерности используемых сеток ia и ja, которые варьировались в диапазоне от 500 до 3000.

В операциях (4), (5) имеет место «сцепление» значений вычисляемой функции, причем следует обратить внимание на важную особенность алгоритма: при вычислении массива G значения его элементов «связаны» по первому индексу и независимы по второму, а при вычислении массива H значения его элементов «связаны» по второму и независимы друг от друга по первому индексу. Другими словами, элементы массива G, расположенные вдоль одного столбца матрицы, являются зависимыми между собой при полной независимости всех столбцов друг от друга. Аналогично элементы массива H, расположенные вдоль одной строки матрицы, являются зависимыми между собой при полной независимости всех строк друг от друга.

Нахождение элементов массивов G и H, т. е. реализация (4), (5), приводит к решению систем уравнений методом прогонки:

$$G_{ij} = \left\{ I - \text{INVERSION} \begin{pmatrix} i = \overline{1, ia} \\ i = \overline{ia, 1} \end{pmatrix} \right\}_j , \tag{6}$$

$$H_{ij} = \left\{ J - \text{INVERSION} \begin{pmatrix} j = \overline{1, ja,} \\ j = \overline{ja, 1} \end{pmatrix} \right\}_i . \tag{7}$$

Символическая запись (6) и (7) означает следующее. Для операции (6) при любом значении ндекса j (в области своего определения) осуществляется двухшаговая процедура прогонки: на первом шаге при последовательном возрастании индекса i от 1 до ia вычисляются «бегущим счетом» массивы прогоночных коэффициентов α_i и β_i; на втором шаге при последовательном убывании индекса i от ia до 1 вычисляются значения G в i-узлах. Граничные условия задачи ставятся для прогоночных коэффициентов при $i = 1$ и для функций при $i = ia$. Таким образом, в сеточном пространстве проводятся операции вдоль любого столбца сначала «снизу вверх» (прогоночные коэффициенты), затем «сверху вниз» (функции) с последовательным перебором всех столбцов.

Для операции (7) процедура «развернута на 90 градусов». При любом значении индекса i (в области своего определения) осуществляется двухшаговая процедура прогонки: на первом шаге при последовательном возрастании индекса j от 1 до ja вычисляются «бегущим счетом» массивы прогоночных коэффициентов γ_j и δ_j; на втором шаге при последовательном убывании индекса j от ja до 1 вычисляются значения H в j-узлах. Граничные условия задачи ставятся для прогоночных коэффициентов при $j = 1$ и для функций при $j = ja$. Таким образом, в сеточном пространстве проводятся операции вдоль любой строки сначала «слева направо» (прогоночные коэффициенты), затем «справа налево» (функции) с последовательным перебором всех строк.

Подчеркнем, что перебор столбцов для (6) и перебор строк для (7) может осуществляться в произвольном порядке, даже неупорядоченно вследствие их независимости. Это прямо указывает на существование естественных способов распараллеливания операций в индексном пространстве: по индексу j для (6) и индексу i для (7).

Для (6) возможно естественное использование числа процессоров

$$P_c = ja, \tag{8}$$

для (7)

$$P_c = ia. \tag{9}$$

При ограниченном числе процессоров, например, $P_c \ll ja$ и $P_c \ll ia$, с невозможностью выполнения (8), (9) требуется разбиение индексного пространства на P_c вертикальных полос для (6) и P_c горизонтальных полос для (7).

Эта параллелизация в использованных выше терминах является «крупноблочной»; возможно также применение «мелкозернистого» распараллеливания операций внутри собственно процедуры прогонки при проведении внутренних операций (6), (7). В принципе при использовании суперкомпьютеров с существенным числом процессоров возможно одновременное применение этих двух способов.

Следует кратко остановиться и еще на некоторых возможных способах распараллеливания прогонки. В [4] предложен эффективный в теоретическом плане метод параллелизации, учитывающий порядок проведения операций в прогонке по двум направлениям, позволяющий почти одновременно, с некоторым определенным «запаздыванием», проводить прогонки сразу в двух направлениях индексного пространства. Сначала выполняется шаг «снизу вверх», а затем после первого такта шага «сверху вниз» прогонки «по столбцам» можно начинать шаг «слева направо» прогонки «по строкам». Однако этот метод требует высокой синхронизации всего процесса и, по-видимому, пока трудно реализуем на практике.

В ряде случаев оказывается необходимым применение глобального (крупноблочного) сегментирования расчетной области не на полосы, идущие от границы до границы, а на клетки, «разрезающие» область на подобласти как прилегающие к границам, так и находящиеся внутри нее. Этот способ прерывает последовательное проведение операций от границы к границе, на которых существуют естественные физико-математические граничные условия. Внутри области возникают искусственные границы, на которых нет естественных граничных условий задачи. В [3] предлагаются некоторые процедуры «сшивки» операций прогонки на этих внутренних границах. Несмотря на некоторую искусственность и сложность в реализации, данный метод может оказаться небесполезным, а в ряде задач даже безальтернативным.

Для оценки ускорения q, которое возможно получить на компьютере из p процессоров, можно воспользоваться законом Амдала:

$$q = \frac{k}{f + (1-f)/p}, \tag{10}$$

где f – доля операций, требующих последовательного выполнения. Интервал изменения $f \in [0,1]$, крайние случаи в значениях f соответствуют полностью параллельным ($f = 0$) и полностью последовательным ($f = 1$) программам. Заметим, что даже при $p \to \infty$ рост ускорения ограничен пределом $q \to 1/f$, и если в некоторых подпрограммах, составляющих код, значение f велико, то следует заняться «расшивкой этих узких мест». Множитель k в законе Амдала есть коэффициент, корректирующий (уменьшающий, $k \in [0,1]$) возможное ускорение вычислительного процесса из-за задержек, связанных с маршрутизацией, обменом и подкачкой данных, в том числе режимов ожидания с возможностью «зависания» некоторых процессоров и возникновением тупиковых ситуаций. Этот коэффициент весьма эмпиричен, и его более или менее точное значение может быть определено лишь в вычислительном эксперименте на конкретном алгоритме в конкретной многопроцессорной системе. Поэтому (10) достаточно точно дает значение q лишь в «идеальном» ($k = 1$) смысле, без учета замедления вычислительного процесса из-за обменов данными.

Это замедление зависит от значительного числа факторов (о некоторых см. далее), в частности не в последнюю очередь от опыта и квалификации программиста, необходимых для тщательного согласования структуры алгоритмов и реализующих их кодов с особенностями архитектуры параллельных вычислительных систем.

В заключение этого раздела рассмотрим некоторый технологический аспект проблемы, связанный с порядком выборки данных из общей памяти и пересылкой их в локальную память каждого процессора при параллелизации прогонки. В кодировке языка Си, который используется для численных экспериментов, развертка двумерных массивов в одномерные осуществляется «по строкам». В связи с этим при проведении распараллеливания «по столбцам» в операции (6) будет осуществляться пересылка данных, идущих подряд в общей памяти, а при распараллеливании «по строкам» в операции (7) требуется выборка из общего массива элементов, адрес каждого из них смещен по памяти (для двумерных таблиц смещение равно размерности первого индекса, для трехмерных – зависит от расположения и размерностей индексов). Потери на проведение выборки могут быть велики и существенно снижать общую производительность. Кроме того, в зависимости от архитектуры конкретного компьютера могут появляться задержки, являющиеся прямым следствием ограниченной пропускной способности каналов передачи данных, и при операциях с векторными функциями типа G и H требуется специальная организация интерфейсов, иначе часть времени (иногда значительная) будет неизбежно потрачена впустую на ожидание подкачки для запуска операций с зацеплением.

Необходимая для этого глобальная реорганизация хорошо функционирующего на однопроцессорной ЭВМ комплекса программ, отлаженного в течение длительного времени ценой больших интеллектуальных усилий, может оказаться неприемлемой для разработчиков как в психологическом аспекте, так и в чисто практическом плане, когда «проще и удобнее» создать новые методы, чем адаптировать для многопроцессорных вычислительных систем старые алгоритмы.

Некоторые вычислительные эксперименты. Для анализа возможных ускорений различных типов прогонок был проведен ряд численных экспериментов на двух многопроцессорных вычислительных системах.

Вычислительный кластер Института вычислительных технологий (ИВТ) СО РАН состоит из восьми узлов со следующей конфигурацией: M/b ASUS P3B-F 440BX; CPU Intel Pentium-III 550 MHz; DIMM 128 Mb; SDRAM SPD 100 MHz; HDD 6.4 Gb (сервер), 7 HDD 1 Gb. Коммуникационное и сетевое оборудование: Switch Intel InBusiness SS101TX8EU; 8Ether-Express PRO100. Программное обеспечение: операционная система Linux RedHat 7.0; MPICH 1.2.0. Пиковая производительность составляет 1 GF/s.

Вычислительная система Института вычислительной математики и математической геофизики СО РАН Siemens Nixdorf состоит из двух SMP-серверов RM600 E30 с общим полем оперативной памяти 4.2 Gb и 1.1 Gb на базе 8 и 3 процессоров MIPS R10000 250 MHz. Коммуникационное и сетевое оборудование представлено Nbase NH 2016 MegaSwitch G. Программное обеспечение Reliant UNIX 5.44C20, MPICH 1.2.0. Пиковая производительность 3 GF/s.

В связи с большим количеством проведенных вычислительных экспериментов и ограниченным объемом данной статьи далее приводятся только

ключевые результаты распараллеливания операций алгоритмов прогонок по различным координатным направлениям.

Условия эксперимента. Исследовались три задачи, отличающиеся друг от друга только сегментацией полной расчетной области, т. е. ее «нарезкой»:

1) сегментация в виде P одинаковых горизонтальных полос, прогонки ведутся вдоль X-направления, задача (4), алгоритм (6);

2) сегментация в виде P одинаковых вертикальных полос, прогонки ведутся вдоль Y-направления, задача (5), алгоритм (7);

3) сегментация в виде P одинаковых горизонтальных полос, прогонки ведутся вдоль Y-направления, задача (4), алгоритм (7) («разрезанные» прогонки со сшивкой решений по [3]).

Решение в каждом из P сегментов проводилось своим процессором, количество P процессоров в эксперименте варьировалось до максимума, обеспечиваемого возможностями системы на момент его проведения.

Цель эксперимента – определить эффективность распараллеливания, т. е. определить ускорение вычислительного процесса с учетом потерь на маршрутизацию в зависимости от значений P, ia, ja.

Эксперименты должны были дать ответ на следующие вопросы:

1. Какое ускорение процесса может быть получено для каждого типа сегментации?

2. Поскольку выборки из массивов G_{ij} и H_{ij} по первому и второму индексам требуют различного времени, каковы его затраты на них при распараллеливании?

3. Какова эффективность распараллеливания при «разрезанной» прогонке? Может быть, этот вид параллелизации вообще нельзя применять?

4. Поскольку вычислительные системы производят виртуальное размещение данных во внешней памяти, возможно скачкообразное изменение ускорения процесса. Это происходит в случае, когда заказанная задачей (вычислителем) размерность массива требует памяти *большей, чем оперативная, для одного процессора, но при распределении на P процессоров* своя часть сегментного массива *на каждом отдельном процессоре размещается в оперативной памяти.* Каково это ускорение?

5. Насколько отличаются времена работы процессоров при одинаковой арифметической нагрузке на каждый из них? Где образуются «узкие места» при заданной конфигурации процессорного пространства?

6. Каковы задержки в маршрутизации данных, вызываемые нахождением в многопроцессорной системе других, «чужих», задач?

Ответы на поставленные вопросы представляются весьма важными при принятии решения о способах распараллеливания больших программных комплексов, важной составляющей частью которых являются алгоритмы разных типов прогонок.

Приступим к последовательному анализу результатов. Введем для краткости изложения следующие обозначения: $T_p^k(ia, ja)$. Здесь T – время счета задачи в секундах; индекс $k \in [1,3]$ нумерует тип сегментации; индекс p равен числу задействованных процессоров; ia и ja – размерности сеток по X- и Y-направлениям. Определим также коэффициент ускорения

$$q_{p1,\,p2}^{k1,\,k2} = T_{p1}^{k1} \Big/ T_{p2}^{k2}$$

для задач с одной и той же размерностью сеток $(ia)_1 = (ia)_2$ и $(ja)_1 = (ja)_2$.

Таблица 1

Siemens RM600	Одно-процессорный запуск	Двухпроцессорный запуск		Четырехпроцессорный запуск			
Тип параллелизации	$p=1$	$p=1$	$p=2$	$p=1$	$p=2$	$p=3$	$p=4$
$k=1$	4,04	3,11	3,33	1,70	1,17	1,67	2,54
$k=2$	2,81	2,20	1,87	1,26	0,55	0,57	1,09
$k=3$	2,81	15,76	9,26	9,03	7,72	6,83	12,30

В табл. 1 приводятся значения $T_p^1(3000, 1000)$, $T_p^2(1000, 3000)$, $T_p^3(3000,1000)$, полученные на вычислительной машине Siemens RM600.

Во всех трех задачах для каждого из процессоров число арифметических операций одинаково: для двух процессоров 1000 узлов делится на 2, а для четырех – соответственно на 4. Массив размерностью $3 \cdot 10^6$ требует памяти меньшей, чем оперативная, поэтому особых неожиданностей не наблюдается. Выполняются естественные условия для всех k, следующие из закона Амдала (10):

$$T_1^k > T_2^k > T_4^k, \tag{11}$$

$$q_{12}^{kk} < 2; \quad q_{24}^{kk} < 2; \quad q_{14}^{kk} < 4. \tag{12}$$

Чтобы ответить на первый вопрос эксперимента, сравним данные построчно, т. е. для задач с одинаковой сегментацией. Определяя коэффициенты ускорений из табл. 1, имеем (с точностью до 0,1)

$$q_{12}^{11} = 1{,}3; \quad q_{24}^{11} = 1{,}8; \quad q_{14}^{11} = 2{,}4, \tag{13}$$

$$q_{12}^{22} = 1{,}3; \quad q_{24}^{22} = 1{,}7; \quad q_{14}^{22} = 2{,}2. \tag{14}$$

В (13) и (14) для расчета ускорений использованы времена работы первых процессоров. Из (13), а также из (14) следует, что для задач $k=1$ и $k=2$ переход с одно- на двухпроцессорный счет, для краткости обозначаемый далее как $(1 \to 2)$-переход, менее эффективен, чем $(2 \to 4)$-переход:

$$q_{12}^{11} < q_{24}^{11}; \quad q_{12}^{22} < q_{24}^{22}. \tag{15}$$

Заметим также, что (1→ 4)-переход в относительном смысле еще менее эффективен, чем (1→ 2)-переход:

$$q_{12}^{11} > q_{14}^{11}/2; \qquad q_{12}^{22} > q_{14}^{22}/2. \tag{16}$$

Таким образом, из (13)–(16) можно сделать следующие выводы:

а) эффективность ($p1 \to p2$)-переходов при одних и тех же значениях $p1$ и $p2$ практически одинакова для первого и второго типов сегментаций расчетной области;

б) зависимость ускорения (имеется в виду относительное ускорение $p2/p1 = \text{const}$) от числа задействованных процессоров носит немонотонный характер также для обоих типов сегментаций: эффективность (1→ 2)-перехода ниже эффективности (2→ 4)-перехода, а эффективность (относительная) (1→ 4)-перехода еще ниже, чем у (1→ 2)-перехода.

Если первый результат являлся ожидаемым и естественным, то второй – представляется не вполне ясным и, по-видимому, связан с конкретной архитектурой системы. В целом из табл. 1 следует, что потери на маршрутизацию достаточно велики и достигают в некоторых случаях 50 %.

Отдельно следует рассмотреть нижнюю строку табл. 1. Сегментация области, поперечная к направлению прогонок, увеличивает время решения задачи: $q_{12}^{33} = 0{,}2$ и даже $q_{14}^{33} = 0{,}3$. Это плата за неудачный алгоритм [3], в котором необходимо проводить искусственную «сшивку» прогонок. Однако переход с двух- на четырехпроцессорный счет дает положительное ускорение: $q_{24}^{33} = 1{,}7$, т. е. алгоритм [3] все-таки распараллеливается с потерями на маршрутизацию около 30 %.

Для ответа на второй вопрос экспериментирования поколоночный анализ данных приводит к следующим выводам: горизонтально-полосная ($k = 1$) сегментация менее эффективна, чем вертикально-полосная ($k = 2$) примерно в 1,5 раза:

$$q_{11}^{12} = 1{,}4; \qquad q_{22}^{12} = 1{,}6; \qquad q_{44}^{12} = 1{,}3. \tag{17}$$

Исключительно неэффективным оказался третий вид сегментации ($k = 3$). Например, $q_{22}^{32} = 7{,}9$, а $q_{44}^{32} = 7{,}2$.

Чтобы завершить анализ этого типа параллелизации и принять окончательное решение о допустимости его применения при распараллеливании больших вычислительных комплексов (вопрос стоит об эффективности или неэффективности крупномасштабного геометрического вида декомпозиции для комплексов типа [6, 7] в целом), следует более детально просмотреть калькуляцию затрат на каждый этап распараллеливания прогонки с «разрезанием» лучей прогоночных направлений и последующей сшивкой решений на границах разрезов. В табл. 2 приведены значения $T_2^3(3000,1000)$, полученные на системе Siemens RM600, и значения $T_2^3(3000,3000)$, $T_4^3(3000,3000)$, полученные на кластере ИВТ. Во-первых, отметим, что времена счета на каждом из процессоров различаются. В целом (см. табл. 2, строка 7) различие достигает 10 %, но по отдельным позициям оно существенно больше. Так, этап № 2 на процессоре 1 занимает на 15 % времени больше, чем на процессоре 2, зато этап № 6 идет на процессоре 2 медленнее, чем на процессоре 1, почти в 2 раза. Это не вполне ясное, но устойчивое явление, по-видимому,

Таблица 2

№ п/п	Процесс	Siemens RM600		Процессоры кластера ИВТ СО РАН					
		Двухпроцессорный запуск		Двухпроцессорный запуск		Четырехпроцессорный запуск			
		$p=1$	$p=2$	$k11$	$k14$	$k11$	$k14$	$k15$	$k16$
1	Инициализация	0,97	0,77	1,91	1,09	0,85	0,90	1,79	0,93
2	Вычисление массивов в слоях	6,07	5,30	163,72	256,68	6,53	7,87	10,30	7,57
3	Подготовка к пересылке	0,01	0,01	0,30	0,18	0,01	0,01	0,01	0,45
4	Пересылка	0,13	0,01	1,34	1,34	0,25	0,25	0,25	0,23
5	Вычисление граничных слоев	0,01	0,01	4,80	9,13	13,11	17,04	24,54	17,58
6	Вычисление результатов	2,30	4,20	486,70	865,41	5,56	6,64	9,72	6,43
7	Итого (без инициализации)	8,51	9,52	638,88	1132,76	25,47	31,95	45,00	32,29

связано с особенностями архитектуры и функционирования многопроцессорной системы.

Подчеркнем, что на кластере ИВТ этот эффект выражен еще больше. В частности, помещенный в табл. 2 результат $T_2^3(3000,3000)=1132,76$ есть время счета процессора $k14$, время параллельно работающего над этой задачей процессора $k11$ составляло «всего» 638,88, т. е. почти в 1,5 раза меньше.

Во-вторых, из табл. 2 видно, что основное время расходуется не на системные операции (маршрутизацию и обмены), а на сугубо алгоритмические, присущие самому методу. Вследствие этого кардинально сократить время не представляется возможным ни при какой оптимизации. Резюмируя вышеизложенное, можно заключить: крупноблочное геометрическое распараллеливание, использующее этот тип параллелизации прогонки, абсолютно неэффективно и применяться не должно, за исключением безальтернативных ситуаций.

Таким образом, ответ на третий вопрос эксперимента отрицателен: сегментацию на полосы, поперечные направлению прогонки, применять для распараллеливания операций не следует, поскольку существуют более эффективные геометрические способы.

Таблица 3

Процессоры кластера ИВТ	Одно-процессорный запуск	Двухпроцессорный запуск		Четырехпроцессорный запуск			
Тип параллелизации	$k1$	$k1$	$k14$	$k1$	$k14$	$k15$	$k16$
$k=1$	223,06	2,34	2,44	1,15	1,20	1,18	1,20
$k=2$	11,74	1,58	1,75	0,79	0,84	0,67	0,67
$k=3$	11,74	653,16	1090,19	24,71	30,20	45,81	30,59

На четвертый вопрос эксперимента был получен весьма интересный и неожиданный ответ. В табл. 3 показаны значения $T_p^k(3000,3000)$ – это ситуация, когда использование сетки $9\cdot10^6$ превышает возможности размещения массивов в оперативной памяти процессоров кластера ИВТ.

Вообще говоря, факт сильного ускорения задачи для вычислителя может оказаться «приятным сюрпризом»: если физико-математическая сущность задачи требует от вычислителя безальтернативного использования сеток порядка $S=10^7$ и больше, заведомо превышающих возможности оперативной памяти с соответственно большим временем решения, то применение многопроцессорных компьютеров может ускорить решение не только потому, что задействованы P процессоров, проводящих одновременно арифметические операции, но и потому, что распределенный на каждый процессор массив размерности S/P уже может быть размещен в оперативной памяти без необходимости подкачки и обменов данными из внешней памяти.

На первый взгляд невероятное и кажущееся совершенно неправдоподобным ускорение при переходе с одного процессора на два: $q_{12}^{11}=95, q_{12}^{22}=7{,}4$, что значительно превосходит положенное по закону Амдала число 2, оказалось после специальных контрольных пусков, действительно, имеющим место. Таковы потери времени на выборку элементов из массива, расположенного во внешней памяти, и пересылку их в оперативную.

Ускорения процессов при переходе с двух- на четырехпроцессорную систему достаточно стандартны: $q_{24}^{11}=1{,}95$, а $q_{24}^{22}=1{,}88$, и лежат вблизи значения 2 с потерей времени на маршрутизацию всего от 3 до 6 %. Заметим также, сравнивая времена счета, что второй тип распараллеливания эффективней первого, как отмечалось и ранее. Эта эффективность примерно одинакова для разного числа процессоров: $q_{22}^{12}=1{,}48$, а $q_{44}^{12}=1{,}42$.

Наконец, анализируя третий тип распараллеливания, имеем, помимо чрезмерно больших значений времени счета, еще и замедление вычислительного процесса: $q_{12}^{33}=0{,}01$.

(Заметим, что $T_1^3=T_1^2$, т. е. времена решения задач $k=2$ и $k=3$ совпадают, как и должно быть, поскольку совпадают сами задачи вследствие того, что при $p=1$ область, естественно, не сегментируется.)

Времена счета производят, действительно, сильное впечатление: при однопроцессорном счете $T_1^3 = 11{,}74$, а при двухпроцессорном $T_2^3 = 653{,}16$ для первого процессора и $T_2^3 = 1090{,}19$ для второго. Такой скачок определяется двумя факторами, на которые уже указывалось выше: «разрезкой» прогонки и превышением величины используемых массивов над объемом оперативной памяти. Однако априори трудно предположить, насколько большим может быть этот скачок времени счета. Ускорение $(2 \to 4)$-перехода тоже достаточно нестандартно: $q_{24}^{33} = 26{,}4$ и $q_{24}^{33} = 36{,}1$ для процессоров $k11$ и $k14$ соответственно, что также значительно превосходит число 2, как предел в законе Амдала. Объяснение такое же, как и для $(1 \to 2)$-перехода в задачах $k = 1$ и $k = 2$; помимо ускорения, определяемого проведением арифметических операций вдвое большим числом процессоров, играет роль тот факт, что «уменьшенные» вдвое объемы массивов уже могут размещаться в оперативной памяти, причем второй фактор существеннее первого. Таким образом, исследование четвертого вопроса эксперимента дало исчерпывающую и весьма интересную информацию, которая может быть полезной при параллелизации больших программных комплексов и использовании в них сеток со значительным числом узлов.

Перейдем к анализу пятого вопроса эксперимента. Рассмотрим в табл. 1–3 позиции, относящиеся к четырехпроцессорным запускам, из которых численно будем далее анализировать среднюю строку, относящуюся ко второму типу сегментации, как наиболее оптимальному (заметим, что информация, относящаяся к другим типам, также интересна).

Из табл. 1 следует, что медленнее всего осуществляются операции на первом процессоре Siemens RM600, где $T_4^2 = 1{,}26$, а наиболее быстро – на втором, где $T_4^2 = 0{,}55$, т. е. в 2,3 раза быстрее. Поскольку число арифметических действий в этих процессорах одинаково, то следует сделать вывод, что замедление вызвано маршрутизацией данных и связано с архитектурой системы. Аналогичный факт (см. табл. 3) имеет место на кластере ИВТ: на процессоре $k14$ операции осуществляются медленнее $(T_4^2 = 0{,}84)$, чем на процессорах $k15$ и $k16$ $(T_4^2 = 0{,}67)$ примерно в 1,3 раза.

Тот же самый факт влияния системы на время исполнения задач в каждом процессоре обнаруживается и для задачи с «разрезанной» прогонкой (см. табл. 2, строка № 7). Здесь как для двухпроцессорного (Siemens RM600 и кластер ИВТ), так и четырехпроцессорного счета времена отличаются, иногда весьма существенно. Так, времена счета в двухпроцессорном варианте на Siemens RM600 отличаются всего в 1,1 раза, а на кластере ИВТ – в 1,7 раз, а в четырехпроцессорном счете – до 1,8 раз.

Однако следует подчеркнуть, что не существует фиксированных «абсолютно быстрых» и «абсолютно медленных» процессоров (в разных запусках их «положение» менялось произвольным образом) и обнаружить какую-либо систему в этом не удалось. Данный факт иллюстрирует табл. 4, в которой помещены значения $T_4^2(3000,3000)$, полученные в десяти проведенных подряд пусках одной и той же задачи на кластере ИВТ и машине Siemens RM600, и приводятся значения времени работы каждого из четырех процессоров, «арифметическая нагрузка» которых была одинакова. Для кластера ИВТ максимальное и минимальное значения времени счета составляют соответственно 1,25 (процессор $k16$, пуск 10) и 1,11 (процессор $k11$, пуск 3) с

Таблица 4

Задача 1	Процессоры кластера ИВТ				Процессоры Siemens RM600			
Пуск	$k11$	$k14$	$k15$	$k16$	$p1$	$p2$	$p3$	$p4$
1	1,15	1,20	1,18	1,20	1,70	1,17	1,67	2,54
2	1,14	1,22	1,19	1,20	1,22	1,72	2,23	1,23
3	1,11	1,24	1,18	1,23	1,19	0,75	0,75	1,60
4	1,13	1,21	1,17	1,25	1,21	1,08	1,75	0,71
5	1,13	1,19	1,18	1,22	0,82	0,70	1,54	1,16
6	1,13	1,21	1,20	1,20	1,09	0,71	2,10	0,78
7	1,11	1,18	1,17	1,17	1,72	0,70	0,93	0,72
8	1,15	1,23	1,19	1,23	1,60	0,71	0,67	1,55
9	1,12	1,18	1,17	1,21	1,20	1,19	0,70	1,70
10	1,23	1,22	1,18	1,25	1,53	1,21	2,00	1,37

разницей около 10 %. Для Siemens RM600 максимальное и минимальное значения времени счета составляют соответственно 2,54 (процессор $p1$, пуск 1) и 0,67 (процессор $p3$, пуск 8) со значительным отличием в 3,8 раза.

Заметна разница времени как вдоль каждой строки (один пуск, разные процессоры), так и вдоль каждого столбца (один процессор, разные пуски). Первый эффект связан с архитектурой ЭВМ и соответственно маршрутизацией данных и в определенной степени «напоминает» вычислителю, который создал свою виртуальную сеть (линейку, кольцо) процессоров, что система может использовать реальные конфигурации маршрутов, одни из которых могут быть более, другие – менее оптимальными. В данной таблице построчная разница T_4^2 не превышает 10 % для кластера ИВТ, но для Siemens RM600 доходит до 70 %.

Второй эффект более неприятен для вычислителя и связан с нахождением в системе других задач, «тормозящих» исполнение его задачи. В данном случае максимум посторонней задержки не превышает 5 % для каждого процессора кластера ИВТ и эта задержка весьма заметна для процессоров Siemens RM600 (до 200 %), и, по-видимому, можно прогнозировать ситуации, когда этот эффект будет существенно более значительным.

Авторы выражают благодарность В. Э. Малышкину и В. И. Шелехову за полезные обсуждения и постоянные консультации, а также Г. С. Хакимзянову за внимание к данной работе.

СПИСОК ЛИТЕРАТУРЫ

1. **Ковеня В. М., Тарнавский Г. А., Черный С. Г.** Применение метода расщепления в задачах аэродинамики. Новосибирск: Наука, 1990.
2. **Парийский Б. С.** Об экономии памяти ЭВМ и времени счета при дифференциальной прогонке // ЖВМиМФ. 2000. **40**, № 2. С. 332.
3. **Яненко Н. Н., Коновалов А. Н., Бугров А. Н., Шустов Г. В.** Об организации параллельных вычислений и распараллеливании прогонки // Числ. методы мех. сплошн. среды. 1978. 9, № 6. С.139.
4. **Тарнавский Г. А., Шпак С. И.** Схема распараллеливания операций решения систем алгебраических уравнений методом многомерной скалярной прогонки // Вычисл. методы и программирование. 2000. **1**. С. 21;
   Интернет-журнал, http://num-meth.sscc.msu.ru
5. **Воеводин Вл. В.** Курс лекций в Международном университете г. Дубна // http://www.parallel.ru
6. **Тарнавский Г. А., Шпак С. И.** Декомпозиция методов и распараллеливание алгоритмов решения задач аэродинамики и физической газовой динамики: вычислительная система «ПОТОК-3» // Программирование. 2000. № 6. С. 45.
7. **Вшивков В. А., Краева М. А., Малышкин В. Э.** Параллельная реализация метода частиц // Программирование. 1997. № 2. С.39.

Институт вычислительных технологий СО РАН,
Институт теоретической и прикладной механики СО РАН,
Новосибирский государственный университет,
E-mail: vsh@ict.nsc.ru

Поступила в редакцию
13 ноября 2001 г.