УДК 519.6

ПРИМЕНЕНИЕ ЯЗЫКА НОРМА ДЛЯ РЕШЕНИЯ ЗАДАЧ НА ВЛОЖЕННЫХ СЕТКАХ

А. Н. Андрианов[1]

В работе рассматривается вопрос о применении языка Норма при построении параллельных программ для задач математической физики, использующих адаптивные вложенные структурные сетки. Приводятся сравнительные результаты выполнения построенной параллельной программы на многопроцессорной вычислительной системе с распределенной архитектурой для различных размерностей сеток ($N = 64$ и $N = 128$) и для различных конфигураций процессоров (от 1 до 512).

Ключевые слова: язык Норма, параллельные программы, структурные сетки, математическая физика, многопроцессорные вычислительные системы, адаптивные вложенные сетки.

1. Постановка задачи. Язык Норма [1] предназначен для записи алгоритмов решения задач математической физики на статических структурных сетках. В данной работе будет рассмотрена возможность применения языка Норма для записи алгоритмов решения задач математической физики, использующих адаптивные вложенные сетки.

В качестве исходной рассматривается задача трехмерного численного моделирования развития крупномасштабной конвективной неустойчивости внутри протонейтронной звезды, возникшей в результате действия неравновесного процесса нейтронизации вещества [2, 3]. Система уравнений адиабатической гидродинамики с учетом гравитации решается с помощью явной консервативной TVD разностной схемы годуновского типа со вторым порядком по пространству и времени. Для повышения точности результатов используется метод вложенных сеток. Расчетная область покрывается набором вложенных друг в друга сеток с одинаковым числом ячеек, но последовательно уменьшающимся в 2 раза абсолютным размером. Для выполнения условия устойчивости необходимо, чтобы двум шагам вычисления на сетке k-го уровня соответствовал один шаг вычислений по времени на сетке $(k-1)$-го уровня. Пересчет всех величин в каждой ячейке на сетке $(k-1)$-го уровня совершался простым усреднением 8 значений этих величин на сетке k-го уровня. Граничные значения, необходимые для вычисления величин на сетке k-го уровня, получались в результате монотонной интерполяции по 27 значениям этих величин на сетке $(k-1)$-го уровня. В вычислениях использовалось три уровня вложенных сеток (G^0, G^1, G^2), каждая размерностью $N \times N \times N$ (значение параметра N при расчетах исходной задачи варьировалось от 64 до 128) и с постоянным шагом по пространству (схема сетки приведена на рис. 1).

Схему расчета на каждом временном шаге можно описать следующим образом. Сначала определяются граничные значения для сетки G^k по значениям в соответствующих узлах сетки G^{k-1}. Далее вычисляются значения расчетных величин во внутренних узлах сетки G^k. В заключение полученные результаты усредняются и возвращаются в соответствующие узлы сетки G^{k-1}. На сетке каждого уровня вычисления расчетных величин проводятся по одной и той же схеме.

В качестве параметров задачи задаются размер сетки, координаты точки, начиная с которой помещается левый нижний угол вложенной сетки, и число используемых процессоров.

2. Распределение вычислений и данных. Кратко опишем схему распределения вычислений и данных, используемую при реализации языка Норма. Одним из ключевых понятий языка Норма является понятие области. Понятие области введено в языке Норма для представления понятия индексного пространства. Область — это совокупность целочисленных наборов $\{i_1, \ldots, i_n\}$, $n > 0$, $i_j > 0$, $j = 1, \ldots, n$, каждый из которых задает координаты точки n-мерного индексного пространства. С каждым направлением (осью координат) n-мерного пространства задачи связывается уникальное имя — *имя индекса* (имя оси координат индексного пространства).

Следует отметить, что область определяет значения координат точек индексного пространства, а не значения расчетных величин в этих точках. Например, если требуется вычислить значения величины $Y_{i,j}$,

[1] Институт прикладной математики им. М. В. Келдыша РАН, Миусская пл., 4, 125047, Москва; e-mail: and@a5.kiam.ru

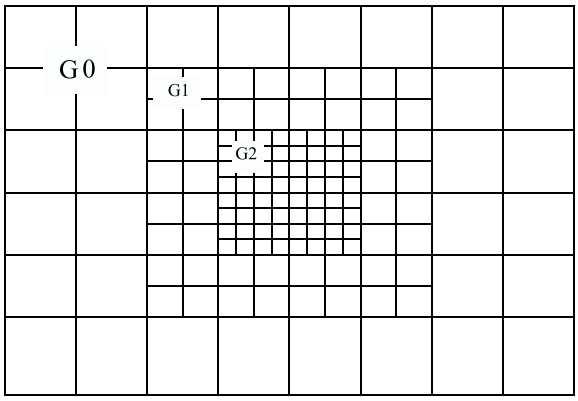

Рис. 1

$i, j = 1, \ldots, n$, на некоторой сетке $X_{i,j}$, $i, j = 1, \ldots, n$, введенной при решении задачи (например, формулой вида $X_{i,j} = F(h, i, j)$, F — заданная функция, h — заданный параметр), то следует:

(1) описать область, состоящую из точек $i, j = 1, \ldots n$;

(2) описать на этой области величины X и Y;

(3) задать на этой области правило вычисления значений сетки $X_{i,j}$: $X_{i,j} = F(h, i, j)$ и правило вычисления значений $Y_{i,j}$: $Y_{i,j} = G(X_{i,j})$ (считается, что функция G также некоторым способом задана).

Области используются в описаниях сеточных величин и при задании вычислений в операторах. Каждая сеточная величина связывается с указанной в описании областью. Эта область определяет имена индексов, которые могут использоваться в индексных выражениях при обращении к данной величине, и совокупность точек индексного пространства, в которых величина может принимать значения.

При описании вычислений в операторах область, указанная в заголовке оператора, задает совокупность точек, в которых требуется вычислить значение сеточной величины.

Для программ, которые предполагается выполнять на распределенных вычислительных системах, вводится описание индексов распределения. Это описание служит для отображения двух индексных направлений индексного пространства задачи на матрицу процессорных элементов (ПЭ) распределенной системы. При компиляции Норма-программы данное описание приводит к распределению между ПЭ системы как данных, так и управления и автоматической генерации операторов обмена данными между ПЭ, если это необходимо. Распределению подлежат величины, участвующие в расчетах и имеющие индексы, совпадающие с указанными в описании **DISTRIBUTION INDEX**.

Например, если задано описание

DISTRIBUTION INDEX $i = 1..8, \quad j = 1..8,$

то все переменные, определенные на областях с индексами i и j, будут равномерно распределены по $\text{ПЭ}_{i,j}$ с виртуальными номерами строк $i = 1..8$ и номерами столбцов $j = 1..8$ матрицы ПЭ.

Таким образом, программисту, использующему для описания решения своих задач язык Норма, при переходе на распределенную вычислительную систему достаточно определить с помощью описания индексов распределения конфигурацию вычислительной системы (линейка или матрица) и число процессоров по каждому направлению.

При трансляции Норма-программ для распределенных вычислительных систем применяется следующая схема распределения данных и вычислений.

Обозначим верхнюю границу по направлению i (i — индекс распределения) для величины X^k через U^i_k. Пусть $U^i = \max\limits_k U^i_k$. Процедура распределения данных основывается на размещении постоянного числа компонентов всех массивов в каждом процессоре (по фиксированному индексу распределения). Пусть по индексу i используется m процессоров. Тогда число компонентов массива по направлению i задается значением $i_s = \left[U^i/m\right] + 1$.

Аналогично определяется параметр j_s для индекса распределения j. Такое размещение массивов позволяет существенно упростить решаемую задачу определения параметров для стандартных (например, из библиотеки MPI) вызовов процедур передачи сообщений.

При построении фрагмента распределенной программы, реализующей вычисление на области O некоторой переменной X, определенной на области D, необходимо определить:

1) номера процессоров, которые должны выполнять данный фрагмент;

2) границы изменения параметров циклов, обеспечивающих обход области O;

3) описание массива X;

4) вид и расположение операторов передачи сообщений (обменов), которые обеспечивают доступ к значениям величин, используемым в расчетных формулах для вычисления X.

Транслятор с языка Норма решает все перечисленные выше задачи автоматически.

3. Обмен информацией между сетками. Процесс коммуникации между соседними уровнями сеток в рассматриваемой задаче можно описать следующим образом. На каждом временно́м шаге в процессе выполнения основной вычислительной программы определяются граничные значения для сетки G^k на основании значений в соответствующих узлах сетки G^{k-1}. По окончании процесса вычислений на сетке G^k полученные результаты усредняются и возвращаются в соответствующие узлы сетки G^{k-1}. Таким образом, процесс коммуникации соседних уровней сеток можно разделить на два этапа, работающих в "разных направлениях": пересылка граничных значений и возврат усредненных значений.

При реализации процесса коммуникации на распределенной вычислительной системе необходимо решить две задачи. Первая состоит в определении того, в каких процессорах надо проводить вычисление границ и усреднение. Эта задача обусловлена тем, что в общем случае аргументы для вычисления значений и сами значения находятся в различных процессорах. Например, в вычислении усредненных значений практически используются все процессоры системы, на которых решается задача. В то же время, результаты размещаются только во внутренних процессорах.

Вторая задача состоит в определении параметров (в какой процессор и сколько значений надо передать) для коммуникационных процедур.

Вначале рассмотрим задачу о том, в каких процессорах проводить операции вычисления граничных значений и операции усреднения.

3.1. Вычисление граничных значений. Граничные значения вычисляются в точках с координатами:

$$
\begin{aligned}
&(i = 1:2; && j = 3:N-2; && k = 3:N-2),\\
&(i = N-1:N; && j = 3:N-2; && k = 3:N-2),\\
&(i = 3:N-2; && j = 1:2; && k = 3:N-2),\\
&(i = 3:N-2; && j = N-1:N; && k = 3:N-2),\\
&(i = 3:N-2; && j = 3:N-2; && k = 1:2),\\
&(i = 3:N-2; && j = 3:N-2; && k = N-1:N).
\end{aligned}
$$

Для этих вычислений требуются аргументы в точках с координатами:

$$
\begin{aligned}
&(i = Mx-1:Mx+1; && j = My-1:My+N42+1; && k = Mz-1:Mz+N42+1),\\
&(i = Mx+N42:Mx+N42+2; && j = My-1:My+N42+1; && k = Mz-1:Mz+N42+1),\\
&(i = Mx-1:Mx+N42+1; && j = My-1:My+1; && k = Mz-1:Mz+N42+1),\\
&(i = Mx-1:Mx+N42+1; && j = My+N42:My+N42+2; && k = Mz-1:Mz+N42+1),\\
&(i = Mx-1:Mx+N42+1; && j = My-1:My+N42+1; && k = Mz-1:Mz+1),\\
&(i = Mx-1:Mx+N42+1; && j = My-1:My+N42+1; && k = Mz+N42:Mz+N42+2),
\end{aligned}
$$

где $N42 = (N-4)/2$, (Mx, My, Mz) — координата точки, в которую помещается левый нижний угол вложенной сетки.

Всего необходимо вычислить значения в $6 \times \left(2 \times (N-4)^2\right) = 12 \times (N-4)^2$ точках, причем в расчете принимают участие пять величин. Поэтому общее число вычисленных значений равно $60 \times (N-4)^2$. Для проведения этих вычислений необходимо $\dfrac{6 \times \left(3 \times (N+2)^2\right)}{4} \times 5 = 22.5 \times (N+2)^2$ значений аргументов. Поэтому целесообразно проводить вычисления в процессорах, в которых размещаются граничные значения, так как объем пересылок в этом случае меньше.

3.2. Вычисление усредненных значений. Аналогично рассматривается вопрос о том, в каких процессорах проводить усреднение значений на вложенной сетке. Усреднение проводится по точкам $i = 3..N-2$; $j = 3..N-2$; $k = 3..N-2$. Таким образом, требуется вычислить $(N-4)^3/8$ значений. Для их вычисления

требуется (с учетом повторного использования) $(N-2)^3$ аргументов. Очевидно, что в данном случае вычисление усредненных значений надо проводить в процессорах, в которых находятся аргументы для их вычисления, а получившиеся средние рассылать по соответствующим процессорам.

3.3. Определение параметров коммуникационных процедур. В качестве примера, иллюстрирующего возникающие проблемы при реализации коммуникаций между сетками различных уровней, остановимся на задаче возврата усредненных значений. В усреднении участвуют все точки k-го уровня сетки, кроме граничных ячеек.

Исходную запись, определяющую процесс усреднения вычисленных значений, можно представить следующим образом. Необходимо вычислить значения величины $X_{i,j}^{k-1}$ (где верхний индекс указывает на принадлежность к сетке определенного уровня) по следующему правилу:

$$X_{ij}^{k-1} = F\Big(X^k_{2(i-Mx)+1,\,2(j-My)+1}\Big) \qquad \begin{matrix} i = Mx+1,\ldots,Mx+\dfrac{n-4}{2}, \\ j = My+1,\ldots,My+\dfrac{n-4}{2}, \end{matrix}$$

где функция F имеет следующий вид:

$$F(X_{ij}) = \frac{X_{ij} + X_{i+1,j} + X_{i,j+1} + X_{i+1,j+1}}{4.0}$$

(для простоты изложения рассматриваем двумерный случай).

Такая запись недопустима в языке Норма: по определению языка Норма в качестве индексных выражений допускаются выражения вида $i \pm c$. Для реализации коммуникационных процедур был использован интерфейс Норма-программ и Фортран-программ, а также механизм определения "карты приема" (для процедур приема информации) и "карты рассылки" (для процедур рассылки). В "карте приема" указываются номера процессоров, от которых принимаются значения, и порядковые номера ячеек, в которые принимаются эти значения. В связи с тем, что сетки являются статическими, определение соответствующих "карт" можно проводить один раз, до начала основного счета. Рассмотрим способ определения значений для "карты рассылки" и "карты приема". Введем некоторые соотношения, которые будем использовать в дальнейшем.

Пусть матрица значений некоторой переменной распределена на сетку процессоров таким образом, что по направлению i в каждый процессор попадает i_s точек. Тогда координату i_r можно представить в виде

$$i_r = (ipr-1) \times i_s + i_p,$$

где ipr — номер процессора, в который попала точка i_r, а i_p — порядковый номер этой точки в процессоре с номером ipr. Номер процессора, в который попала точка с координатой i_r, задается формулой

$$ipr = \left[\frac{i_r - 1}{i_s}\right] + 1.$$

С учетом введенных формул номера процессоров, используемых для отправки усредненных значений, составляют диапазон от $\left[\dfrac{2}{i_s}\right] + 1$ до $\left[\dfrac{n-3}{i_s}\right] + 1$.

Определим номер процессора, в который необходимо передать значение из точки, находящейся в процессоре с номером ipr и имеющей координату i. Представим координату точки i в виде

$$i = (ipr-1) \times i_s + i_p,$$

где i_p — порядковый номер точки i в процессоре с номером ipr. Значение в точке i требуется в точке с координатой

$$i' = \frac{i + 2 \times Mx - 1}{2}.$$

Определим номер процессора, в котором находится точка i', подставляя вместо координаты точки i ее выражение:

$$\left[\frac{\dfrac{(ipr-1)\times i_s + i_p + 2\times Mx - 1}{2} - 1}{i_s}\right] + 1 = \left[\frac{(ipr-1)\times i_s + i_p + 2\times Mx - 3}{2\times i_s}\right] + 1.$$

Таким образом, схему отправки значений можно представить в виде цикла:

$$\begin{aligned}&\textbf{DO}\quad i_p = 1, i_s\\ &\qquad \text{Send}\left(X(i_p), \ldots, \left[\frac{(ipr-1)\times i_s + i_p + 2\times Mx - 3}{2\times i_s}\right] + 1\right)\\ &\textbf{ENDDO}\end{aligned}$$

Очевидно, такая схема не является эффективной из-за того, что вызов процедуры обмена производится для отправки только одного (по направлению i) значения. Более эффективной выглядела бы схема, в которой значения, отправляемые в один и тот же процессор, были сгруппированы и их отправка проводилась одним оператором.

Определим, сколько последовательно расположенных точек должны быть отправлены в один и тот же процессор. Пусть точки с порядковыми номерами $i_p, \ldots, i_p+k$ отправляются в один и тот же процессор. Определим условие, которому, в этом случае, должно удовлетворять значение k:

$$\left[\frac{(ipr-1)\times i_s + i_p + 2\times Mx - 3}{2\times i_s}\right] + 1 = \left[\frac{(ipr-1)\times i_s + i_p + k + 2\times Mx - 3}{2\times i_s}\right] + 1.$$

Представим числитель $(ipr-1)\times i_s + i_p + 2\times Mx - 3 = 2\times i_s \times p + c$, где $0 \leqslant c < 2\times i_s$. Тогда

$$\left[\frac{2\times i_s\times p + c}{2\times i_s}\right] + 1 = \left[\frac{2\times i_s\times p + c + k}{2\times i_s}\right] + 1, \quad \text{или} \quad \left[\frac{c+k}{2\times i_s}\right] = 0.$$

Отсюда следует ограничение на значение величины k: $k < 2\times i_s - c$. Значения, которые может принимать величина i_p, находятся в диапазоне $[1, i_s]$. Нетрудно показать, что каждый процессор отправляет необходимые значения не более чем в два процессора. Тогда схему отправки значений можно представить в следующем виде:

$$\begin{aligned}&k = 2\times i_s - c - 1\\ &\text{Send}\left(X(1:k), \ldots, \left[\frac{(ipr-1)\times i_s + 1 + 2\times Mx - 3}{2\times i_s}\right] + 1\right)\\ &\text{Send}\left(X(k+1:i_s), \ldots, \left[\frac{(ipr-1)\times i_s + i_s + 2\times Mx - 3}{2\times i_s}\right] + 1\right)\end{aligned}$$

В отличие от предыдущей схемы, здесь используется только два оператора отправки сообщений. Фактически весь диапазон изменения величины $i_p - [1, i_s]$ разбивается на два диапазона: $[1, k]$ и $[k+1, i_s]$. Каждый оператор отправки сообщений, в последнем случае, пересылает все точки, принадлежащие одному диапазону.

Рассмотрим вопрос определения параметров для процедур приема значений. В приеме участвуют процессоры, номера которых лежат в диапазоне от $\left[\frac{Mx}{i_s}\right] + 1$ до $\left[\frac{Mx + \frac{n-4}{2} - 1}{i_s}\right] + 1$.

Определим номер процессора, от которого необходимо принимать значения. Пусть точка $i' \in G^{k-1}$ принадлежит процессору с номером ipr. Представим координату точки i' в виде

$$i' = (ipr-1)\times i_s + i_p, \quad 1 \leqslant i_p \leqslant i_s,$$

где i_p — порядковый номер точки i' в процессоре с номером ipr. Тогда требуемое значение (от сетки G^k) находится в процессоре с номером

$$\begin{aligned}\left[\frac{2\times\big((ipr-1)\times i_s + i_p - Mx\big) + 1 - 1}{i_s}\right] + 1 &= 2\times(ipr-1) + \left[\frac{2\times(i_p - Mx)}{i_s}\right] + 1 =\\ &= 2\times ipr - 1 + \left[\frac{2\times(i_p - Mx)}{i_s}\right].\end{aligned} \tag{1}$$

Схема приема сообщений должна быть согласована со схемой отправки. При отправке сообщений мы группировали данные, отправляемые в один процессор. Поэтому и при приеме необходимо придерживаться такой же схемы: принимать все данные от одного процессора одним оператором приема.

Определим, сколько точек, которые требуются в процессоре с номером ipr, находятся в процессоре с номером, заданным (1). Пусть точка с порядковым номером i_p принадлежит процессору с номером ipr, а точка с порядковым номером $i_p + k$ также находится в этом процессоре. Тогда для того чтобы требуемые значения для этих точек находились в процессоре с номером, заданным в (1), необходимо, чтобы выполнялось равенство

$$\left[\frac{2 \times (i_p - Mx)}{i_s}\right] = \left[\frac{2 \times (i_p + k - Mx)}{i_s}\right]. \tag{2}$$

Из (2) можно получить ограничение на значение величины k. Обозначим $2 \times (i_p - Mx) = d \times i_s + c$, где $0 \leqslant c < i_s$. Тогда (2) принимает вид

$$\left[\frac{d \times i_s + c}{i_s}\right] = \left[\frac{d \times i_s + c + 2 \times k}{i_s}\right], \quad \text{или} \quad \left[\frac{c}{i_s}\right] = \left[\frac{c + 2 \times k}{i_s}\right].$$

Так как $c < i_s$, то последнее равенство справедливо при

$$k < \frac{i_s - c}{2}. \tag{3}$$

Таким образом, если для точки с порядковым номером i_p в процессоре с номером ipr требуются значения из процессора с номером, заданным (1), то точкам с порядковыми номерами $i_p{+}1, \ldots, i_p{+}k$, требуются значения из того же процессора. При этом для значения k должно быть выполнено условие (3). Как и при отправке сообщений, можно показать, что максимальное число процессоров, от которых необходимо принимать сообщения, не превосходит трех.

На рис. 2 и 3 приводятся карты приема и отправки усредненных значений для сетки размерностью $N = 64$ при счете задачи на матрице процессоров размерностью 4×4. В первом столбце таблицы указывается номер процессора, который принимает (отправляет) значения. Во втором столбце указан номер процессора, который отправляет (принимает) значения. В третьем столбце указывается диапазон значений индекса i. В четвертом столбце указывается диапазон значений индекса j.

Процессор приема	От кого	Диапазон по направлению i	Диапазон по направлению j
2 2	1 1	2–8	2–8
	1 2	2–8	9–16
	2 1	9–16	2–8
	2 2	9–16	9–16
2 3	1 3	2–8	1–8
	1 4	2–8	9–15
	2 3	9–16	1–8
	2 4	9–16	9–15
3 2	3 1	1–8	2–8
	3 2	1–8	9–16
	4 1	9–15	2–8
	4 2	9–15	9–16
3 3	3 3	1–8	1–8
	3 4	1–8	9–15
	4 3	9–15	1–8
	4 4	9–15	9–15

Рис. 2. Карта приема усредненных значений

Аналогично рассматривается вопрос построения “карты приема” и “карты рассылки” для пересылки граничных значений. Следует отметить, что способ реализации обменов на основании карт приема и отправки является достаточно универсальным и, по-видимому, может быть автоматизирован и поддержан набором соответствующих “стандартных” подпрограмм.

Процессор отправки	Кому	Диапазон по направлению i	Диапазон по направлению j
1 1	2 2	3 – 16	3 – 16
1 2	2 2	3 – 16	1 – 16
1 3	2 3	3 – 16	1 – 16
1 4	2 3	3 – 16	1 – 13
2 1	2 2	1 – 16	3 – 16
2 2	2 2	1 – 16	1 – 16
2 3	2 3	1 – 16	1 – 16
2 4	2 3	1 – 16	1 – 16
3 1	3 3	1 – 16	3 – 16
3 2	3 2	1 – 16	1 – 16
3 3	3 3	1 – 16	1 – 16
3 4	3 3	1 – 16	1 – 13
4 1	3 2	1 – 13	3 – 16
4 2	3 2	1 – 13	1 – 16
4 3	3 3	1 – 13	1 – 16
4 4	3 3	1 – 13	1 – 13

Рис. 3. Карта отправки усредненных значений

4. Результирующая программа. Для написания параллельной программы, реализующей алгоритм решения исходной задачи, использовался язык Норма и язык Фортран.

Текст Норма-программы занимает 730 строк. При изменении конфигурации процессоров, на которых мы хотим считать задачу, необходимо заменить строки программы с описанием индексов распределения. Конечно, можно считать, что это накладывает определенные трудности для разработчика программы, вызванные необходимостью перетранслировать текст Норма-программы при изменении числа процессоров. Однако заметим, что время трансляции Норма-программы в нашем случае составляет приблизительно 3.5 сек. Поэтому мы считаем, что указанные трудности вполне преодолимы.

Язык Фортран использовался для составления ряда вспомогательных программ (например, для вычислений, заданных в одной точке индексного пространства, или программ, осуществляющих коммуникации между сетками различных уровней). Текст Фортран-программ — 2000 строк.

Для обмена информацией между процессорами использовались процедуры приема (передачи) сообщений с блокировкой (MPI_SEND и MPI_RECV). При этом использовался простейший вариант размещения операторов передачи сообщений: требуемые значения отправлялись адресату как только они были вычислены, а следом за оператором MPI_SEND размещался оператор MPI_RECV. Таким образом, перекрытия вычислениями процесса передачи данных нет. Общее количество автоматически построенных операторов отправки (приема) равно 56. Операторы, относящиеся к реализации функций минимума и максимума, в данном случае мы не рассматриваем. Для сетки с размерностью $N = 64$ и для матрицы процессоров 4×4 длины сообщений распределились следующим образом: 2 сообщения с длиной 300 слов, 4 — 1500 слов, 32 — 1800 слов, 4 — 2100 слов и 14 — 2400 слов. Слово состоит из 4 байт.

5. Сравнительные характеристики выполнения программы. В заключение приводятся графики, в которых представлены сравнительные характеристики исполнения программы на ЭВМ МВС–1000М [4].

Измерения времени проводились с помощью вызовов процедуры MPI_WTIME. Измерялись общее время исполнения основного итерационного цикла программы и времена, затрачиваемые на работу процедур MPI_SEND и MPI_RECV. Счет каждого варианта программы повторялся пять раз. В качестве исходной информации для построения предлагаемых графиков выбиралось минимальное время исполнения программы. При проведении численных экспериментов в качестве параметра сетки выбирались значения $N = 64$ и $N = 128$. Для $N = 64$ итерации исполнялись 1000 раз, для $N = 128$ — 100 раз. Расчеты проводились при различном числе процессоров.

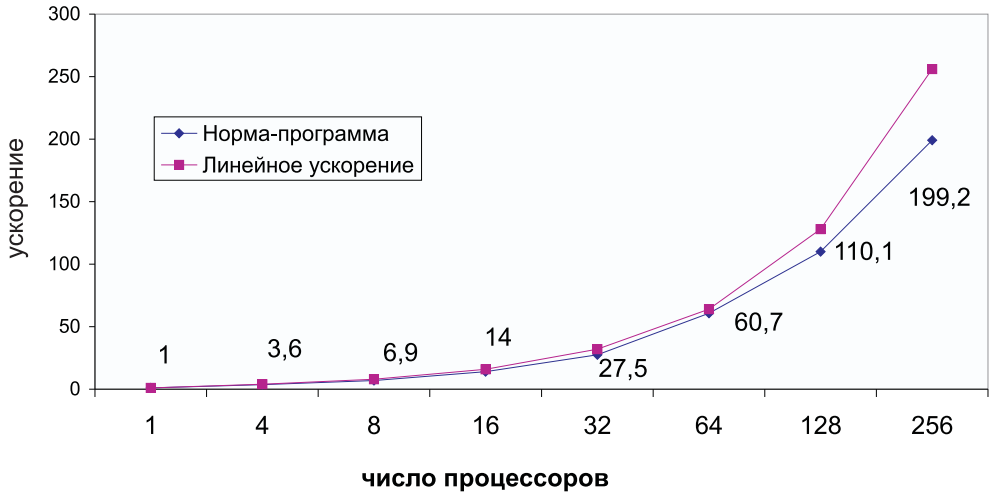

Рис. 4. Ускорение времени счета задачи ($N = 64$)

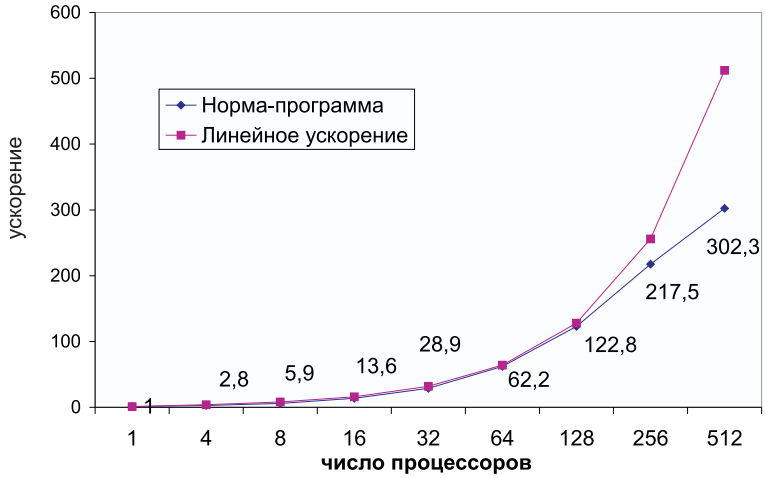

Рис. 5. Ускорение времени счета задачи ($N = 128$)

На рис. 4 и 5 иллюстрируется ускорение времени счета задачи для $N = 64$ и $N = 128$. На каждом из рисунков приводятся по два графика. Первый из них соответствует ускорению, полученному для Норма-программы и рассчитываемому по формуле $Speedup = T_1/T_p$, где T_1 — время счета задачи на одном процессоре, а T_p — время счета задачи на p процессорах. Второй график соответствует линейному ускорению, т.е. такому ускорению, при котором использование p процессоров должно приводить к сокращению времени счета в p раз.

Время счета задачи на каждом i-ом процессоре состоит из времени, затрачиваемому на передачу данных между процессорами $T^i_{\rm com}$ и времени, затрачиваемому непосредственно на вычисления $T^i_{\rm run}$. Для каждой конфигурации из p процессоров вычислим следующие величины: $T_{\rm com}(p) = \sum_{i=1}^{p} T^i_{\rm com}$ (суммарное время по всем p процессорам, затрачиваемое только на передачу информации); $T_{\rm run}(p) = \sum_{i=1}^{p} T^i_{\rm run}$ (суммарное время по всем p процессорам, затрачиваемое только на вычисления); $T_{\rm rez}(p) = \sum_{i=1}^{p} (T^i_{\rm com} + T^i_{\rm run})$ (суммарное время по всем p процессорам, затрачиваемое на решение задачи).

На рис. 6 и 7 иллюстрируется время счета и обменов задачи для $N = 64$ и $N = 128$. На каждом из рисунков приводятся по три графика. Первый график соответствует зависимости суммарного времени $T_{\rm com}(p)$, затрачиваемого на передачу информации, от числа используемых процессоров. Второй график соответствует зависимости суммарного времени, затрачиваемого непосредственно на счет $T_{\rm run}(p)$, от числа используемых процессоров. На третьем графике представлена зависимость общего времени $T^p_{\rm rez}$ счета от числа используемых процессоров.

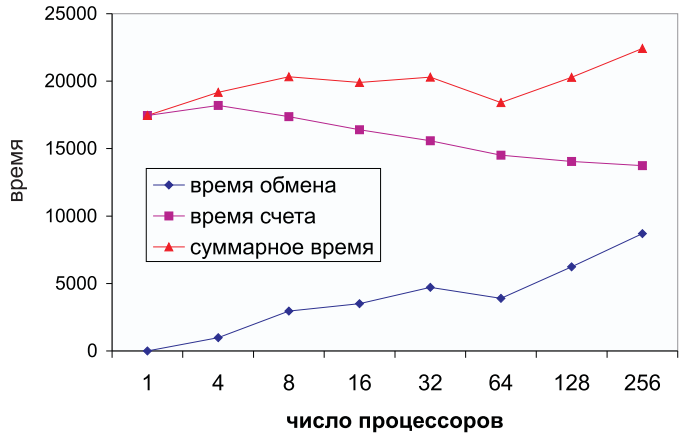

Рис. 6. Время счета и обменов задачи ($N = 64$)

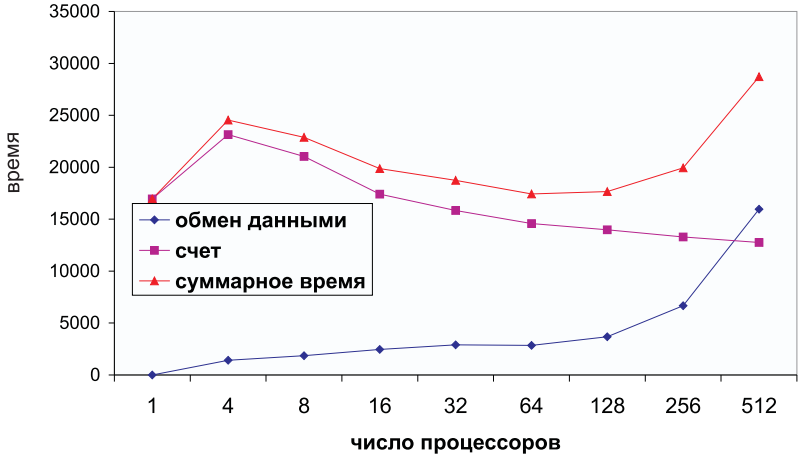

Рис. 7. Время счета и обменов задачи ($N = 128$)

Для оценки эффективности параллельных программ используется величина $E_p^1 = \dfrac{T_1}{T_p \times p} \times 100\,\%$. При этом общее время счета задачи на процессоре состоит из времени T_{run}, затрачиваемого непосредственно на вычисления, и времени T_{com}, затрачиваемого на обмен данными между процессорами. Конечно, мы хотим определить, как влияют затраты, связанные с обменом данными между процессорами, на общее время исполнения параллельной программы, то есть какую часть от общего времени вычисления составляет T_{com}. При использовании большого числа процессоров для счета программы величина T_{run} начинает сокращаться. Это обусловлено двумя причинами. Во-первых, каждый процессор (при увеличении общего числа процессоров) проводит вычисления на меньшем объеме данных. Во-вторых, сокращение объемов вычислений позволяет использовать возможности сверхбыстрой памяти (кэш-памяти) процессорных элементов. На последнее никак не влияет качество построенной параллельной программы.

Рассмотрим другую метрику эффективности:

$$E_p^2 = \frac{T_{\text{run}} \times p}{(T_{\text{run}} + T_{\text{com}}) \times p} \times 100\,\% = \frac{T_{\text{run}}}{T_{\text{run}} + T_{\text{com}}} \times 100\,\% = \left(1 - \frac{T_{com}}{T_{\text{run}} + T_{\text{com}}}\right) \times 100\,\%\,.$$

Заметим, что вторая величина всегда меньше чем 100 %; она показывает, как сильно влияют на общее время исполнения программы накладные расходы, связанные с передачей данных между процессорами. На рис. 8 представлены два графика. Первый график соответствует традиционному определению эффективности E_p^1, второй график соответствует величине E_p^2.

На рис. 9 представлены два графика. График с названием “число передаваемых слов” иллюстрирует возрастание объемов передаваемой информации (V_p) между всеми используемыми при счете задачи

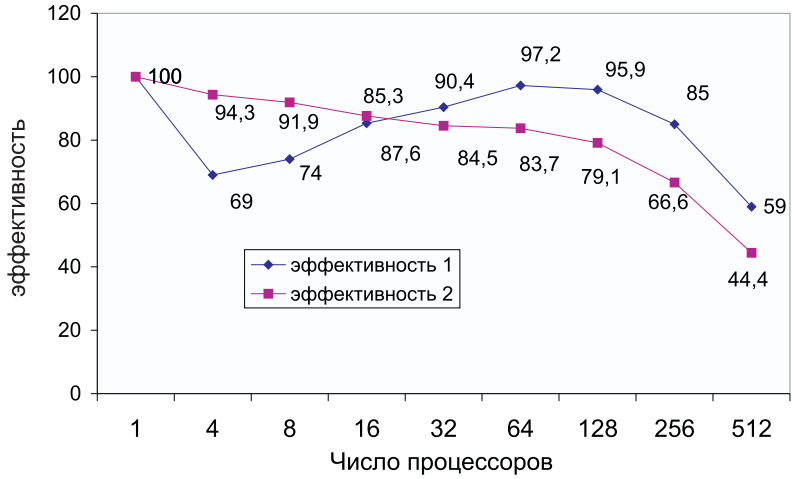

Рис. 8. Эффективность исполнения программы ($N = 128$)

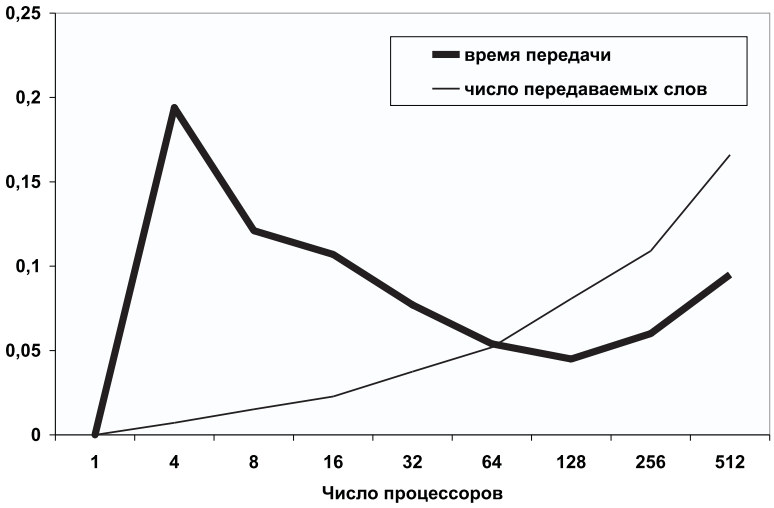

Рис. 9. Время передачи данных между процессорами и число передаваемых данных для сетки $N = 128$

процессорами. Второй график соответствует величине $T_{\text{comworld}} = T_{\text{com}}(p)/V_p$, т.е. среднему времени, затрачиваемому на передачу одного слова (4 байта), где $T_{\text{com}}(p) = \sum_{i=1}^{p} T_{\text{com}}^i$.

В заключение я хотел бы выразить глубокую признательность К. Н. Ефимкину за внимание и помощь при выполнении данной работы, а также поблагодарить С. Д. Устюгова за постановку проблемы и помощь при отладке и счете рассматриваемой задачи на параллельной вычислительной системе.

Работа выполнена при финансовой поддержке РФФИ (проект № 01–01–00411).

СПИСОК ЛИТЕРАТУРЫ

1. http://www.keldysh.ru/norma/
2. *Sazhin V.M., Ustyugov S.D., and Chechetkin V.M.* Pisma Zh. Teor. Fiz. 1996. **64**. 817.
3. *Ustyugov S.D. and Chechetkin V.M.* Supernovae explosions in the presence of large-scale convective instability in a rotating protoneutron star // Astronomy Reports. 1999. **43**, 11. 718–726.
4. http://www.jscc.ru

Поступила в редакцию
09.02.2002