РОССИЙСКАЯ АКАДЕМИЯ НАУК
ОРДЕНА ЛЕНИНА
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
им.М.В.КЕЛДЫША

УДК 523.16

П.В.Кайгородов, О.А.Кузнецов

АДАПТАЦИЯ СХЕМЫ РОУ–ОШЕРА ДЛЯ КОМПЬЮТЕРОВ С МАССИВНО–ПАРАЛЛЕЛЬНОЙ АРХИТЕКТУРОЙ

Москва – 2002

Аннотация

Рассмотрен метод адаптации вычислительной схемы Роу-Ошера для компьютера с массивно-параллельной архитектурой. Исследованы различные способы разбиения счетной области на подобласти и их влияние на производительность параллельного вычислительного кода. Приведены результаты тестирования производительности параллельного вычислительного кода в зависимости от числа процессоров и типа разбиения.

Abstract

We describe the adaptation of Roe-Osher scheme for the computers with massive-parallel architecture. Various methods of division of computation domain onto subdomains as well as its influence on the computational productivity were investigated. Results of testing of computational productivity in dependence on the number of processors and for different type of division of computation domain are given.

Введение

Потребность в вычислительных ресурсах растет с каждым днем, однако рост производительности вычислительных систем совершенно недостаточен для обеспечения запросов исследователей. Современные высокопроизводительные вычислительные машины слишком дороги и недоступны для большинства научных центров. В то же время, на рынке представлено множество моделей персональных ЭВМ, производительность которых сравнительно невелика, однако, благодаря массовому производству, соотношение цена/производительность выгодно отличает их от специализированных вычислительных систем. Кроме того, на данный момент существуют широко доступные технологии передачи данных, позволяющие создавать сети персональных ЭВМ, отличающиеся высокой скоростью и малыми задержкам прохождения данных.

В течение последних лет было создано множество свободно распространяемых программных комплексов, позволяющих облегчить развертывание вычислительных систем и существенно снизить затраты на программное обеспечение. Существующие продукты данного класса включают в себя ядро операционной системы, обеспечивающее поддержку оборудования, компиляторы с языков высокого уровня, коммуникационные библиотеки и множество утилит, облегчающих программирование и отладку.

Все это позволяет создавать так называемые “кластеры” – наборы сравнительно дешевых серийных ЭВМ, соединенных высокопроизводительной сетью передачи данных и работающих под управлением специализированного ПО. Подобные системы позволяют получить достаточно высокую производительность на большинстве задач, при этом оставаясь доступными по стоимости для многих пользователей.

Рассмотрим особенности кластерной архитектуры. Начнем с традиционных (однопроцессорных) систем. В традиционной вычислительной системе процессор имеет доступ к нескольким уровням оперативной памяти. Современные компьютеры имеют как минимум два таких уровня, как-то оперативную память (ОЗУ) и сверх-оперативную, или кэш (cache) память

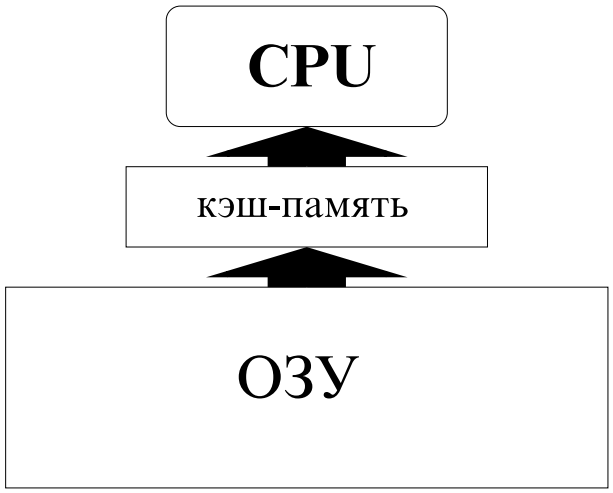

Рис. 1: Однопроцессорный компьютер.

(см. рис. 1). При считывании данных из памяти процессор прежде всего обращается к кэш-памяти, скорость доступа к которой может быть в несколько раз выше, чем к ОЗУ. В случае отсутствия необходимых данных процессор переходит в состояние ожидания до тех пор, пока данные не будут загружены в кэш-память из основной памяти.

Теперь рассмотрим многопроцессорные системы, которые подразделяются на два класса – симметричные (SMP) и массивно-параллельные (MPP) системы. SMP системы построены следующим образом: несколько процессоров, каждый из которых имеет свою сверх-оперативную память, объединены общей шиной данных и используют общее пространство ОЗУ (см. рис. 2). Это существенно упрощает программирование (наиболее распространенной библиотекой, облегчающей программирование SMP машин, является **OpenMP**) и позволяет эффективно обрабатывать большие объемы данных. К недостаткам SMP следует отнести их плохую масштабируемость ("чистые" SMP имеют, как правило, не более 16 процессоров) и сложность технической реализации механизмов, обеспечивающих актуальность данных, находящихся в кэш-памяти разных процессоров. Последнее приводит к существенному удорожанию производства SMP систем с большим числом процессоров.

Вычислительные системы, имеющие MPP-архитектуру (кластеры), со-

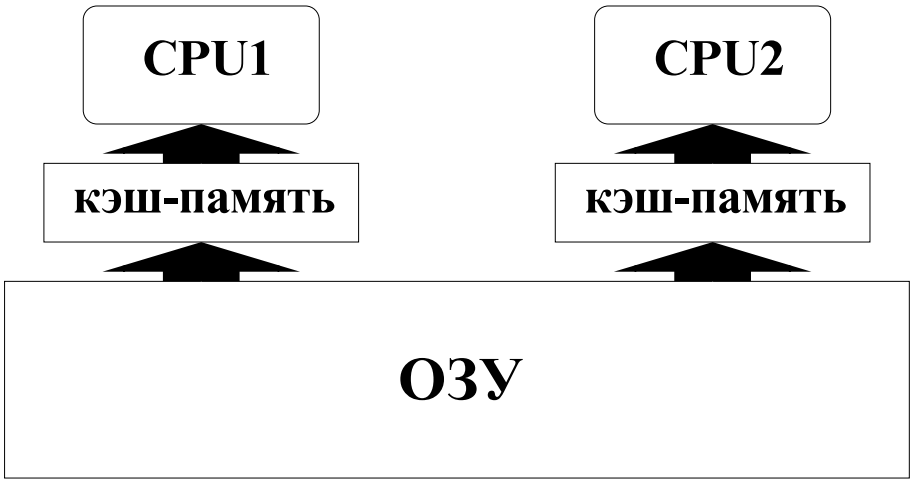

Рис. 2: SMP-архитектура.

стоят из нескольких вычислительных узлов, каждый из которых содержит процессор, имеющий собственную оперативную и сверхоперативную память (см. рис. 3). Для обмена информацией используется высокопроизводительная сеть передачи данных. Таким образом создается некая виртуальная область памяти, доступ к которой может получить любой процессор, но не непосредственно, а по явному запросу и при поддержке операционной системы. Некоторые современные MPP-системы содержат до нескольких десятков тысяч процессоров и традиционно лидируют по производительности, далеко опережая большинство SMP-систем. К достоинствам MPP-систем можно отнести их масштабируемость и сравнительно низкую стоимость в расчете на единицу вычислительной мощности. Недостатком этой архитектуры является сложность программирования, кроме того, характеристики сети передачи данных накладывают ограничения на класс задач, к решению которых могут быть привлечены кластеры.

В 2001-м году в Институте Астрономии РАН был построен вычислительный кластер с пиковой производительностью 28.8 GFlops. Данный кластер состоит из 18 узлов, каждый из которых содержит процессор Pentium-4 1600 MHz, 512 MB памяти и диск размером 30 Gb. Узлы объединены локальной сетью FastEthernet на базе коммутатора 3Com. Все программное обеспечение, используемое на кластере, является некоммерческим, что позволило заметно удешевить весь комплекс. Благодаря ис-

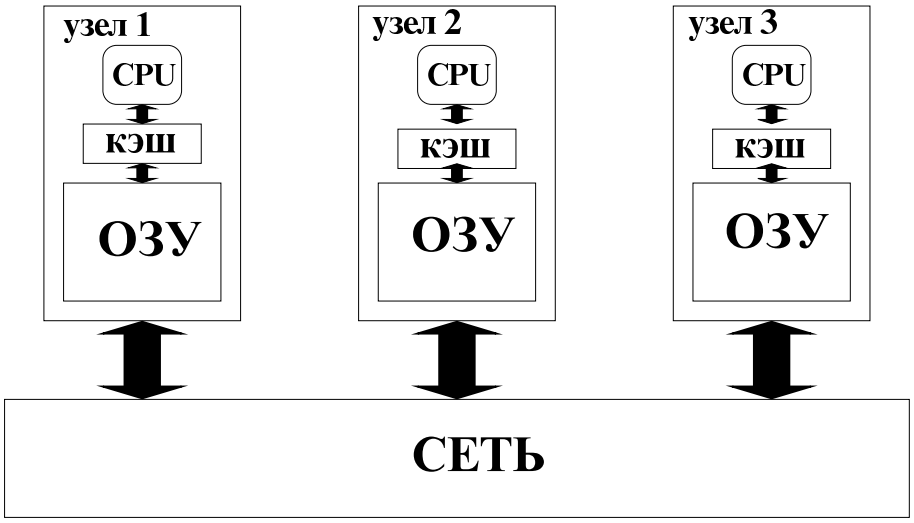

Рис. 3: MPP-архитектура.

пользованию оптимизирующего компилятора фирмы Intel (лицензия для некоммерческого использования) удалось добиться существенного прироста производительности вычислительных кодов.

Для проведения газодинамических расчетов на данном кластере нами были проведены работы по адаптации явной схемы годуновского типа для многопроцессорного компьютера.

1. Схемы годуновского типа

Рассмотрим систему уравнений трехмерной гравитационной газовой динамики для идеального газа:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \boldsymbol{v}\,\mathrm{grad}\,\rho + \rho\,\mathrm{div}\,\boldsymbol{v} = 0\,, \\ \dfrac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{v}\cdot\nabla)\boldsymbol{v} + \dfrac{1}{\rho}\,\mathrm{grad}\,p = -\,\mathrm{grad}\,\Phi\,, \\ \dfrac{\partial \varepsilon}{\partial t} + \boldsymbol{v}\,\mathrm{grad}\,\varepsilon + \dfrac{p}{\rho}\,\mathrm{div}\,\boldsymbol{v} = 0\,, \\ p = (\gamma-1)\rho\varepsilon\,. \end{cases} \tag{1}$$

В дивергентном виде эта система может быть записана как

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \operatorname{div} \rho \boldsymbol{v} = 0\,, \\ \dfrac{\partial \rho \boldsymbol{v}}{\partial t} + \operatorname{div}(\rho \boldsymbol{v} \otimes \boldsymbol{v} + p) = -\rho \operatorname{grad} \Phi\,, \\ \dfrac{\partial \rho(\varepsilon + |\boldsymbol{v}|^2/2)}{\partial t} + \operatorname{div} \rho \boldsymbol{v}(\varepsilon + p/\rho + |\boldsymbol{v}|^2/2) = -\rho \boldsymbol{v} \operatorname{grad} \Phi\,, \\ p = (\gamma - 1)\rho\varepsilon\,. \end{cases} \tag{2}$$

Здесь ρ – плотность, $\boldsymbol{v} = (u, v, w)$ – вектор скорости, p – давление, ε – внутренняя энергия, Φ – гравитационный потенциал, γ – показатель адиабаты (отношение теплоемкостей). Для решения таких задач часто используются численные схемы, построенные по типу классической схемы Годунова. Их суть состоит в использовании точного (или приближенного) решения задачи Римана о распаде произвольного разрыва для нахождения величин потоков. Для набора уравнений идеальной газовой динамики такое решение может быть найдено полуаналитически, что делает метод Годунова весьма подходящим для данного класса моделей. Впервые данная схема была построена в работе [1] и может быть записана в общем виде как

$$\frac{\hat{\boldsymbol{q}}_i - \boldsymbol{q}_i}{\tau} + \frac{\boldsymbol{F}_{i+1/2} - \boldsymbol{F}_{i-1/2}}{h} = 0\,,$$

где $\boldsymbol{q}$ – вектор, состоящий из газодинамических величин (ρ, ρu, ρv, ρw, $\rho(\varepsilon + |\boldsymbol{v}|^2/2)$), $\boldsymbol{F}$ – вектор потоков соответствующих величин, τ – временной шаг, h – шаг пространственной сетки. Значком ($\hat{}$) помечен вектор, состоящий из значений величин на следующем временном шаге. Потоки определяются из решения задачи о распаде произвольного разрыва следующим образом:

$$\boldsymbol{F}_{i+1/2} = \frac{1}{\tau} \int\limits_{t_0}^{t_0+\tau} \boldsymbol{F}(\boldsymbol{q}^*(x_{i+1/2}, t))\, dt \approx \boldsymbol{F}(\boldsymbol{q}^*(x_{i+1/2}, t_0))\,.$$

Для линейного уравнения переноса вида

$$\frac{\partial q}{\partial t} + a\frac{\partial q}{\partial x} = 0 \tag{3}$$

величина $q^*(x_{i+1/2}, t)$ определяется как

$$q^*(x_{i+1/2}, t) = \begin{cases} q_L & \text{если} \quad a < 0\,, \\ q_R & \text{если} \quad a > 0\,, \end{cases}$$

где q_L и q_R – значения величин, соответственно слева и справа от границы $i + 1/2$. Потоки в этом случае определяются как

$$F_{i+1/2} = aq^* = a^+ q_L + a^- q_R\,,$$

где $a^+ = \max(a, 0)$, $a^- = \min(a, 0)$.

Для системы гиперболических уравнений вида

$$\frac{\partial \boldsymbol{q}}{\partial t} + \mathcal{A}\frac{\partial \boldsymbol{F}}{\partial x} = 0$$

потоки определяются следующим образом:

$$\boldsymbol{F}_{i+1/2} = \frac{\boldsymbol{F}_i + \boldsymbol{F}_{i+1}}{2} - \frac{1}{2}|\mathcal{A}| \cdot (\boldsymbol{q}_{i+1} - \boldsymbol{q}_i)\,.$$

Примером схемы годуновского типа является схема Роу. Данная схема, основанная на приближенном решении задачи Римана, впервые была представлена в работе [2]. Матрица $\mathcal{A}$ выражается через усредненное значение $\boldsymbol{q}^*(\boldsymbol{q}_L, \boldsymbol{q}_R)$ как

$$\mathcal{A}(\boldsymbol{q}_L, \boldsymbol{q}_R) = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{q}}(\boldsymbol{q}^*)\,,$$

а $\boldsymbol{q}^*$ находится из условия:

$$\boldsymbol{F}_R - \boldsymbol{F}_L = \mathcal{A}(\boldsymbol{q}_L, \boldsymbol{q}_R) \cdot (\boldsymbol{q}_R - \boldsymbol{q}_L)\,,$$

откуда для системы уравнений газодинамики (2) получаются выражения:

$$\rho^* = \sqrt{\rho_L \rho_R}, \quad \boldsymbol{v}^* = \frac{\sqrt{\rho_L}\boldsymbol{v}_L + \sqrt{\rho_R}\boldsymbol{v}_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}\,, \quad h^* = \frac{\sqrt{\rho_L}h_L + \sqrt{\rho_R}h_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}\,,$$

где $h = \varepsilon + p/\rho + |\boldsymbol{v}|^2/2$ – удельная полная энтальпия. Окончательно потоки записываются как:

$$\boldsymbol{F}_{i+1/2} = \frac{\boldsymbol{F}_i + \boldsymbol{F}_{i+1}}{2} - \frac{1}{2}\sum_m |\lambda^m(\boldsymbol{q}^*)|\Delta s^m_{i+1/2}\boldsymbol{r}^m(\boldsymbol{q}^*)\,,$$

$$\Delta s^m_{i+1/2} = \boldsymbol{l}^m(\boldsymbol{q}^*)(\boldsymbol{q}_{i+1} - \boldsymbol{q}_i)\,,$$

где λ^m – набор собственных значений, а $\boldsymbol{r}^m$ и $\boldsymbol{l}^m$ – соответственно правые и левые собственные вектора матрицы $\mathcal{A}$.

Проведенные выше схемы имеют первый порядок аппроксимации. Для повышения порядка и для уменьшения численной диффузии используются TVD схемы с так называемыми ограничителями потоков. Для простого уравнения переноса (3) поток можно записать как (для простоты будем считать, что $a > 0$):

$$F_{i+1/2} = aq_i + \frac{1}{2}a\left(1 - \frac{\tau a}{h}\right)\alpha(R)(q_{i+1} - q_i)\,,$$

где

$$R = \frac{q_i - q_{i-1}}{q_{i+1} - q_i}\,.$$

Переменная R называется анализатором гладкости, а функция $\alpha(R)$ – ограничителем потока. Для построения схем с порядком аппроксимации выше первого, необходимо, чтобы $\alpha(1) = 1$.

В работах [3, 4, 5] было показано, что для обеспечения монотонности необходимо соблюдения условия:

$$0 \leqslant \alpha(R) \leqslant \mathrm{minmod}(2, 2R)\,,$$

где

$$\mathrm{minmod}(x, y) = \frac{1}{2}(\mathrm{sign}(x) + \mathrm{sign}(y))\,\min(|x|, |y|).$$

Этому условию удовлетворяют схемы:

$\alpha(R) = \mathrm{minmod}(1, R)$	'minmod' [4]
$\alpha(R) = \max(0, \min(1, 2R), \min(2, R))$	'superbee' [6]
$\alpha(R) = \max\left(0, \min\left(\frac{1+R}{2}, 2, 2R\right)\right)$	'MUSCL' [7]
$\alpha(R) = \frac{R + \lvert R\rvert}{1 + R}$	Van Leer [8]

Для получения более высоких порядков аппроксимации, функцию $\alpha(R)$ можно задать следующим образом [5, 9], получив схему Роу-Ошера:

$$\alpha(R) = \begin{cases} 0\,, & R \leqslant 0 \\ \left(\dfrac{1+\phi}{2}\beta + \dfrac{1-\phi}{2}\right) R\,, & 0 \leq R \leqslant 1/\beta \\ \dfrac{1+\phi}{2} + \dfrac{1-\phi}{2} R\,, & 1/\beta \leqslant R \leqslant \beta \\ \dfrac{1+\phi}{2} + \dfrac{1-\phi}{2}\beta\,, & R \geqslant \beta > 1 \end{cases}$$

или

$$\alpha(R) = \frac{1+\phi}{2}\,\mathrm{minmod}(1, \beta R) + \frac{1-\phi}{2}\,\mathrm{minmod}(\beta, R).$$

где ϕ и β – некие параметры, удовлетворяющие условию $1 < \beta \leqslant \beta_{max}$, где $\beta_{max} = (3-\phi)/(1-\phi)$. Анализ данной схемы показывает [5, 9], что при выборе $\phi = 1/3$ получается схема третьего порядка аппроксимации, при всех других значениях схема имеет второй порядок.

Окончательно для системы уравнений газовой динамики потоки в схеме Роу-Ошера имеют вид

$$\begin{aligned} \boldsymbol{F}_{i+1/2} \;=\; & \boldsymbol{F}^I_{i+1/2} + \frac{1+\phi}{4}\sum_m \mathrm{minmod}\left(\boldsymbol{F}^{m+}_{i+1/2}, \beta \boldsymbol{F}^{m+}_{i-1/2}\right) \\ & + \frac{1-\phi}{4}\sum_m \mathrm{minmod}\left(\beta \boldsymbol{F}^{m+}_{i+1/2}, \boldsymbol{F}^{m+}_{i-1/2}\right) \\ & - \frac{1+\phi}{4}\sum_m \mathrm{minmod}\left(\boldsymbol{F}^{m-}_{i+1/2}, \beta \boldsymbol{F}^{m-}_{i+3/2}\right) \\ & - \frac{1-\phi}{4}\sum_m \mathrm{minmod}\left(\beta \boldsymbol{F}^{m-}_{i+1/2}, \boldsymbol{F}^{m-}_{i+3/2}\right)\,, \end{aligned}$$

где

$$\boldsymbol{F}^{m\pm}_{i+1/2} = \left[\lambda^m(\boldsymbol{q}^*)\right]^{\pm} \Delta s^m_{i+1/2} \boldsymbol{r}^m(\boldsymbol{q}^*)\,,$$

а потоки для схемы первого порядка $\boldsymbol{F}^I$ определены ранее.

Для обеспечения устойчивости вышеприведенных схем, шаг по времени τ выбирается из условия Куранта-Фридрихса-Леви (CFL). Данное условие накладывает следующее ограничение:

$$\tau = C \cdot \min(\tau_x, \tau_y, \tau_x)\,, \qquad \begin{aligned} \tau_x &= \min_{i,j,k}\left(\frac{u_{i,j,k} + c^s_{i,j,k}}{h^x_{i,j,k}}\right)\,, \\ \tau_y &= \min_{i,j,k}\left(\frac{v_{i,j,k} + c^s_{i,j,k}}{h^y_{i,j,k}}\right)\,, \\ \tau_z &= \min_{i,j,k}\left(\frac{w_{i,j,k} + c^s_{i,j,k}}{h^z_{i,j,k}}\right)\,, \end{aligned} \qquad c^s = \sqrt{\gamma\frac{p}{\rho}}\,,$$

здесь h^x, h^y, h^z – пространственный шаг вычислительной сетки по трем направлениям. Константа C здесь – число Куранта, которое выбирается, исходя из особенностей конкретной схемы.

2. Адаптация схемы Роу-Ошера для компьютеров с многопроцессорной архитектурой

Схема Роу-Ошера, представленная выше, позволяет получать решения задач газодинамики с хорошей степенью точности. Однако, при использовании высоких пространственных разрешений, данная схема становится слишком ресурсоемкой. Таким образом, большинство современных задач может моделироваться только с использованием суперкомпьютеров – машин, имеющих множество параллельно работающих процессоров. Однако, требуется специальная адаптация вычислительного кода к параллельной архитектуре.

Явные методы, к которым относятся и методы годуновского типа, могут быть адаптированы к многопроцессорной архитектуре путем разбиения вычислительной сетки на подобласти. При этом каждая область обрабатывается одним процессором, а взаимодействие процессов требуется только при переходе к следующему временному шагу.

Так как рассматриваемые схемы годуновского типа относятся к классу явных численных схем, для вычисления значений в любой из ячеек на сле-

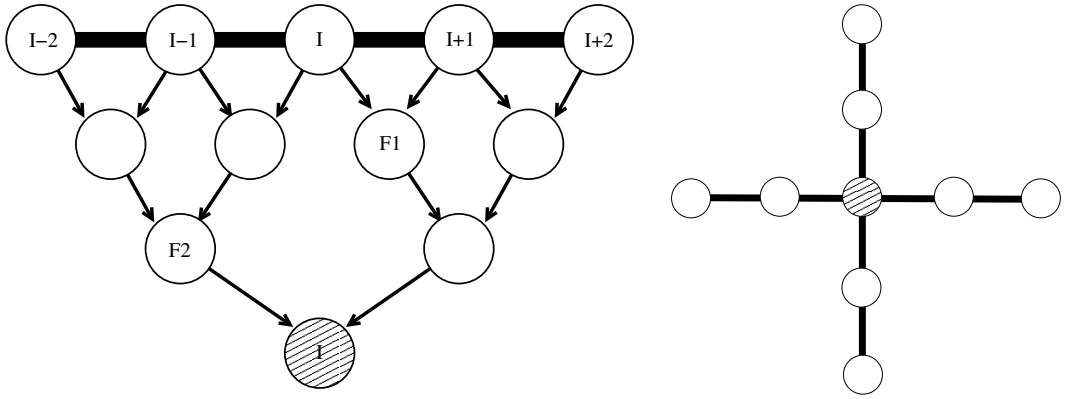

Рис. 4: Пятиточечный шаблон.

дующем временном шаге необходимо знать значения нескольких соседних ячеек.

Шаблон вычислительной схемы определяет, какие именно ячейки необходимы для вычисления нового значения в узле с координатами i, j, k. В случае схемы Роу-Ошера используется простой пятиточечный шаблон (см. рис. 4). При разбиении обрабатываемого объема на подобласти, неизбежно возникнут ячейки, для обсчета которых необходимо будет знать значения ячеек, лежащих за пределами их подобласти. Другими словами, для получения правильного ("сшитого" на границах подобластей) решения, процессам, обрабатывающим соседние подобласти, придется обмениваться информацией.

Таким образом, для адаптации вычислительного кода к многопроцессорной архитектуре можно обойтись модификацией процедур, обеспечивающих наложение граничных условий. В качестве примера, рассмотрим два процесса, обрабатывающих смежные области (см. рис. 5). Фактически, каждую область обрабатывает отдельный вычислительный код, не содержащий модификаций по сравнению с однопроцессорным вариантом.

Для наложения граничных условий обычно вводятся дополнительные ячейки, которые формально находятся за границей области и не обрабатываются кодом. Величины в этих ячейках задаются либо постоянными (фиксированные граничные условия), либо являются функциями от величин в соседних "реальных" ячейках.

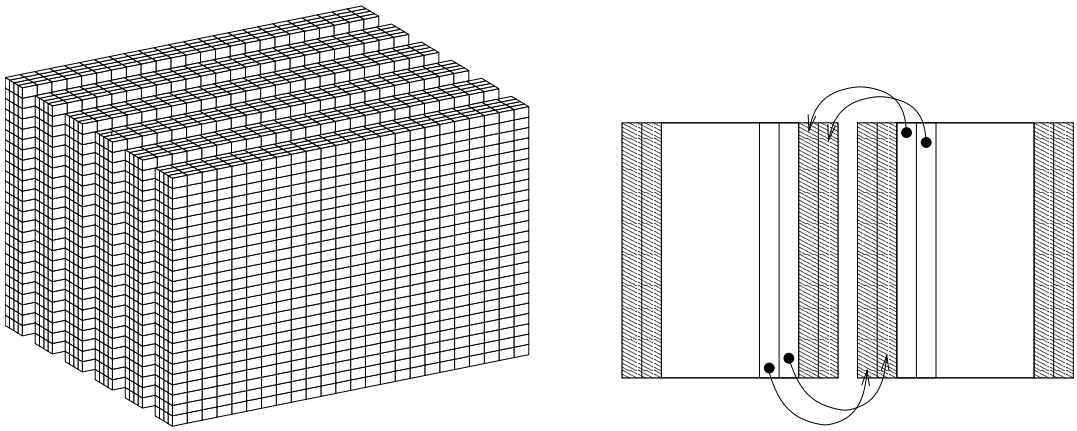

Рис. 5: Одномерное разбиение.

При разбиении на подобласти, кроме внешних границ, в расчетной зоне возникают внутренние границы, разделяющие подобласти. Для того, чтобы сумма решений в двух смежных подобластях полностью совпадала с решением однопроцессорного варианта кода, для явных схем годуновского типа необходимо и достаточно правильно задать граничные условия на внутренних границах. В качестве внешних ячеек, находящихся за такой границей, необходимо брать ячейки соседнего процесса, непосредственно прилегающие к данной границе с другой стороны (на рис. 5 области таких ячеек заштрихованы).

Как правило, кроме пересылки данных о приграничных узлах, процессам необходимо в ходе расчетов обмениваться некой дополнительной информацией. Например, для обеспечения сшивания решения на границах областей, все процессы должны использовать одинаковое значение для временного шага τ. В практических расчетах τ обычно выбирается автоматически, например, исходя из условия Куранта-Фридрихса-Леви. Наиболее рациональный способ вычисления τ следующий: каждый процесс вычисляет свое собственное значение τ, исходя из значений величин в своей подобласти, после чего процессы должны выбрать наименьшее значение τ и использовать его на следующем шаге. Алгоритм такого выбора может заметно влиять на производительность расчетов, особенно при большом числе процессоров и больших задержках при передаче данных.

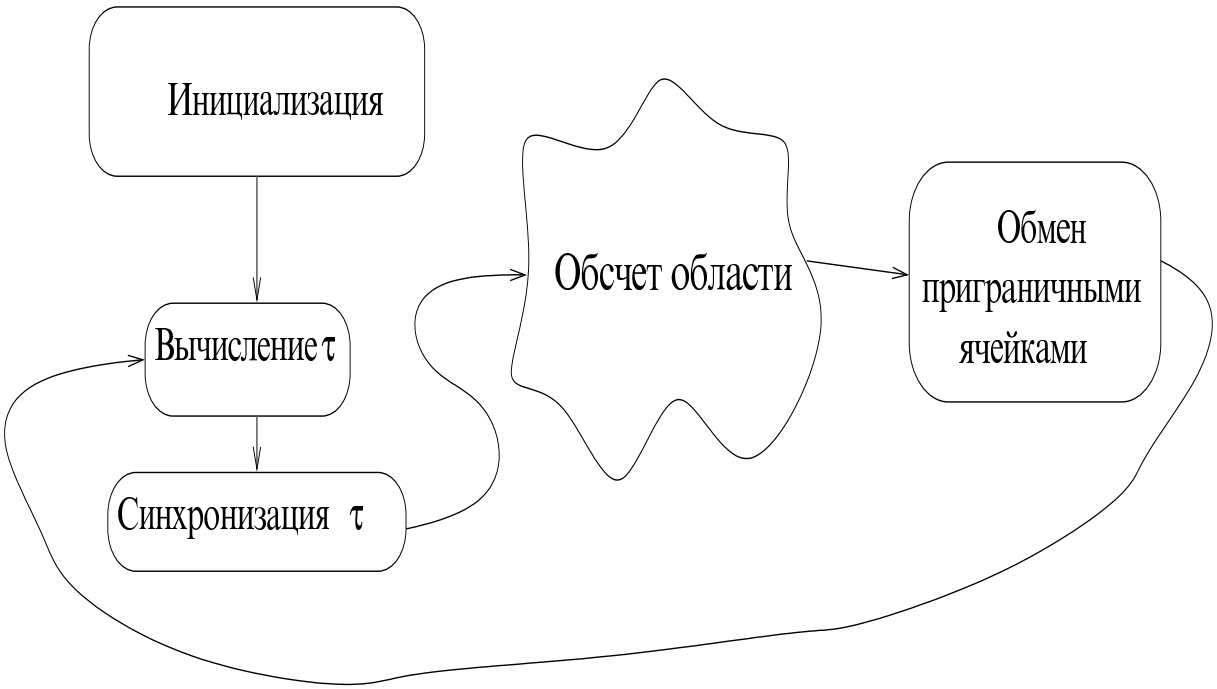

Рис. 6: Блок-схема работы одного процесса.

Способы оптимизации этого алгоритма будут рассмотрены ниже. Упрощенная блок-схема алгоритма работы кода представлена на рис. 6.

Очевидно, что простую вычислительную сетку, имеющую разрешение $N_x \times N_y \times N_z$ узлов можно разделить на подобласти множеством различных способов. Остановимся на трех, наиболее простых методах:

2.1 Деление вдоль одной оси

Наиболее простой в реализации способ деления вычислительной области – разрезать ее на слои вдоль одной из осей, по числу процессоров (см. рис. 5). Таким образом, если в наличии имеется P процессоров, мы получим P областей с размерностью $\frac{N_x}{P} \times N_y \times N_z$. Для получения решения на момент времени t^{n+1} в каждой из областей необходимо произвести вычисления по одной из разностных схем, приведенных выше. Легко видеть, что такое решение не может быть независимым для различных подобластей. Внутри каждой области существуют узлы, для вычисления величин в которых необходимо знать значения величин в узлах, не принадлежащих данной подобласти. Шаблон вычислительной схемы определяет, какие именно уз-

лы необходимы. В случае простого пятиточечного шаблона (см. рис. 4) для сшивания решений необходимо использовать величины из двух соседних узлов с каждой стороны.

Отсюда возникает необходимость в пересылке двух слоев ячеек с каждой стороны подобласти, граничащей с подобластями соседних процессов. Легко видеть, что объем данных, предназначенных для пересылки между двумя процессами на каждом временном шагу, может быть вычислен [1)] как

$$L_{bytes} = 4 \times N_{var} \times 8 \times N_y \times N_z \,.$$

2.2 Деление вдоль двух и трех осей

Очевидно, что деление вдоль одной оси, представленное выше, не может считаться оптимальным по количеству передаваемых данных. При увеличении числа процессоров, толщина слоя выделенного каждому из них убывает, в то время как количество передаваемых данных не изменяется. Это в конечном счете приводит к так называемому насыщению – рост производительности вычислений замедляется и со временем выходит на постоянный уровень, при котором добавление новых процессоров уже практически не влечет за собой роста производительности. Если учесть, что синхронизация τ с ростом числа процессоров начинает отнимать все больше и больше времени, может сложиться ситуация, при которой с добавлением процессоров производительность падает.

Следующим по сложности реализации следует считать такое разбиение, при котором вычислительная область разрезается сначала вдоль одной оси, после чего получившиеся слои разрезаются вдоль другой оси на несколько долей. Таким образом, область оказывается разделенной на множество “столбиков”, каждый из которых является подобластью, обрабатываемой одним процессором (см. рис. 7). При этом для областей (на рис. 7 они выделены двойной штриховкой), значения в ячейках вычисля-

[1)] для вычислений с двойной точностью.

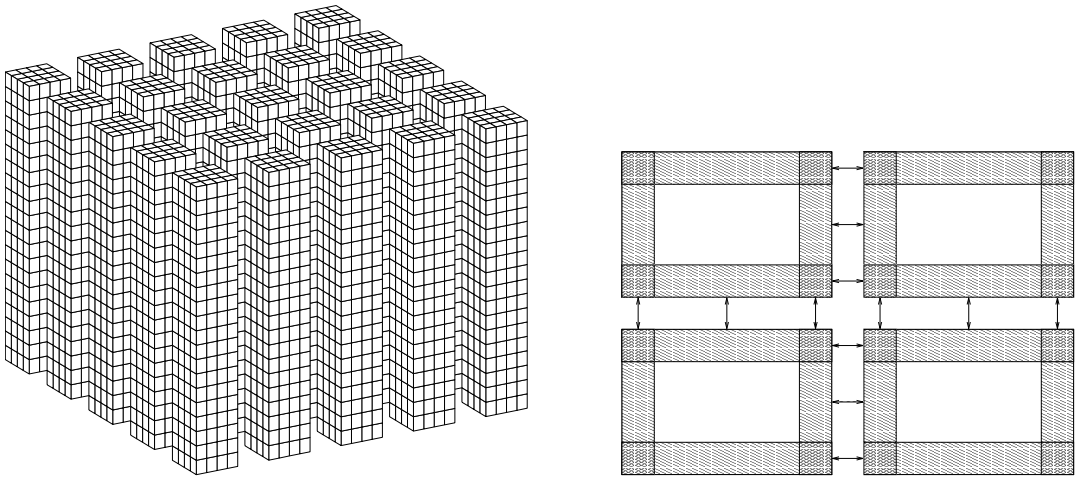

Рис. 7: Двумерное разбиение.

ются с использованием данных, полученных сразу от двух соседних процессов – для вычисления потоков вдоль соответствующих направлений.

Легко видеть, что количество данных, передающихся от процесса к процессу, при таком разбиении получается равным

$$L_{bytes} = 8 \times N_{var} \times 8 \times N_z \times \left(\frac{N_x}{P_x} + \frac{N_y}{P_y} \right)$$

где P_x и P_y – число разбиений вдоль осей x и y соответственно. Видно, что при подобном способе разбиения количество передаваемых данных убывает с числом процессоров, однако для расчета уже нельзя брать произвольное число процессоров – это число должно раскладываться как минимум на два множителя. Тем не менее, эффект насыщения должен присутствовать и здесь – это связано как с особенностями сетей передачи данных (скорость передачи максимальна при пересылке больших объемов данных), так и с потерями на синхронизацию τ. Однако этот предел отстоит гораздо дальше, чем в случае с одномерным разбиением, что позволяет использовать существенно большее число процессоров при расчете.

Разбиение вдоль трех осей (см. рис. 8) совершенно аналогично, количество передаваемых данных при нем равно

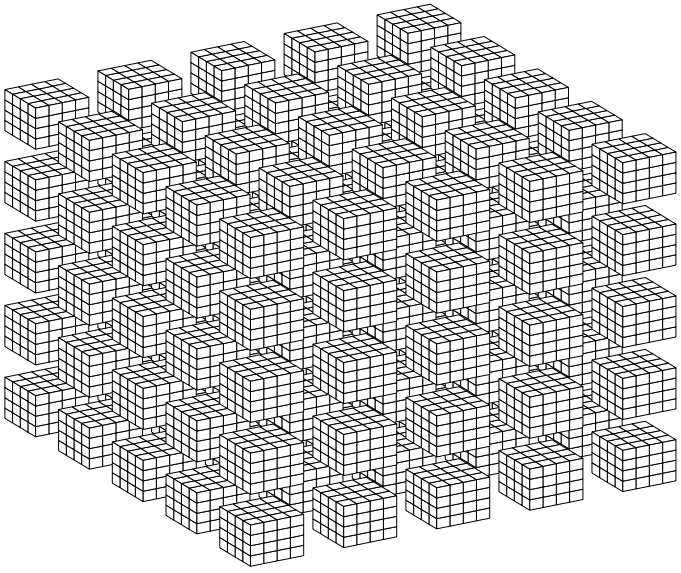

Рис. 8: Трехмерное разбиение.

$$L_{bytes} = 12 \times N_{var} \times 8 \times \left(\frac{N_x}{P_x}\frac{N_y}{P_y} + \frac{N_x}{P_x}\frac{N_z}{P_z} + \frac{N_y}{P_y}\frac{N_z}{P_z} \right).$$

При этом насыщение наступает, соответственно, при большем числе процессоров, чем в вышеприведенных случаях.

2.3 Синхронизация τ

Как было сказано выше, каждый процесс перед обсчетом очередного временного шага вычисляет собственное τ исходя из условия CFL. Однако, для обеспечения сшивки решений двух соседних процессов, необходимо равенство используемых ими τ. Это приводит к необходимости обмена между всеми процессами. Очевидно, что выбор минимального значения среди τ, полученных по условию CFL в подобластях, эквивалентен применению условия CFL ко всей области.

Наиболее простой и очевидный способ такого выбора – передать значения τ со всех процессов машины одному, выбранному заранее (например, процессу с номером 0), после чего этот процесс, выбрав из всех полученных значений минимальное, разошлет его всем остальным процессам.

Количество пересылок при таком алгоритме равно удвоенному числу процессоров. Существуют, однако, более оптимальные алгоритмы решающие данную задачу.

Сеть, соединяющая вычислительные узлы кластера строится, как правило, на базе коммутаторов, обеспечивающих независимую передачу данных между произвольными парами узлов. Используя этот факт, можно существенно ускорить передачу данных за счет параллелизации пересылок. Наиболее эффективным, очевидно, является алгоритм двоичного дерева. Суть этого алгоритма состоит в следующем: выделяется один процесс (процесс уровня 0), который ожидает поступления данных от двух подчиненных процессов (процессы уровня 1), выбирает наименьшее значение из своего τ и двух полученных по сети, после чего передает результат обоим подчиненным процессам. Каждый процесс уровня 1, в свою очередь, имеет два подчиненных процесса уровня 2. Таким образом τ, передаваемое процессу уровня 0, является минимальным из значений τ данного процесса и τ принятых от процессов уровня 2. Данная схема распространяется и на процессы уровня 2, 3, ... , так что в конечном итоге в древовидную структуру (см. рис. 9) включаются все процессы задачи. Количество пересылок при таком подходе получается равным

$$N_{trans} = 2\left(|\log_2(P)| + 1\right),$$

учитывая и обратные пересылки (распространение общего τ по сети). Замена линейной зависимости логарифмической позволяет увеличить порог насыщения по числу процессоров при любом типе разбиения.

Исходя из особенностей конкретной задачи, можно еще немного сократить количество пересылок, воспользовавшись тем фактом, что передача данных по сети происходит фиксированными порциями – "пакетами". Как правило, пакет имеет размеры порядка 1 Kb. При обмене приграничными ячейками передается сравнительно большое количество данных, которые с большой вероятностью не укладываются в целое количество пакетов. При добавлении к передающимся данным одного числа (τ), количество пакетов не изменится и время передачи, соответственно, не возрастет.

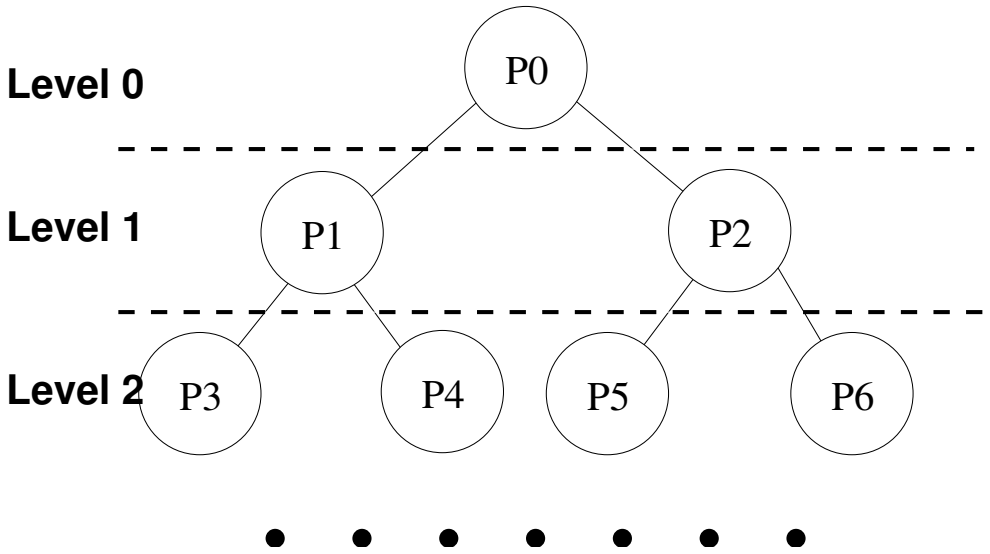

Рис. 9: Синхронизация временно́го шага τ.

Таким образом, группируя процессы по уровням с учетом взаимного расположения их подобластей, можно сэкономить одну-две пересылки, что может составлять существенный процент от общего количества передач при небольшом числе процессоров.

3. Результаты тестирования производительности

Нами был взят трехмерный вычислительный код [10], использующий схему Роу-Ошера, описанную выше. Данный код предназначен для моделирования процессов аккреции в двойной звездной системе. Моделирование производится в декартовой системе координат. Так как задача симметрична относительно плоскости $z = 0$, при расчетах учитывается только половина пространства.

Код был адаптирован к многопроцессорной архитектуре путем разбиения области по двум осям. При этом сохранилась возможность исследования одномерного разбиения, так как оно является частным случаем используемого разбиения. При расчетах использовалось разрешение сетки $121 \times 121 \times 61$ узлов с двойной точностью. Измерялось время, необходимое для расчета 100 временных шагов. Результаты тестирования приведены в таблице 1. Более наглядно, результаты тестирования представлены на

графиках (рис. 10, 11).

В таблице 1 и на графиках показаны следующие величины:

Np	–	число процессоров
Px	–	число разбиений по оси X
Py	–	число разбиений по оси Y
N	–	число ячеек
T	–	время счета (сек)
Tn	–	нормированное время счета (сек)
Eff	–	эффективность
Spd	–	ускорение

Здесь $Spd = T(Np)/T(1)$, $Eff = Spd/Np \cdot 100\%$, Np – число процессоров, задействованных при счете задачи, $T(Np)$ – время счета задачи на Np процессорах.

Так как сетка во многих случаях не может быть разделена на подобласти целочисленно, при различных разбиениях общее число ячеек $(N_x \times N_y)$ может немного различаться (графа таблицы N). Нормированное время счета, таким образом, представляет собой некое гипотетическое время работы программы, при количестве ячеек, равном $121 \times 121 \times 61$. Ускорение в данном случае представляет собой отношение нормированного времени счета ко времени счета задачи одним процессором, а эффективность – ускорение, деленное на количество процессоров. Таким образом, эффективность демонстрирует влияние накладных расходов (передача данных по сети) на время счета задачи.

В таблице 1 представлены всевозможные варианты разбиения, как одномерные ($P_y = 1$), так и двумерные. На графиках 10 (эффективность работы параллельной программы) и 11 (ускорение работы кода в зависимости от числа процессоров) точки соответствующие одномерным разбиениям обозначены звездочками, двумерные – квадратами.

На графике 10 можно наглядно наблюдать эффекты влияния кэш-памяти на скорость счета. При все более мелком разбиении сетки,

Np	Px	Py	N	T	Tn	Eff	Spd
1	1	1	14641	945.3	945.3	100%	1
2	2	1	14762	462.1	458.3	103%	2.06
3	3	1	14641	302.3	302.3	104%	3.13
4	4	1	15004	238.0	232.3	101%	4.07
	2	2	14884	225.7	222.0	106%	4.26
5	5	1	15004	191.9	187.2	100%	5.05
6	6	1	15004	165.7	161.7	97%	5.84
	3	2	14762	156.2	154.9	101%	6.10
7	7	1	14883	146.8	144.3	93%	6.55
8	8	1	15004	130.8	127.6	92%	7.41
	4	2	15128	124.1	120.1	98%	7.87
9	9	1	14641	116.1	116.1	90%	8.13
	3	3	14641	110.3	110.3	95%	8.56
10	10	1	15004	108.1	105.4	89%	8.96
11	11	1	15125	99.3	96.1	89%	9.82
12	12	1	15004	91.7	89.5	87%	10.5
	6	2	15128	87.6	84.8	92%	11.14
	3	4	15004	87.3	85.2	92%	11.09
13	13	1	14641	85.5	85.5	84%	11.04
14	14	1	15730	85.5	79.6	84%	11.87
	7	2	15006	77.4	75.5	89%	12.51
15	15	1	15004	78.9	77.0	81%	12.27
	5	3	15004	74.6	72.8	86%	12.98
16	16	1	15972	77.5	71.0	83%	13.29
	8	2	15128	71.4	69.1	85%	13.66
	4	4	15376	78.2	74.5	79%	12.68
17	17	1	14883	72.1	70.9	78%	13.31
18	18	1	15730	73.5	68.4	76%	13.81
	9	2	14762	68.6	68.0	77%	13.88
	6	3	15004	66.9	65.3	80%	14.47

Табл. 1: Результаты тестирования.

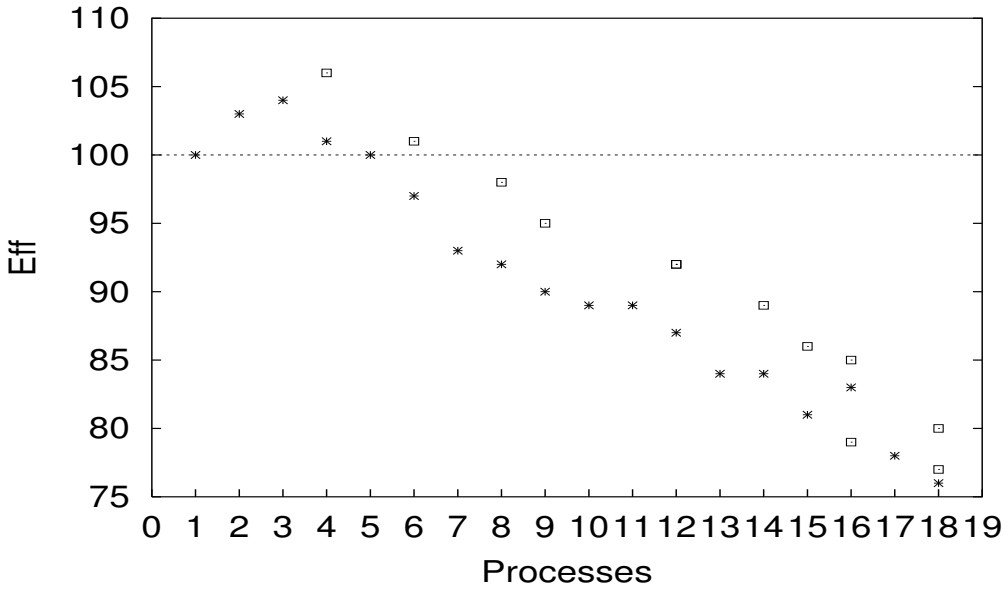

Рис. 10: Эффективность работы параллельной программы.

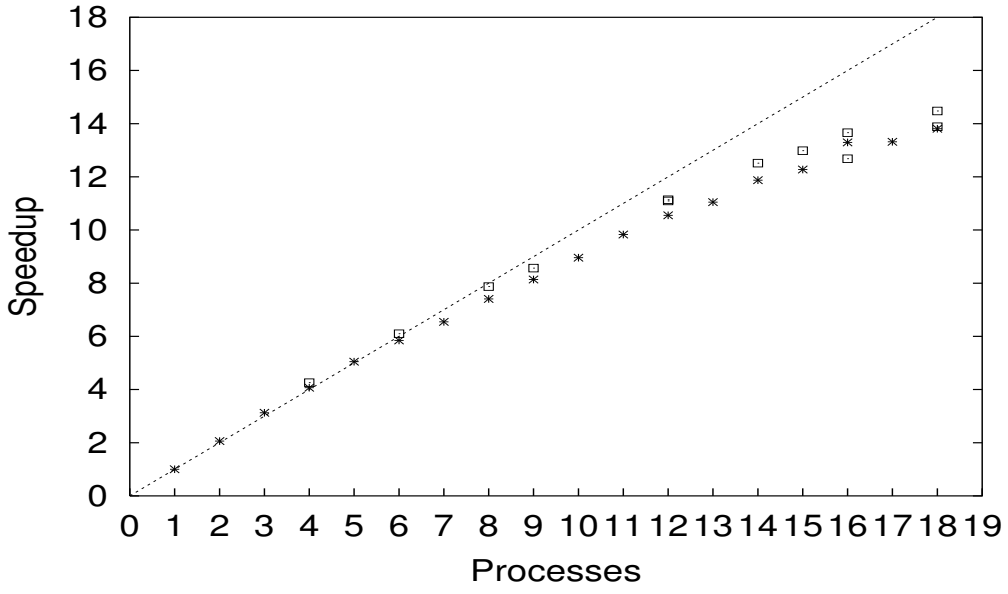

Рис. 11: Ускорение работы в зависимости от числа процессоров.

число ячеек, обрабатываемых одним процессором, уменьшается, что повышает эффективность использования кэш-памяти. В то же время, накладные расходы, источником которых являются задержки передачи данных, при малом числе процессоров не вносят еще существенного вклада в производительность. Все это приводит к росту эффективности при малом числе процессоров. Наибольшая эффективность (106% !), чего и следовало ожидать, достигается на 4-х процессорах при двумерном разбиении. При внимательном рассмотрении, видна некая ступенчатость графика 10. Данная форма графика обязана своим происхождением как сложной многоуровневой системе кэш-памяти процессора Intel Pentium-4, так и изменению количества пакетов данных, передаваемых по сети.

Описанный код был также протестирован на МВС-1000М, суперкомпьютере МСЦ РАН. Для тестов была использована сетка разрешением $121 \times 121 \times 32$ ячеек. Для сокращения числа тестов, были взяты только двумерные разбиения, по возможности симметричные (такие, чтобы число разбиений по обеим осям было, по возможности, примерно равным). Суперкомпьюьер МВС-1000М содержит 768 процессоров и работает в многопользовательском режиме. Было запущено 107 тестов, при этом число используемых процессоров варьировалось от 4-х до 306. Результаты тестирования приведены на графиках 13 и 14. Видно, что при числе процессоров < 50 производительность кода растет почти линейно. При большом числе процессоров сетка, распределяемая каждому из них, очень сильно уменьшается [2)] и время прогона начинает зависеть от множества труднопредсказуемых величин: распределения задач по процессорам кластера, загрузки коммутационной сети и т.д. Этим и объясняется такой разброс значений на графиках при большом числе процессоров. Однако видно, что производительность в среднем продолжает расти почти линейно даже при числе процессоров ~ 300.

[2)] например, при использовании 306 процессоров (102 по оси x и 3 по оси y) каждому процессору выделялось всего 5 ячеек по оси x.

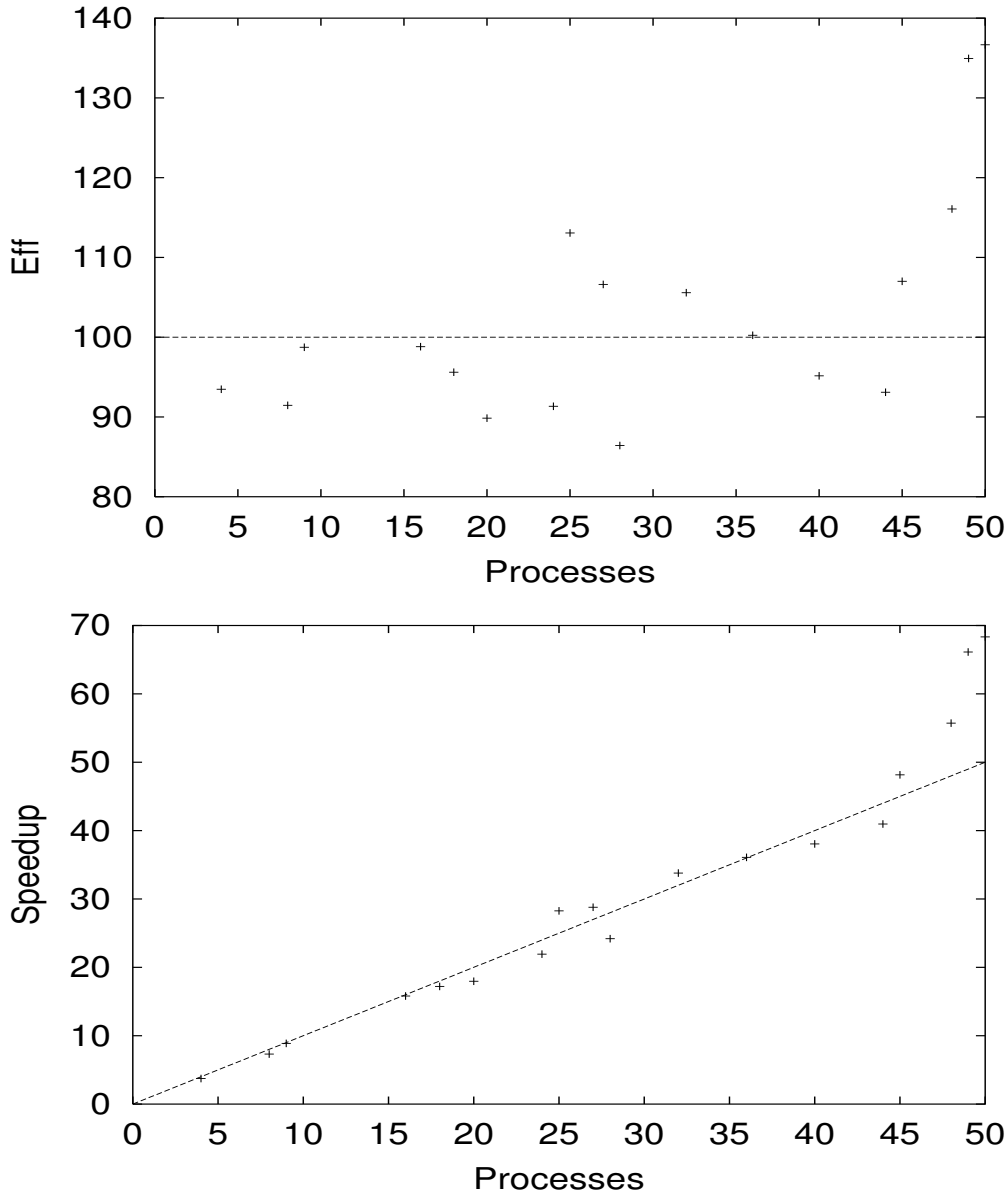

Рис. 12: Эффективность и ускорение работы в зависимости от числа процессоров для расчетов на параллельном компьютере МВС-1000М, установленном в МВЦ РАН.

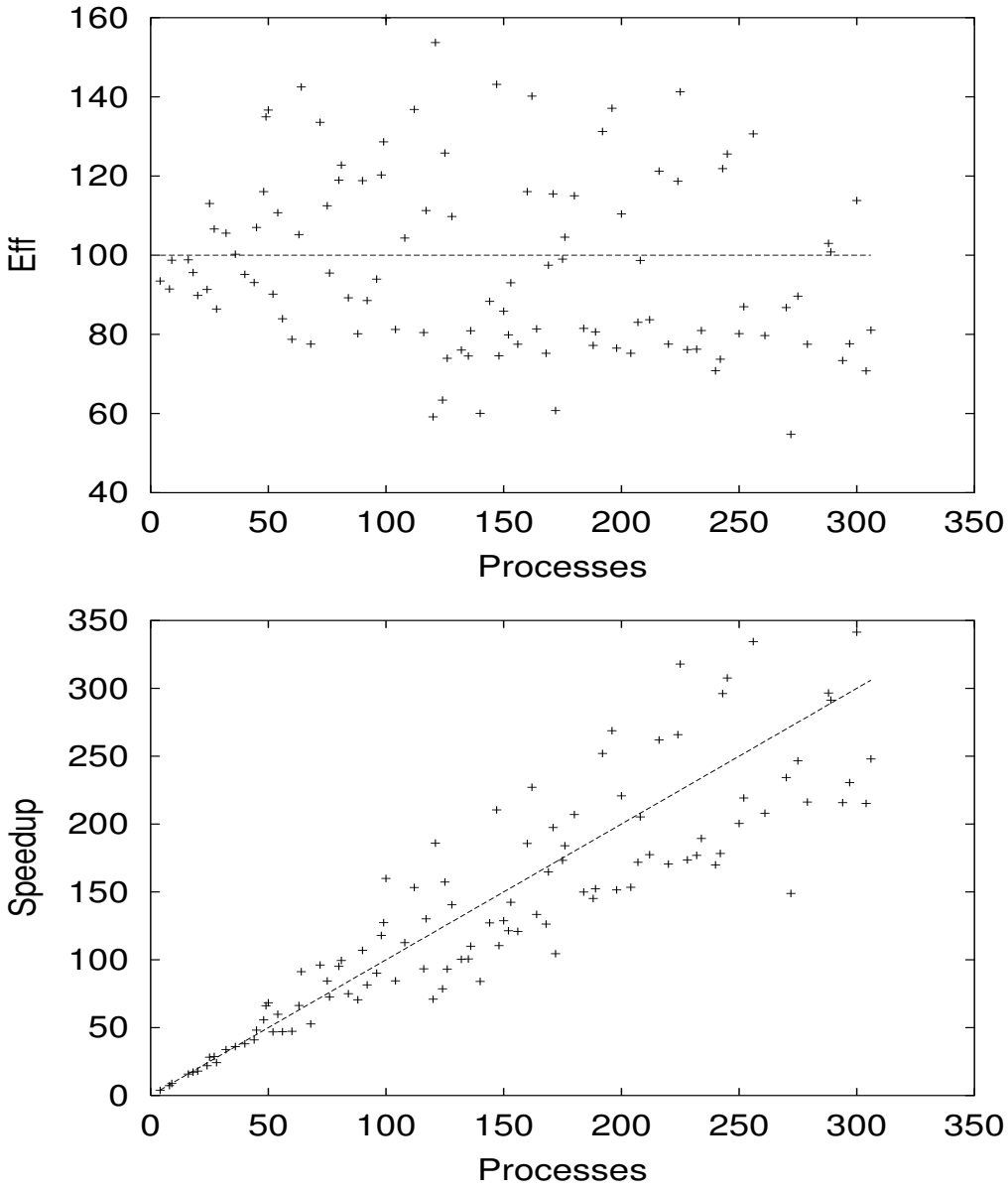

Рис. 13: То же самое для большего числа процессоров.

4. Выводы

Рассматривая результаты, приведенные в разделе 3, можно с уверенностью утверждать, что построение и использование кластеров для газодинамических расчетов явными годуновскими методами вполне оправдано как с точки зрения стоимости расчетов, так и с точки зрения эффективности. Современные программно-аппаратные средства позволяют добиться очень хорошей масштабируемости на упомянутом классе задач при числе процессоров порядка 20 и выше. Различные методы разбиения вычислительной зоны на подобласти могут приводить к существенно различной эффективности работы кода. Существует некий порог, ограничивающий максимальное количество процессоров, которые могут эффективно использоваться при обсчете задачи. При использовании машин с числом процессоров, бóльшим данного порога, ускорение будет падать при увеличении числа процессоров. Абсолютная величина порога может быть определена только с учетом как аппаратных особенностей конкретного компьютера, так и особенностей вычислительного кода. Изменение алгоритма работы кода позволяет существенно увеличить данный порог.

Приведенные графики прироста производительности показывают, что при максимальном использованном числе процессоров (18 для кластера ИНАСАН и 300 для суперкомпьютера МВС-1000М, установленном в МВЦ РАН) рост все еще не очень существенно отличается от линейного. Это говорит о том, что о достижении порога эффективности используемого кластера говорить еще рано.

Настоящая работа выполнена при поддержке Российского фонда фундаментальных исследований (гранты №№ 02-02-16088, 02-02-17642, 00-01-00392), а также Программы Президиума РАН “Математическое моделирование”. Авторы признательны Д.В.Бисикало за полезные обсуждения.

Литература

[1] Годунов С.К., 1959, Мат. Сборник, **47**(89), 271.

[2] Roe P.L., 1986, Ann. Rev. Fluid Mech., **18**, 337.

[3] Harten A., 1983, J. Comp. Phys., **49**, 357.

[4] Sweby P.K., 1984, SIAM J. Numer. Anal., **21**, 995.

[5] Вязников К.В., Тишкин В.Ф., Фаворский А.П., 1989 Мат. Моделирование, 1989, **1**, № 1, 95.

[6] Roe P.L., 1983, in Proc. 1983 AMS–SIAM Summer Seminar on Large Scale Computing in Fluid Mechanics, Lectures in Applied Math., **22**, 163, Philadelphia: SIAM.

[7] van Leer B., 1977, J. Comp. Phys., **23**, 276.

[8] van Leer B., 1974, J. Comp. Phys., **14**, 361.

[9] Chakravarthy S.R., Osher S., 1985, AIAA Pap., № 85-0363.

[10] Boyarchuk A.A., Bisikalo D.V., Kuznetsov O.A., Chechetkin V.M., 2002, Mass transfer in close binary stars, London: Taylor & Frances.

Оглавление