НЕКОТОРЫЕ ПОДХОДЫ К ПОСТРОЕНИЮ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ ВЫЧИСЛИТЕЛЬНОЙ ГИДРОДИНАМИКИ*

С.В.Тимченко

НИИ ПММ при Томском госуниверситете

Введение

В последнее время, в связи с широким распространением сравнительно недорогих многопроцессорных систем, резко вырос интерес к параллельным методам вычислительной гидродинамики. Чрезвычайно быстрый рост производительности современных компьютеров дал реальную возможность приступить к решению задач гидро- и газодинамики, которые еще совсем недавно были недоступны для численного моделирования. При этом необходимо отметить, что применение традиционных подходов при решении задач математической физики приводит к тому, что используется лишь малая часть современных вычислительных мощностей.

Следуя [1], отметим четыре требования к вычислительным алгоритмам, обеспечивающие их успешную адаптацию к архитектуре многопроцессорных вычислительных систем с распределенной памятью:

1) *внутренний параллелизм*, позволяющий разбить исходный алгоритм на некоторое количество подзадач (по возможности совпадающее с числом процессоров);
2) обеспечение *равномерности загрузки* процессоров;
3) *минимизация обмена информацией* между процессорами;
4) *простота* алгоритма.

Если первое из этих требований носит алгоритмический характер, то остальные три связаны в значительной степени с экономической стороной вопроса. Действительно, только существенное снижение стоимости расчетов может побудить исследователя перейти на новую вычислительную технику и новые вычислительные алгоритмы. Если в стоимость вычислений закладывать, в частности, оплату труда исследователя, то простота алгоритма (и, как следствие, скорость переноса программ с последовательных компьютеров на параллельные) становится чрезвычайно существенной.

С точки зрения практической методологии при построении параллельных алгоритмов часто выделяют следующие три подхода:

* Работа выполнена при поддержке РФФИ, грант №02-01-01022.

1. *Конвейерное решение.* При решении многих нестационарных задач возможна следующая декомпозиция исходного алгоритма – каждый процессор получает решение некоторой подзадачи в некоторый момент времени, после чего это решение передается другим процессорам для использования при решении других подзадач, а данный процессор продолжает решать свою подзадачу для последующих моментов времени.
2. *Геометрическое решение.* При нахождении решения многих гидродинамических задач приходится сталкиваться с регулярными геометрическими структурами с пространственно-ограниченным взаимодействием. Благодаря этой однородности исходные данные могут быть единообразно распределены между процессорами, каждый из которых будет находить решение поставленной задачи в отведенной ему геометрической подобласти при ограниченном обмене информацией с другими процессорами. Как правило, этот обмен связан с необходимостью аппроксимации производных на границах подобластей.
3. *Коллективное решение.* При решении многих физических задач необходимо неоднократно выполнять одну и ту же часть программы с различными, слабосвязанными друг с другом наборами входных данных. Например, при решении различных задач оптимизации естественным является распределение по процессорам программы вычисления целевой функции.

1. Пример геометрического параллелизма

Рассмотрим задачу расчета течений в пространственном турбулентном пограничном слое (ТПС). На основе принципа геометрического параллелизма построим численный алгоритм для решения этой задачи на параллельной вычислительной системе с распределенной памятью.

1.1. Постановка задачи

Рассмотрим задачу пространственного обтекания затупленных тел сложной формы гиперзвуковым потоком вязкого однородного газа под углами атаки и скольжения. Для численного решения задачи будем использовать криволинейную систему координат $\{x^i\}$, нормальным образом связанную с поверхностью обтекаемого тела: x^3 отсчитывается по нормали к телу, а x^1 и x^2 некоторым образом выбраны на его поверхности. В системе координат $\{x^i\}$ соответствующие уравнения ТПС имеют следующий безразмерный вид:

$$D_i(\rho u^i a^{1/2} a_{(ii)}^{-1/2}) = 0,$$

$$\rho\,(Du^{\alpha} + A^{\alpha}_{\beta\delta}u^{\beta}u^{\delta}) = -\frac{\gamma-1}{2\gamma}a^{\alpha\beta}a^{1/2}_{(\alpha\alpha)}D_{\beta}P + D_3(\mu D_3 u^{\alpha}),$$

$$\rho DT = \frac{\gamma-1}{\gamma}D^{*}P + D_3\left(\frac{\mu}{\mathrm{Pr}}D_3 u^{\alpha}\right) + 2\mu B_{\alpha\beta}D_3 u^{\alpha}D_3 u^{\beta},$$

$$p = \rho T, \mu = \mu_l + \Gamma\mu_t, \left(\frac{\mu}{\mathrm{Pr}}\right) = \left(\frac{\mu}{\mathrm{Pr}}\right)_l + \Gamma\left(\frac{\mu}{\mathrm{Pr}}\right)_t,$$

$$D_i = \partial/\partial x^i, D^{*} = u^{\alpha}a^{-1/2}_{\alpha\alpha}D^{\alpha},$$

$$D = D^{*} + u^3 D_3, B_{\alpha\beta} = a^{\alpha\beta}a^{-1/2}_{(\alpha\alpha)}a^{-1/2}_{(\beta\beta)},$$

$$A^{\alpha}_{\alpha\alpha} = \frac{a_{12}}{a}\left(\frac{\partial\sqrt{a_{(\alpha\alpha)}}}{\partial x^{\beta}} + \frac{a_{12}}{a_{(\alpha\alpha)}}\frac{\partial\sqrt{a_{(\alpha\alpha)}}}{\partial x^{\alpha}} - \frac{1}{\sqrt{a_{(\alpha\alpha)}}}\frac{\partial a_{12}}{\partial x^{\alpha}}\right),$$

$$A^{\alpha}_{12} = A^{\alpha}_{21} = \frac{1}{2a}\left[\sqrt{a_{11}a_{22}}\left(1 + \frac{a^2_{12}}{a_{11}a_{22}}\right)\frac{\partial\sqrt{a_{(\alpha\alpha)}}}{\partial x^{\beta}} - 2a_{12}\frac{\partial\sqrt{a_{(\beta\beta)}}}{\partial x^{\alpha}}\right],$$

$$A^{\beta}_{\alpha\alpha} = \frac{\sqrt{a_{(\beta\beta)}}}{a}\left(\frac{\partial a_{12}}{\partial x^{\alpha}} - \sqrt{a_{(\alpha\alpha)}}\frac{\partial\sqrt{a_{(\alpha\alpha)}}}{\partial x^{\beta}} - \frac{a_{12}}{\sqrt{a_{(\alpha\alpha)}}}\frac{\partial\sqrt{a_{(\alpha\alpha)}}}{\partial x^{\alpha}}\right), \alpha \neq \beta.$$

Данная система решается с граничными условиями на поверхности тела и на внешней границе пограничного слоя:

$$x^3 = \infty : u^{\alpha} = u^{\alpha}_e(x^1, x^2), T = T_e(x^1, x^2);$$

$$x^3 = 0 : u^{\alpha} = u^{\alpha}_w(x^1, x^2), T = T_w(x^1, x^2), (\rho u^3)_w = G(x^1, x^2).$$

Здесь все линейные размеры отнесены к характерному линейному размеру L, а нормальная координата x^3 – к $L/\sqrt{\mathrm{Re}}$; $\sqrt{2c_{p0}T_0}u^{\alpha}$, $\sqrt{2c_{p0}T_0}u^{\alpha}/\sqrt{\mathrm{Re}}$ – физические компоненты вектора скорости; PP_0, $\rho\rho_0, TT_0, \mu\mu_0$ – соответственно давление, плотность, температура и коэффициент вязкости; Г – коэффициент перемежаемости; $a_{\alpha\beta}$ – ковариантные компоненты первой квадратичной формы поверхности тела. Нижние индексы *w, 0, e* здесь и далее соответствуют значениям газодинамических параметров на поверхности тела, в набегающем потоке и на внешней границе пограничного слоя соответственно. Нижние индексы *l* и *t* соответствуют ламинарным и

турбулентным значениям параметров. $\mathrm{Re} = \sqrt{2c_{p0}T_0}\rho_0 L/\mu_0$ – число Рейнольдса, $\mathrm{Pr} = \mu c_p/\lambda$ – число Прандтля.

При определении коэффициента турбулентной вязкости использовалась алгебраическая модель Себеси-Смита, а коэффициент перемежаемости Г в переходной области течения вычислялся на основе формул Чена и Тайсона.

Значения $u^\alpha = u_e^\alpha(x^1,x^2), T_e(x^1,x^2)$ на внешней границе пограничного слоя определяются из решения следующей системы уравнений:

$$\rho_e(Du_e^\alpha + A_{\beta\delta}^\alpha u_e^\beta u_e^\delta) = -\frac{\gamma-1}{2\gamma}a^{\alpha\beta}a_{(\alpha\alpha)}^{1/2}D_\beta P,$$

$$\rho_e DT_e = \frac{\gamma-1}{\gamma}D^* P,$$

где $P(x^1,x^2)$ – известное распределение давления по поверхности тела. В гиперзвуковом случае для определения *P* используется формула Ньютона:

$$P = \cos^2\alpha,$$

где α – угол между вектором скорости набегающего потока и нормалью к поверхности тела.

В рамках данной постановки рассматриваются течения в пространственном пограничном слое около поверхности трехосных эллипсоидов и гиперболоидов, обтекаемых под углами атаки и скольжения.

1.2. Общий алгоритм решения задачи

Для численного решения исходной задачи использовалась специально выбранная система координат на поверхности тела (подробно описанная в [2]), представляющая собой систему координат полярного типа x^1 – радиальная и x^2 – угловая координата с началом в критической точке $x^1 = 0$. Для удобства численного интегрирования с учетом особенности выбранной системы координат осуществлялся переход к новым переменным типа переменных Дородницына, позволяющим разрешить особенности в исходных уравнениях на критической линии. Использовалась неявная конечно-разностная схема, имеющая четвертый порядок аппроксимации по поперечной координате, второй порядок по окружной координате и первый порядок точности по маршевой координате, представляющая собой обобщение на трехмерный случай схемы Петухова и использующая переменный шаг по нормальной координате. Методические расчеты, проведенные на последовательности сгущающихся сеток, показали, что для

достижения точности в 1% по распределениям теплового потока вдоль поверхности тела достаточно использовать сетку, содержащую 15 точек по нормальной координате, 64 точки по окружной координате и 30 точек по маршевой координате. Поэтому в дальнейшем все серийные расчеты были проведены на этой сетке.

В конвективном операторе производные по маршевой координате заменяются разностями против потока, а производные по окружной координате заменяются центральными разностями на основе решения, полученного на предыдущей итерации на текущей окружности. Так как данная система координат вырождается в критической точке, то для решения исходных уравнений в критической точке использовалась невырожденная криволинейная система координат. После определения решения в невырожденной системе координат производился пересчет в старую систему координат.

Используемый метод решения фактически является вариантом метода релаксации по строкам, построенного на основе метода прогонки из работы Петухова. Несмотря на то, что этот метод является неявным, он очень хорошо может быть адаптирован для параллельных компьютерных систем с распределенной памятью, так как декомпозиция расчетной области в окружном направлении x^2 позволяет получать решение в каждой подобласти на текущей итерации независимо от других подобластей, а из-за использования высокого порядка аппроксимации по поперечной координате количество обменов между отдельными процессорами сравнительно невелико.

1.3. Декомпозиция расчетной области

Вышеописанный численный метод решения уравнений ТПС реализуется в виде программного кода 3DTBL, предназначенного для использования на сети из N компьютеров, обменивающихся пакетами данных при помощи пакета MPI. Вычислительная область Ω разбита на N непересекающихся подобластей Ω_j (j =1,..., N). Каждая подобласть дискретизируется на K сеточных точек в направлении x^2:

$$\Omega = \bigcup_{J=1}^{N} \Omega_j, \Omega_j = \left\{ \left(x^1, x^2, x^3 \right) \mid 0 \le x^1 \le x^1_*; x^2_j \le x_2 \le x^2_{j+1}; 0 \le x^3 \le x^3_e \right\}.$$

1.4. Общая стратегия

Для удобства дальнейшего изложения введем следующие обозначения: *SEND(T_i, T_j, A)* и *RECV(T_i, T_j, A)*, которые соответственно означают посылку (получение) структуры данных на узле кластера T_i на (из) узел(ла) кластера T_j

(нижние индексы i и j отличаются на единицу). Этим операторам соответствуют простейшие функции MPI: *MPI_SEND* и *MPI_RECV*. Будем также считать, что индексная арифметика выполняется по модулю N. Выражение A^m_j означает, что структура данных A относится к точке сетки с номером m в окружном направлении на узле T_j . В сеточных точках с номерами от 1 до K решение вычисляется с использованием процессора с тем же самым номером, а необходимые для замыкания задачи в размещенной на этом кластере вычислительной подобласти структуры данных A^0_j и A^{K+1}_j передаются с "соседних" процессоров. Тогда схема вычислений может быть представлена следующим способом:

1**.** Решение в критической точке независимо получается на каждом процессоре. На этой стадии алгоритм не параллелен.

2. При получении решения системы уравнений ТПС на боковой поверхности тела для конечно-разностного представления производных в окружном направлении необходимо знать значения неизвестных функций на предыдущей итерации в соседних сеточных точках в направлении x^2. Для этого перед каждой итерацией каждый нечетный процессор сначала посылает необходимые значения функций левым и правым "соседним" процессорам, а потом получает соответствующую информацию от них. Каждый четный процессор, наоборот, первоначально получает информацию от правого и левого "соседних" процессоров и после этого посылает соответствующие данные обратно. Посылка и получение информации на процессоре T_j могут быть представлены следующим способом:

IF(odd(j))
    SEND(T_j, T_{j+1}, A^K_j)
    SEND(T_j, T_{j-1}, A^1_j)
    RECV(T_j, T_{j+1}, A^{K+1}_j)
    RECV(T_j, T_{j-1}, A^0_j)
ENDIF
IF(even(j))
    RECV(T_j, T_{j-1}, A^0_j)
    RECV(T_j, T_{j+1}, A^{K+1}_j)
    SEND(T_j, T_{j-1}, A^1_j)
    SEND(T_j, T_{j+1}, A^K_j)
ENDIF

Здесь предикаты *odd* и *even* дают значение "истина" в случаях нечетности и четности своих аргументов соответственно.

3. При этом шаге каждый процессор получает решение уравнений ТПС в вышеописанной подобласти на текущей итерации полностью, независимо от других процессоров.

4. После окончания текущей итерации необходимо проверить определенный критерий *D* сходимости итерационного процесса. Этот процесс заканчивается при достижении сходимости для всех точек x^1=const, т.е. при реализации условия $\forall j : \left| Y_j^{(i)} - Y_j^{(i-1)} \right| < \delta$, где δ – некоторое число; *i* – номер текущей итерации; *j* – номер процессора; Y_j – вектор неизвестных функций. Для синхронизации вычислительных процессов на различных процессорах необходимо одновременное завершение итераций на всех процессорах, для чего при помощи функции *MPI_REDUCE* вычисляется величина $D = \bigwedge_j D_j$, где $D_j = \left(\left| Y_j^{(i)} - Y_j^{(i-1)} \right| < \delta \right)$. Если значение *D*="ложь", то выполняется шаг 2. В противоположном случае переменная x^1 увеличивается на Δx^1, и после этого выполняется шаг 2.

Более подробно данный алгоритм (в том числе для гетерогенных кластерных систем) рассматривается в работе [3].

1.5. Эффективность параллельного алгоритма

Важнейшей характеристикой любого параллельного алгоритма является его эффективность *E,* т.е. величина

$$E = \frac{t_1}{N t_N},$$

где t_i – время решения задачи на *i* процессорах.

В данной задаче проблема уменьшения объема обмена данных между процессорами решается за счет использования схемы высокого порядка аппроксимации в направлении x^3, по которому производится декомпозиция расчетной области. В результате число обменов между процессорами невелико по сравнению с объемом вычислений. Поэтому можно дать приближенную оценку эффективности:

$$E = \frac{1 + K \cdot L}{N + K \cdot L},$$

где *K* – число точек вдоль тела; *L* – число точек в окружном направлении; *N* – число процессоров. Снижение эффективности параллельного алгоритма с увеличением числа процессоров объясняется исключительно наличием нераспараллеленного участка в программе (начальный участок – получение решения в критической точке – дублируется на всех процессорах). Поэтому

увеличение числа точек в продольном либо окружном направлении приводит к увеличению эффективности распараллеливания. Некоторые результаты расчетов на кластере Томского госуниверситета приведены в таблице.

Эффективность параллельного алгоритма, %

Число процессоров	Сетка K=30,L=64	Сетка K=20,L=64	Сетка K=30,L=32
2	99	99	99
4	98	98	97
8	97	96	94
16	94	93	88

2. Пример коллективного решения

Как показывает анализ литературы [4-6], большинство параллельных вычислительных алгоритмов в той или иной мере основано на декомпозиции вычислительной области на ряд подобластей, в каждой из которых решение ищется на отдельном процессоре. Особенно широко распространены параллельные алгоритмы для явных разностных схем. Однако применение

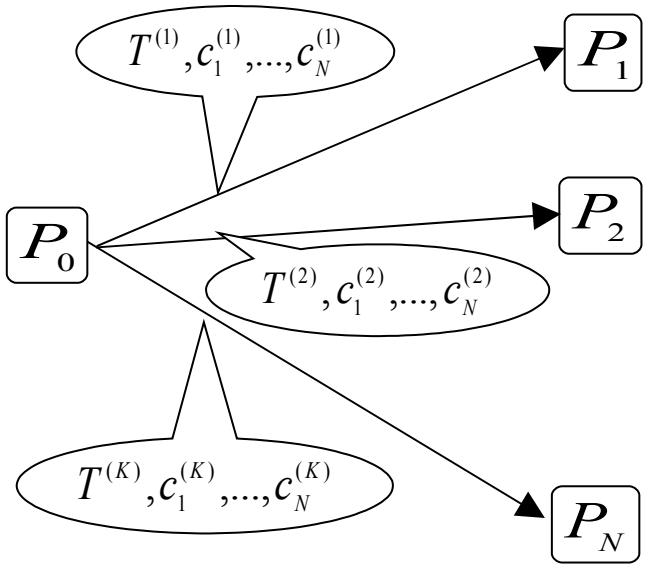

Рис.1. Рассылка нулевым процессом значений температуры и концентраций при помощи функции MPI_SCATTER

различных быстрых методов решения задач линейной алгебры, таких как прогонка, препятствует в этих случаях применению декомпозиции при построении параллельных алгоритмов.

При решении задач гидродинамики многокомпонентных химически

неравновесных реагирующих смесей основное время вычислений тратится на расчет коэффициентов переноса – коэффициентов вязкости, теплопроводности и диффузии. Этот факт позволяет предложить другой подход к построению параллельных вычислительных алгоритмов:

1. Основной вычислительный блок размещается на корневом (нулевом) процессоре, на остальных процессорах размещается программа расчета коэффициентов переноса.
2. После инициализации параллельного приложения (инициализации системы передачи сообщений) на корневом (нулевом) процессоре запускается основной вычислительный блок.
3. На каждой итерации корневой процесс рассылает подчиненным процессам необходимую для расчета коэффициентов переноса информацию (значения температуры и концентраций). В используемой параллельной библиотеке передачи сообщений MPI это осуществляется функцией *MPI_SCATTER* (рис. 1).
4. После расчета коэффициентов переноса их значения собираются на корневом процессе при помощи функции *MPI_GATHER* (рис. 2).

Другим характерным примером построения параллельных алгоритмов на основе коллективного решения являются многие методы глобальной оптимизации (например эволюционные и генетические алгоритмы и методы управляемого случайного поиска [5-6]).

Список литературы

1. *Четверушкин Б.Н.* Кинетически-согласованные схемы в газовой динамике. Новая модель вязкого газа, алгоритмы, параллельная реализация, приложения. – М.: Изд-во Моск. ун-та, 1999. – 232 с.
2. *Бородин А.И., Пейгин С.В.* Пространственный тонкий вязкий ударный слой при отсутствии в течении плоскостей симметрии //Изв. АН СССР. МЖГ. – 1989. – № 2. – С.150-158.
3. *Пейгин С.В., Тимченко С.В.* Расчет пространственного турбулентного пограничного слоя на сети компьютеров различной производительности //ЖВМиМФ. – 1998. – Т.38, №3. – С. 510 - 519.
4. Parallel Computation Fluid Dynamics. New Trends and Advances /J.Periaux et al. (ed). Elsevier Science B.V. Amsterdam, 1995. – 560 p.

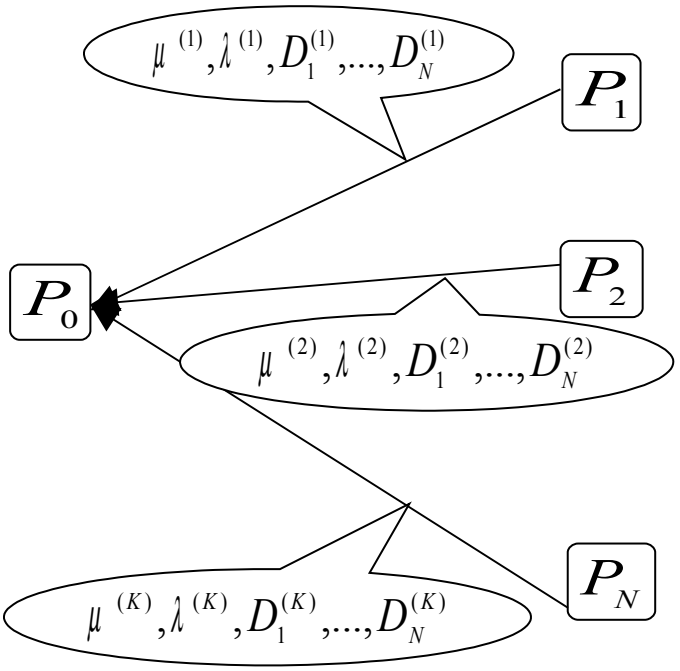

Рис.2. Получение нулевым процессом значений коэффициентов переноса при помощи функции MPI_GATHER

5. Parallel Computation Fluid Dynamics. Resent Developments and Advances Using Parallel Computers /J.Periaux et al. (ed). Elsevier Science B.V. Amsterdam, 1998. – 682 p.
6. Parallel Computation Fluid Dynamics. Developments and Applications of Parallel Technology /J.Periaux et al. (ed). Elsevier Science B.V. Amsterdam, 1999. – 534 p.