УДК 681.3:518.5

ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА "ПОТОК-3": ОПЫТ ПАРАЛЛЕЛИЗАЦИИ ПРОГРАММНОГО КОМПЛЕКСА. ЧАСТЬ 1. ИДЕОЛОГИЯ РАСПАРАЛЛЕЛИВАНИЯ

Г. А. Тарнавский[1], **В. Д. Корнеев**[2], **Д. А. Вайнер**[3], **Н. М. Покрышкина**[4], **А. Ю. Слюняев**[4], **А. В. Танасейчук**[4], **А. Г. Тарнавский**[3]

Рассматриваются технологические аспекты параллелизации вычислительной системы "Поток-3", предназначенной для моделирования задач аэродинамики и физической газовой динамики. Предложены и исследованы способы и проблемы глобального распараллеливания программного комплекса по главным входным параметрам, а также C-, L- и V-типы параллелизации процедур основного итерационного ядра системы для локального ускорения операций.

1. Введение. В работах [1, 2] была разработана общая стратегия параллелизации большого вычислительного комплекса, предназначенного для компьютерного моделирования задач аэродинамики и физической газовой динамики. В [3, 4] рассматривались различные схемы распараллеливания операций многомерной скалярной прогонки, составляющих основной алгоритм численного решения системы дифференциальных уравнений Навье–Стокса динамики вязкого теплопроводного газа, и проводились многоцелевые вычислительные эксперименты по определению эффективных способов распараллеливания и их реализации на различных мультипроцессорных системах. В [5] изучались некоторые аспекты организации проведения расчетов задач в предметной области, связанной с обеспечением численного метода начальными данными, и частично обсуждались проблемы вывода полученной информации и размещения ее на долгосрочное хранение в базах данных.

Настоящая работа представляет идеологию и опыт параллелизации вычислительной системы и многосторонний анализ примененных различных типов распараллеливания.

2. Глобальная параллелизация. Из рассмотренных в [1, 2] общих видов параллелизации, связанных с физико-математической, геометрической и технологической декомпозицией полной задачи на ряд исполняемых параллельно подзадач на первом этапе реорганизации однопроцессорного программного комплекса "Поток-2" в параллельную вычислительную систему "Поток-3", ориентированную на многопроцессорный счет с возможностью ее использования широким кругом пользователей, было применено несколько способов распараллеливания. Одним из них является глобальная параллелизация системы по главным входным параметрам.

Рассмотрим физическую сущность этого способа. Вычислительный комплекс обеспечивает решение задачи обтекания летательного аппарата потоком газа в широком диапазоне определяющих параметров. Главными из них (здесь имеются в виду только физические параметры, определяющие основные режимы течения; алгоритмические параметры, хотя на них так же может быть распространена излагаемая ниже идеология, здесь не рассматриваются) являются: форма поверхности тела, тепловое условие на нем, высота и скорость полета, термодинамические свойства газа. Для ясности изложения выберем два из них, например, высоту H и скорость V полета. Как правило, для изучения процессов обтекания необходимо провести значительное количество расчетов с вариацией как H, так и V. Такой цикл расчетов может быть организован в едином запуске вычислительного комплекса с параллелизацией "по глобальному параметру", схема которой приведена на рис. 1.

Здесь показан ряд уровней распараллеливания, образующий некоторый односвязный граф: древовидную ветвящуюся непересекающуюся систему связей, выходящих из единого корня — входа в программ-

[1] Институт теоретической и прикладной механики СО РАН, ул. Институтская, 4/1, 630090, г. Новосибирск; e-mail: tarnav@itam.nsc.ru

[2] Сибирский суперкомпьютерный центр СО РАН, просп. Лаврентьева, 6, 630090, г. Новосибирск; e-mail: korneev@ssd.sscc.ru

[3] Новосибирский государственный университет, ул. Пирогова, 2, 630090, г. Новосибирск; e-mail: dmv@mail.ru; tarnavsky@chat.ru

[4] Новосибирский государственный технический университет, просп. Маркса, 20, 630092, г. Новосибирск; e-mail: zuo@hotbox.ru; vinny@ngs.ru; inko@mail.ru

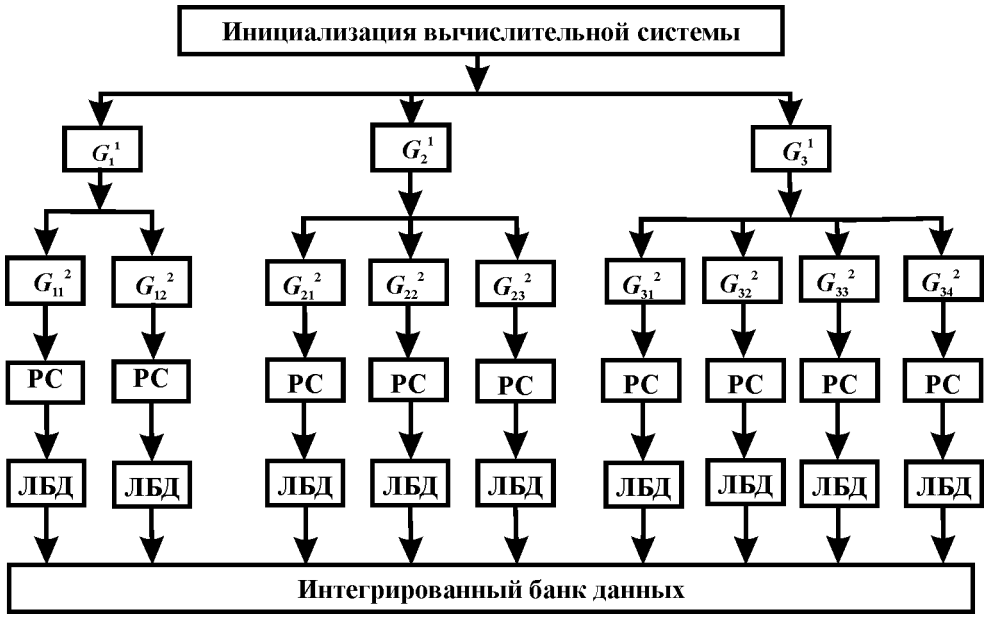

Рис. 1. Глобальная параллелизация вычислительной системы "Поток-3"

ный комплекс. Каждый уровень представлен символом

$$G^m_{g_1,\dots,g_m} : \ m \in [1, M];\ g_1 \in [1, g_{1F}];\ \dots\ g_m \in [1, g_{mF}], \tag{1}$$

где введены следующие обозначения (все индексы целочисленные):

m — номер горизонтального уровня графа (отсчитываемого от корня), или, говоря другими словами, номер вида глобального параметра, по которому производится параллелизация (например, $G^1 \equiv H$ (высота полета), $G^2 \equiv V$ (скорость полета) и т.д.);

M — глубина распараллеливания (число горизонтальных уровней графа), т.е. длина списка физико-математических параметров задачи, по которым производится параллелизация;

g_{1F} — длина списка (число вариантов) значений параметра первого уровня;

g_1 — текущий номер в этом списке;

g_{mF} — длина списка (число вариантов) значений параметра в одной подгруппе m-уровня;

g_m — текущий номер в этом списке.

Заметим, что в общем случае длины списков во всех подгруппах m-уровня могут быть различными и определяться принадлежностью к той или иной подгруппе предыдущего уровня, что формализованно может быть представлено рекуррентной формулой

$$g_{mF} = g_{mF}(g_{m-1}), \quad m = \overline{2, M}. \tag{2}$$

Такая ситуация показана на рис. 1, иллюстрирующем двухуровневое глобальное распараллеливание. На первом G^1-уровне длина списка параметров $g_{1F} = 3$, т.е. организуется решение задачи с тремя значениями высоты полета H_1, H_2, H_3. На втором G^2-уровне длины списков параметров определяются принадлежностью к той или иной подгруппе (ветви графа): для первой длина списка (число вариантов) $g_{1,2F} = 2$, для второй — $g_{2,2F} = 3$, для третьей — $g_{3,2F} = 4$. В поясняющих рис. 1 терминах это означает заказ на организацию расчета в варианте с высотой полета $H = H_1$ двух подвариантов со значениями скорости полета $V = V_{11}$ и V_{12}, в варианте с $H = H_2$ — трех подвариантов со значениями $V = V_{21}, V_{22}, V_{23}$, в варианте с $H = H_3$ — четырех подвариантов со значениями $V = V_{31}, V_{32}, V_{33}, V_{34}$. После организации файлов вариантов входных параметров с "индивидуальными" значениями $G^1_{g_1}$, $G^2_{g_1,g_2}$ каждый файл (окончательная, наиболее удаленная от корня графа структура) передает данные в программный комплекс РС и инициирует его процессорную систему. Подчеркнем еще раз, что обмена данными в процессе исполнения всего задания в целом не происходит между процессорными подгруппами G^2, ни тем более между процессорными блоками G^1. В этом состоит сущность данного типа параллелизации — распараллеливание по глобальным параметрам для программного комплекса, т.е. по основным входным параметрам (как правило, скалярам), определяющим физико-математический смысл задачи.

По окончании каждого процесса результаты вычислений, имеющие физический смысл и необходимые для дальнейшего анализа, передаются в локальную базу данных (ЛБД). После полного завершения всех процессов ЛБД объединяются в интегрированный банк данных.

Вообще говоря, процессорное время, затрачиваемое на исполнение операций решения одиночной задачи (независимого процесса), существенно зависит от используемых начальных данных: может оказаться целесообразным вначале заканчивать процесс, например G^m_{11}, и затем принимать его результаты в качестве стартовых условий для процесса G^m_{12} (подробнее см. [5]). Однако этот вопрос лежит вне рамок настоящей работы; здесь будет предполагаться полная независимость всех процессов $G^m_{g_1,g_2}$. В этом случае имеет место параллелизация SIMD-типа: единые инструкции (весь программный комплекс, обозначенный как PC на рис. 1) — множественность данных (список $G^m_{g_1,g_2}$) вообще без обмена данными между процессами.

Минимально требуемое для такого распараллеливания число процессоров P_G определяется суммой длин списков всех подгрупп уровня, наиболее удаленного от корня графа. В представленном на рис. 1 частном случае двух уровней имеем

$$P_G = \sum_{g_1=1}^{3} g_{2F}(g_1). \tag{3}$$

В общем случае вид формулы (3) усложняется:

$$P_G = \sum_{g_{m-1}=1}^{g_{(m-1)F}} g_{MF}(g_{m-1}) \tag{4}$$

и требует привлечения рекуррентной формулы (2).

Данный тип параллелизации, т.е. распараллеливание по глобальным параметрам, с точки зрения собственно MPI-подхода принципиальных трудностей не вызвал, не было также проблем и конфликтов в отладке, однако при разработке выявился ряд технически сложных проблем унифицированного входа в систему, конфигурации процессорного пространства и организации автоматизированного сбора со всех задействованных процессоров полученной информации в интегрированный банк данных вычислительной системы.

Кратко рассмотрим проблемы организации входа в систему и ветвление по параметрам. Из более чем 50 глобальных входных параметров вычислительной системы "Поток-3", определяющих физический (высота и скорость полета, тепловой режим тела и температура его поверхности и т.п.), математический ("геометрический" тип задачи — плоская, осесимметричная, пространственная, вид системы координат — декартова, сферическая и т.д.), алгоритмический (топология вычислительной сетки — размерность, коэффициенты сгущения узлов, итерационные параметры и т.п.) и режимно-технический (управление вводом-выводом информации) смыслы задачи, были отобраны 12 "основных" (перебор вариантов), распараллеливание по которым наиболее целесообразно. Максимальная глубина распараллеливания M, т.е. количество уровней параллелизации (см. рис. 1 — уровни G^1 и G^2), для первого этапа была выбрана равной 4 (отладка проводилась, как правило, для $M = 2$). Был построен граф параллелизации, позволяющий в произвольном порядке размещать основные параметры по уровням: для приведенных выше примеров ($G^1 \equiv H$) вначале определялся список высот полета, для которых необходимо получить решение, а затем для каждого элемента этого списка H_{g1} определялся свой список скоростей полета ($G^2 \equiv V$), причем списки V_{g_1,g_2} могут быть разными для каждого H_{g_1} как по длине (количеству вариантов), так и по их конкретным значениям. Возможна и обратная параллелизация ($G^1 \equiv V$, $G^2 \equiv H$): вначале определяется список скоростей полета, для которых необходимо получить решение, а затем для каждого элемента этого списка V_{g_1} — список высот полета H_{g_1,g_2}.

Следует отметить, что устойчивое функционирование этого графа в смысле разумной постановки задачи и запуска системы специалистами в области вычислительной аэродинамики (выбранными в качестве тест-группы), не связанными с разработкой данного комплекса, обеспечивалось только до уровня $M = 2$, что, по-видимому, и должно быть реализовано в системе, ориентированной на широкие круги пользователей, поскольку использование более глубоких уровней $M = 3$ или $M = 4$ вызывало существенные затруднения. Однако собственно идеология и наработки, полученные в ходе разработки и опытной эксплуатации распараллеливания по глобальным параметрам с использованием графа многоуровневого ветвления, могут быть полезны при создании параллельных вычислительных систем в других областях знания.

Другой проблемой этого типа параллелизации была проблема организации потоков выходной информации. Даже отдельная задача в прикладном аспекте порождает большой файл данных, требующих

структуризации: "заголовок" (список метаданных, т.е. параметров решенной задачи, полностью ее характеризующий и позволяющий однозначно ее идентифицировать и отличить от других; это особо важно при проведении не одиночных расчетов, а циклов для комплексного изучения какой-либо научной или прикладной проблемы), "том" (графика или таблицы), "глава" (таблицы различных физических величин — плотности, давления, температуры в поле течения, аэродинамические характеристики летательного аппарата и т.п.), "страница" (часть "главы", представляющая физическую функцию в какой-либо подобласти параметров) и др., вплоть до "строки" или "колонки" (см. любые таблицы по газовой динамике, например ставшие классическими [6]). Это структурирование должно обеспечивать в дальнейшем достаточно простой доступ к полученной информации и возможность сравнения данных из разных "томов".

В принципе здесь может быть использована какая-нибудь достаточно надежная СУБД типа Oracle, Access или Paradox, однако этот вопрос еще требует своего решения. В представляемой разработке использовалась специальная система маркировки выходной информации, применяемая при записи файлов во временный банк данных проводимого расчета, содержащая главный и вспомогательный ключи.

Главный ключ — индивидуальный (неповторяющийся) идентификатор проводимого расчета составлялся из семи символов: одной буквы и шести цифр. Буквы H, B, N и F (Head, Body, Nearwake, Farwake) являлись краткими наименованиями баз данных, в которых размещались результаты расчетов в различных геометрических сегментах задачи (подробнее см. [1]); первые три цифры показывали виртуальный номер процессора, на котором проводился расчет варианта; последние три цифры обеспечивали сквозную нумерацию варианта в H-, B-, N- или F-базе данных, составляющей интегрированный банк данных. При такой структуре имен имеют место некоторые ограничения: максимально используемое число процессоров, как и число записей в одном томе базы данных, не должно превышать 999. Кроме того, на момент исполнения задания образуются дополнительные каналы связи, помеченные литерами C, L, S и G (Chemical, Local, Standard, Graphics), соответствующими:

1) каналу обмена данными между сегментами "химия", "горение", "турбулентность" и "газовая динамика" вычислительного комплекса;

2) каналу передачи данных из локальной БД текущего расчета в любой модуль комплекса по его запросу;

3) каналу ввода стандартных данных "Ближний след" (Nearwake);

4) каналу записи данных для последующей графической обработки.

При работе с базами имена файлов для каналов передачи информации составлялись из восьми символов, первый из которых — буквы W, R, U или P — означали тип потока данных (Write — запись, Read — чтение, Universal — запись и чтение, Print — файл печати данных), остальные семь совпадали с символами главного ключа. Так, например, символ WH001004 означает, что канал предназначен для записи в базу данных результатов расчета обтекания головной части тела, полученных на процессоре с виртуальным номером 001, и что эта запись размещается в БД под номером 004.

Главный ключ дополняется вспомогательными ключами — метаданными проведенного расчета (главными входными параметрами, определяющими физический смысл задачи, такими, как уже указывавшиеся выше высота H и скорость V полета). Главный ключ, в отличие от вспомогательных ключей, однозначно идентифицирует запись в БД, поскольку в БД, естественно, может содержаться несколько записей с одним и тем же значением (например, H).

Вообще говоря, эта система, далекая от завершения, хотя и позволила определенным образом более или менее осознанно управлять потоками информации и селектировать получаемые выходные данные различных процессов (а их может быть значительное количество), оказалась громоздкой и не вполне удобной при эксплуатации, требовала внимания администратора системы и высокой дисциплины вычислителей, поскольку в противном случае в БД появлялись записи с одинаковыми именами. Это вызывало необходимость создания временных БД, их просмотра администратором и только затем переноса записей в БД долгосрочного хранения.

В целом проблеме фиксирования информации, получаемой на мультипроцессорных вычислительных системах, в настоящее время практически не уделяется должного внимания, поскольку использование МВС вычислителями-прикладниками зачастую осуществляется в режиме типа "распараллелил программу — запустил задачу — посмотрел результаты — и все (в лучшем случае записал числа на CD)". Аналогичными являются и проблемы визуализации (графического представления) решений, полученных при глобальной параллелизации по входным параметрам, однако этот вопрос слишком объемен и лежит вне рамок настоящей работы.

3. Локальная параллелизация. Локальное распараллеливание, относящееся по классификации [1] к технологическому типу параллелизации "низшего уровня", предназначено для ускорения процесса вычи-

сления в подпрограммах, составляющих основное алгоритмическое ядро PC (см. рис. 1) вычислительного комплекса "Поток 3" [3]:

$$\{PC\} = \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{i=1}^{I} \sum_{j=1}^{J} k - \mathrm{OPERATIONS}\,(i, j;\, n), \tag{5}$$

$$k - \mathrm{OPERATIONS}\,(i, j;\, n) \Longrightarrow F_{ij}^{kn}. \tag{6}$$

Символическая запись (5) – (6) означает следующее. Результатом функционирования k-й процедуры алгоритма, реализованной в виде отдельной подпрограммы

$$\mathrm{SUBROUTINE}\,(k,\ \text{input parameters},\ \text{output parameters}), \quad k \in [1, K], \tag{7}$$

на n-ом итерационном слое t_n (n-ом шаге по времени t)

$$t_n = \varphi_n(n), \quad n \in [1, N] \tag{8}$$

является двумерный массив F_{ij}^{kn} — прямоугольная таблица чисел (для краткости изложения рассматривается двумерная задача предметной области — плоская и осесимметричная), где i и j нумеруют соответственно по каждому (y, x)-координатному направлению узлы расчетной сетки, дискретизирующей непрерывную дифференциальную задачу:

$$y_i = \varphi_y(i), \quad i \in [1, I], \tag{9}$$

$$x_j = \varphi_x(j), \quad j \in [1, J]. \tag{10}$$

При использовании равномерного пространственно-временного шаблона формулы (8) – (10) принимают вид $t_n = \tau n$; $y_i = i\Delta y$; $x_j = j\Delta x$, где τ, Δy и Δx — шаги по времени, y- и x-координатам.

Четырехкратная сумма (5), разделяя каждым отдельным суммированием различные по математическому и алгоритмическому смыслу операции OPERATIONS, призвана обратить внимание на их большое количество. Две внутренние суммы с индексами i и j в (5) символизируют операции в каждом (i, j)-узле расчетной сетки OPERATIONS (i, j). Средняя сумма, помеченная индексом k, означает использование набора отдельных процедур (7), реализующих различные аспекты k-OPERATIONS вычислительного алгоритма на отдельном итерационном шаге n k-OPERATIONS $(i, j;\, n)$. И, наконец, внешняя сумма с индексом n означает, что для завершения решения всей задачи в целом алгоритм должен сделать N глобальных итерационных циклов.

Подпрограммы k-OPERATIONS $(i, j;\, n)$ многократно исполняются в главном итерационном процессе решения сложной системы дифференциальных уравнений и содержат внутри себя циклические операторы перебора индексов; внутри некоторых циклов содержится значительное число арифметических операций.

Для ускорения исполнения всех k-процедур главного итерационного блока программного комплекса были разработаны и использованы C-, L-, V- и W-типы параллелизации, соответствующие распараллеливанию операций "по столбцам", "по строкам", "по вектору" и "без распараллеливания" ("Column"-, "Line"-, "Vector"- и "Without"-parallelizations).

Условная схема параллелизации основного вычислительного ядра программного комплекса "Поток-3" (обозначенного на рис. 1 символом PC) представлена на рис. 2.

Каждый из типов параллелизации был ориентирован на индивидуальные алгоритмические и реализующие их программные особенности каждой k-процедуры с целью получить максимальный выигрыш в ускорении вычислительного процесса с минимизацией, насколько это возможно, времени информационного интерфейса: подготовки к распараллеливанию процессором-диспетчером P_0, рассылки заданий и данных по процессорам-исполнителям P_1, P_2, P_3 и т.д., сбора данных после исполнения заданий, их реорганизации в единый массив данных и передачи управления в следующую $(k+1)$-процедуру.

Заметим, что количество и порядок следования C-, L-, V- и W-типов распараллеленных процедур определяется алгоритмом и конкретным видом реализующих его подпрограмм.

Рассмотрим подробнее сущность каждого типа параллелизации (ниже, если это не оговаривается особо, имеется в виду параллельная реализация подпрограмм комплекса, ориентированного на многопроцессорные системы с распределенной памятью).

Неявный конечно-разностный алгоритм главного итерационного ядра вычислительного комплекса "Поток-3" основан на расщеплении системы дифференциальных уравнений Навье–Стокса

$$\frac{\partial F}{\partial t} = W\Big(F,\ \frac{\partial F}{\partial x},\ \frac{\partial F}{\partial y},\ \frac{\partial^2 F}{\partial x^2},\ \frac{\partial^2 F}{\partial y^2},\ \frac{\partial^2 F}{\partial x \partial y}\Big), \tag{11}$$

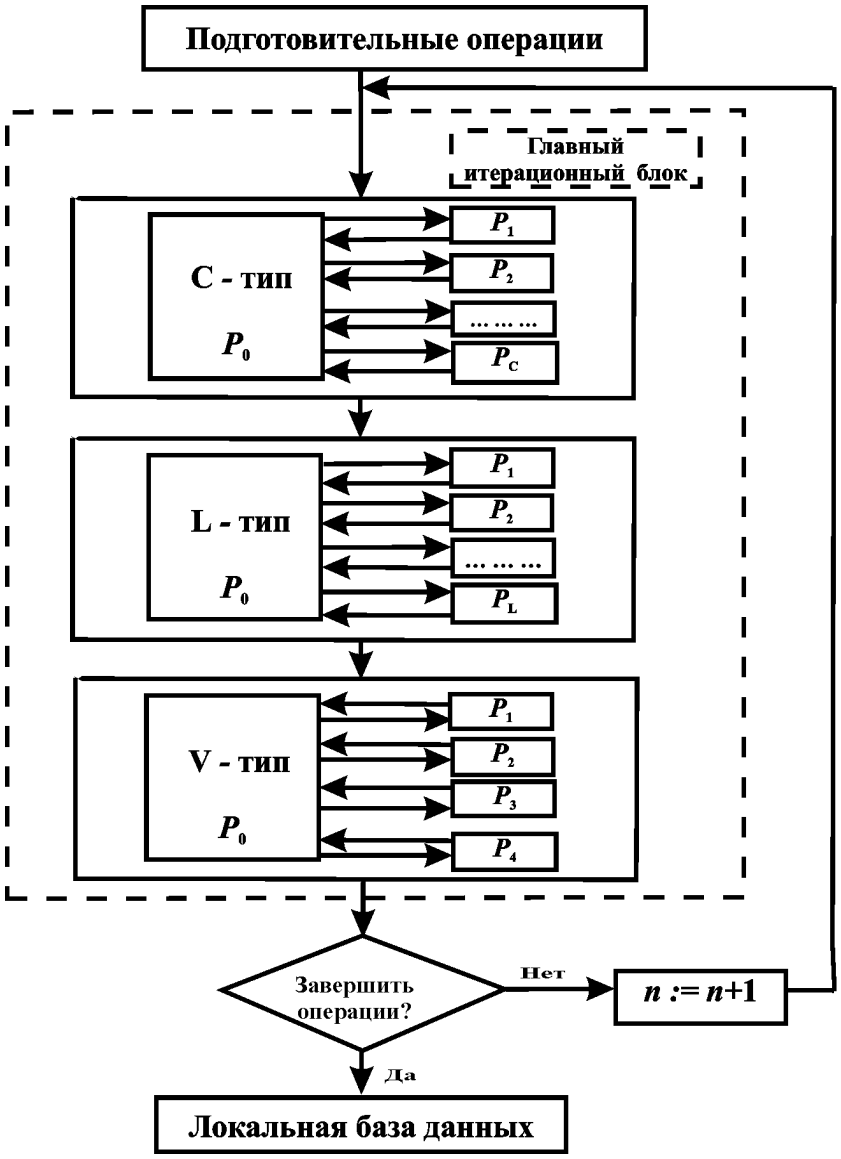

Рис. 2. Локальная параллелизация основного алгоритмического ядра вычислительной системы "Поток-3"

$$F = F(t, x, y) \tag{12}$$

по координатным направлениям x и y:

$$W = W_1 + W_2 + W_3. \tag{13}$$

Эти уравнения применяются для моделирования нестационарных задач аэродинамики и физической газовой динамики.

Расщепление (13) производится таким образом, чтобы разделить, по возможности, члены, содержащие координатные производные:

$$W_1 = W_1\Big(F, \frac{\partial F}{\partial x}, \frac{\partial^2 F}{\partial x^2}\Big), \tag{14}$$

$$W_2 = W_2\Big(F, \frac{\partial F}{\partial y}, \frac{\partial^2 F}{\partial y^2}\Big), \tag{15}$$

$$W_3 = W_3\Big(F, \frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial^2 F}{\partial x^2}, \frac{\partial^2 F}{\partial y^2}, \frac{\partial^2 F}{\partial x \partial y}\Big), \tag{16}$$

где W_1 содержит производные только по x, W_2 — только по y, а W_3 является некоторой "координатно-нерасщепляющейся" частью.

Блоки системы уравнений W_1 и W_2 разделяются далее по физическим процессам ("динамика" и "диссипация"):

$$W_1 = W_{11} + W_{12}, \tag{17}$$

$$W_2 = W_{21} + W_{22}. \tag{18}$$

При этом зависимости W_{11}, W_{12}, W_{21}, W_{22} от вектора искомых функций F, записанные в матричной форме

$$W_{11} = A_{11}F, \tag{19}$$

$$W_{12} = A_{12}F, \tag{20}$$

$$W_{21} = A_{21}F, \tag{21}$$

$$W_{22} = A_{22}F, \tag{22}$$

характеризуются следующими свойствами: матрицы A_{11} и A_{21} являются диагональными, а матрицы A_{12} и A_{22} имеют существенную заполненность ненулевыми элементами (даже без диагонального преобладания).

Конкретный вид формул (11) – (22) для дальнейшего изложения значения не имеет и здесь не приводится (более подробно см. [7, 8]).

Данная структура метода расщепления (11) – (22) детерминирует в общей стратегии параллелизации программного комплекса тактику локального (технологического по терминологии [1]) распараллеливания подпрограмм, реализующих этапы численного алгоритма (11) – (18) с учетом их особенностей (19) – (22).

4. C-тип параллелизации. Этот тип распараллеливания части вычислительного алгоритма ориентирован на ускорение вычислений в подпрограммах, реализующих расчет членов уравнений типа (15), т.е. содержащих производные (точнее, конечно-разностные операторы) только по y-координатному направлению. В сеточном (дискретном) пространстве операций (6) данный тип операндов может быть записан (индекс k для простоты записи снят) как

$$F_{ij}^{n+\nu} = \mathrm{OPERATIONS}\,(F_{ij}^{n},\ F_{i\pm1,j}^{n},\ F_{i\pm2,j}^{n}). \tag{23}$$

Запись (23) означает, что по известным значениям F на n-слое вычисляются значения на $(n+\nu)$-слое. Символ ν означает некоторый "дробный" (например, $\nu = \dfrac{1}{6}$, $\nu = \dfrac{2}{6}$ и т.д.) этап перехода на $(n+1)$ итерационный слой, однако в представляемой работе с неизбежной беглостью изложения конкретизация и уточнения алгоритма нецелесообразны (подробнее см. [7]), поскольку для дальнейшего это значения не имеет.

В операциях типа (23) имеет особое, даже ключевое значение то обстоятельство, что при вычислении массива $F_{ij}^{n+\nu}$ используются элементы массивов F_{ij}^{n} только с вариацией первого индекса (i, $i \pm 1$, $i \pm 2$): имеет место сцепление операций по первому (i) индексу и их независимость по второму (j) индексу. Это означает возможность проведения операций одновременно (параллельно) по всем колонкам (столбцам) расчетной сетки, что показано на рис. 3, иллюстрирующем сущность C-типа ("Column") распараллеливания.

Общее сеточное пространство R

$$\{R_{ij}\} = i \otimes j \tag{24}$$

в области (9) – (10) может быть разделено на P_C подпространств ("вертикальных колонок"), в каждой из которых операции (23) являются независимыми процессами и могут быть выполнены параллельно:

$$F_{ij}^{n+\nu} = \sum_{p=1}^{P_C} {}^{p}F_{ij}^{n+\nu}. \tag{25}$$

Число независимых процессов P_C, естественно, должно удовлетворять условию

$$P_C \leqslant J. \tag{26}$$

Таким образом, для ускорения вычислений в (23) могут быть использованы P_C процессоров (машин).

Размеры колонок (см. рис. 3) могут быть разными, поскольку в общем случае при $P_C < J$ отношение J/P_C не является целым числом, но, очевидно, разбиение R на почти одинаковые колонки оптимально,

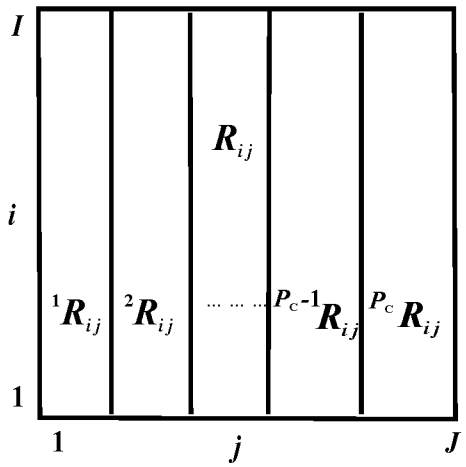

Рис. 3. C-тип параллелизации вычислений

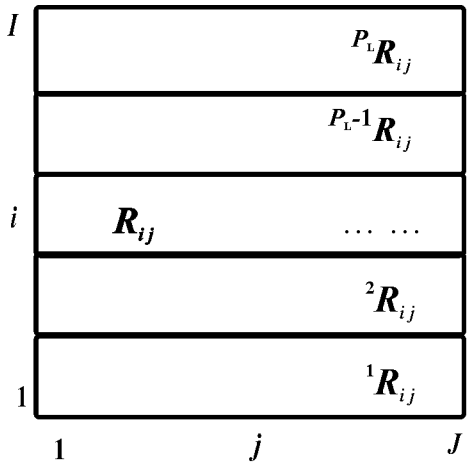

Рис. 4. L-тип параллелизации вычислений

поскольку при этом число операций в колонках почти одинаково; тем самым время ожидания общего завершения всех процессов, т.е. операций (23) и (25) в R, минимизируется.

Реализация этого типа распараллеливания в программном комплексе "Поток-3" была осуществлена следующим образом. По заданному количеству процессоров P_C особый процессор-диспетчер P_0 производит сегментацию R-пространства на P_C подпространств, т.е. разделяет все задействованные в (23) массивы F_{ij} на P_C подмассивов $^pF_{ij}^{n+\nu}$ и рассылает их в P_C машин, которые осуществляют с ними операции (23) и затем возвращают их процессору-диспетчеру, составляющему по (25) полный массив $F_{ij}^{n+\nu}$.

5. L-тип параллелизации. Этот вид распараллеливания части вычислительного алгоритма ориентирован на ускорение вычислений в подпрограммах, реализующих расчет членов типа (14), т.е. содержащих производные (точнее, конечно-разностные операторы) только по x-координатному направлению. В сеточном (дискретном) пространстве операций (6) данный тип может быть записан (индекс k для простоты записи снят) как

$$F_{ij}^{n+\mu} = \text{OPERATIONS}\,(F_{ij}^{n},\ F_{i,j\pm1}^{n},\ F_{i,j\pm2}^{n}). \tag{27}$$

Так же, как и выше, запись (27) означает, что по известным значениям F на n-слое вычисляются значения на $(n+\mu)$-слое. Здесь μ означает некоторый промежуточный этап перехода на $(n+1)$ итерационный слой. В операциях (27) имеет место, в отличие от операций (23), сцепление вычислений по второму (j) индексу и их независимость по первому (i) индексу. Это означает возможность проведения операций одновременно (параллельно) по всем линиям (строкам) расчетной сетки, что показано на рис. 4, иллюстрирующем сущность L-типа ("Line") распараллеливания. Общее сеточное R-пространство (24) в области (9) – (10) может быть разделено на P_L подпространств ("горизонтальных полос"), в каждом из которых операции (27) являются независимыми и могут быть выполнены параллельно:

$$F_{ij}^{n+\mu} = \sum_{p=1}^{P_L} {}^pF_{ij}^{n+\mu}. \tag{28}$$

Число независимых процессов P_L, очевидно, должно удовлетворять условию

$$P_L \leqslant I. \tag{29}$$

Таким образом, для ускорения вычислений в (27) могут быть использованы P_L процессоров (машин).

Размеры полос (см. рис. 4) могут быть разными, поскольку в общем случае при $P_L < I$ отношение I/P_L не является целым числом, но, естественно, разбиение R на почти одинаковые полосы оптимально, поскольку при этом число операций в полосах почти одинаково; тем самым время ожидания общего завершения всех процессов, т.е. операций (27) – (28) в R, минимизируется.

Реализация этого типа распараллеливания в программном комплексе "Поток-3" аналогична реализации C-типа. По заданному количеству процессоров P_L особый процессор-диспетчер P_0 осуществляет

распределение R-пространства на P_L подпространств, т.е. разделяет все задействованные в (27) массивы F_{ij} на P_L подмассивов ${}^pF_{ij}$ и направляет их в P_L машин, которые осуществляют с ними операции (27) и затем возвращают их процессору-диспетчеру, составляющему по (28) полный массив $F_{ij}^{n+\mu}$.

Несмотря на идеологически практическое тождество ("вертикальные колонки" и "горизонтальные полосы"), C- и L-типы параллелизации весьма существенно отличаются при их компьютерной реализации на многопроцессорных системах. Рассмотрим конкретно функционирование процессора-диспетчера (см. рис. 5), распределяющего элементы полного массива a_{ij} ($I = 4$, $J = 4$) по процессорам-исполнителям ($P_C = P_L = 3$).

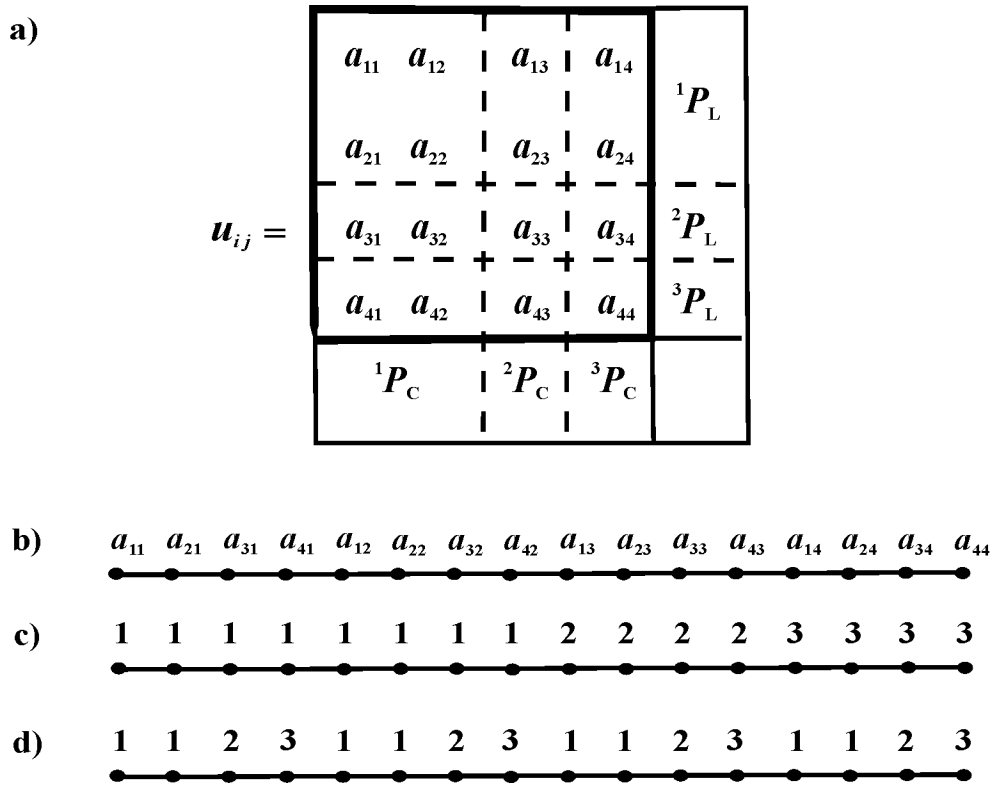

Рис. 5. Реконфигурация процессорного пространства: a) двумерное представление массива a_{ij}, $i \in [1,4]$; $j \in [1,4]$; b) одномерная Фортран-развертка двумерного массива a_{ij}; c) распределение элементов массива a_{ij} при C-типе параллелизации на трех процессорах; d) распределение элементов массива a_{ij} при L-типе параллелизации на трех процессорах

На рис. 5 a показано традиционное математическое представление массива a_{ij} с распределением его элементов вдоль строк (вариация j) и вдоль столбцов (вариация i). В компьютерном представлении двумерный массив a_{ij} реконфигурируется в некоторое адресное пространство, которое можно для иллюстрации трактовать в геометрическом смысле как одномерную прямую линию (рис. 5 b) с последовательным расположением элементов массива a_{ij}. В начале этой линии расположен элемент a_{11}. Дальнейшая расстановка элементов зависит от языка программирования. В языке Фортран двумерный массив a_{ij} "развертывается" в одномерный вариацией первого индекса (i) с фиксированным вторым (j); затем сменой j на $j+1$ и снова вариацией i во всем своем диапазоне изменения и т.д.: a_{11}, a_{21}, a_{31}, ... (см. рис. 5 b). Таким образом, развертка осуществляется "по столбцам" (в языке Си — наоборот). При C-типе параллелизации ("по колонкам") процессор-диспетчер при разделении массива a_{ij} на P_C подмассивов (см. рис. 5 c, где $P_C = 3$) определяет в адресном пространстве (см. рис. 5 b) расположение соответствующих элементов a_{ij}.

На рис. 5 c над узлами, изображающими элементы a_{ij}, точнее, их адреса, поставлены цифры, соответствующие номерам процессоров-получателей этих элементов, выполняющих над ними арифметические операции (23) и по их окончании возвращающих эти подмассивы диспетчеру для "сборки" в общий массив (25). В общем адресном пространстве адреса элементов этих подмассивов располагаются "плотными группами", образуя "односвязные подмножества", что существенно упрощает работу с ними и минимизирует процессорное время, необходимое на реконфигурацию массива a_{ij} в подмассивы ${}^pa_{ij}$, $p \in [1,3]$.

При L-типе параллелизации ("по строкам") процессор-диспетчер при разделении массива a_{ij} на P_L подмассивов (см. рис. 5 d, где $P_L = 3$) определяет в адресном пространстве (см. рис. 5 b) расположение соответствующих элементов a_{ij}. На рис. 5 d над узлами, изображающими элементы a_{ij}, точнее, их адреса, поставлены цифры, соответствующие номерам процессоров-получателей этих элементов, выполняющих над ними арифметические операции (27) и по их окончании возвращающих эти подмассивы диспетчеру для "сборки" в общий массив (28). В общем адресном пространстве адреса элементов ${}^p a_{ij}$ располагаются по-другому, изолированными группами, образуя некоторые "многосвязные подмножества". Расстояния между адресами первых элементов этих групп равны в индексном пространстве значениям высоты (ширины) "полосы" (см. рис. 4).

В приведенном на рис. 4 примере имеем "ширину полосы" или 2, или 1, т.к. среднее значение ширины

$$S = \frac{I}{P_L} = \frac{4}{3}\,. \tag{30}$$

Таким образом, элементы 1, 2 и 3 подмассивов будут располагаться группами по два элемента для первой полосы и уединенно — для второй и третьей полосы, а интервалы между элементами одной и той же группы, например первой, будут равны

$$\mathrm{Int} = \frac{J(P_L - 1)}{P_L}\,. \tag{31}$$

Отметим, что в случаях, когда I не только не делится нацело на P_L, но и используются полосы разной ширины, формула (31) существенно усложняется.

После выборки элементов массива a_{ij} и рассылки их диспетчером по соответствующим процессорам-получателям данные элементы, образующие некоторое одномерное пространство, реорганизуются в двумерные подмассивы ${}^p a_{ij}$. По окончании выполнения над ними операций (27), подмассивы ${}^p a_{ij}$ возвращаются диспетчеру, который по (28) интегрирует их в общий массив a_{ij}, т.е. в адресном пространстве (рис. 5 b) размещает новые значения элементов массива a_{ij} по соответствующим позициям.

В связи с вышесказанным можно констатировать, что L-тип параллелизации в практической реализации значительно более сложен, чем C-тип, требует существенного внимания от программиста и бóльших компьютерных ресурсов (времени реконфигурации вычислительного пространства).

6. V-тип параллелизации. Этот вид распараллеливания части вычислительного алгоритма ориентирован на ускорение вычислений в подпрограммах, реализующих расчет членов уравнений типа (19) и (21) и может дополнять в них C- и L-типы параллелизации. V-тип ("Vector") параллелизации детерминируется диагональностью матриц A_{11} и A_{21} в (19) и (21), что приводит к независимости в некоторых подпрограммах комплекса "Поток-3" операций вычисления компонент вектора $F = (F_1, F_2, F_3, F_4)$ (см. (12)). Для трехмерной задачи вектор F имеет пять компонент (подробнее см. [7]).

Этот тип распараллеливания достаточно прост в реализации: процессор-диспетчер рассылает необходимые данные четырем процессорам-исполнителям, которые после проведения необходимых операций возвращают новые значения массивов $F_{1_{ij}}^{n+\mu}$, $F_{2_{ij}}^{n+\mu}$, $F_{3_{ij}}^{n+\mu}$, $F_{4_{ij}}^{n+\mu}$ (здесь, как и выше, символом μ обозначен некоторый "дробный" шаг продвижения решения в общем алгоритме от F^n к F^{n+1}).

Обозначим для общности число процессоров, необходимых для этого типа параллелизации, через P_V (плюс процессор-диспетчер, хотя в данном случае процесс организации данных, их рассылки и последующего приема весьма незатратен по компьютерным ресурсам и диспетчеру можно "поручить" работу над одной из компонент вектора F, уменьшая на единицу число процессоров-исполнителей). Рассмотрим кратко некоторые технические вопросы, связанные с соотношением значений P_C, P_L и P_V. Если первые два могут принимать любые положительные значения, то значение P_V жестко детерминировано: $P_V = 4$ для двумерной и $P_V = 5$ для трехмерной задачи.

Если выбрать значения P_C или P_L большими (этот выбор определяется только возможностями доступа вычислителя к той или иной многопроцессорной системе с различными техническими данными), то в момент исполнения подпрограмм, параллелизованных по V-типу, будут находиться в режиме ожидания, т.е. простаивать, $(\max(P_C, P_L) - P_V)$ процессоров, что не вполне оптимально.

Заметим, что подобная проблема существует и для соотношения значений P_C и P_L, которая может быть решена простым выбором $P_C = P_L$. Вообще говоря, этот вопрос более сложен, чем кажется на первый взгляд, и весьма дискуссионен, поскольку требования к эффективности решения научной задачи и оптимальности использования ресурсов ЭВМ могут находиться в сильном противоречии. Разумеется, никто не станет возражать против того, чтобы научная задача была решена эффективно с минимальными затратами компьютерных ресурсов, но если нет оптимума между этими полюсами, то следует, на наш

взгляд, предпочесть эффективное решение с существенно "непродуктивным" использованием МВС, чем "экономное" использование МВС для получения неэффективного решения.

Возвращаясь после данного отступления к конкретной проблеме, можно определить некоторый способ оптимизации соотношения (P_C, P_L) и P_V. Процедуры, допускающие V-тип параллелизации, в алгоритмах системы "Поток-3" допускают также или C-тип, или L-тип распараллеливания. В связи с этим можно: или вообще отказаться от V-типа; или дополнительно к V-типу применить C-тип с другими, чем в общесистемном режиме, значениями $P'_C \neq P_C$, такими, чтобы выполнялось $P_V \cdot P'_C = P_C$ для C-типа параллелизации; или для L-типа со значениями $P'_L \neq P_L$, такими, чтобы выполнялось $P_V \cdot P'_L = P_L$.

7. W-тип параллелизации. В главном вычислительном ядре программного комплекса "Поток-3", реализуемом в итерационном цикле, т.е. с многократно исполняемыми подпрограммами, существует единственная подпрограмма, выполняющая операции (16), параллелизация вычислений в которой затруднительна или нецелесообразна по всем рассмотренным выше C-, L- и V-типам. Для общности терминологии этот тип условно назван W-типом ("Without" — без распараллеливания).

Поскольку эта подпрограмма достаточно заметна по затратам времени вычислений (около 20 %) в общем потоке подпрограмм (5), следует обсудить теоретические и практические возможности ее распараллеливания. Подчеркнем, что в этом разделе речь идет о способах локального ускорения вычислений (технологического распараллеливания — см. [1]), т.е. операций в подпрограммах. Широко применяющееся крупномасштабное геометрическое распараллеливание, заключающееся в декомпозиции полной расчетной области на ряд подобластей, в которых производятся одни и те же операции с последующей сшивкой решений, лежит совершенно в иной плоскости и здесь не рассматривается (см. [1, 2]).

Основным препятствием, затрудняющим использование C- и L-типов параллелизации, является наличие в (16) смешанных производных $\dfrac{\partial^2 F}{\partial x\,\partial y}$, дискретный аналог которых на конечно-разностной сетке приводит к сцеплению значений во всех ее узлах:

$$F_{ij}^{n+\nu} = \mathrm{OPERATIONS}\,(F_{ij}^n,\ F_{i\pm1,j}^n,\ F_{i\pm2,j}^n,\ F_{i,j\pm1}^n,\ F_{i,j\pm2}^n). \tag{32}$$

Вообще говоря, эту трудность можно преодолеть, применяя при параллелизации (32) сегментирование расчетной области

$$\{R\}_{ij},\quad i \in [1, I],\quad j \in [1, J] \tag{33}$$

на P подобластей "с перекрытием":

$$\{R\}_{ij}^p:\quad i \in [I_1^p - 2, I_2^p + 2],\quad j \in [J_1^p - 2, J_2^p + 2],\quad p \in [1, P]. \tag{34}$$

Вычисления производятся в их "центральной части"

$$\{R\}_{ij}^p:\quad i \in [I_1^p, I_2^p],\quad j \in [J_1^p, J_2^p],\quad p \in [1, P], \tag{35}$$

а значения F вдоль границ по периметрам

$$\{DR\}_{ij}^p = \{R\}_{ij}^p - \{R_C\}_{ij}^p \tag{36}$$

используются для расчета конечно-разностных аналогов первых и вторых производных вблизи этих границ. Естественно, должно выполняться

$$\{R\}_{ij} = \sum_{p=1}^{P} \{R_C\}_{ij}^p \tag{37}$$

и

$$I_1^1 = 1,\quad J_1^1 = 1,\quad I_2^P = I,\quad J_2^P = J. \tag{38}$$

Вопрос об организации счета (32) вблизи границы полной расчетной области (33) и, соответственно, организации счета в приграничных сегментах (35) здесь не рассматривается, поскольку определяется общей алгоритмикой решения дифференциальной задачи и постановкой краевых условий.

Сегментация (33) – (38) может быть применена для крупномасштабного геометрического распараллеливания всего алгоритма решения (11) – (12) в целом (см. [1]), однако в [2, 4] показана ее неэффективность для конкретных вычислительных алгоритмов программного комплекса "Поток-3", опирающихся на решение систем алгебраических уравнений методом многомерной скалярной прогонки. Для вычислений же (32), не связанных с прогонкой, в принципе этот тип распараллеливания также может быть

применен, однако при этом требуется специальная организация операций, нарушающая однородность алгоритма (32) из-за появления фиктивных границ внутри расчетной области. Кроме того, возникает необходимость организации обмена данными между сегментами с высокой степенью синхронизации этого процесса. Заметим, что обмен данными собственно между сегментами при C-, L- и V-типах параллелизации не производится; осуществляется только пересылка данных между процессором-диспетчером и процессорами-исполнителями.

Операции (32) могут быть в принципе распараллелены по V-типу, поскольку (32) допускает запись в векторной форме

$$F_{ij}^{n+\nu} = AF^n, \tag{39}$$

где A — дифференциальный матричной оператор.

Однако матрица A в (39) весьма заполнена ненулевыми элементами, причем таким образом, что вычисления одной из компонент вектора F, отвечающая закону сохранения энергии, требует значительно бóльших затрат, чем вычисление других компонент, выражающих законы сохранения массы и координатных проекций импульсов, занимая около 85 % от объема всех вычислений. Поэтому ускорение операций при V-типе распараллеливания не даст выигрыша во времени более чем в 15 %, а с учетом потерь на коммуникации и того меньше.

Таким образом, C-, L- и V-типы параллелизации, разработанные и примененные для локального ускорения операций в подпрограммах главного вычислительного ядра программного комплекса "Поток-3", не являются эффективными для распараллеливания некоторых (в данном комплексе — одной) процедур. Возможно, здесь окажется более эффективным применение методов параллелизации для многопроцессорных комплексов с общей памятью, однако вопрос о совместном использовании систем параллельного программирования MPI и OpenMP является существенно открытым.

Завершая данную работу, следует отметить, что, опираясь на изложенные в [9] принципы, можно провести еще более глубокое распараллеливание программного комплекса "Поток-3", в частности, организовать параллелизацию выполнения арифметических операций внутри циклов каждого из представленных C-, L- и V-типов распараллеливания.

В заключение авторы считают своим приятным долгом выразить благодарность В. Э. Малышкину, Н. В. Кучину, В. А. Вшивкову и Г. С. Хакимзянову за полезные обсуждения и внимание к работе.

Работа выполнена при поддержке РФФИ (проекты № 00–07–90297, 01–01–00781 и 02–01–00097).

СПИСОК ЛИТЕРАТУРЫ

1. *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. 45–57.
2. *Тарнавский Г.А., Вшивков В.А., Тарнавский А.Г.* Параллелизация алгоритмов и кодов вычислительной системы "Поток-3" // Программирование. 2003. № 1. 1–20.
3. *Тарнавский Г.А., Шпак С.И.* Схемы распараллеливания операций решения систем алгебраических уравнений методом многомерной скалярной прогонки // Вычислительные методы и программирование. 2000. **1**. 21–29. (http://www.srcc.msu.su/num-meth)
4. *Вшивков В.А., Тарнавский Г.А., Неупокоев Е.В.* Параллелизация алгоритмов прогонки: многоцелевые вычислительные эксперименты // Автометрия. 2002. № 4. 74–86.
5. *Тарнавский Г.А., Тарнавский А.Г.* Современные компьютерные технологии и неединственность решений задач газовой динамики // Симметрия и дифференциальные уравнения. Красноярск: Изд-во ИВТ СО РАН, 2002. 209–213.
6. *Любимов А.Н., Русанов В.В.* Течения газа около тупых тел. М.: Наука, 1970.
7. *Ковеня В.М., Тарнавский Г.А., Черный С.Г.* Применение метода расщепления в задачах аэродинамики. Новосибирск: Наука, 1990.
8. *Тарнавский Г.А., Шпак С.И.* Проблемы численного моделирования сверхзвукового ламинарно-турбулентного обтекания тел конечного размера // Математическое моделирование. 1998. **10**, № 6. 53–74.
9. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. СПб: БХВ-Петербург, 2002.

Поступила в редакцию
23.12.2002