УДК 519.6

РАСПАРАЛЛЕЛИВАНИЕ МЕТОДИКИ "Д" ДЛЯ РЕШЕНИЯ ДВУМЕРНЫХ ЗАДАЧ ГАЗОВОЙ ДИНАМИКИ С ДИНАМИЧЕСКОЙ БАЛАНСИРОВКОЙ АРИФМЕТИЧЕСКОЙ НАГРУЗКИ ПРОЦЕССОРОВ

И. М. Епишков, П. В. Егоров
(ФГУП "РФЯЦ-ВНИИЭФ", г. Саров Нижегородской области)

Описываются алгоритмы многофрагментной блочно-регулярной декомпозиции, а также основные принципы динамической балансировки арифметической нагрузки процессоров при проведении расчетов в многопроцессорном режиме по лагранжевой методике Д. Представлены результаты тестовых расчетов, демонстрирующие применимость реализованных алгоритмов в методике Д.

Ключевые слова: методика Д, многофрагментная блочно-регулярная декомпозиция, динамическая балансировка арифметической нагрузки процессоров.

Введение

Методика Д [1] предназначена для решения двумерных нестационарных задач газовой динамики в переменных Лагранжа на структурированных сетках. Как известно, для повышения точности расчетов и более аккуратного численного моделирования сложных газодинамических течений необходимо использовать счетные сетки с большим количеством ячеек, что влечет за собой значительное увеличение календарных сроков расчетов. Один из путей сокращения этих сроков — проведение расчетов на многопроцессорных вычислительных комплексах.

В 2009 г. в методике Д была реализована возможность проводить расчеты в многопроцессорном режиме с использованием модели распределенной памяти MPI [2]. Эффективность многопроцессорного счета в большинстве случаев не превышала 50 %, и требовались новые алгоритмы и подходы, которые смогли бы ее повысить.

Первая причина столь низкой эффективности была связана с особенностью способа декомпозиции задачи по процессорам. В методике Д использовался наиболее распространенный способ отображения задач газовой динамики на вычислительное поле многопроцессорной системы — геометрическая декомпозиция. Данный метод предполагает разбиение сетки, покрывающей математические области, на множество *параобластей*, каждая из которых рассчитывается на отдельном процессоре. Если распределение счетных узлов сетки между областями равномерное, алгоритмы регулярной (матричной) декомпозиции достаточно часто позволяют получать приемлемую эффективность многопроцессорного счета. Однако в расчетах по методике Д используется многообластной счет, в математической постановке число областей может составлять до нескольких десятков, причем дисбаланс по числу счетных узлов сетки между областями может быть очень существенным. Поскольку на каждом процессоре рассчитывался лишь один фрагмент задачи, то с использованием стандартного алгоритма регулярной декомпозиции оптимальная сбалансированность арифметической нагрузки, а следовательно, высокая эффективность многопроцессорного счета были труднодостижимы.

Вторая причина низкой эффективности связана с неравномерностью вычислений, приходящихся на разные счетные ячейки и узлы, как в пределах одной области, так и между различными областями в целом. При этом в процессе счета количество вычислений может меняться достаточно существенно, что вызывает дисбаланс арифметической нагрузки процессоров. Как следствие, эффективность многопроцессор-

ного счета снижается, а общее время счета задачи увеличивается. В связи с этим в процессе счета возникает необходимость выполнить новую декомпозицию с учетом равномерного распределения арифметической нагрузки между процессорами и продолжить расчет далее. Если такая процедура происходит без остановки счета задачи, она называется *динамической балансировкой*. В методике Д данная возможность ранее отсутствовала.

Отметим, что многофрагментная блочно-регулярная декомпозиция не является новой и в том или ином виде встречается в ряде работ [3—5]. В данной статье обосновывается выбор именно этого подхода к организации декомпозиции задачи по процессорам при проведении расчетов по методике Д. Приводятся особенности реализации алгоритма декомпозиции, а также основные принципы алгоритма динамической балансировки арифметической загрузки процессоров. На тестовых расчетах исследуется время счета и эффективность с использованием новых алгоритмов и без них.

1. Многофрагментная блочно-регулярная декомпозиция

Многофрагментная блочно-регулярная декомпозиция является важной составляющей алгоритма динамической балансировки. Отметим, что в методике Д декомпозиция по процессорам осуществляется в автоматическом режиме. В дальнейшем будем называть декомпозиционным блоком фрагмент геометрии, рассчитываемый на одном процессоре.

Основной причиной, по которой предпочтение отдается именно многофрагментной блочно-регулярной декомпозиции, является возможность получения более сбалансированной арифметической нагрузки при использовании произвольного числа процессоров и математических областей по сравнению с обычной регулярной декомпозицией, которую иллюстрирует рис. 1 (цифры на рисунке — номера процессоров, рассчитывающих область).

Для регулярной схемы разбиения сетки по процессорам характерны жесткие зависимости между параллельными рядами декомпозиционных блоков. В случае, когда количество вычислений для каждого счетного узла не зависит от его положения в области, равномерное разбиение по обоим направлениям приведет к хорошо сбалансированной декомпозиции. Однако гораздо

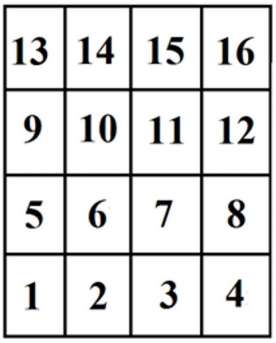

Рис. 1. Регулярная декомпозиция области

чаще количество вычислений, приходящихся на разные счетные узлы, существенно различается. Источниками такого различия могут быть особенности физической постановки, разные процессы, происходящие в разных областях задачи, используемые уравнения состояния и пр.

Поскольку распределение арифметической нагрузки по счетным узлам области является произвольным, достичь сбалансированной декомпозиции методом регулярного разбиения не всегда удается. На рис. 2 представлена область, в одной части которой (закрашенной) арифметическая нагрузка существенно больше, чем в другой (незакрашенной).

Достичь сбалансированной декомпозиции для представленной на рис. 2 области можно, используя многофрагментную блочно-регулярную декомпозицию (рис. 3), которая позволяет добиться сгущения декомпозиционных блоков в первой части и разрежения — во второй.

Другим существенным преимуществом многофрагментной блочно-регулярной декомпозиции (недостижимым при регулярной декомпозиции)

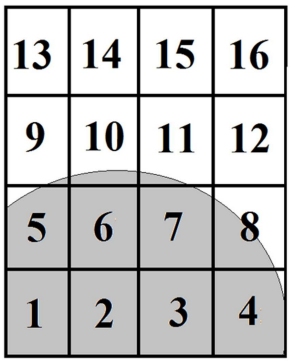

Рис. 2. Регулярная декомпозиция для случая неоднородной арифметической нагрузки

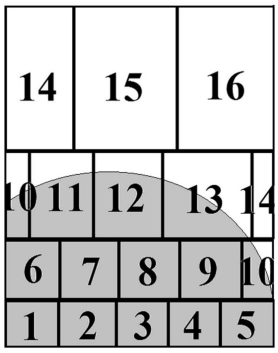

Рис. 3. Многофрагментная блочно-регулярная декомпозиция для случая неоднородной арифметической нагрузки

является возможность выполнять распределение произвольного количества областей на произвольное число процессоров, получая при этом хорошую сбалансированность арифметической нагрузки. Эта особенность обеспечивается тем, что в данном случае каждый процессор может рассчитывать несколько декомпозиционных блоков, принадлежащих различным областям. В примере, приведенном на рис. 3, таким свойством обладают процессоры с номерами 10 и 14.

Многофрагментная блочно-регулярная декомпозиция получается в результате разбиения области на блоки в двух направлениях: главном (число узлов сетки в этом направлении больше) и вспомогательном (число узлов сетки в этом направлении меньше). Первое разбиение выполняется для главного направления, охватывая всю область (как для регулярной декомпозиции). При этом формируются декомпозиционные слои (рис. 4). Второе разбиение выполняется для вспомогательного направления и только в рамках одного декомпозиционного слоя (в отличие от регулярной декомпозиции, когда разбиение по второму направлению выполнялось также с охватом всей области).

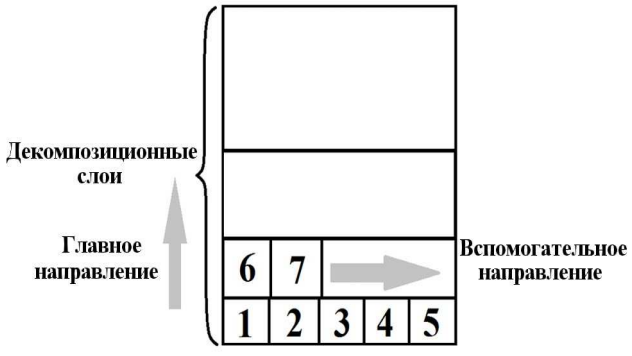

Рис. 4. Формирование многофрагментной блочно-регулярной декомпозиции

Рассмотрим более подробно алгоритм многофрагментной блочно-регулярной декомпозиции. Данный алгоритм выполняется при условии известного значения T_{ij}^k для каждого счетного узла сетки $(i,\ j)$ каждой математической области k (в начальный момент распределение нагрузки принимается равномерным). Это позволяет определить суммарную арифметическую нагрузку задачи:

$$T_\Sigma = \sum_{k=1}^{N} \sum_{i,j} T_{ij}^k,$$

а также арифметическую нагрузку, приходящуюся на один процессор:

$$T_{\text{пр}} = \frac{T_\Sigma}{N_{\text{пр}}}.$$

Здесь N — общее число областей; $N_{\text{пр}}$ — количество процессоров.

Исходя из цели равномерного распределения нагрузки по процессорам счетные узлы каждой математической области k должны быть распределены на N^k процессоров:

$$N^k = \text{int}^* \left(\frac{\sum\limits_{i,j} T_{ij}^k}{T_\Sigma} N_{\text{пр}} \right).$$

В соответствии с полученным числом процессоров для каждой математической области определяется число декомпозиционных слоев $N_{\text{сл}}^k$. Для этого можно воспользоваться разными подходами. Авторами было принято решение вычислять число декомпозиционных слоев, исходя из равномерного числа строк и столбцов сетки в каждом декомпозиционном блоке. Это позволяет избежать сильных геометрических перекосов и, как следствие, сократить возможное число соседствующих процессоров.

Таким образом, должно выполняться следующее условие:

$$\frac{I^k}{N_{\text{сл}}^k} = \frac{J^k}{\left(\dfrac{N_{\text{сл}}^k}{N^k} \right)},$$

*int — функция получения целой части числа.

где I^k, J^k — количество узлов сетки соответственно в главном и вспомогательном направлениях k-й математической области. Отсюда получаем число декомпозиционных слоев в области:

$$N^k_{\text{сл}} = \min\left\{ \text{int}\left(\sqrt{N^k \frac{I^k}{J^k}}\right), N^k \right\}.$$

Толщина декомпозиционного слоя выбирается так, чтобы распределение арифметической нагрузки по декомпозиционным слоям было равномерным. Декомпозиционные блоки составляются для процессоров последовательно из текущего декомпозиционного слоя текущей математической области. Переход к следующему процессору выполняется после того, как для текущего процессора набирается арифметическая нагрузка, равная $T_{\text{пр}}$. Если декомпозиционный блок, который набирается для рассматриваемого процессора, не "помещается" в текущем декомпозиционном слое (см. процессор 10 на рис. 3), то создается новый блок, который переносится на следующий декомпозиционный слой и добирается уже за его счет (или за счет следующей математической области). В этом случае процессор будет рассчитывать несколько декомпозиционных блоков. После того как набирается $T_{\text{пр}}$ для текущего процессора, происходит перерасчет оставшейся арифметической нагрузки, поскольку, как правило, существует некоторая погрешность при выполнении декомпозиции.

В целом многофрагментную блочно-регулярную декомпозицию можно представить в виде декомпозиционной схемы, имеющей иерархическую структуру (рис. 5), конечным звеном в которой является декомпозиционный блок.

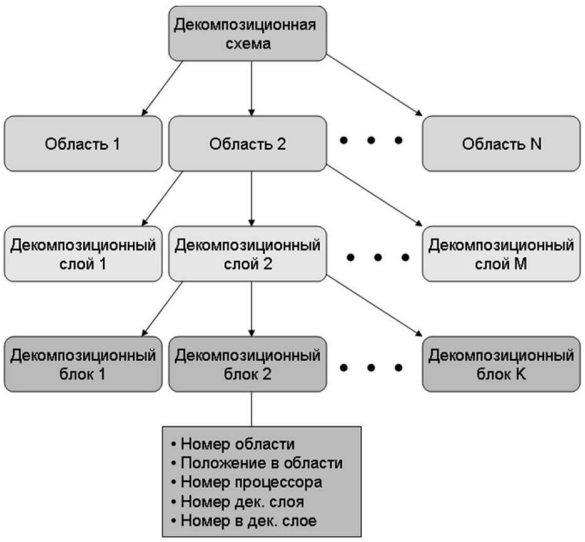

Рис. 5. Структура декомпозиционной схемы

2. Межпроцессорные обмены

В методике Д используются явные конечно-разностные схемы. Для расчета газодинамических величин в счетной ячейке требуется информация из соседних с ней ячеек сетки. Необходимость получения дополнительных данных от соседних декомпозиционных блоков обусловлена спецификой алгоритмов решения уравнения движения и алгоритмов сглаживания скоростей [1]. В этих алгоритмах для вычисления значений параметров в граничных узлах сетки рассматриваемого декомпозиционного блока используется информация об узлах и ячейках сетки, которые выходят за пределы данного блока. Поэтому декомпозиционные блоки расширяются на несколько *фиктивных* слоев, данные в которых необходимо обновлять с соседних процессоров на каждом временном шаге. На данный момент в методике Д используется два фиктивных слоя.

Декомпозиционные блоки связаны между собой таблицей связи *ExchData*, которая имеет списки для отправки (*ExchDataOut*) и списки для приема (*ExchDataIn*) данных. Все обмены между процессорами соседних декомпозиционных блоков происходят во время расчета шага в соответствии с этими списками. Для межпроцессорных обменов используются асинхронные буферизированные MPI-операции [2] отправки и приема данных. Оперативная память для данных буферов выделяется перед расчетом временных шагов. Среднее число ячеек (узлов) сетки, пересылаемых с одного процессора на другой, в стандартных расчетах по методике Д обычно не превышает нескольких тысяч.

На рис. 6 приведен пример, демонстрирующий, каким образом составляется таблица межпроцессорной связи *ExchDataIn*. Математическая область размечена на декомпозиционные блоки. Серым цветом отмечен декомпозиционный блок 5-го процессора, для которого составляется таблица приема. Можно увидеть, что этот блок имеет связи (отмечены черным цветом) с соседними блоками: с 1-м процессором посредством 1-го блока, со 2-м процессором посредством 2-го блока, с 6-м процессором посредством 3-го блока и т. д.

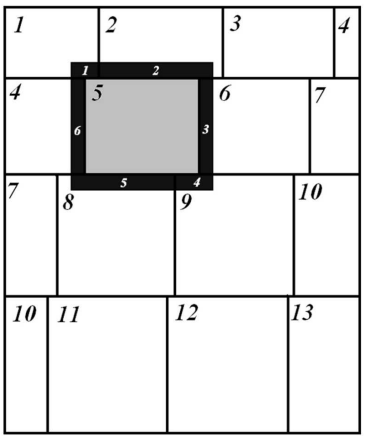

Рис. 6. Формирование таблиц межпроцессорных обменов

3. Алгоритм динамической балансировки

Отметим, что основным условием для выполнения алгоритма динамической балансировки является возможность размещения в оперативной памяти каждого счетного процессора данных, которые ему необходимо рассчитывать, в двойном размере. Причины этого будут рассмотрены ниже.

Как было сказано в разд. 1, условием выполнения многофрагментной блочно-регулярной декомпозиции является наличие информации об арифметической нагрузке T_{ij}^k при расчете временного шага для каждого счетного узла сетки (i, j) каждой математической области k. Однако на практике определение времени счета отдельно для каждого узла сетки является задачей труднореализуемой. Поэтому в качестве T_{ij}^k используется удельная арифметическая нагрузка процессора, на котором рассчитывается счетный узел сетки (i, j):

$$T_{ij}^k = \frac{T_{\text{сч}}^m}{\sum\limits_{r=1}^{N_{\text{бл}}} I_r^m J_r^m}, \qquad (1)$$

где $T_{\text{сч}}^m$ — *полезное* время счета (без учета времени межпроцессорных коммуникаций) временного шага на процессоре m, к которому относится рассматриваемый узел; $N_{\text{бл}}$ — количество декомпозиционных блоков на процессоре m; I_r^m и J_r^m — число строк и столбцов сетки декомпозиционного блока r на процессоре m.

Рассмотрим данный алгоритм более подробно:

1. В качестве начального приближения принимаем равномерное распределение арифметической нагрузки ($T_{ij}^k = 1$). В соответствии с этим строим многофрагментную блочно-регулярную декомпозицию. Данная декомпозиция будет геометрически равномерной, т. е. на каждом процессоре будет рассчитываться приблизительно одинаковое количество узлов сетки.
2. Рассчитываем несколько временных шагов, замеряя полезное время счета $T_{\text{сч}}^m$ для каждого процессора m.
3. Определяем дисбаланс проведения многопроцессорного счета по формуле

$$\delta = \frac{\sum\limits_{m=1}^{N_{\text{пр}}} |T_{\text{сч}}^{\max} - T_{\text{сч}}^m|}{N_{\text{пр}} T_{\text{сч}}^{\max}}, \qquad (2)$$

   где $T_{\text{сч}}^{\max}$ — наибольшее время счета на процессоре.
4. Если дисбаланс удовлетворяет заданному критерию (обычно задается условие $\delta < 0{,}15$), то переходим к п. 2, если нет — к п. 5.
5. По формуле (1) определяем нагрузку процессора при расчете одного счетного узла и устанавливаем распределение по ячейкам T_{ij}^k.
6. Получив следующее приближение распределения арифметической нагрузки, вновь выполняем многофрагментную блочно-нерегулярную декомпозицию согласно правилам, изложенным ранее.
7. Переходим к п. 2.

Как видно из описания алгоритма, итерационный процесс коррекции декомпозиции выполняется в ходе расчета газодинамических шагов, а процедура улучшения декомпозиции происходит только в случае низкой эффективности счета.

Остановимся на некоторых технических моментах представленного алгоритма.

Передекомпозиция. Как сказано выше, в алгоритме многофрагментной блочно-регулярной декомпозиции вместо вычислительной нагрузки процессора, приходящейся на каждый рассчитываемый на нем счетный узел, используется средняя вычислительная нагрузка этого процессора. Поэтому целесообразнее использовать не распределение арифметической нагрузки по ячейкам

сетки в явном виде, а декомпозиционную схему, из которой можно получить информацию о средней нагрузке процессора, приходящейся на один счетный узел при счете временного шага, для каждого декомпозиционного блока.

В соответствии с этой идей можно ввести две декомпозиционные схемы: первая будет соответствовать текущей (новой) декомпозиции, а вторая — предыдущей (старой). Процедура расчета новой декомпозиции опирается на использование старой декомпозиции. Старая декомпозиционная схема содержит информацию обо всех декомпозиционных блоках, которые были ранее, о том, какие процессоры их рассчитывали, а также о вычислительной нагрузке каждого процессора, приходящейся на один узел. Используя эти данные как информацию о распределении арифметической нагрузки, можно по правилам, изложенным в разд. 1, составить новую декомпозиционную схему. Данная схема вносит некоторые изменения в распределение декомпозиционных блоков по процессорам, поэтому в результате передекомпозиции, как правило, каждый процессор начинает "отвечать" за новый участок расчетной сетки.

Процедура обменов данными при передекомпозиции выполняется в соответствии с таблицей связи *DecompData*, которая для старой декомпозиционной схемы определяет последовательность отправки данных (*DecompDataOut*), а для новой декомпозиционной схемы — последовательность приема данных (*DecompDataIn*). Каждая из этих таблиц содержит набор связей текущего процессора с остальными процессорами. Каждая такая связь представляет собой упорядоченный список сеточных блоков, которые образуются в результате пересечения декомпозиционных блоков старой и новой декомпозиционных схем. Процедура упорядочения сеточных блоков необходима для того, чтобы последовательности отправки и приема данных были одинаковыми. Таким образом, чтобы отправить данные для новой декомпозиции на конкретный процессор, необходимо получить соответствующую запись из таблицы *DecompDataOut* и для всех сеточных блоков из списка в этой записи выполнить отправку сеточных данных для каждой ячейки и каждого узла. Аналогично, чтобы принять данные, соответствующие старой декомпозиции, необходимо воспользоваться таблицей *DecompDataIn*.

Таким образом, происходит глобальный обмен данными между процессорами с помощью буферизированных асинхронных MPI-операций. Каждый процессор выделяет память под сеточные массивы для новых декомпозиционных блоков и принимает от соответствующих процессоров все необходимые газодинамические данные. При этом объем пересылаемой информации на другие процессоры может варьироваться в зависимости от области пересечения старой и новой декомпозиций.

Отметим, что в двумерном случае можно не беспокоиться о недостатке оперативной памяти на процессоре, так как при декомпозиции двумерных задач среднее число узлов сетки, приходящихся на один процессор, в методике Д не превышает 30 000. Таким образом, можно утверждать, что в оперативной памяти одного процессора могут одновременно поместиться как старые, так и новые данные, которые будут приниматься от других процессоров. После проведения обменов массивы со старыми данными удаляются из оперативной памяти.

4. Задача о сжатии газа сферической оболочкой в двухобластной постановке

Данная задача служит тестом на определение эффективности расчета для многообластного случая, когда геометрия задачи содержит контактные границы, т. е. имеет место разбалансирующий фактор, которым является расчет контактных узлов в многопроцессорном режиме.

В задаче об одномерном сжатии сферической газовой полости тяжелой сферической оболочкой радиус газовой сферы R_1 намного больше толщины тяжелой сферической оболочки, наружный радиус которой равен R_2: $0 < R_2 - R_1 << R_1$, а именно $R_1 = 5{,}0$; $R_2 = 5{,}5$.

В начальный момент $t_{\text{нач}} = 0$ оба слоя (газ и оболочка с начальными постоянными по толщине плотностями 1 и 20 г/см^3 соответственно) покоятся и имеют нулевое давление и нулевую внутреннюю энергию. На наружной лагранжевой границе задается постоянное давление $p(t) = p_{\text{гран}} = 1{,}0$ ГПа, $t \geq 0$. Используется уравнение состояния вида $P = (\gamma_{k-1})\rho E$, $\gamma_1 = \gamma_2 = 1{,}31$.

Газовая полость и тяжелая оболочка задавались отдельными математическими областями.

Проводилось две серии расчетов до момента времени 100 мкс: с числом счетных узлов

сетки 800 тыс. и 6 млн. Варьировалось число процессоров. Первая серия расчетов выполнялась с использованием регулярной декомпозиции (старый вариант) и многофрагментной блочно-регулярной декомпозиции с динамической балансировкой и без нее. Во второй серии расчетов использовался только новый алгоритм декомпозиции с динамической балансировкой и без нее. Дополнительно сравнивались среднее время выполнения программы динамической балансировки и среднее время выполнения одного счетного шага.

Для всех расчетов оценивались ускорение S_N и эффективность многопроцессорного счета E_N в зависимости от количества используемых процессоров, рассчитанные по формулам

$$S_N = \frac{t_1}{t_N}; \quad E_N = \frac{t_1}{N t_N} \cdot 100\,\%,$$

где t_1 — время счета на одном процессоре; t_N — время счета на N процессорах.

На рис. 7 показаны графики ускорения и эффективности многопроцессорного счета для первой серии расчетов (800 тыс. узлов сетки) в зависимости от количества процессоров. Видно, что новый алгоритм многофрагментной блочно-регулярной декомпозиции с использованием динамической балансировки с увеличением числа процессоров позволяет существенно повысить как эффективность счета (до 3 раз), так и ускорение по сравнению с регулярной декомпозицией. Также новые алгоритмы позволяют использовать в расчетах большее число процессоров, сохраняя при этом приемлемую эффективность многопроцессорного счета.

Для варианта с использованием многофрагментной блочно-регулярной декомпозиции с динамической балансировкой в табл. 1 приведено сравнение времен выполнения счетного шага и программы балансировки вычислительной нагрузки. При этом N обозначает количество используемых процессоров; $t_{\text{ш}}$ — среднее время выполнения счетного шага, $t_{\text{бал}}$ — среднее время выполнения балансировки (сбор общей информации, вычисление новой декомпозиции и обмен данными). Из таблицы видно, что при использовании более 48 процессоров время балансировки не превышает времени выполнения одного расчетного шага. При использовании числа процессоров менее 48 время балансировки может превышать время одного счетного шага до 1,5 раз. Однако несмотря на это, время, затраченное на

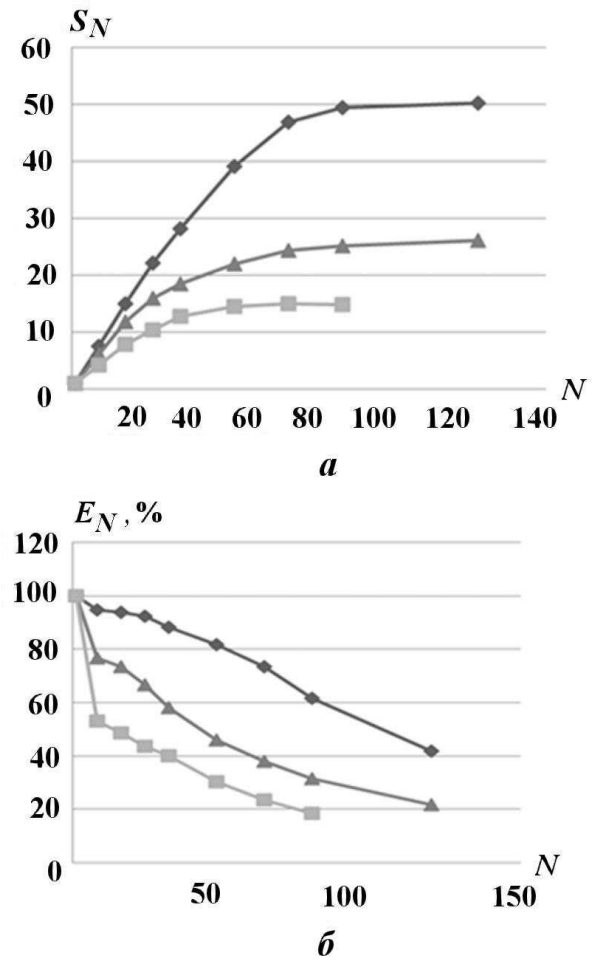

Рис. 7. Расчеты с 800 тыс. ячеек. Ускорение *(а)* и эффективность *(б)* счета: —♦— — новый алгоритм с балансировкой; —▲— — новый алгоритм без балансировки; —■— — старый алгоритм

Таблица 1

Среднее время выполнения счетного шага и динамической балансировки в первой серии расчетов

N	Среднее число узлов сетки на 1 процессоре	$t_{\text{ш}}$, с	$t_{\text{бал}}$, с
1	800 000	0,881	—
8	100 000	0,116	0,15
16	50 000	0,058	0,08
24	33 333	0,039	0,06
32	25 000	0,031	0,04
48	16 666	0,022	0,02
64	12 500	0,019	0,01
80	10 000	0,018	0,01
120	6 666	0,017	0,01

выполнение алгоритма балансировки, компенсируется тем ускорением, которое получается в результате перераспределения арифметической нагрузки между процессорами (см. рис. 7). Отметим, что алгоритмы динамической балансировки в данном тесте использовались не чаще, чем раз в 100 счетных шагов.

На рис. 8 показаны графики ускорения и эффективности многопроцессорного счета для расчетов на сетке с 6 млн узлов. Видно, что динамическая балансировка позволяет получить ускорение счета более чем в 2 раза, а также использовать большее число процессоров, сохраняя при этом приемлемую эффективность многопроцессорного счета.

В табл. 2 для этой серии расчетов приведено сравнение времени выполнения счетного шага и времени балансировки вычислительной нагрузки. Из таблицы следует, что при использовании оптимального числа процессоров в расчете (256 при ускорении в 132 раза и эффективности 51 %) время выполнения программы балансировки не превышает времени выполнения одного счетного шага. При этом видно, что с уменьшением числа узлов сетки, приходящихся на один процессор, время выполнения процедуры балансировки также сокращается, так как сокращается объем данных, которыми нужно обмениваться с другими процессорами.

Отметим, что время вычисления новой декомпозиции во всех расчетах не превышает 1 % от общего времени выполнения программы балансировки. Основные временные затраты приходятся на копирование данных в/из обменного буфера и MPI-операции.

В проведенных тестовых расчетах по формуле (2) определялся дисбаланс многопроцессорного счета (вычислительной нагрузки между процессорами) как с динамической балансировкой, так и без нее. При отключении динамической балансировки дисбаланс возрастал с течением времени счета до 0,4—0,8 (в зависимости от расчета). При использовании динамической балансировки значение дисбаланса не превышало 0,15, так как, если это происходило, проводилась балансировка вычислительной нагрузки с применением алгоритмов, описанных в разд. 3, и формировалась новая многофрагментная блочно-регулярная декомпозиция.

На рис. 9 (см. также цветную вкладку) представлен пример декомпозиции расчетной сетки по процессорам и распределение поля давлений

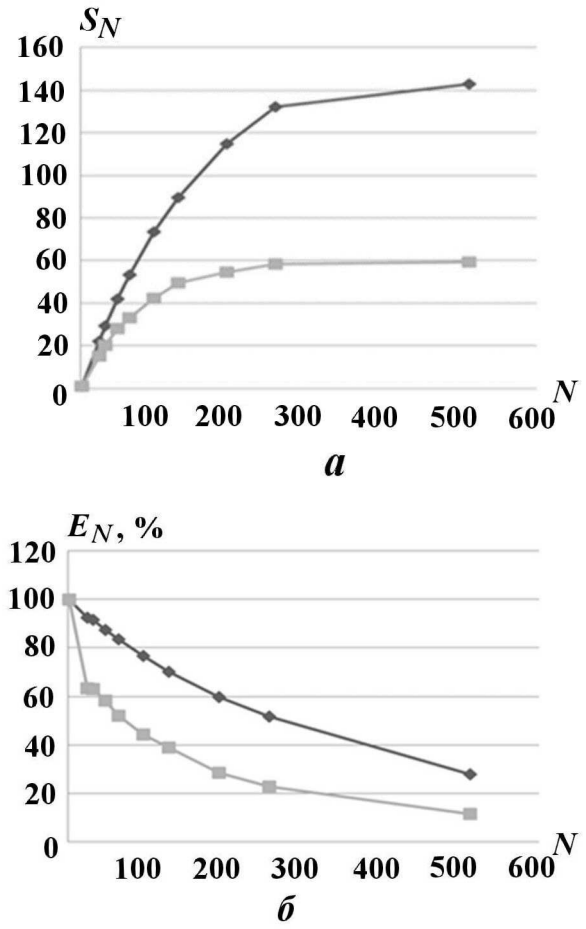

Рис. 8. Расчеты с 600 млн ячеек. Ускорение *(а)* и эффективность счета *(б)*: —♦— — новый алгоритм с балансировкой; —■— — новый алгоритм без балансировки

Таблица 2

Среднее время выполнения счетного шага и динамической балансировки во второй серии расчетов

N	Число узлов сетки на 1 процессоре	$t_{ш}$, с	$t_{бал}$, с
1	6 000 000	2,81	—
24	250 000	0,126	0,37
32	187 500	0,096	0,31
48	125 000	0,067	0,17
64	93 750	0,052	0,14
96	62 500	0,038	0,09
128	46 875	0,031	0,06
192	31 250	0,028	0,04
256	23 437	0,027	0,02
512	11 718	0,025	0,02

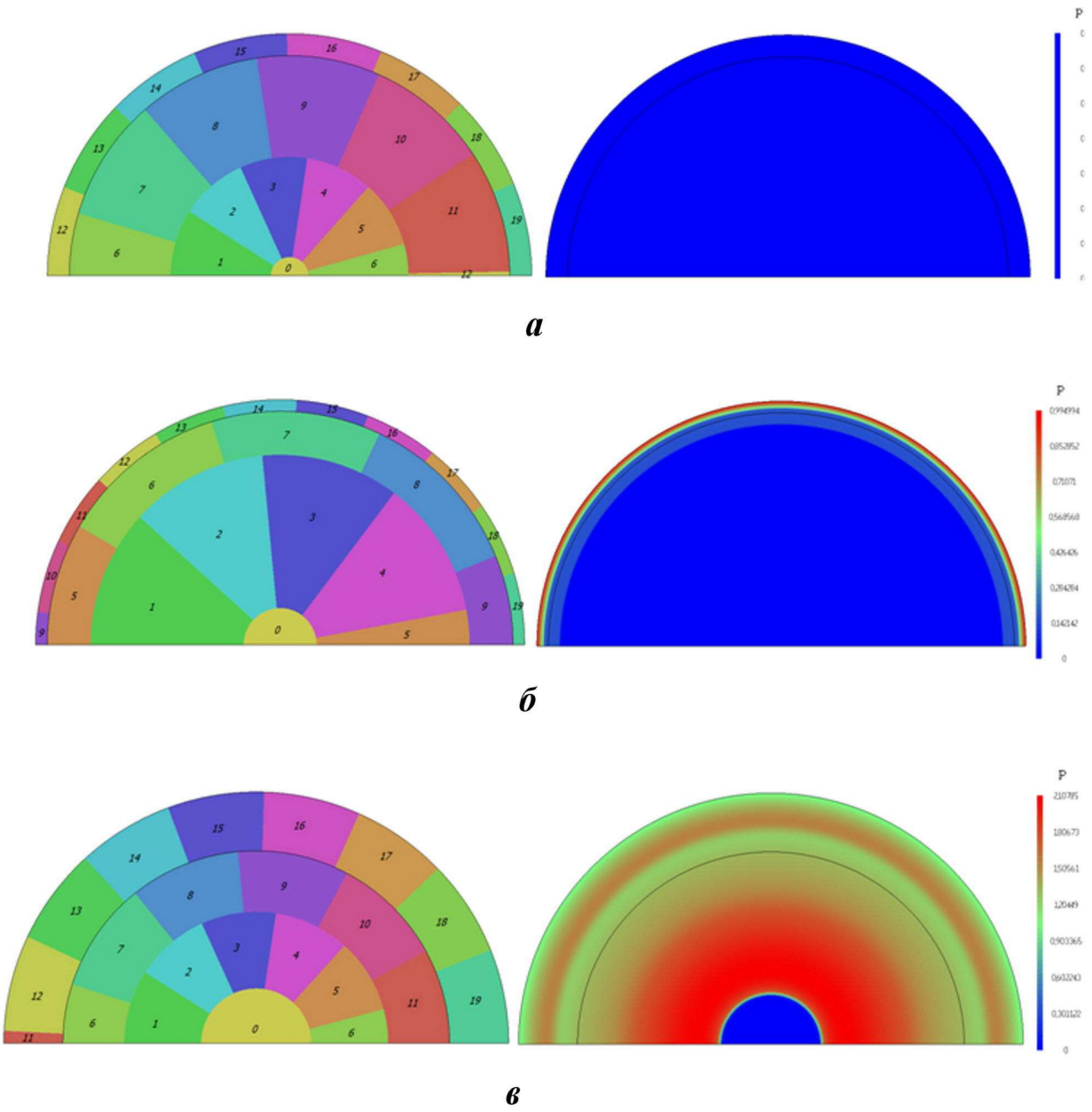

Рис. 9. Декомпозиция по процессорам и распределение поля давлений: *а* — $t = 0$; *б* — $t = 50$ мкс; *в* — $t = 100$ мкс

на моменты времени $t = 0; 50; 100$ мкс (использовалось 20 процессоров). Видно, как по мере продвижения фронта волны происходит перераспределение декомпозиционных блоков в сторону их сгущения в возмущенных областях.

Заключение

В методике Д реализованы алгоритмы динамической балансировки арифметической нагрузки между процессорами, а также алгоритмы многофрагментной блочно-регулярной декомпозиции. Все алгоритмы выполняются в автоматическом режиме. На тестовых расчетах показано, что использование многофрагментной блочно-регулярной декомпозиции позволяет более сбалансированно распределять данные по процессорам по сравнению с обычной регулярной декомпозицией, а в результате динамической балансировки удается не только получить ускоре-

ние счета, но и повысить эффективность благодаря автоматической передекомпозиции данных в процессе проведения расчета.

Список литературы

1. *Софронов И. Д., Делов В. И., Дмитриева Л. В. и др.* Методика Д для расчета многомерных задач механики сплошной среды в переменных Лагранжа на регулярной сетке // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 1999. Вып. 4. С. 42—50.
2. MPI Documents. http://www.mpi-forum.org/docs/docs.html.
3. *Pmar A., Aykanat C.* Sparse matrix decomposition with optimal load balancing // Proc. of 4th Int. Conf. on High-Performance Computing. IEEE Xplore Digital Library. 1997. P. 224—229.
4. *Ujaldon M., Sharma Sh., Zapata E., Saltz J.* Experimental evaluation of efficient sparse matrix distributions // Proc. of 10th Int. Conf. on Supercomputing (ICS'96). 1996. New York: ACM, 1996. P. 78—85.
5. *Пронин В. А.* Методы распараллеливания двумерных задач газодинамики на неструктурированных сетках с переменной топологией в методике МЕДУЗА // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 2011. Вып. 1. С. 54—67.

Статья поступила в редакцию 26.06.13.
Исправленный вариант — 02.04.14.

PARALLELING OF THE "D" CODE FOR 2D GAS DYNAMICS SIMULATIONS WITH DYNAMIC BALANCING OF ARITHMETIC PROCESSOR LOAD / I. M. Epishkov, P. V. Egorov (FSUE RFNC-VNIIEF, Sarov, Nizhny Novgorod region).

The paper describes multi-fragment regular-block decomposition algorithms and the basic principles of dynamic balancing of processor arithmetic load in multiprocessor simulations using the Lagrangian code D and presents the results of test simulations demonstrating the applicability of the algorithms in the code.

Keywords: D code, multi-fragment regular-block decomposition, dynamic balancing of processor arithmetic load.