ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ ИЗМЕНЕНИЯ СТРУКТУРЫ РАСЧЕТНОЙ СЕТКИ В ЗАДАЧАХ НА НЕСТРУКТУРИРОВАННЫХ СЕТКАХ*

А.Н. Андрианов

В работе рассматриваются проблемы, которые возникают при написании параллельных программ для многопроцессорных систем с распределенной памятью, для численных методов, использующих динамически изменяемые неструктурные сетки. Представлены вопросы, возникающие при распараллеливании операций изменения структуры расчетной сетки. Приводимые результаты опираются на опыт построения параллельных программ решения нестационарных задач двумерной газовой динамики на основе использования свободно-лагранжевого метода на треугольных сетках переменной структуры.

1. ВВЕДЕНИЕ

Тут текст Метод НЕСЛА [1] (неявный свободно-лагранжев) - это метод решения двумерных нестационарных магнитогазодинамических задач, основанный на использовании треугольной лагранжевой сетки переменной структуры и использующий неявные полностью консервативные операторно-разностные схемы МГД. Необходимость изменения структуры расчетной сетки определяется тем обстоятельством, что лагранжева сетка, связанная с движущейся средой, может искажаться в процессе расчета.

В данной работе мы рассматриваем проблемы, которые приходится решать при построении параллельной версии программы, реализующей указанный метод расчета. Исходным материалом для работы послужили тексты последовательных программ расчета, предоставленные авторами метода [1]. При выполнении данной работы, в основном, приходилось опираться на те структуры данных, которые заложили авторы последовательной версии программы. Такой подход позволял надеяться на минимальные изменения текстов исходных программ и основное внимание сосредоточить на разработке программ, поддерживающих параллельное выполнение расчета.

2. БАЗОВЫЕ СТРУКТУРЫ ДАННЫХ

Основное понятие, используемое в численном методе с использованием неструктурных сеток и, соответственно, в расчетной программе – сетка, состоящая из узлов и ячеек.

Сетка может рассматриваться как граф, вершинами которого являются узлы, связи между узлами представляются дугами. Узлы, между которыми есть связь, называются соседними. В дальнейшем будем называть граф, представляющий сетку, графом узлов. Ячейка является треугольником и задается списком узлов сетки, расположенных в ее вершинах. Ячейка является минимальным треугольником в том смысле, что она не может содержать внутри себя другую ячейку.

В узлах сетки вычисляются расчетные величины, значения которых могут зависеть от значений расчетных величин в соседних узлах, в ячейках вычисляются ячеечные функции, значения которых могут зависеть от значений в узлах-вершинах данной ячейки или соседних к ней вершин. Конечно, в программах могут существовать и другие структуры. Например, ребра графа, специальные структуры для представления граничных узлов и т.п.

Расчет узловой /ячеечной величины, в общем случае, можно рассматривать как вычисление некоторого соотношения, аргументами которого являются значения в соседних узлах/ячейках. В случае структурных сеток адресация к значениям из соседних узлов/ячеек задается *неявным* (заданием индексного выражения с некоторым смещение) образом. При расчете на неструктурной сетке адресация к значениям в соседних узлах/ячейках задается *явным* образом, и для такой адресации необходимо в программе иметь структуру, задающую для каждого узла список соседних к нему узлов.

Все узлы и ячейки нумеруются целыми числами. Конечно, нумерация узлов и ячеек отлична друг от друга. Порядок нумерации произвольный, обычно он определяется математическим методом построения сетки. Граф узлов представляется в виде двумерного массива, в котором строка с номером i содержит номера узлов, соседних к узлу с номером i. Для удобства программирования вводится массив, в котором строка с номером i содержит номера ячеек, в которые входит узел с номером i. В отдельном массиве содержится информация об узлах, составляющих ячейку сетки. Подробно представленные структуры описаны в [2].

Распараллеливание последовательной программы состоит в распределении узлов и ячеек по процессорам параллельной системы. И какой способ распределения мы бы не выбрали, в параллельной программе должны быть сохранены все структуры, содержащиеся в последовательной программе. Конечно, данные структуры сократятся в

*Работа выполнена при финансовой поддержке гранта Президента РФ для ведущих научных школ НШ-2139.2008.9

объеме, но принципы работы с этими структурами должны оставаться такими же, как и в случае последовательной программы.

Высокая эффективность вычислений на параллельной системе с распределенной архитектурой обычно достигается в случае:

1. равномерной загрузки процессоров вычислениями
2. минимизацией накладных расходов на обмены данными между процессорами.

3. ИЗМЕНЕНИЕ СТРУКТУРЫ СЕТКИ

Операции над расчетной сеткой, выполняются в процессе вычислений при возникновении определенных условий, определяемых численным методом [1], и изменяют структуру сетки.

Под перестройкой сетки понимается изменение ее структуры с сохранением свойства треугольности в любой части расчетной области. Введение новой сетки в части расчетной области в процессе расчета предполагает решение двух задач: изменение самой структуры сетки (перестройка структуры) и задание на новой сетке значений сеточных функций, определенных в ячейках (пересчет значений).

В соответствии с формальными критериями на каждом шаге по времени осуществляется многопроходная процедура перестройки сетки: в окрестности каждого узла проводится анализ качества сетки и при необходимости инициируется локальная перестройка.

3.1 ОПРЕДЕЛЕНИЕ ОПЕРАЦИЙ ИЗМЕНЕНИЯ СТРУКТУРЫ СЕТКИ

Локальная перестройка сетки сводится к проведению последовательности операций, которые называются элементарными. В методе НЕСЛА, приняты три элементарные операции:

1.*Разрыв связи между узлами в введением новой связи между их общими соседями*. Эту операцию иллюстрирует рис. 1, здесь вместо связи AC устанавливается новая связь BD.

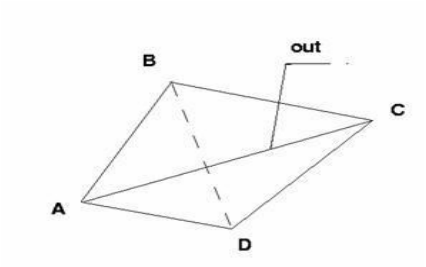

Рис.1

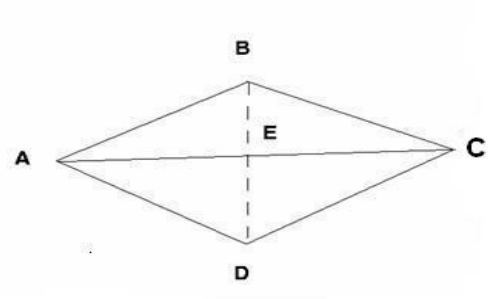

Рис.2

2.*Введение в сетку нового узла на связь между существующими узлами*. Эту операцию иллюстрирует рис. 2, здесь в сетку вводится новый узел E. Вместо связи AC вводятся четыре новые связи (AE, BE, CE, DE).

3.*Исключение узла из сетки с соответствующей коррекцией связей*. Эта операция показана на рис.3a и рис.3b.

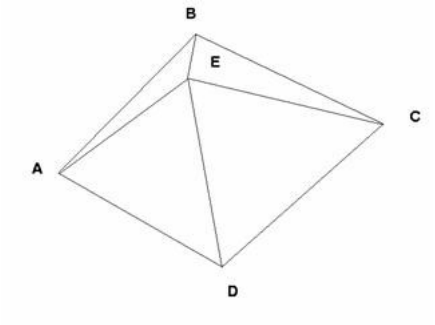

Рис. 3a

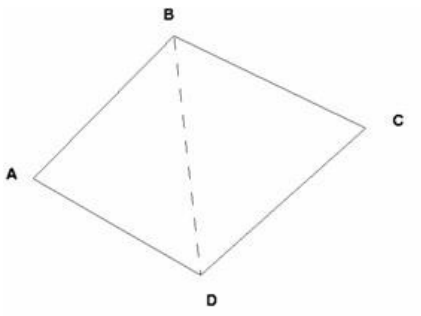

Рис. 3b

На рис.3a изображена часть сетки до применения операции исключения. В общем случае узел E может иметь другое количество соседей, не равное 4, как на рис. 3a. На рис.3b представлена та же часть сетки после при-

менения этой операции. Здесь из сетки исключается узел E (при этом исключаются связи AE, BE, CE, DE) и устанавливается новая связь BD.

Необходимо отметить, что выполнение операций над сеткой включает как определение применимости операции, так и перевычисление значений расчетных переменных в ячейках.

Исходными данными для проведения операций являются две ячейки с общей стороной. На рис.4 это ячейки с номерами 1 и 2. В эти ячейки входят узлы с номерами от 1 до 4. В дальнейшем будем называть "исходными первого порядка" ячейки с номерами от 1 до 14, т.е. ячейки, в которые входят узлы с номерами от 1 до 4.

В ячейках, в которые входят эти узлы (а это ячейки с номерами от 1 до 14) при проведении операции изменения структуры сетки ***изменяются*** значения расчетных переменных. Узловые точки и ячейки, являющиеся соседними к "исходным первого порядка", назовем "исходными второго порядка".

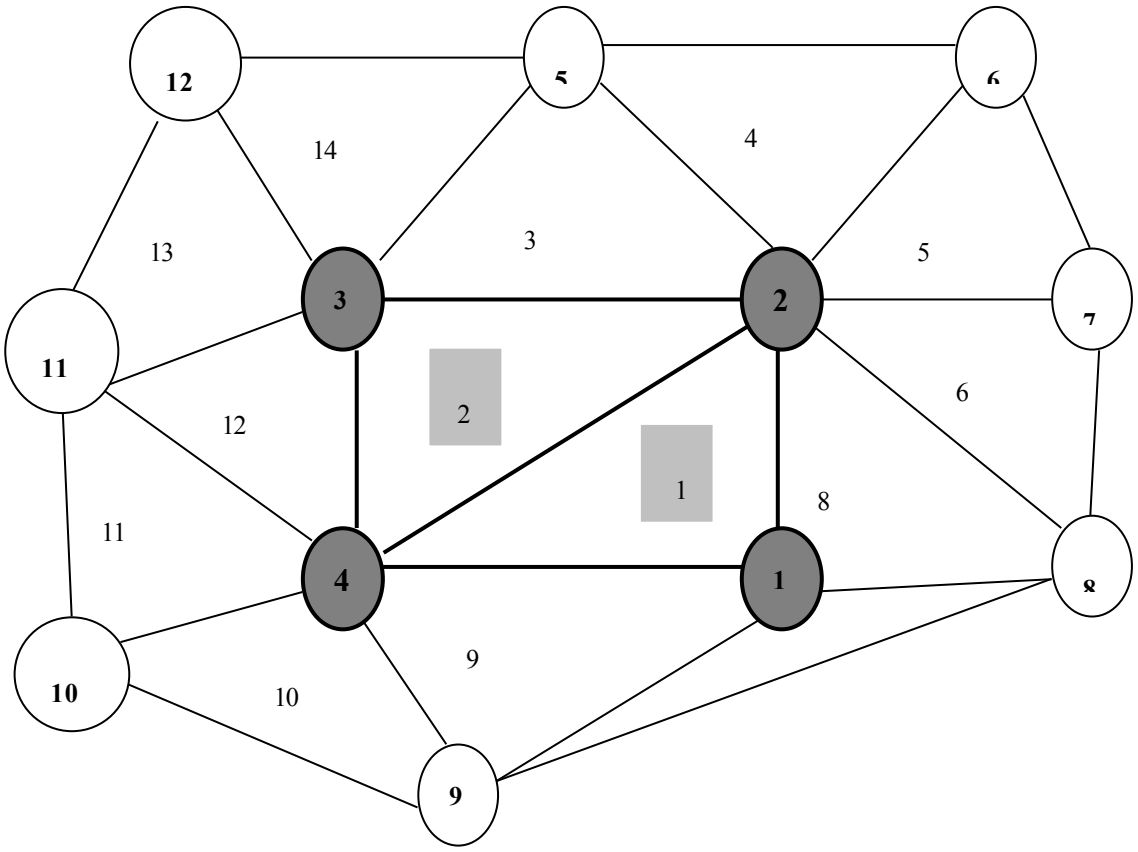

Рис. 4

Чтобы не перегружать рисунок часть ячеек, в которые входят узлы с номерами от 5 до 12, опущены. Совокупность значений, заданных в “исходных первого и второго порядков” назовем "***контекстом операции***".

При выполнении операции изменяются значения в ячейках первого порядка и ячейках второго порядка, то есть существует зависимость по чтению из памяти и записью в память. Эти свойства существенны при определении методов организации параллельных вычислений для программ рассматриваемого класса, так как, фактически, они определяют основные потоки информации, возникающие после распределения графа узлов между процессорами вычислительной системы с распределенной памятью.

3.2 ПАРАЛЛЕЛЬНОЕ ВЫПОЛНЕНИЕ ОПЕРАЦИЙ ИЗМЕНЕНИЯ СТРУКТУРЫ СЕТКИ

При реализации операции изменения структуры сетки на параллельной ЭВМ с распределенной памятью возникает следующие проблемы

1. где выполнять операцию в том случае, когда контекст операции принадлежит нескольким процессорам;
2. как формировать локальные структуры данных для проведения операции;
3. локальная балансировка распределения узлов после выполнения операции.

Конечно, существуют и другие проблемы при решении данной задачи, но мы остановимся на рассмотрении перечисленных выше задач.

3.2.1 ГДЕ ПРОВОДИТЬ ОПЕРАЦИИ ИЗМЕНЕНИЯ СТРУКТУРЫ СЕТКИ

Представленный ниже рисунок приводится для иллюстрации сложности задачи параллельной перестройки сетки. Схематично представлена сетка, распределенная на четыре процессора. Пусть верхний квадрант соответствует процессору с номером 1.

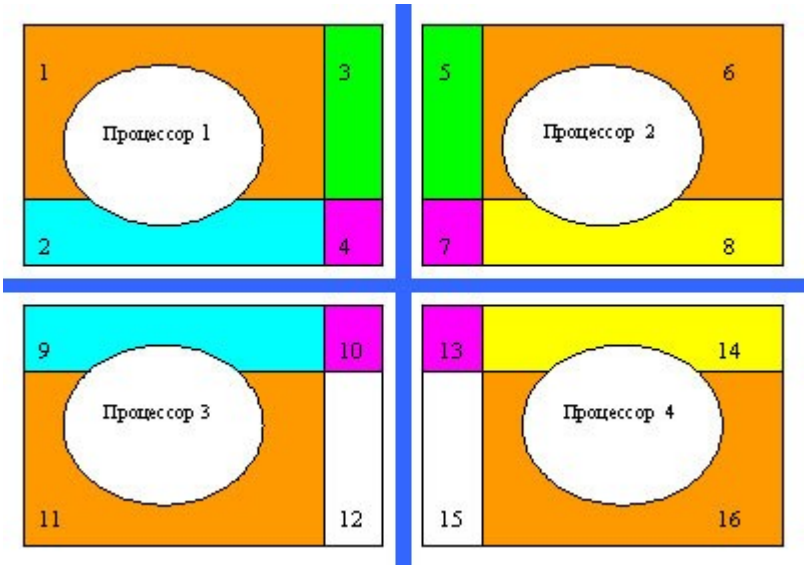

Рис. 5

Пусть исходная сетка распределена на четыре процессора. Фрагмент сетки, распределенный в первый процессор, состоит из 4 частей (на рисунке они обозначены цифрами 1-4). В части, обозначенной цифрой 1, контекст операций полностью состоит из узлов и ячеек, содержащихся в первом процессоре. В части, обозначенной цифрой 2, контекст операций состоит из узлов и ячеек, содержащихся как в первом процессоре, так и 3-м процессоре. В части, обозначенной цифрой 3, контекст операций состоит из узлов и ячеек, содержащихся как в первом процессоре, так и во втором процессоре. И, наконец, в части, обозначенной цифрой 4, контекст операций состоит из узлов и ячеек, содержащихся как в первом процессоре, так и в остальных процессорах. Аналогично можно рассмотреть и остальные части других процессоров. Очевидно, что операции перестройки в частях 1,6,11,16 можно проводить параллельно в каждом процессоре. Но дальше начинает влиять тот факт, что контекст состоит из узлов и ячеек, принадлежащих разным процессорам. Например, для частей 2 и 9 в операции задействованы значения из первого и третьего процессоров (аналогично можно рассматривать и части 3,5; 8,14; 12,15;). В этом случае процедура планирования, т.е. в каком процессоре проводить операцию, какие данные необходимо в этот процессор передавать и куда размещать полученные результаты, приобретает существенную сложность. Для приведенных частей в операциях задействованы данные только из двух процессоров. Более трудная задача стоит в том случае, когда операции надо проводить для частей с номерами 4,7,10,13. В этом случае при планировании необходимы данные из четырех процессоров. Конечно, мы представили упрощенный пример, но в общем случае, когда распределение узлов по процессорам произвольное (например, только с учетом минимизации числа связей между процессорами), задача изменения структуры сетки становится очень трудоемкой.

На представленном ниже рисунке выбрано такое распределение исходной сетки на линейку процессоров, в котором каждый процессор с номером k имеет соседние узлы только в процессорах с номерами k-1 и k+1. При таком распределении, в основном, контекст операции будет находиться только в двух соседних процессорах.

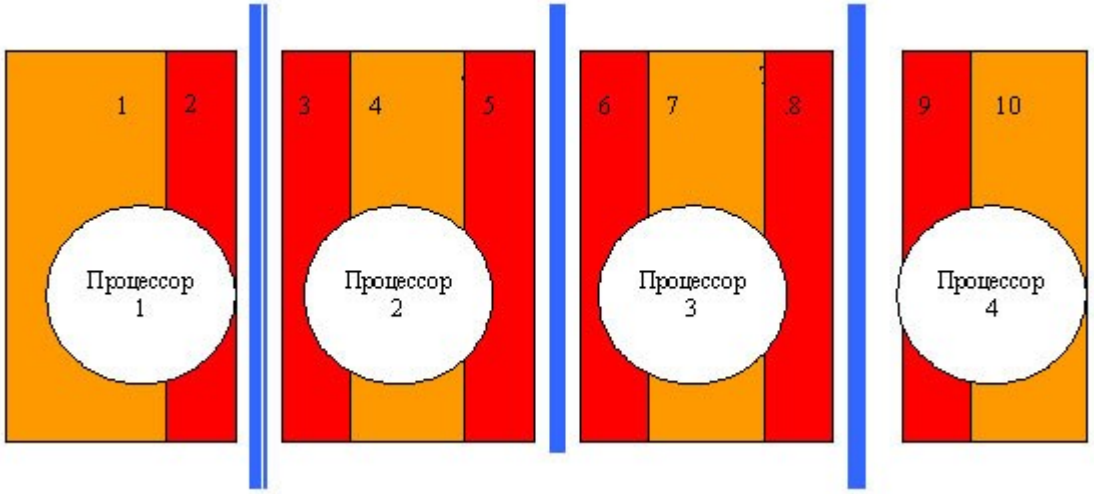

Рис. 6

В части, обозначенной цифрой 1, контекст операций полностью состоит из узлов и ячеек, содержащихся в первом процессоре. В части, обозначенной цифрой 2, контекст операций со стоит из узлов и ячеек, содержащихся как в первом процессоре, так и в правом (втором) процессоре. В части, обозначенной 4, контекст операций полностью состоит из узлов и ячеек, содержащихся во втором процессоре. В части, обозначенной цифрой 3, контекст операций состоит из узлов и ячеек, содержащихся как в первом процессоре, так и во втором процессоре. Если объединить фрагменты сеток, принадлежащих частям 2 и 3, то можно проводить операцию изменения структуры для получившегося фрагмента сетки.

Основной принцип, положенный в основу нашей реализации, состоит в том, что все операции разделяются на две группы. К первой относятся операции, чей контекст полностью принадлежит одному процессору. Ко второй группе относятся операции, когда контекст размещен только в двух процессорах. Случай, когда контекст содержится более чем в двух процессорах, в данной работе является недопустимым.

Когда контекст разделяется только двумя процессорами, можно сделать простой выбор. Мы считаем, что в этом случае операцию должен выполнять процессор с меньшим номером. Конечно, процессор с максимальным номером не выполняет операцию. Зато каждый процессор, по-прежнему, может выполнять операции независимо от других процессоров.

3.2.2 ПОРЯДОК ПРОВЕДЕНИЯ ОПЕРАЦИИ ИЗМЕНЕНИЯ СТРУКТУРЫ СЕТКИ

Общая схема проведения операций перестройки сетки состоит из двух шагов. На первом шаге проводятся операции, чей контекст целиком принадлежит процессору. На втором шаге проводятся операции, в которых контекст разделяется двумя процессорами.

Операции, у которых контекст целиком содержится в процессоре, могут выполняться независимо друг от друга в различных процессорах. Более того, изменения в последовательных исходных программах незначительны. Они состоят в том, что надо провести проверку того, что контекст полностью содержится в текущем процессоре.

Если контекст операции содержит узлы (ячейки) принадлежащие двум процессорам, то последовательность выполнения перестройки сетки состоит в следующем. Для проведения вычисления на сетке, состоящей из узлов, принадлежащих процессорам с номерами k и k-1 (k=2,…p, где p число процессоров на которых решается задача) формируются локальные структуры данных (на время проведения операции), которые размещаются в процессоре с номером k-1. Конечно, исходные структуры должны быть сохранены в некоторой специальной области сохранения. В результате проведения операции изменяется структура графа узлов (ячеек). В этом случае надо определить, к какому из двух процессоров относятся эти изменения, и провести коррекцию исходных структур в этом процессоре. Если выполненные изменения затрагивают граф, распределенный на процессор с номером k, то необходимая информация отправляется в этот процессор и там производится соответствующая коррекция. Операции изменения, как и на первом шаге перестройки, проводятся в каждом процессоре независимо друг от друга. Процессор с номером p не принимает участие в выполнении операций.

При проведении перестройки второй шаг может быть разбит на два последовательных подшага. При этом в начале операции перестройки будут выполнять все процессоры с нечетными номерами, а только после этого все процессоры с четными номерами. Такое разбиение вызвано проблемами, которые могут возникать при формировании локальных структур данных (п. 3.2.3).

ЗАМЕЧАНИЕ. Хотелось подчеркнуть, что при таком порядке (параллельном) выполнения операций изменения структуры сетки результат (новая сетка) может отличаться от того, как это было бы получено с помощью последовательной программы. Здесь можно провести аналогию с последовательным и параллельным суммированием. Корректность такого изменения порядка должна быть обеспечена самим математическим методом, выбранным для решения задачи.

3.2.3 ФОРМИРОВАНИЕ ЛОКАЛЬНЫХ СТРУКТУР ДАННЫХ

Программа, в которой решается задача изменения сетки, использует общие структуры данных – представление графа сетки, описание ячеек, вектора расчетных переменных и т.п. На первом шаге решения задачи изменения структуры сетки, когда контекст операции полностью находится в текущем процессоре, никаких изменений в структуры данных вносить не надо.

Более сложная ситуация возникает в том случае, когда контекст состоит из фрагментов сетки, содержащихся в двух соседних процессорах. Неформально, мы выделяем фрагменты сетки задачи каждого из двух процессоров, участвующих в операции, и объединяем их. Для получившей подсетки создаются необходимые структуры – граф узлов, граф ячеек, вектора расчетных переменных и т.п. и размещаются они в том процессоре, в котором будет решаться задача изменения структуры сетки.

Кратко рассмотрим действия, которые необходимо выполнять, когда контекст состоит из ячеек и узлов, размещенных в двух процессорах. Назовем *левой границей процессора* с номером k совокупность узлов, которые связаны с узлами, размещенными в процессоре с номером k-1. Аналогично определим и правую границу процессора. Как отмечали в п. 3.2.1, контекст, который содержит узлы, содержащиеся в процессорах с номерами k и k-1, будет решаться в процессоре с номером k-1. Поэтому в процессоре с номером k-1 выделим подграф, начиная с правой границы процессора.

Выделение производится следующим образом. В начале выделяем узлы правой границы процессора. Затем выделяем все узлы, принадлежащие текущему процессору и которые соседствуют с узлами правой границы. Назовем получившуюся совокупность узлов *подграфом первого порядка*. Далее выделяем узлы, соседствующие с подграфом первого порядка. Получаем подграф второго порядка. И завершаем выделение построением подграфа третьего порядка. Конечно, в этот подграф включаются и все ячейки, в которые входят выбранные узлы. Выбранный подграф назовем *левым локальным подграфом перестройки*. Аналогично в процессоре с номером k строится *правый локальный подграф перестройки*. При этом исходными данными для построения правого локального подграфа являются узлы левой границы процессора. Поскольку операция изменения структуры сетки должна проводится в процессоре с номером k-1, то необходимо передать правый локальный подграф из процессора с номером k в процессор с номером k-1. В процессор с номером k-1 передаются и все связанные с выбранными узлами и ячейками значения расчетных переменных. Далее из указанных двух подграфов строится объединение и получившийся граф является основой для проведения операций изменения структуры для сетки, примыкающей к правой границе процессора с номером k-1 и примыкающей к левой границе процессора с номером k.

В получившемся графе вводится новая локальная нумерация узлов и ячеек. Это во многом определяется тем, что для операций изменения структуры сетки в данном случае используются те же программы, что и в случае, когда контекст целиком содержится в одном процессоре.

Далее остановимся на ограничениях предлагаемой организации схемы расчета. При построении локального графа перестройки в каком-либо процессоре может возникнуть ситуации, когда при построении локального подграфа перестройки для процессоров с номерами k-1 и k требуются узлы (ячейки), принадлежащие процессору с номером (k-2 или k+1).

Определим *ширину графа* на процессоре как минимальный путь от левой границы процессора до правой границы процессора. Если эта величина меньше 8, то один и тот же узел может оказаться как в левом локальном подграфе перестройки, так и в правом локальном подграфе перестройки. Поскольку эти подграфы используются в разных процессорах, которые должны проводить независимо друг от друга операции изменения структуры сетки, то такая ситуация может привести к некорректной работе программы. Таким образом, если ширина графа меньше 8, то проводить параллельную работу всеми процессорами недопустимо.

Однако можно разбить вычисления на 2 подшага. На первом работу будут выполнять только процессоры с нечетными номерами. Для работы такого процессора (с номером k) требуется левый локальный подграф из текущего процессора и правый локальный подграф от процессора с номером k+1. Для корректной работы в этом случае достаточно, чтобы ширина графа на процессоре была не меньше 4.

В случае, когда ширина графа на каком-то процессоре окажется меньше 4, необходимо проводить глобальное перераспределение узлов сетки.

3.3 УПРАВЛЕНИЕ СВОБОДНОЙ И ЗАНЯТОЙ ПАМЯТЬЮ

Появление и удаление узлов (ячеек) ставит вопрос об организации процесса корректировки структур, поддерживающих задание сетки исходной задачи. В исходной программе это реализовано достаточно просто. С каждым узлом (номер - i) связывается признак (jsort(i)[1]). Если значение признака отлично от -1, то узел с номером i принимает участие в расчете. В противном случае узел является свободным и не принимает участие в расчете. Вновь появившийся узел получает номер k, являющийся минимальным номером из узлов, не принимающих участие в расчете. При этом значение jsort(k) становится отличным от -1. Если узел с номером k удаляется из расчета, то jsort(k) принимает значение -1. Процесс управления памятью, описанный в последовательной программе перенесен в параллельную версию без изменений.

ЗАМЕЧАНИЕ. Достаточно простой механизм добавления/удаления узлов базируется на структурах, введенных авторами последовательной версии программы. Можно говорить о некотором “разбазаривании” памяти при размещении графов узлов и ячеек, но это полностью себя оправдывает простотой решения задачи добавления и удаления новых узлов и ячеек.

3.4 ДИНАМИЧЕСКАЯ БАЛАНСИРОВКА ЗАГРУЗКИ

Исходное распределение базовых структур (и, конечно, вычислений) основывается на равномерном распределении узлов по процессорам, на которых решается задача. В процессе решения задачи число узлов меняется. Это может привести к разбалансировке вычислений - в некоторых процессорах число узлов может увеличиваться, а в некоторых, наоборот, уменьшаться. Очевидно, что для такого случая необходимо предусмотреть процедуру нового распределения узлов по процессорам. Такая процедура должна учитывать как *текущее состояние распределения узлов*, так и определение того, *какие узлы из каких процессоров и куда должны быть перемещены*. При этом, перемещение узлов означает, что необходимо "перебросить" в другой процессор и все узловые переменные. Очевидно и то, что "переброска" узлов приводит и к изменению распределения ячеек по процессорам. В результате нового распределения изменяются и базовые структуры алгоритма.

Вызов процедуры, анализирующей распределение узлов, проводится через определенный интервал расчета[2]. Если разница между максимальным и минимальным значениями узлов в процессорах превосходит некоторый заданный параметр, то происходит вызов процедуры непосредственно проводящей перераспределения узлов сетки. Алгоритм процедуры основывается на принципе “сообщающихся сосудов”. Одним из основных результатов выполнения этой процедуры состоит в том, что после завершения работы процедуры принцип соседства (т.е. каждый процессор имеет соседей только слева и справа от себя) сохраняется.

В качестве иллюстрации затрат на проведение такой операции приведем следующие временные результаты расчета. Исходная сетка состояла из 44622 узлов и 88201 ячеек. Проводился расчет 300 итерационных шагов на 24 процессорах. В результате выполнено (всеми процессорами) операций разрыва связи между узлами - 12708, операций введения в сетку нового узла – 4228 и операций удаления узла – 4055. В результате получилась сетка состоящая из 44868 узлов и 88683 ячеек. Динамическая балансировка выполнялась 9 раз и ее выполнение заняло 1.22 сек. Проверка балансировки проводилась каждые 10 шагов и условием ее применения состояло в том, чтобы минимальное число узлов на процессоре отличалось от максимального больше, чем на 50 узлов (при начальном распределении каждый процессор содержал 1860 узлов). Важным показателем является и число перемещаемых узлов и ячеечных переменных задачи. В нашем случае перемещение одного узла состоит в пересылке 44 (из них 24 относятся к параметрам химической кинетики) узловых значения, а для ячейки — 5.

ЛИТЕРАТУРА:

1. Н.В. Арделян, К.В.Космачевский “Неявный свободно-лагранжев метод расчета двумерных магнитогазодинамических течений.” Математическое моделирование, М., Издательство Московского университета, 1993. с.25-44.
2. А.Н. Андрианов, К.Н.Ефимкин “Подход к параллельной реализации численных методов на неструктурированных сетках. “ Вычислительные методы и программирование”, М., Издательство Московского университета, 2007. т.8, с.6-17, http://num-meth.srcc.msu.su/.

[1] Величина jsort(i) используется также для определения, является ли точка с номером i граничной или внутренней.

[2] Расчет состоит в последовательности итерационных шагов. Интервал, через который проводится вызов процедуры, задается пользователем в виде параметра расчета.