УДК 519.6
DOI: 10.53403/9785951505309_2022_27_1_246

Методы балансировки вычислительной нагрузки в методике ТИМ

С. С. Соколов, И. Г. Новиков, А. А. Воропинов, Т. Н. Половникова

Рассматриваются методы балансировки вычислительной нагрузки, реализованные в рамках методики ТИМ. Описывается набор критериев для проведения динамической и квазидинамической балансировок. Динамическая балансировка заключается в переносе расчета ячеек с одного процесса на другой, квазидинамическая – в полной переинициализации данных для задачи без остановки ее расчета и построении новой декомпозиции. Рассмотрены области применения алгоритмов динамической и квазидинамической балансировок. Применение методов балансировки вычислительной нагрузки позволяет эффективно загрузить процессорное поле, выделенное на задачу, и ускорить счет.

Введение

Методика ТИМ [1, 2] предназначена для решения многомерных нестационарных задач механики сплошных сред с использованием лагранжева подхода на неструктурированных многогранных сетках произвольного вида. По данной методике можно рассчитывать широкий набор процессов: газовую динамику и упругопластичность, детонацию, теплопроводность, магнитную гидродинамику с учетом диффузии магнитного поля, многопотоковую газодинамику и др. По методике ТИМ рассчитываются задачи в многообластной постановке. При этом, как правило, каждое вещество выделяется в отдельную область. Между областями решается задача контактного взаимодействия. Смесь веществ не допускается.

Для повышения точности расчетов необходимо выполнять численное моделирование на сетках с большим количеством ячеек. Такие расчеты требуют значительного календарного времени. Один из путей его сокращения – проведение расчетов в параллельном режиме счета.

Для методики ТИМ используется метод трехуровневого распараллеливания [3]. На первом (верхнем) уровне осуществляется распараллеливание счета по математическим областям в модели распределенной памяти с использованием интерфейса передачи сообщений MPI [4]. На втором уровне также с помощью MPI распараллеливается счет внутри математической области по параобластям. На третьем (нижнем) уровне осуществляется распараллеливание итераций счетных циклов в модели общей памяти с использованием интерфейса OpenMP [5]. Эти подходы могут применяться как вместе в различных сочетаниях, так и раздельно при расчете одной задачи.

Эффективное выполнение счетных программ на многопроцессорных машинах с распределенной памятью требует декомпозиции данных по процессам таким образом, чтобы распределение вычислительной нагрузки было равномерным, а количество межпроцессных обменов – минимальным. При этом в процессе расчета сбалансированность вычислительной нагрузки со временем может ухудшаться. Проблема балансировки вычислительной нагрузки при проведении расчетов встречается во многих параллельных приложениях (см., например, [6–9]).

Балансировку вычислительной нагрузки для сеточных методик, использующих декомпозицию по пространству, можно классифицировать следующим образом:

1. Статическая балансировка. Выполняется на этапе начала расчета при построении декомпозиции с учетом весовых функций. Весовая функция отражает объем вычислений, необходимых для расчета элемента сетки. В методике ТИМ это время, которое требуется для расчета данной ячейки.

2. Квазидинамическая балансировка. Заключается в построении полностью новой декомпозиции и переходе на нее без остановки счета. В методике ТИМ для квазидинамической балансировки применяются алгоритмы, аналогичные статической балансировке.

3. Динамическая балансировка. Заключается в передаче расчета отдельных элементов сетки между вычислительными устройствами. В методике ТИМ такой перенос осуществляется по ячейкам и называется переброской.

В процессе численного моделирования работы различных конструкций по методике ТИМ количество ячеек не фиксировано, структура сетки может изменяться, выполняются различающиеся по объему вычислений процессы (алгоритмы поддержания качества счетной сетки, решение уравнений состояния веществ, кинетических уравнений – кинетики детонации, кинетики разрушения и т. д.). Поэтому, чтобы эффективно загрузить процессорное поле, выделенное на задачу, и ускорить счет, а также улучшить качество декомпозиции, необходима балансировка вычислительной нагрузки.

Алгоритмы декомпозиции со статической балансировкой не могут полностью обеспечить оптимальное распределение вычислительной нагрузки по процессам. Это объясняется тем, что сами алгоритмы декомпозиции имеют определенную точность – обычно около 5 %. Однако декомпозиция сначала делается для математических областей, затем для параобластей внутри области, и при этом каждый из этапов выполняется с точностью 5 %. Негативный вклад в сбалансированность при статической балансировке связан и с тем, что используемая весовая функция является приближенной, так как получается усреднением времени для расчета по ячейкам. При декомпозиции со статической балансировкой также трудно учесть дополнительные ограничения (которые будут рассмотрены ниже), так как это требует решения полной оптимизационной задачи. В итоге разбалансированность может составлять до 10 %.

В то же время опыт проведения расчетов показал, что основной проблемой является не получение плохой декомпозиции, а постепенное ухудшение ее свойств в процессе проведения расчета. Одним из путей решения данной проблемы является формирование и переход на новую декомпозицию (передекомпозиция). Однако, с одной стороны, выполнение передекомпозиции является дорогостоящей операцией, а с другой – результирующая декомпозиция может не улучшить ситуацию в случае использования сложных критериев для оценки ее качества. В таких случаях удобнее использовать локальное улучшение декомпозиции путем переброски ячеек между параобластями (динамическая балансировка вычислительной нагрузки). Эта операция может также использоваться для улучшения сбалансированности в процессе вычислений, когда еще не достигнута критическая ситуация для выполнения передекомпозиции данных.

1. Расчет весовой функции

Один из первых вопросов, который возникает при разработке алгоритмов балансировки, связан с определением весовой функции, отражающей вычислительную нагрузку. Для методики ТИМ использование априорных оценок для весовой функции затруднительно, так как целый ряд алгоритмов характеризуется изменяющейся вычислительной нагрузкой, причем отличие от шага к шагу может достигать нескольких раз. Поэтому весовая функция должна определяться динамически, непосредственно в процессе проведения расчета. Это делается исходя из времени выпол-

нения разных счетных блоков на основании временных засечек. Другая проблема связана с изменяющимся в процессе счета количеством ячеек, которые непосредственно влияют на объем вычислений. К тому же некоторые алгоритмы распространяются не на одну конкретную ячейку, а на их набор – такая ситуация возникает, например, при решении систем линейных уравнений. Также важно заменить, что обращение к процедурам засечек времени является достаточно дорогостоящей операцией, поэтому чрезмерная детализация (например, определение времени расчета одного уравнения в одной конкретной ячейке) может приводить к существенному росту общего времени расчета.

В связи с этим при определении весовой функции в методике ТИМ должны выполняться следующие требования:

1) определение на основе времени выполнения различных блоков;

2) учет средневзвешенного объема вычислений за несколько шагов. При этом вклад за последние шаги должен быть значительно выше, чем за предыдущие 100 и более шагов;

3) привязка к ячейке сетки и области;

4) учет алгоритмов, имеющих привязку не одной, а к целому набору ячеек;

5) достаточно редкие обращения к процедурам получения временных засечек.

С учетом вышеизложенного в методике ТИМ принята следующая схема расчета весовой функции:

1. Вес области принимается равным суммарному весу ее ячеек.

2. При первоначальной декомпозиции, а также в других случаях, когда значение весовой функции становится неактуальным, вес ячейки принимается равным единице.

3. Весовая функция для ячейки состоит из двух частей: w_S – веса на для текущем шаге и W_c – общего веса.

4. Для определения w_S используется значение t_S – время, необходимое для расчета данной ячейки на текущем шаге, которое определяется следующим образом: $t_S = \sum_{j=1}^{K} \left(T_j / N_j \right)$, где K – количество счетных блоков, в которых участвует ячейка; T_j – время расчета блока j; N_j – количество ячеек, участвующих в расчете блока j.

5. В счетных блоках засечки времени могут не выполнять. В этом случае считается, что в расчете блока участвуют все ячейки с одинаковой нагрузкой. С другой стороны, допускается дополнительное «утяжеление» отдельных ячеек по каким-то другим законам (например, если ячейка участвует в расчете контактного взаимодействия). Для согласования времени счета области и ячеек в конце счетного шага производится расчет окончательного веса ячейки w_S:

$$w_S = \begin{cases} t_S \dfrac{T_A}{\sum_{i=1}^{K_c} \left(t_S \right)_{c_i}}, & \text{если } \sum_{i=1}^{K_c} \left(t_S \right)_{c_i} > T_A, \\ t_S + \dfrac{T_A - \sum_{i=1}^{K_c} \left(t_S \right)_{c_i}}{K_c}, & \text{если } \sum_{i=1}^{K_c} \left(t_S \right)_{c_i} \le T_A, \end{cases}$$

где T_A – время на расчет всей области, K_c – количество ячеек в области, $\left(t_S \right)_{c_i}$ – время расчета (ненормированный вес) ячейки i на текущем шаге.

6. Общий вес ячейки пересчитывается в конце каждого счетного шага по формуле

$$W_c = \alpha W_c + w_S,$$

где $\alpha < 1$ – некоторая константа, определяющая вклад за предыдущие шаги. Например, при $\alpha = 0{,}955$ за 100 шагов вклад текущего w_S уменьшится примерно в 100 раз. Такая схема позволяет, с одной стороны, сгладить влияние конкретного шага, а с другой стороны, нивелирует влияние шагов, сделанных «давно».

7. При балансировке в качестве весовой функции ячеек используются значения W_c.

2. Декомпозиция и статическая балансировка

При запуске расчета производится операция декомпозиции. Для методики ТИМ декомпозиция по пространству осуществляется в два этапа.

На первом этапе для каждой математической области определяется количество и размер параобластей. При этом решается модифицированная задача о минимизации времени обработки (minimum makespan problem [10]) с разделением заданий на подзадания разного веса. На данном этапе для балансировки используются веса математических областей. На втором этапе для каждой математической области формируется разбиение на параобласти (количество параобластей и их вес являются результатом первого этапа декомпозиции). Эта задача сводится к решению задачи о разрезании графа на подграфы. При формировании графа его вершины (точки) соответствуют ячейкам сетки, а ребра – соседству между ячейками. В методике ТИМ ячейка является основным счетным элементом, поэтому декомпозиция для мелкозернистого распараллеливания строится по ячейкам.

Для решения задачи о разрезании графа на подграфы используются алгоритмы библиотек MeTiS [11] и SCOTCH [12]. Также реализован собственный алгоритм топологическо-геометрической декомпозиции.

2.1. Статическая балансировка

Статическая балансировка является базовым алгоритмом, выполняющимся на этапе запуска задачи на счет. В процессе проведения расчета накапливается информация о времени, необходимом для расчета различных ячеек. Данная информация сохраняется вместе с контрольной точкой и используется в следующем запуске для определения весовой функции. Главным недостатком такого подхода является то, что он плохо согласуется с задачами, рассчитываемыми за небольшое количество запусков. Для таких задач приходится делать специальные запуски для накопления временных засечек. Также существенным недостатком является то, что данный подход не позволяет сбалансировать вычислительную нагрузку в тех задачах, в которых она меняется значительно и довольно быстро в пределах одного запуска. Указанные недостатки ограничивают область применения данного подхода.

2.2. Топологическо-геометрическая декомпозиция

Топологическо-геометрическая декомпозиция основана на топологическом соседстве ячеек между собой. Помимо соседства ячеек сетки используется еще и геометрический критерий, ограничивающий попадание ячеек в параобласть при их наборе. Построение декомпозиции по топологическо-геометрическому алгоритму гарантирует односвязаность параобластей и относительную гладкость границ.

Декомпозиция области проводится с учетом топологического соседства ячеек с использованием координат центров ячеек и координат узлов сетки таким образом, чтобы число ячеек в каждой параобласти было примерно одинаковым либо сбалансированным по вычислительной нагрузке.

Алгоритм разбиения области на параобласти строится следующим образом.

Зная количество ячеек сетки N, можно получить количество ячеек $NPar$ в каждой параобласти $p = \overline{1, nP}$:

$$NPar(p) = \begin{cases} \dfrac{N}{nP} + 1 & \text{для первых } m \text{ параобластей,} \\ \dfrac{N}{nP} & \text{для остальных } nP - m \text{ параобластей,} \end{cases}$$

где m – остаток от деления N на nP.

Далее определяются $X_{\min}$ и $X_{\max}$, $Y_{\min}$ и $Y_{\max}$, $Z_{\min}$ и $Z_{\max}$ – глобальные минимумы и максимумы координат узлов исходной области. Находится ближайший к вершине параллелепипеда узел $(X_{\min}, Y_{\max}, Z_{\max})$ области. Этот узел становится начальным (опорным) в разбиении.

Далее определяется стартовая ячейка, одной из вершин которой является начальный (опорный) узел, и ее центр r_0. Также определяется ее максимальная диагональ (расстояние между самыми удаленными друг от друга вершинами ячейки). Вычисляется некоторый максимальный параметр ΔR увеличения радиуса сферы, в пределах которой ячейки в параобласть будут набираться итерационно. Этот параметр выбран эмпирически и составляет $4-5$ максимальных диагоналей стартовой ячейки.

Начиная от стартовой ячейки, по топологическому соседству производится набор ячеек-претендентов в параобласть, которые попадают в сферу радиусом $R = \Delta R i$: $|\vec{r}_k - \vec{r}_0| < R$, где $\vec{r}_k$ – центр k-й ячейки-претендента, i – номер итерации увеличения радиуса при наборе параобласти.

Если центр ячейки k находится в пределах заданной сферы, то для ячейки устанавливается признак принадлежности к параобласти p и счетчик количества ячеек K увеличивается на 1.

Если больше нет ячеек, попадающих в сферу радиусом R, а количество ячеек $K < NPar(p)$, то выполняется переход к следующей итерации по увеличению радиуса сферы: $i = i + 1$. Количество итераций по увеличению радиуса для экономии времени ограничено пятью пустыми итерациями подряд. Под пустой итерацией понимается итерация, на которой не произошло добавление ячеек в рассматриваемую параобласть.

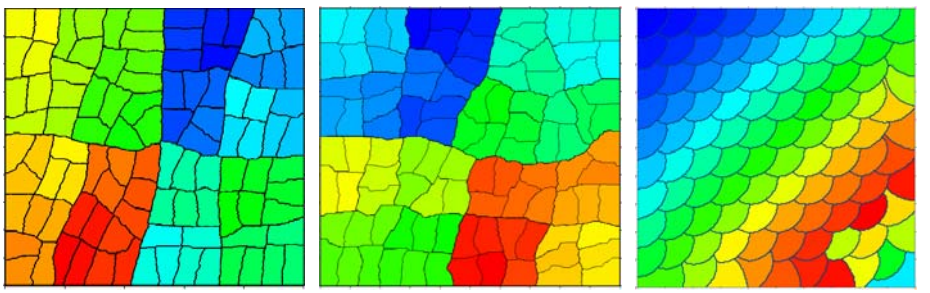
Завершается набор ячеек в параобласть p тогда, когда будет выполнено условие $K \geq NPar(p)$ или выполнены все итерации по увеличению радиуса, включая пустые.

Данный подход к разбиению области на параобласти не обладает недостатками чисто геометрического подхода и является наиболее подходящим для декомпозиции данных по процессам в методике ТИМ.

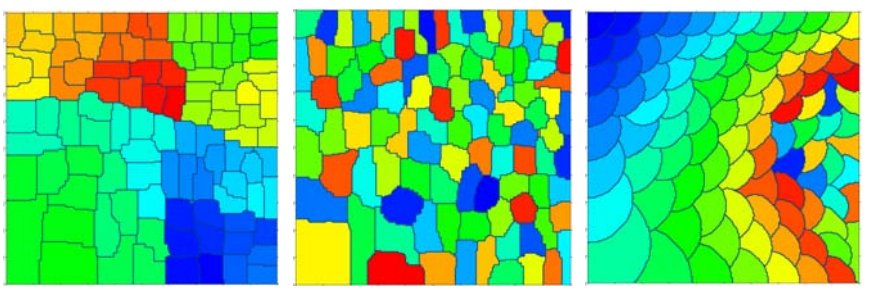
2.3. Примеры применения различных алгоритмов декомпозиции

Примеры применения различных алгоритмов декомпозиции двумерной сетки на 100 параобластей при условии равного веса у всех ячеек представлены на рис. 1,*а*.

Примеры применения различных алгоритмов декомпозиции двумерной сетки на 100 параобластей при условии, что вес ячеек равен расстоянию от левого нижнего угла до центра ячейки, представлены на рис. 1,*б*.



а



б

Рис. 1. Примеры декомпозиций двумерной сетки: а – равный весу всех ячеек; б – вес ячейки равен расстоянию от центра до левого нижнего угла; слева – с помощью библиотеки SCOTCH, в центре – с помощью библиотеки MeTiS, справа – топологическо-геометрическая декомпозиция

Время выполнения декомпозиции с помощью библиотек SCOTCH и MeTiS и топологическо-геометрической декомпозиции, а также коэффициент разбалансированности, полученный при запуске трехмерных задач на сетке в 10, 40 и 70 млн ячеек и декомпозиции на 100 компактов [3], представлены в табл. 1.

Таблица 1

Время выполнения декомпозиции

Алгоритм	Коэффициент разбалансированности, %	Время выполнения декомпозиции, с		
		10 млн ячеек	40 млн ячеек	70 млн ячеек
MeTiS	1,3	8,7	47	99
SCOTCH	0,99	27	137	282
Топологическо-геометрическая декомпозиция	0,29	78	274	417

3. Квазидинамическая балансировка вычислительной нагрузки

Квазидинамическая балансировка вычислительной нагрузки в методике ТИМ во многом является развитием подхода статической балансировки. В этом случае на счетном шаге оценивается текущая декомпозиция с точки зрения сбалансированности. Если текущая декомпозиция несбалансирована, то производится сохранение контрольной точки, полная очистка памяти, считывание сохраненных данных с построением новой декомпозиции.

Такой подход не требует выполнения остановок, и благодаря этому решается ряд проблем, присущих статической балансировке. Например, для задач, где существенно меняется вычислительная нагрузка, передекомпозиция может быть сделана несколько раз за один запуск. Важным преимуществом данного подхода является возможность использования существующих алгоритмов сохранения контрольной точки, декомпозиции и инициализации параллельного счета. Также получение новой декомпозиции позволяет рассматривать всю задачу сразу, уходя от локальных операций.

Недостатком данного подхода является необходимость сохранения и чтения контрольной точки, а также построение новой декомпозиции. Эти операции для больших задач часто являются довольно дорогостоящими по времени, и могут занимать существенную долю от времени выполнения расчета. По этой причине операция передекомпозиции должна выполняться достаточно редко, чтобы выигрыш в скорости счета превысил накладные расходы. В результате данный подход плохо применим для задач, где вычислительная нагрузка меняется интенсивно на длительном протяжении расчета.

3.1. Критерии квазидинамической балансировки вычислительной нагрузки

Образование струйных и вихревых течений, нарушение связности областей, изломы параобластей вблизи границы между параобластями (*параллельной границы*), происходящие в процессе численного моделирования конструкций в задаче, могут приводить к увеличению межпроцессных обменов. Алгоритмы квазидинамической балансировки используются для устранения указанных недостатков.

Для повышения технологичности проведения расчетов были разработаны и внедрены следующие критерии-анализаторы квазидинамической балансировки вычислительной нагрузки.

1. *Разбалансированность по вычислительной нагрузке*. Данный критерий является классическим при расчетах двумерных и трехмерных задач. Если вычислительная нагрузка прцесса больше средней загруженности по процессорам (разница более 30 %), то выполняется квазидинамическая балансировка.

На рис. 2 приведен пример декомпозиции (при условии равного веса и равного размера у всех ячеек). Если каждая параобласть рассчитывается на своем процессе, то на процессе, рассчитывающем самую большую параобласть (слева), будет превышение средней вычислительной нагрузки.

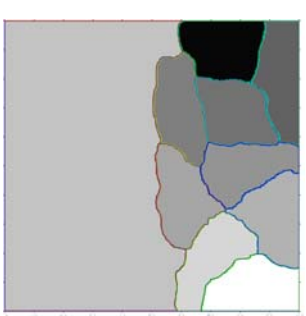
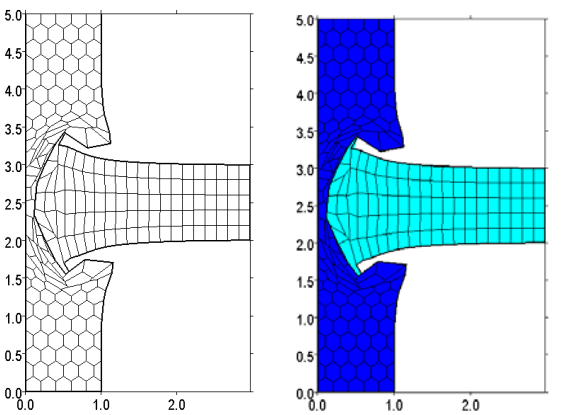


Рис. 2. Пример декомпозиции, приводящей к разбалансированности по вычислительной нагрузке

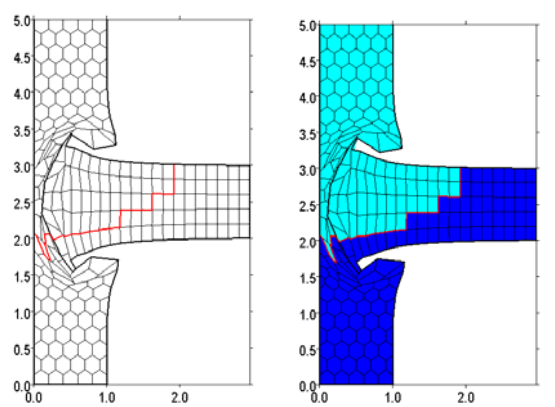
2. *Параобласть близка к перебитию*. Перебитие параобласти приводит к нарушению ее связности. Если параобласть перебивается, то изменяется граф обменов между параобластями. Это требует специальных глобальных операций, снижающих производительность, как при непо-

средственном перебитии параобласти, так и при регулярном счете. По этой причине желательно не допускать такой ситуации.

На рис. 3 представлено сечение цилиндра, который врезается в преграду. Цилиндр и преграда принадлежат разным параобластям. В процессе внедрения цилиндра преграда становится тоньше. Если в месте пробития толщина преграды равна одной ячейке (см. рис. 3,*а*), то на следующем шаге произойдет перебитие параобласти, что приведет к появлению несвязных параобластей. Чтобы этого избежать, необходимо выполнить передекомпозицию всей задачи (см. рис. 3,*б*).



а

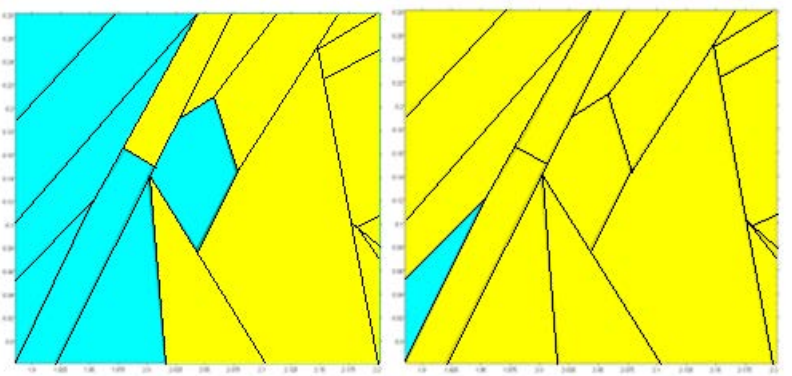


б

Рис. 3. Применение критерия «параобласть близка к перебитию»:
а – ситуация на шаге, предшествующем пробитию преграды цилиндром;
б – декомпозиция после квазидинамической балансировки

3. *Неудовлетворительные геометрические свойства ячейки*. Предполагается, что ситуация не может быть исправлена алгоритмами динамической балансировки (рис. 4).

Рассматриваются только ячейки, у которых параграничных (ллежащих на границе параобластей) ребер больше, чем внутренних. Для того чтобы в результирующей декомпозиции уменьшить объем передаваемых данных, целесообразно выбирать ячейки с максимальной поверхностью раздела с соседней параобластью (в графе декомпозиции для соответствующей вершины количество разрезанных ребер больше, чем неразрезанных). Оценивается отношение количества параграничных ребер ячейки к количеству внутренних ребер. Если оно больше допустимого (заданного пользователем), то необходимо выполнить квазидинамическую балансировку.



а б

Рис. 4. Длина внутренних ребер ячейки меньше, чем длина внешних ребер:
а – до передекомпозиции, б – после передекомпозиции

4. *Ячейка содержит несколько особых точек*. В этом случае также может требоваться изменение коммуникационного графа. Особая точка – это узел сетки, в котором сходятся несколько границ между областями. В процессе численного моделирования по методике ТИМ может возникнуть ситуация, когда ячейка содержит несколько особых точек. В этих случаях необходима передекомпозиция всей задачи, чтобы избежать вырождения фрагмента границы, а значит, изменения коммуникационного графа задачи.

4. Динамическая балансировка вычислительной нагрузки

Динамическая балансировка вычислительной нагрузки основана на переброске ячеек между соседними параобластями непосредственно при проведении расчета. К плюсам динамической балансировки относится то, что при ее выполнении не требуется останавливать задачу и выполнять дорогостоящую операцию сохранения и чтения контрольной точки, а также полную передекомпозицию задачи. Однако динамическая балансировка не лишена недостатков, ограничивающих ее применение.

В частности, довольно часто при разбалансированности вычислительной нагрузки наиболее и наименее загруженные параобласти не являются соседними. По смыслу динамической балансировки переброска ячеек в этом случае напрямую невозможна. Если же ее организовать, то это приведет к несвязности параобласти. Таким образом, возможны ситуации, когда динамическая балансировка не может напрямую решить проблему разбалансированности или потребуется множество операций переброски ячеек.

Другую проблему для динамической балансировки представляет ситуация, когда вычислительная нагрузка в процессе численного расчета меняется очень резко. Такая ситуация возможна, например, на старте задачи, когда первоначальная декомпозиция выполняется без использования весовых функций. В результате первоначальная декомпозиция может быть несбалансированной, вплоть до того, что некоторые MPI-процессы должны полностью сменить набор ячеек. В этом случае объем передаваемых данных между параобластями чрезвычайно большой, и может оказаться, что быстрее выполнить передекомпозицию, т. е. квазидинамическую балансировку.

4.1. Определение операции переброски ячеек

Алгоритмы распараллеливания в методике ТИМ построены по ячейкам, и с ними связаны основные вычислительные затраты. В результате декомпозиции каждая ячейка относится строго к одному компакту [3], на основе которого строится параобласть. Первоначальное распределение ячеек по компактам определяет номер параобласти, где соответствующая ячейка рассчитывается.

В процессе счета по какой-либо причине возможно изменение принадлежности ячейки параобласти. Из исходной параобласти ячейка удаляется, и функция ее расчета передается другой параобласти.

Отметим, что переброска ячеек возможна только между параобластями одной математической области, так как в разных математических областях сетки полностью не зависят друг от друга.

4.2. Бинарность операции переброски ячеек

Как видно из общего описания, операция переброски ячеек всегда бинарна, т. е. расчет ячейки передается из одной параобласти в другую. По этой причине на этапе переброски ячеек проще рассматривать бинарные алгоритмы, работающие с парой параобластей.

Очевидно, что переброска произвольных ячеек нецелесообразна. В частности, если выполнять переброску внутренних ячеек, то это приводит к ухудшению качества декомпозиции. Параобласть может стать несвязной с эффектом «оторванных» ячеек, может измениться схема взаимодействия между параобластями. Все это в конечном счете снижает эффективность распараллеливания.

Необходимо оптимально выполнять переброску ячеек между соседними параобластями, имеющими общую параллельную границу. Более того, перебрасывать нужно только ячейки, которые непосредственно примыкают к границе. Это упрощает саму операцию переброски и позволяет существенно повысить качество результирующей декомпозиции.

Результаты применения операции переброски ячеек за несколько шагов представлены на рис. 5.

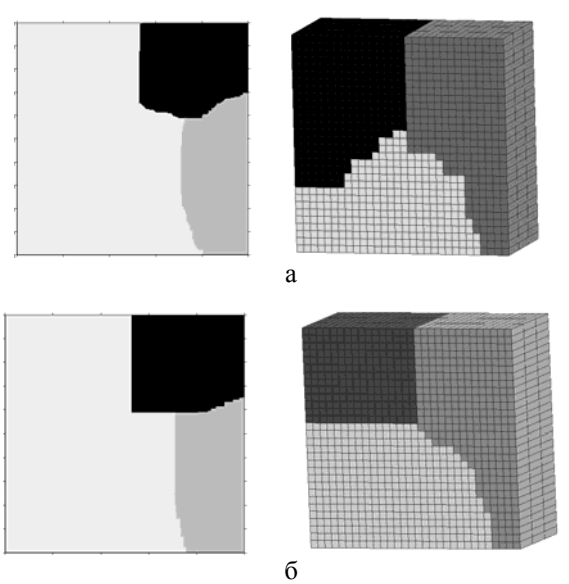


Рис. 5. Применение операции переброски ячеек: а – начальные декомпозиции, б – декомпозиции через несколько шагов; слева – двумерный случай, справа – трехмерный случай

4.3. Критерии динамической балансировки вычислительной нагрузки

Набор критериев для проведения динамической балансировки вычислительной нагрузки можно разделить на два класса:

1) определение необходимости улучшения качества декомпозиции;

2) выбор ячеек для переброски.

Понятно, что данные критерии связаны между собой. Критерии второго класса выполняются только в случае, если выполнен хотя бы один из критериев первого класса. Однако разделение критериев упрощает реализацию алгоритмов и согласование выполнения критериев между собой.

К критериям первого класса относятся:

– разбалансированность по вычислительной нагрузке;

– близость параобласти к перебитию.

К критериям второго класса относятся:

– неоптимальное отношение количества внутренних и граничных ребер (2D) или граней (3D);

– неоптимальное отношение длин внутренних и граничных ребер (2D) и площадей внутренних и граничных граней (3D);

– наличие ячейки с максимальным весом (на счет которой тратится максимальное количество времени);

– наличие ячейки неудовлетворительного качества для алгоритмов поддержания качества сетки.

Выбор критериев динамической балансировки вычислительной нагрузки позволяет улучшить качество декомпозиции, что приводит к эффективной загрузке процессорного поля, выделенного на задачу, и ускорению счета.

5. Тестовые расчеты с использованием методов балансировки вычислительной нагрузки

Применение методов балансировки вычислительной нагрузки тестировалось на двумерных и трехмерных задачах.

Задача 1. Рассмотрим три тестовых расчета трехмерной задачи, содержащей две математические области, с искусственно заданной разбалансировкой: 14, 19, 27 %. Задача состоит из 10050 ячеек. Задача считалась до момента времени 0,1 на 6 процессах. В результате применения методов балансировки разбалансированность по вычислительной нагрузке в конце каждого расчета составила около 10 %.

На рис. 6 показана начальная декомпозиция сетки. В табл. 2 приведено ускорение счета задачи при использовании методов балансировки вычислительной нагрузки. Небольшое ускорение счета задачи во втором и третьем тестах обусловлено большим объемом перебрасываемых ячеек. Расчет задачи показывает действенность данных алгоритмов.

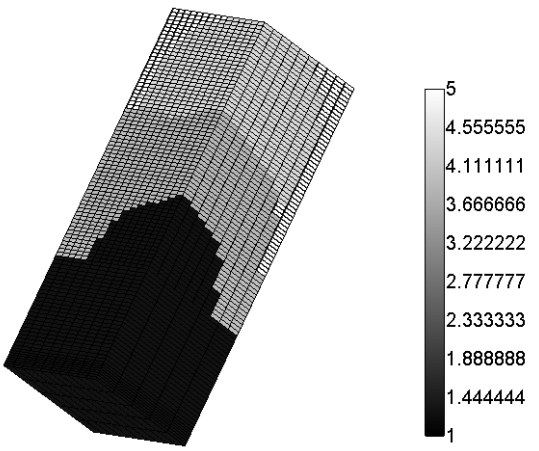


Рис. 6. Начальная декомпозиции сетки для задачи 1

Таблица 2

Ускорение счета задачи 1, полученное с использованием методов балансировки вычислительной нагрузки

Номер расчета	Разбалансированность по процессам, %	Время счета задачи		Ускорение, %
		без методов балансировки	с использованием методов балансировки	
1	14	136,5	117,0	14
2	19	138	126	8,7
3	27	140	127	9

Задача 2. Применение методов балансировки вычислительной нагрузки тестировалось на модельной трехмерной задаче о плоской волне [13], содержащей 1 млн ячеек, при расчете на различном количестве вычислительных узлов. Для тестирования была использована шестигранная сетка. Задача считалась до момента времени $t = 400$ с с использованием квазидинамической и динамической балансировок вычислительной нагрузки с различным набором критериев. Ускорение счета задачи и уменьшение разбалансированности по процессам на конечный момент времени, полученные при расчете с использованием методов балансировки вычислительной нагрузки на различном количестве узлов, представлены в табл. 3.

Таблица 3

Ускорение счета задачи 2 и уменьшение разбалансированности по процессам

Количество узлов	Без методов балансировки		С использованием методов балансировки		Уменьшение разбалансированности по процессам, %	Ускорение счета, %
	дисбаланс, %	время счета задачи	дисбаланс, %	время счета задачи		
10	16	76,953	7	70,791	9	8
20	17	59,854	11	51,299	6	14,3
30	22	51,036	19	47,384	3	7,8
50	24	50,112	21	47,854	3	4,6
100	33	55,838	22	55,648	10	0,3

Как видно из результатов, в ряде случаев небольшое уменьшение разбалансированности приводит к более высокому росту скорости счета. Это объясняется тем, что в результате выполнения алгоритмов балансировки было потрачено меньше времени на вспомогательные алгоритмы.

При расчете на 100 узлах выигрыш небольшой из-за того, что алгоритмы балансировки потребовали большого объема операций.

Задача 3. Были проведены тестовые расчеты трехмерной задачи, которая содержит пять математических областей, количество ячеек и параобластей в каждой из которых представлено в табл. 4. Сетка преимущественно шестигранная. Расчет задачи выполнен за 250 шагов с полным аппаратом поддержания качества сетки на 9 вычислительных узлах.

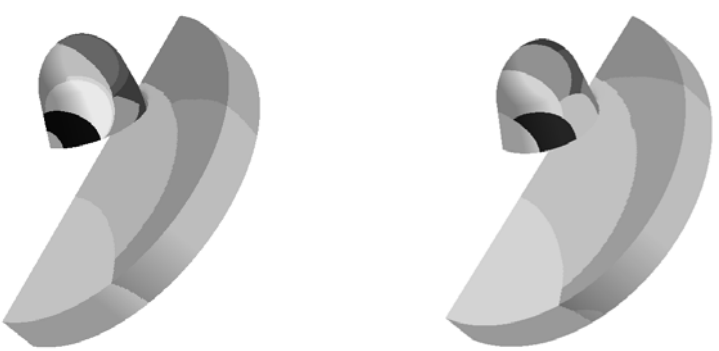
Таблица 4

Количество ячеек по математическим областям в задаче 3

Номер математической области	Количество ячеек	Количество параобластей
1	132505	2
2	61392	3
3	157560	3
4	4648	Параобласти не создавались
5	533830	4
Общее количество ячеек	889935	

При решении данной задачи использовались критерии квазидинамической и динамической балансировок вычислительной нагрузки. При разбалансированности вычислительной нагрузки более 30 % сразу выполнилась передекомпозиция задачи, при разбалансированности больше 10 % и менее 30 % применялись алгоритмы выбора ячеек для переброски на основе следующих критериев: неоптимального отношения по количеству внутренних и внешних граней, неоптимального отношения по площади внутренних и внешних граней, наличия ячейки с максимальным весом, а также ряда критериев, используемых программами поддержания качества сетки. При расчете данной задачи была выполнена 1 квазидинамическая балансировка вычислительной нагрузки и переброшено около 900 ячеек в процессе динамической балансировки.

На рис. 7 приведен пример начальной декомпозиции задачи и результат квазидинамической балансировки вычислительной нагрузки.



а б

Рис. 7. Примеры декомпозиции для задачи 3: а – начальная, б – результат выполнения квазидинамической балансировки (передекомпозиция в процессе счета)

В результате применения всех методов балансировки вычислительной нагрузки задача была рассчитана на 20 % быстрее.

Также были проведены расчеты 1000 шагов данной задачи на 10, 30 и 50 вычислительных узлах по 16 процессорных ядер в каждом с использованием полного аппарата поддержания качества сетки. Ускорение счета задачи с использованием методов балансировки вычислительной нагрузки на различном количестве узлов представлено в табл. 5.

Таблица 5

Ускорение счета задачи 3

Количество вычислительных узлов	Время счета задачи		Ускорение, %
	с использованием алгоритмов балансировки	без использования алгоритмов балансировки	
10	18392	20076	8,3
30	11168	12164	8
50	8119	9324	13

Заключение

Применение методов балансировки вычислительной нагрузки было продемонстрировано на расчете трехмерных многообластных задач.

Применение алгоритмов динамической и квазидинамической балансировок вычислительной нагрузки позволяет более эффективно загрузить процессорное поле, выделенное на задачу, и ускорить счет.

Таким образом, при расчете трехмерных задач получено ускорение до 14,3 %.

В дальнейшем планируется провести исследование на серии сложных расчетов для выбора набора параметров, оптимального для решения широкого класса прикладных задач.

Список литературы

1. Соколов С. С., Панов А. И., Воропинов А. А. и др. Методика ТИМ расчета трехмерных задач механики сплошных сред на неструктурированных многогранных лагранжевых сетках // Вопросы атомной науки и техники. Сер. Матем. моделирование физ. процессов. 2005. Вып 3. С. 37–52.

2. Соколов С. С., Воропинов А. А., Новиков И. Г. и др. Методика ТИМ-2D для расчета задач механики сплошной среды на нерегулярных многоугольных сетках с произвольным количеством связей в узлах // Там же. 2006. Вып. 4. С. 29–43.

3. Воропинов А. А., Соколов С. С. Метод трехуровнего распараллеливания методики ТИМ-2D // Там же. 2013. Вып. 4. С. 70–77.

4. MPI Documents [Electronic resource]. – http://www.mpi-forum.org/docs/docs.html.

5. OpenMP Specifications [Electronic resource]. – http://www.openmp.org/drupal/node/view/8.

6. Якобовский М. В. Введение в параллельные методы решения задач. Глава 8. Динамическая балансировка загрузки процессоров. – М.: Изд-во Московского ун-та, 2013. С. 223–277.

7. Копысов С. П., Новиков А. К., Рычков В. Н. Уровни и алгоритмы динамической балансировки вычислительной нагрузки // Сб. трудов XII Всерос. школы-семинара «Современные проблемы математического моделирования». – Абрау-Дюрсо: Изд-во Южного федер. ун-та, 2006. С. 136–165.

8. Беляев С. П., Дегтяренко Л. И., Турутина И. Ю. Проблемы и возможности разработки параллельных программ с динамической балансировкой вычислительной нагрузки для решения задач механики сплошной среды // Вопросы атомной науки и техники. Сер. Матем. моделирование физ. процессов. 1996. Вып. 4. С. 43–44.

9. Волков К. Н. Балансировка нагрузки процессоров при решении краевых задач механики жидкости и газа сеточными методами // Вычислительные методы и программирование. 2012. Т. 13. С. 107–129.

10. A Polynomial Time Approximation Scheme for Minimum Makespan [Electronic resource]. – http://www.cs.ust.hk/mjg_lib/Classes/COMP572_Fall07/Notes/Min_Makespan.pdf.

11. Karypis G., Kumar V. METIS: a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, version 4.0. Technical report. – Univ. of Minnesota, Dept. of Computer Sci. and Engr., 1998.

12. Pellegrini F., Roman J. SCOTCH: a software package for static mapping by dual recursive bipartitioning of process and architecture graphs. – HPCN-Europe, Springer LNCS 1067, 1996. P. 493–498.

13. Бондаренко Ю. А., Воронин Б. Л., Делов В. И. и др. Описание системы тестов для двумерных газодинамических методик и программ // Вопросы атомной науки и техники. Сер. Матем. моделирование физ. процессов. 1991. Вып. 2. С. 3–14.

Computational Load Balancing Methods in TIM Code

S. S. Sokolov, I. G. Novikov, A. A. Voropinov, T. N. Polovnikova

The paper describes the computational load balancing methods implemented in the TIM code. The set of criteria used for the dynamic and quasi-dynamic balancing is described. The dynamic balancing consists in transferring the calculation of cells from one process to another, the quasi-dynamic balancing consists in constructing a new decomposition and completely re-initializing the problem data with no interrupts of the computing process. The use of the computational load balancing methods allows efficiently loading the field of processors allocated to solve the problem and speeding up the computational process.