УДК 539.4
DOI 10.53403/9785951505071_2022_308

АЛГОРИТМ ДИНАМИЧЕСКОЙ БАЛАНСИРОВКИ НАГРУЗКИ ДЛЯ АДАПТИВНЫХ СЕТОК

Р. В. Муратов

Всероссийский научно-исследовательский институт автоматики им. Н. Л. Духова, Москва

Использование статичных равномерных сеток является нецелесообразным во многих практических задачах. В частности, это справедливо для многомасштабных задач, задач с расчетом кумулятивных процессов и других. Распространенным приемом в таких случаях является использование локально адаптивных сеток. Однако использование сеточной адаптации приводит к новым проблемам при реализации параллельных вычислений. Так, изменение числа ячеек и топологии сетки в течение расчета создает существенный дисбаланс в распределении вычислительной нагрузки между процессами, что не позволяет проводить эффективный параллельный счет. В данной статье предлагается диффузный алгоритм динамической балансировки нагрузки, применимый к адаптивным полярным сеткам. Алгоритм тестируется на модельной задаче, которая имитирует реальные вычисления на адаптивной сетке с различной нагрузкой на ячейки. В серии численных экспериментов демонстрируется масштабируемость метода порядка 70 и 90% для различных модификаций предложенного алгоритма.

Ключевые слова: динамическая балансировка нагрузки, сеточная адаптация.

Введение

В связи с широким распространением высокопроизводительных параллельных вычислительных систем, все более актуальной становится проблема их эффективного использования. Одним из аспектов данной проблемы является эффективное планирование и распределение задач внутри распределенной вычислительной системы с целью оптимального использования вычислительных ресурсов и сокращения времени счета. На практике часто возникают ситуации, когда часть вычислительных ресурсов простаивает, в то время как часть ресурсов перегружена.

Для оптимального использования ресурсов требуется применять методы динамической балансировки нагрузки, которые непосредственно во время счета выравнивают нагрузку между вычислительными потоками. Часто проблемы дисбаланса нагрузки возникают из-за существенной неоднородности исполняемой задачи. В рамках масштабных вычислений с использованием сеточных методов, дисбаланс может быть обусловлен различием вычислительной нагрузки в различных ячейках, к примеру, такая ситуация наблюдается, если в различных ячейках используются коды с различной вычислительной сложностью. Также сильная неоднородность и дисбаланс возникают при сильном изменении топологии сетки в течение вычислений. Такая ситуация наблюдается при использовании сеточной адаптации [1], в этом случае динамическая балансировка нагрузки обязательна.

В данной статье рассматривается задача балансировки нагрузки на иерархической адаптивной сетке. Полная постановка задачи приводится в разделе 2, алгоритм балансировки рассматривается в 3 главе, глава 4 содержит результаты моделирования для тестовой задачи и оценки производительности для предложенного алгоритма балансировки.

Постановка задачи

Рассмотрим двумерную неструктурированную конформную сетку, состоящую из четырехугольных элементов и заполняющую вычислительную область. Под конформной сеткой в данном случае подразумевается сетка, в которой смежные ячейки имеют общую грань, т. е. отсутствуют «висящие» вершины. Данную сетку будем называть базовой, а составляющие ее ячейки — базовыми ячейками. В течение расчета ячейки могут разбиваться на четыре составные части, которые называются дочерними ячейками. Также допускается обратная операция, при которой четыре ячейки, имеющие общую родительскую ячейку, сливаются в одну. Операции адаптации (огрубление и разбиение) могут выполняться как на стадии инициализации начальных данных, так и в ходе расчета. Сетка после проведения процедуры адаптации уже не является конформной, она имеет иерархическую структуру, которую можно описать как лес двумерных деревьев.

Единственное ограничение, которое накладывается на сетку при выполнении адаптации, — необходимость поддерживать баланс 1:2 между соседними ячейками. Другими словами, уровни адаптации смежных ячеек не могут отличаться более чем на единицу. На рис. 1 показаны примеры некорректной (слева) и правильной (справа) сеток. Серым кругом на рис. 1. обозначена область, которая требует максимальной адаптации, а красным пунктиром на некорректной сетке обозначены грани ячеек, на которых не соблюдается баланс 1:2. Ячейки сетки максимального уровня называются листовыми ячейками, они и образуют расчетную область задачи.

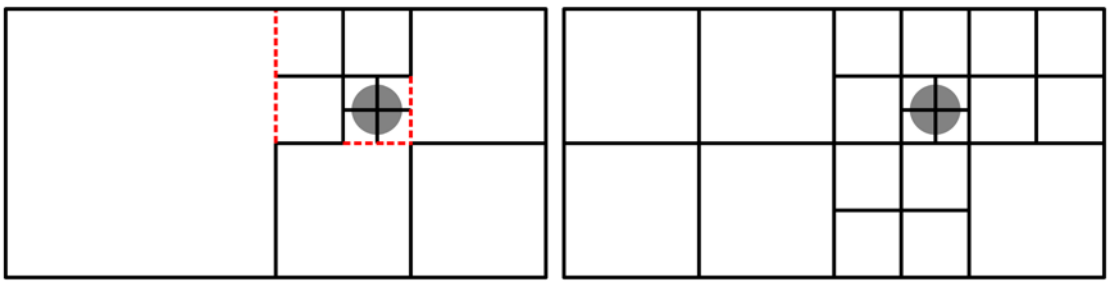

Рис. 1. Примеры правильной (справа) и некорректной (слева) сеток

Для использования возможностей распределенных вычислительных систем, необходимо разделить полученную сетку на несколько частей и распределить эти части между вычислительными потоками. Декомпозиция сетки может быть выполнена один раз в начале вычислений, если проводится расчет на статичной сетке, но такой подход не подходит при работе с адаптивными сетками, поскольку операции добавления и удаления ячеек могут создать существенный дисбаланс нагрузки между процессами. Таким образом, мы приходим к необходимости динамического изменения декомпозиции области в ходе расчета, то есть к необходимости динамической балансировки нагрузки в задачах на адаптивных сетках.

Методы декомпозиции можно разделить на два класса: глобальные и диффузные. Глобальные методы декомпозиции основываются на полной структуре сетки. Одними из лучших методов являются глобальные алгоритмы, основанные на декомпозиции графа [2]. Данные методы могут не только разделить сетку на части с равным числом ячеек, но также минимизировать длину границ между различными подобластями, что позволяет снизить время на коммуникации между процессами. Другим популярным глобальным методом является рекурсивная координатная бисекция [3], для работы алгоритма необходимо знать только центры ячеек, а связность ячеек не играет роли. Глобальные методы декомпозиции хорошо работают для решения задачи статической декомпозиции сетки, но имеют существенные недостатки применительно к задаче динамической балансировки нагрузки. Основной недостаток заключается в том, что глобальные алгоритмы начинают строить декомпозицию каждый раз «с нуля», из-за чего они являются довольно медленными. Также в случаях, когда топология сетки интенсивно меняется в течение расчета, может потребоваться большое число обменных операций: не исключаются случаи, когда в результате новой декомпозиции все ячейки

с одного процесса перемещаются на другой. Также глобальная балансировка не способна учесть сфокусированную нагрузку, что приводит к осцилляциям границ декомпозиции вокруг сфокусированной нагрузки.

Озвученные проблемы не актуальны для диффузных методов балансировки нагрузки. Диффузные алгоритмы основаны на пересылке части нагрузки между процессами в направлении, противоположном градиенту нагрузки (что имитирует закон теплопроводности Фурье) [4]. Применительно к сеточным методам это означает пересылку небольшой доли ячеек между соседними процессами в ходе проведения операции балансировки. Таким образом, декомпозиция не сильно изменяется на этапе балансировки и требуется незначительный обмен ячейками между процессами. Диффузные алгоритмы также позволяют найти хорошо сбалансированную декомпозицию, хотя для этого требуется некоторое число итераций. Интересный вариант диффузного алгоритма рассматривается в статье [5], где в расчетах методом SPH для декомпозиции области используется диаграмма Воронного. Однако в наших приложениях диаграммы Воронного не показали таких хороших результатов, поэтому в данной работе мы предлагаем вариант диффузного алгоритма, который основывается на рекурсивной координатной бисекции.

Описание алгоритма балансировки

Опишем алгоритм на примере двумерной расчетной области, которая имеет форму части кольца

$$K = \{(r, \varphi): R_{in} \le r \le R_{out}, 0 \le \varphi < \alpha\},$$

где внутренний радиус R_{in} может обращаться в ноль. Предположим, что число процессов P является произведением двух чисел $P = P_\varphi \times P_r$. Для разбиения расчетной области мы будем использовать двумерную декомпозицию: P_φ будет числом процессов вдоль угла, а P_r будет числом процессов вдоль радиального направления.

Первоначально область разбивается радиальными линиями под углами Φ_i, где $i = \overline{i, P_\varphi}$ на P_φ частей. Следующим шагом каждый из полученных секторов разбивается на P_r частей дугами, которые имеют радиусы $R_{i,j}$ ($i = \overline{1, P_\varphi}, j = \overline{0, P_r}$). Важно отметить, что набор дуг может отличаться для каждого сектора. Пример подобной декомпозиции приведен на рис. 2. В ходе балансировки радиальные линии и дуги окружностей могут смещаться, что показано стрелками на рисунке.

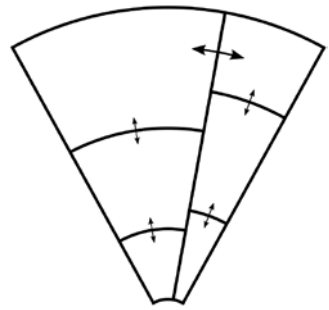

Рис. 2. Декомпозиция на шесть областей

Граничные значения всегда соответствуют границам области

$$\Phi_0 = 0,\ \Phi_{P_\varphi} = \alpha,\ R_{i,0} = R_{in},\ R_{i,P_r} = R_{out},\ i = \overline{1, P_\varphi}.$$

Таким образом, каждый процесс имеет по два индекса $i = \overline{1, P_\varphi}$, $j = \overline{0, P_r}$. Начальное разбиение не имеет значения, поскольку за несколько итераций балансировки алгоритм приводит к сбалансированной декомпозиции, когда число ячеек на каждом процессе примерно равно. Мы рассмотрим также две опции распределения ячеек по процессам. В первом случае предполагается, что листовая ячейка принадлежит некоторому процессу (i, j), если центр соответствующей базовой ячейки расположен внутри области (i, j). Согласно второму методу, мы относим листовую ячейку к процессу (i, j), если она лежит непосредственно в области. Первый способ гарантирует, что все листовые ячейки, имеющие одного родителя, будут расположены на одном процессе. Первый способ будем называть декомпозицией по базовым ячейкам, второй способ – декомпозицией по листовым ячейкам.

Теперь перейдем непосредственно к рассмотрению алгоритма динамической балансировки. Пусть на каждом процессе (i, j) определена нагрузка $w_{i,j}$. Нагрузка может определяться как число

ячеек на процессе, как среднее процессорное время расчета за последние K итераций или другими способами. Далее введем нагрузку на i-ый сектор:

$$W_i = \sum_{j=1}^{P_r} w_{i,j}\,, \qquad i = \overline{1, P_\varphi}.$$

В ходе балансировки сначала рассчитывается смещение радиальных линий по формуле

$$\Delta\Phi_i = D_i^\varphi v_i^\varphi, \qquad v_i^\varphi = \frac{W_{i+1} - W_i}{W_{i+1} + W_i}, \qquad i = \overline{1, P_\varphi - 1}.$$

где v^φ характеризует дисбаланс между смежными секторами и принимает значения от –1 до 1, а параметр D^φ ограничивает движение радиальных линий. В качестве D^φ используется значение в одну десятую ширины смежных секторов

$$D_i^\varphi = 0.1 \max(\Phi_{i+1} - \Phi_i, \Phi_i - \Phi_{i-1})\,, \qquad i = \overline{1, P_\varphi - 1}.$$

Аналогичные формулы используются для смещений дуг внутри секторов

$$\Delta R_{i,j} = D_{i,j}^r v_{i,j}^r, \qquad v_{i,j}^r = \frac{w_{i,j+1} - w_{i,j}}{w_{i,j+1} + w_{i,j}}, \qquad i = \overline{1, P_\varphi}, \qquad j = \overline{1, P_r - 1},$$
$$D_{i,j}^r = 0.1 \max\left(R_{i,j+1} - R_{i,j}, R_{i,j} - R_{i,j-1}\right), \qquad i = \overline{1, P_\varphi}, \qquad j = \overline{1, P_r - 1}.$$

На последнем шаге обновляются значения

$$\Phi_i := \Phi_i + \Delta\Phi_i, \qquad R_{i,j} := R_{i,j} + \Delta R_{i,j}.$$

Аналогичным образом можно производить балансировку в другом порядке: сначала сдвигать дуги окружностей, а затем радиальные линии независимо в каждом кольце. Также алгоритм применим к двумерным прямоугольным областям, в этом случае разбиение производится вдоль координатных осей. Более того, алгоритм может применяться к произвольным областям, для этого необходимо вписать расчетную область в прямоугольник. Также для сложных областей можно выбирать другую систему ортогональных направлений.

Одним из недостатков предложенного алгоритма является необходимость подбора параметров, для которых выполняется соотношение $P = P_\varphi \times P_r$. Однако, технику балансировки можно легко модифицировать и обобщить для возможности разбивать область на произвольное число процессов. Предположим, что помимо общего числа процессов P задается также число столбцов P_x, если значения не кратны, тогда каждый столбец разбивается можно разбить на $\lfloor P/P_x \rfloor$ или $\lfloor P/P_x \rfloor + 1$ частей таким образом, чтобы общее число областей равнялось P (здесь $\lfloor . \rfloor$ означает целую часть).

Пример разбиения прямоугольной сетки на 7 процессов в три столбца приведен на рис. 3. Естественно, теперь необходимо модифицировать формулы для смещения границ столбцов. Изменения коснутся только формулы для расчета нагрузки на столбец.
Новая формула

$$W_i = \frac{1}{P_{y,i}} \sum_{j=1}^{P_{y,i}} w_{i,j}\,, \qquad i = \overline{1, P_x}$$

учитывает также число процессов $P_{y,i}$ в i-ом столбце. Если в каждом столбце содержится одинаковое число процессов, тогда не имеет значения, какую формулу использовать: итоговая формула для смещений одинакова, поскольку множитель $P_{y,i}$ сокращается при вычислении относительного дисбаланса.

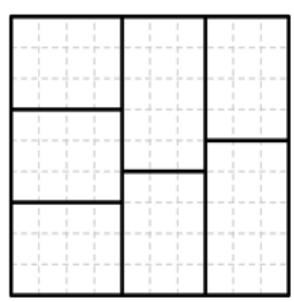

Рис. 3. Разбиение прямоугольной области на 7 процессов

Тестирование алгоритма и производительность

Данный алгоритм балансировки тестируется на модельной задаче. Адаптация и нагрузка вводятся искусственным образом, но за основу взята реальная задача. Базовая сетка представляет собой полярную сетку в полукольце $R_{in} = 10, R_{out} = 10, 0 \le \varphi \le \pi$ с ячейками, приближенными к квадратам, то есть *aspect ratio* ячеек около единицы. Узлы сетки задаются по следующим формулам

$$\varphi_i = \frac{i\alpha}{N_\varphi},\ i = \overline{0, N_\varphi},\ r_j = R_{in}\left(1 + \frac{\alpha}{N_\varphi}\right)^j,\ j = \overline{0, N_r}.$$

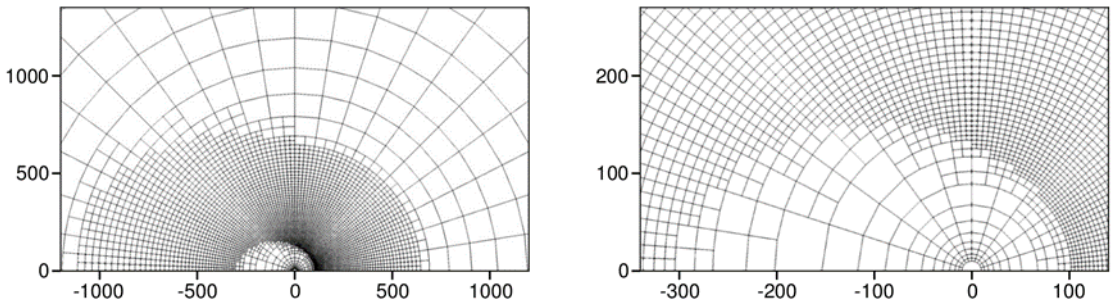

Рис. 4. Центральная часть сетки, разделенная на три области с различными уровнями адаптации. На правом рисунке показано приближение около центра координат

Сторона ячеек у внешней границы примерно в 10^3 раз больше, чем у ячеек в центре области (во столько раз различаются радиусы внутренней и внешней частей). Начальное разрешение по углу $N_\varphi^{base} = 80$, разрешение в радиальном направлении $N_r^{base} = 180$. Предполагается, что расчетная область разделена на три части двумя эллипсами:

$$\left(\frac{x+100}{200}\right)^2 + \left(\frac{y}{150}\right)^2 = 1,\ \left(\frac{x+250}{900}\right)^2 + \left(\frac{y}{700}\right)^2 = 1.$$

Внутренняя часть не адаптируется, внешняя часть имеет первый уровень адаптации, а промежуточный слой имеет третий уровень адаптации. Пример сетки с аналогичным распределением уровней приведен на рис. 4.

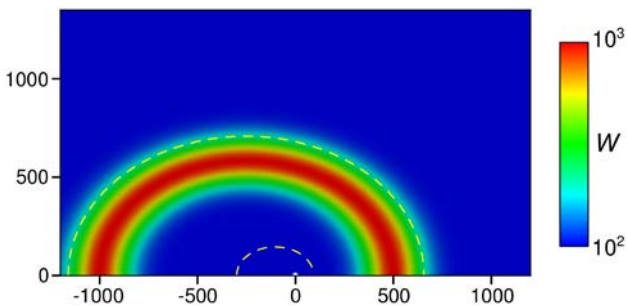

Рис. 5. Распределение нагрузки W в области. Желтой штриховой линией показаны границы области с максимальным уровнем адаптации

После процедуры адаптации сетка содержит около 244 тысяч ячеек, из которых 6 тысяч во внутренней части, 21 тысяча во внешней части, и 217 тысяч или 89 % всех ячеек расположены в промежуточной части с хорошим разрешением. Помимо существенно неоднородной сетки, мы добавляем также искусственную нагрузку к каждой ячейке. Предполагается, что кольцевой слой ячеек, которые лежат на внешней границе кольца с максимальным разрешением, имеет нагрузку в десять раз превышающую нагрузку на остальные ячейки. Используется распределение нагрузки на ячейки, заданное формулой

$$W(x,y) = 10^2 + 10^3 \exp\left(-25\left(\sqrt{\left(\frac{x+250}{750}\right)^2 + \left(\frac{y}{580}\right)^2 - 1}\right)\right).$$

Для моделирования нагрузки каждая ячейка считает функцию синуса в цикле от нуля до целой части $W(x,y)$, где (x,y) являются координатами центра ячейки. Распределение нагрузки показано цветом на рис. 5.

Предполагается, что моделируемый физический процесс протекает достаточно медленно, поэтому значительные изменения топологии сетки и распределения нагрузки занимают множество временных шагов. Таким образом, мы можем использовать статическую сетку и статическую нагрузку для тестирования нашего алгоритма. Нашей целью является показать, что алгоритм позволяет достигнуть устойчивой декомпозиции сетки на процессы за конечное число итераций балансировки, а также показать, что полученное разбиение является хорошо сбалансированным.

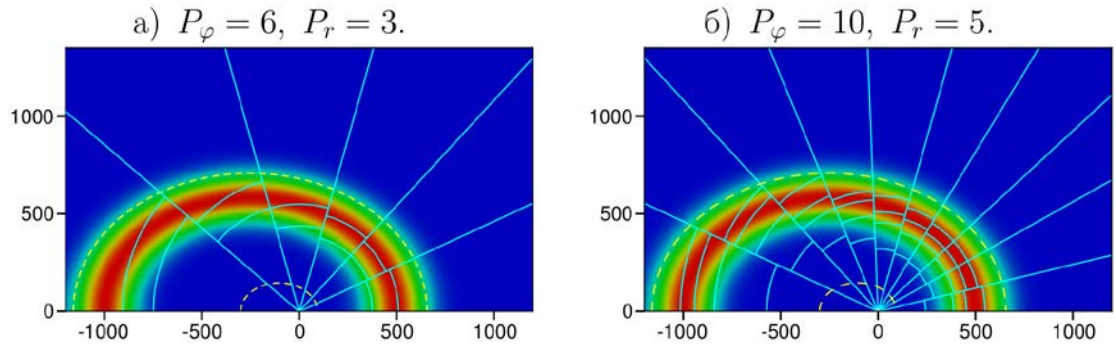

Рис. 6. Равновесное разбиение для двух наборов параметров P_φ и P_r

В начальный момент устанавливается равномерное распределение радиальных линий и дуг

$$\Phi_i = \frac{i\alpha}{P_\varphi},\ R_{i,j} = R_{in} + \frac{j(R_{out}-R_{in})}{N_r}.$$

После этого начинаются итерации балансировки нагрузки. Сначала необходимо уравновесить число ячеек на каждом процессе, для этого алгоритм балансировки нагрузки применяется сотню раз с использованием числа ячеек в качестве нагрузки. После этого начинается модельный расчет, в котором используется искусственная нагрузка. На каждом временном шаге процессы обмениваются фиктивными данными граничных ячеек для моделирования коммуникации между процессами, которая присутствует в реальных задачах. Данные вычисления выполняются на протяжении 10 тысяч итераций с применением алгоритма балансировки каждую двадцатую итерацию.

Десяти тысяч итераций более чем достаточно для получения устойчивого разбиения сетки по процессам. Следующие 10 тысяч шагов используются для сравнения результатов балансировки, по ним рассчитывается среднее процессорное время для выполнения одного шага по времени.

Было проведено две серии экспериментов с различными значениями параметров P_φ и P_r. Используемые параметры приведены в табл. 1. В первой серии экспериментов использовался метод разбиения по базовым ячейкам, в другой серии использовалось разбиение по листовым ячейкам. Устойчивые разбиения для $P_\varphi = 10$, $P_r = 5$ и для $P_\varphi = 6$, $P_r = 3$ показаны на рис. 6. Для других наборов параметров P_φ и P_r распределения аналогичные. Как и ранее, цветом на изображениях показана искусственная нагрузка от 10^2 до 10^3, а границы области с максимальным разрешением показаны желтыми штриховыми линиями. Сплошные цианновые линии обозначают границы между процессами. Представленные распределения рис. 6) соответствуют методу разбиения по базовым ячейкам, разбиение по листовым ячейкам дает визуально неотличимую конфигурацию. Однако такое небольшое различие сильно сказывается на производительности

Таблица 1

Параметры декомпозиции, используемые при тестировании

P	1	2	6	8	9	12	16	18	20	24	28	30	32	36	40	48	50
P_φ	1	2	3	4	3	4	4	6	5	8	7	6	8	9	8	8	10
P_r	1	1	2	2	3	3	4	3	4	3	4	5	4	4	5	6	5

Изменение числа вычислительных узлов при расчете фиксированной задачи означает, что тестируется сильная масштабируемость. Введем метрику для измерения ускорения:

$$S = \frac{T_1}{T_P},$$

где T_P является временем осуществления 10 тысяч итераций на P вычислительных узлах. В общем случае, значение T_P может варьироваться при одном P и различных параметрах декомпозиции P_φ и P_r. Здесь мы пренебрегаем этим и используем исключительно параметры, представленные в табл. 1. Производительность двух методов показана на рис. 7. В обоих случаях масштабируемость близка к линейной. Как и ожидалось, производительность и масштабируемость метода с разбиением листовых ячеек лучше, чем с разбиением по базовым ячейкам. Среднее процессорное время одной итерации около 3 с для единичного узла, при использовании 50 вычислительных узлов среднее время одной итерации равно 0.085 и 0.066 с для метода разбиения по базовым ячейкам и по листовым соответственно. Таким образом, мы получаем масштабируемость методов на 50 узлов 71 и 91 %.

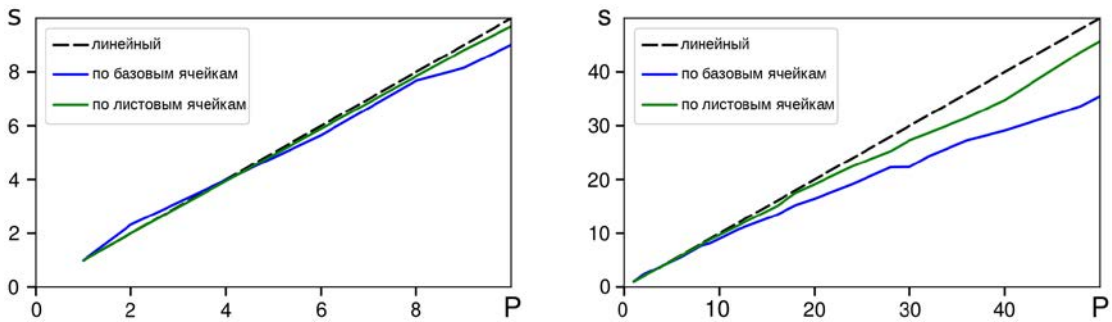

Рис. 7. Графики производительности для двух методов, а также линейный график для сравнения. Рисунок слева демонстрирует масштабируемость при использовании до 10 узлов, рисунок справа – до 50 узлов

Заключение

В статье предлагается метод динамической балансировки нагрузки, разработанный специально для приложений с высоким дисбалансом нагрузки, созданным использованием адаптивных сеток. Предложенный алгоритм показал свою эффективность на модельной задаче, включающей сильную неоднородность сетки и высокую неоднородность нагрузки на узлах. В двух сериях тестов производительности было показано, что на 50 вычислительных узлах достигается ускорение в 36 и 46 раз для различных опций балансировки.

Благодарности

Работа выполнена при финансовой поддержке гранта Российского научного фонда, проект № 21-71-00102.

Литература

1. Lan Z., Taylor V. E., Bryan G. Dynamic load balancing of SAMR applications on distributed systems // Scientific Programming. 2002. Vol. 10, № 2. P. 319–328.

2. Walshaw C., Cross M., Everett M. G. Parallel dynamic graph partitioning for adaptive unstructured meshes // Journal of Parallel and Distributed Computing. 1997. Vol. 47, № 2. P. 102–108.

3. Simon H. D., Teng S.-H. How good is recursive bisection? // SIAM Journal of Scientific Computing. 1997. Vol. 18, № 5. P. 1436–1445.

4. Cybenko G. Dynamic load balancing for distributed memory multiprocessors // Journal of Parallel and Distributed Computing. 1989. Vol. 7. P. 279–301.

5. Egorova M. S., Dyachkov S. A., Parshikov A. N., Zhakhovsky V. V. Parallel SPH modeling using dynamic domain decomposition and load balancing displacement of Voronoi subdomains // Computer Physics Communications. 2019. Vol. 234. P. 112–125.

DYNAMIC LOAD BALANCING ALGORITHM FOR ADAPTIVE MESHES

R. V. Muratov

Dukhov Automatics Research Institute, Moscow

Using of fixed uniform meshes may happen to be impractical for various tasks. For example, it takes place in multiscale problems, simulations of cumulative processes and many other problems. A common solution in these cases is to use an adaptive mesh refinement (AMR) method. However, using of AMR leads to new problem in parallel computation, changing the number of cells creates significant load imbalance between processes and does not allow to work parallel codes efficiently. In this paper we propose a diffusive dynamic load balancing algorithm for an adaptive polar mesh. We test the algorithm on a model problem that simulates calculations on dynamically changing mesh with varying computational work in cells. We consider two modifications of the proposed algorithm and obtain a scalability of application about 70 and 90 % respectively in series of numerical experiments with the model problem.

Key words: dynamic load balancing, adaptive mesh refinement.