УДК 004.09

РАСПАРАЛЛЕЛИВАНИЕ КОНЕЧНО-ОБЪЕМНЫХ ЧИСЛЕННЫХ МЕТОДОВ ДЛЯ СИСТЕМ С ОБЩЕЙ ПАМЯТЬЮ

Гуличева А.А.

МИРЭА - Российский технологический университет, 119454, Россия, г. Москва, проспект Вернадского, 78, e-mail: n-gulicheva@mireaprofkom.ru

Рассматриваются проблемы распараллеливания конечно-объемных численных методов, касающиеся вычисления протекания потоков консервативных величин через границы расчетных ячеек, для систем с общей памятью. Анализируются три подхода для решения данной задачи, для которых проводится теоретическое обоснование, выполняется реализация, тестирование, анализ производительности и масштабируемости предложенных методов. Работа посвящена повышению эффективности работы конечно-объемных численных методов на системах с общей памятью, выполняемой за счет распараллеливания вычислений.

Ключевые слова: конечно-объемные численные методы, распараллеливание вычислений, система с общей памятью, потоки консервативных величин, расчетные ячейки, раскраска графов.

PARALLELIZATION OF FINITE-VOLUME NUMERICAL METHODS FOR SYSTEMS WITH SHARED MEMORY

Gulicheva A.A.

MIREA - Russian Technological University, 119454, Moscow, 78 Vernadskogo Avenue, Russia, e-mail: n-gulicheva@mireaprofkom.ru

The problems of parallelization of finite-volume numerical methods are considered, concerning the calculation of the flow of flows of conservative quantities through the boundaries of computational cells, for systems with shared memory. Three approaches are analyzed for solving this problem, for which a theoretical substantiation is carried out, implementation, testing, analysis of the performance and scalability of the proposed methods are carried out. The work is devoted to increasing the efficiency of finite-volume numerical methods on systems with shared memory, performed by parallelizing calculations.

Keywords: finite volume numerical methods, parallelization of calculations, shared memory systems, fluxes of conserved quantities, calculation grids, graph coloring.

Введение

В настоящее время в мире наблюдается тенденция развития суперкомпьютерных кластерных систем, вычислительные узлы которых состоят из большого количества ядер, и способны производить вычисления на общей памяти с использование огромного числа потоков. Эффективное использование таких вычислительных узлов становится приоритетной задачей. При выполнении вычислений с использованием конечно-объемных численных методов на каждой итерации расчетов требуется проводить две фазы расчетов. На первой фазе внутри каждой расчетной ячейки независимо от остальных ячеек проводится перерасчет локальных физических величин (данная фаза может быть идеально распараллелена в силу независимости производимых расчетов). Во время второй фазы расчетные ячейки обмениваются между собой информацией путем протекания между ними потоков консервативных величин. На данной фазе возможно возникновение конфликтов по доступу к данным расчетных ячеек, и анализ эффективных методов распараллеливания вычислений на данной фазе является предметом настоящей работы.

В настоящей работе основной задачей является реализация распараллеливания вычислений с помощью раскраски графа конфликтов протекания потоков консервативных величин для неструктурированных поверхностных расчетных сеток и для обычных регулярных объемных сеток, ячейки которых являются прямоугольными параллелепипедами.

При параллельном расчете физических величин и протекании их потоков между ячейками на вычислительной машине с общей памятью мы столкнулись со следующей проблемой: при параллельном вычислении протекания потока величины через две границы, инцидентные одной и той же ячейке, наблюдается конфликт по доступу к данным этой ячейки. Поэтому для корректного расчета такие конфликтующие потоки следуют вычислять строго последовательно. Для решения этой проблемы было реализовано несколько способов параллельного вычисления потоков сохраняющихся величин: с помощью OpenMP, с помощью промежуточного сохранения данных на границах ячеек и с помощью раскраски графа конфликтов.

Применение конечно-объемных численных методов для решения задачи

При осуществлении процессов численного моделирования, направленного на решение задач фильтрации и гидродинамики часто возникает необходимость максимально корректно осуществлять учет границ и поверхностей тел, которые могут быть как статическими, так и динамическими. Причем часто в роли примера подобных поверхностей могут выступать крупные трещин в пористых средах, либо свободные поверхности однофазных несжимаемых жидкостей. Для того, чтобы обеспечить учет подобного рода поверхностей при решении подобного рода задач часто использует лангранжевый метод отслеживания, при котором происходит формирование адаптируемой расчетной сетки для осуществления разрешения поверхности. В том случае, когда речь идет про движущиеся поверхности, частым примером которых является свободная поверхность жидкости, то данный подход требует на каждом шаге осуществлять построение новое поверхностной сетки, что приводит к существенному увеличению нагрузки на вычислительную систему. Именно по этой причине было принято решение о рассмотрение основных использования конечно-объемных методов для математических моделей фильтрации и течения неньютоновских несжимаемых жидкостей со свободной поверхностью, позволяющих эффективно учитывать вложенные в сетку границы и получать физически корректные дискретные решения [1].

Практическая значимость выполняемого исследования заключается в рассмотрение методов организации параллельных вычислений для конечно-объемных численных методов в рамках решения практических задач по расчету процессов протекания тонкой пленки по поверхности обтекаемого тела.

Обычно для численного решения задач формулируется математическая модель решения в виде интегральных и дифференциальных уравнений на некоторой многомерной области расчетов. Переход от данной постановки задачи к дискретной математической модели осуществляется заменой функций непрерывного аргумента функциями дискретного аргумента. В результате модель превращается в систему конечноразностных уравнений. Получившаяся модель представляет собой систему алгебраических уравнений, для решения которой с определённой точностью составляется вычислительный алгоритм [6].

Чаще всего все численные методы реализуются через итерационные процедуры, которые выполняются до тех пор, пока не достигнута заданная точность решения. Такой алгоритм решения задачи подразумевает под собой разбиение всего процесса на определенное количество простых шагов (этапов), выполняемых поочередно.

Другими словами, дискретность – набор действий, имеющих строго определенную, предписанную им алгоритмом последовательность. Каждое следующее действие может быть исполнено только при полном завершении предыдущего этапа [5].

В основе решения практических задач, описанных ранее, часто лежит метод конечных элементов, который представляет собой численный метод решения дифференциальных уравнений с частными производными, а также интегральных уравнений, возникающих при решении задач прикладной физики. Метод широко используется для решения задач механики твёрдого деформируемого тела, теплообмена, гидродинамики, электродинамики и топологической оптимизации.

Суть работы данного метода можно наблюдать в его наименовании – для области, в рамках которой будет производиться решение дифференциальных уравнений, производится её разбиение на конечное число подобластей, которые также называют составными элементами области. В каждом элементе произвольным образом осуществляется выбор вида аппроксимирующей функции – в качестве наиболее простого варианта выступает полином первой степени. Вне данного элемента выбранная аппроксимирующая функция равняется нулю. Те значения функции, которые были получены на границах элементов будут являться решением задачи. Поиск коэффициентов аппроксимирующих функций производится на основании условий равенства значений соседних функций на границах элементов. Далее происходит выражение коэффициентов с использованием значений функций в узлах элементов. Следующим шагом происходит составление системы линейных алгебраических уравнений, число которых прямо пропорционально количеству неизвестных значений в узлах, в которых происходит поиск решения исходных систем, в количестве, прямо пропорциональном числу элементов. Данное количество часто ограничивается только вычислительной мощностью, используемой для расчетов вычислительной системы.

С точки зрения вычислительной математики, суть метода конечных элементов заключается в минимизации функционала вариационной задачи посредством совокупности функций, каждая из которых определяется только в рамках собственной подобласти [6].

Методика стала активно использоваться в различных областях – при проектировании зданий и сооружений, процессов движения поверхностей, в частности активно – в гидродинамике. На рисунке 1 представлены учитываемые в модели явления тепло- и массообмена.

При реализации вычислений важной особенностью является тот факт, что выражение пары уравнений частных производных при решении данной задачи происходит их выражение на в двумерных поверхностях, которые попросту встраивают в трехмерные. По результату получается, что первые производных для данных поверхностей вычисляются вдоль данных поверхностей. В рамках метода конечного объема происходит использование данной особенности. Основой вычисления выступает ячейка конечного объема, представляющая собой ячейку двумерно поверхностной сетки, а поверхностная сетка формируется как оболочка трехмерной сетки. Перед началом работы по решению определяющих уравнений необходимо выполнить дискретизацию поверхности всех границ, после чего выполнить построение объемной сетки границ поверхности и области течения. Для этого могут быть использованы расчетные сетки двух видов – структурированные и неструктурированные [4].

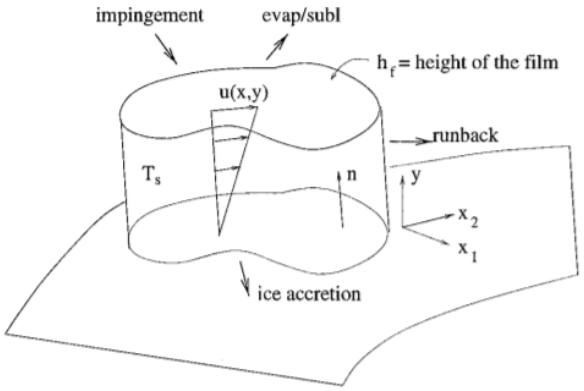

Рисунок 1 – Учитываемые в рамках модели явления тепло- и массообмена

При структурированной сетки для каждой точки сетки происходит однозначная идентификация как её индексами, так и декартовыми координатами. Ячейки данной сетки могут представлять собой четырехугольники в двухмерном представлении, либо шестигранники в трехмерном представлении.

Неструктурированная сетка характеризуется тем, что ячейки сетки, также, как и точки на сетки, не имеют какого-то определенного порядка, что не позволяет однозначно идентифицировать её ячейки. По этой причине составные элементы сетки представляют собой сочетания четырехугольников и треугольников для двумерной сетки, а также сочетания тетраэдров, призм и пирамид для трехмерной сетки. Это выполнено с целью корректного разрешения пограничных слоев.

Генерации сетки с помощью инструментария OpenMP

При генерации структурированной сетки первым этапом осуществляется распределение точек сетки по граничным кривым. За счет этого достигается возможность формирования сетки поверхности, на основании которых может быть выполнено построение объемной сетки.

При генерации неструктурированных сеток чаще всего использовались треугольники и тетраэдры, однако в последнее время большим уровнем популярности становится построение неструктурированных сеток на основании различных типов элементов, включаю сюда шестигранники, призмы и тетраэдры.

Оба варианта построение сеток поверхности обладают своими сильными сторонами. Так, структурированные сетки позволяют легко и быстро получить доступ к любому узлу данной сетки посредством использования индексов. Однако в случае сложных геометрий генерация структурированных сеток будет довольно сложна. В то время как при использовании неструктурированных сеток появляется возможность быстрого построения сложных геометрий, не требуя при этом вмешательства пользователя. В качестве варианта устранения данных недостатков часто используются смешанные сетки, позволяющие добиться объединения данных подходов [1].

Так как процесс построения сеток поверхности представляет собой сложный вычислительный процесс и требует существенных вычислительных мощностей, то с целью ускорения его выполнения требуется распараллеливание вычислений. Для этого применимо к системам с общей памятью часто используют технологию OpenMP. В качестве основы в данном случае будет выступать последовательная программа, которая преобразуется в параллельную структуру посредством специального набора директив, процедур и переменных окружения.

Для распараллеливания программы её текст разбивается на области с параллельным и последовательным типом выполнения, как это представлено на рисунке 2. При запуске программы происходит порождение «основной нити», запускающей процесс исполнения программного кода со стартовой точки. Именно она осуществляет процесс исполнения последовательных областей программы. Параллелизм программы реализуется посредством схемы FORK/JOIN. При входе в параллельную область происходит формирование основной нитью дополнительных нитей с использованием операции FORK. При появлении каждой дополнительной нити её присваивается уникальный номер, а в рамках самой нити происходит выполнение одного и того же программного кода. После того, как параллельная нить завершить свои вычисления, основной нитью осуществляется ожидание завершение вычислений всеми нитями, и далее программа выполняется уже непосредственно в рамках основной нити – происходит выполнение операции JOIN [8].

В рамках параллельной области для переменных в рамках программы производится их разделение на два класса: общие (SHARED) и локальные (PRIVATE) переменные. Для общих переменных характерно их существование только в одном экземпляре для всей программы, а также доступ к ним всегда под одним и тем же именем. Локальные переменные создаются каждая для своей нити, никаким образом, не изменяясь в рамках других нитей.

Реализация максимально эффективной параллельной программы возможна только в том случае, когда все нити будут равномерно загружены полезными вычислениями. Для этого важно осуществить балансировку нагрузки, для чего и применяются различные механизмы OpenMP.

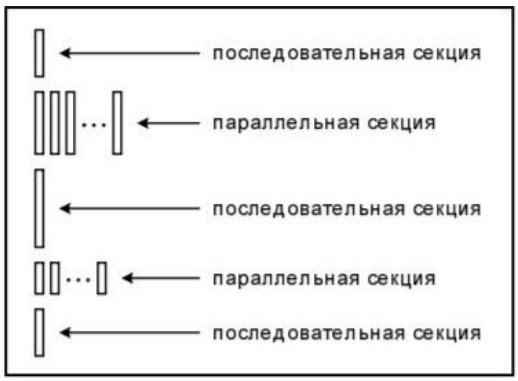

Рисунок 2 – OpenMP. Процесс исполнения программы

Немаловажным моментом является обеспечение синхронизации общих данных, так как с ними будет осуществляться одновременная работа нескольких нитей. По этой причине значительная часть функционала OpenMP направлена как раз на осуществление процессов синхронизации.

Процедуры формирования поверхностных сеток в рамках моделирования физических процессов часто выполняются с применением машин с многопоточной архитектурой. А это говорит о возможности организации вычисления потоков в параллельном режиме. С целью обеспечения данного режима важно обеспечить разрешение конфликтов при осуществлении доступа к данным при их корректировках в результате перетекания потоков веществ. Это возможно реализовать несколькими методиками.

Первый метод подразумевает использованием механизма критических секций OpenMP. Является довольно простой, однако может оказаться критичной для некоторых архитектур.

Второй метод подразумевает сохранение потоков сначала на всех границах, после чего произведение единовременного перерасчета для каждой ячейки. Реализация данного механизма потребует сохранения информации обо всех границах, являющимися для конкретной ячейки входными и выходными, что является довольно весомым недостатком.

Третий метод подразумевает выполнение разбиения границ между расчетными ячейками на подмножества таким образом, чтобы в результате внутри каждого подмножества не было конфликтующих границ. Недостатками данного подхода является необходимость дополнительных действий, связанных с реализацией процедур разбиения границ на подмножества без конфликтов [3].

Распараллеливание вычислений с помощью раскраски графа для неструктурированных поверхностных расчетных сеток

Вводится граф. Вершинами графа являются границы расчетной сетки. Две вершины графа соединены ребром, если соответствующие границы сетки конфликтуют. Далее следует решить задачу о вершинной раскраске получившегося графа.

Из Рисунков 3 и 4 видим, что максимальная степень вершины равна четырем, а это означает, что такой граф можно раскрасить в 4 цвета по теореме 2, планарность это обеспечивает.

Соответственно алгоритм жадной раскраски выдает результат в 5 цветов. Здесь выбирается первая случайная вершина, которой присваивается значение первого цвета, а уже дальше все последующие вершины окрашиваются таким, образом, чтобы цвет не совпадал с уже раскрашенными вершинами, с которыми данная вершина имеет общее ребро. Цвет выбирается из уже использованных цветов, но при невозможности их использования, добавляется новый цвет.

Но мы разработали более оптимальный способ и при его запуске результат получается: 4 цвета.

Теперь опишем алгоритм раскраски в 4 цвета.

1) Так как степень каждой вершины графа не более 4, то можно красить в 5 цветов жадным способом. После этого будем пытаться перекрасить все вершины, покрашенные в цвет 5.

2) Рассмотрим вершину с цветом 5. Если ее нельзя перекрасить, то все ее 4 соседа покрашены в цвета 1, 2, 3, 4 (Рисунок 5).

3) Если можно перекрасить хотя бы одного соседа в цвет, отличный от 5, то саму рассматриваемую вершину после этого можно перекрасить. А значит в наихудшем случае никакого соседа перекрасить также нельзя, значит окрестность рассматриваемой вершины имеет вид, как показано на Рисунке 6.

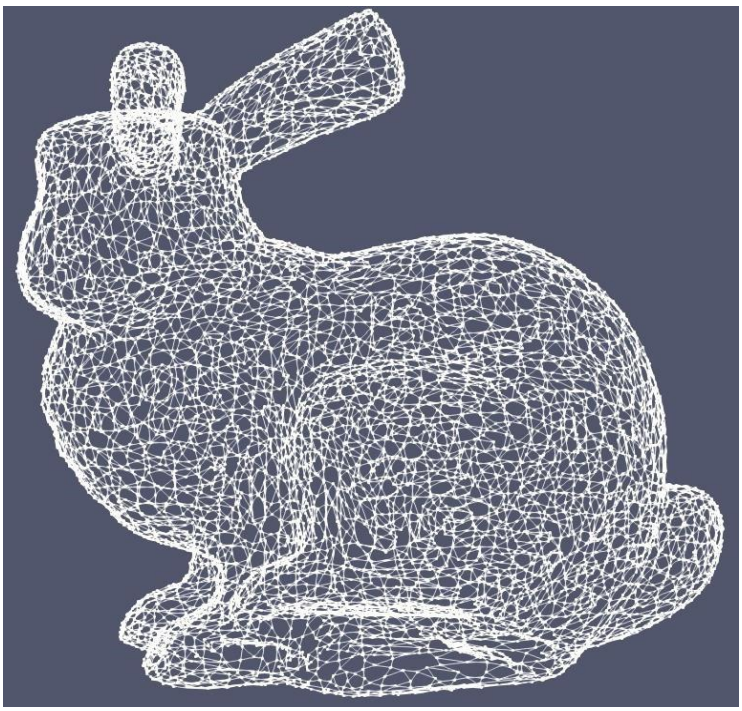

Рисунок 3 – Пример неструктурированной поверхностной сетки «Кролик»

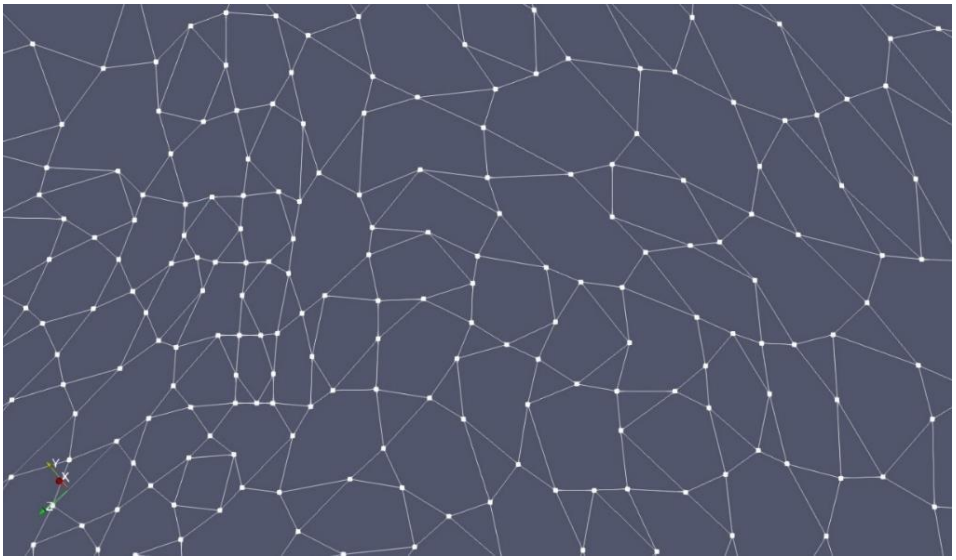

Рисунок 4 – Пример неструктурированной поверхностной сетки «Кролик» вблизи

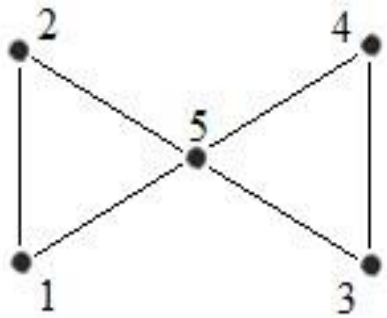

Рисунок 5 – Пример случая раскраски в 5 цветов

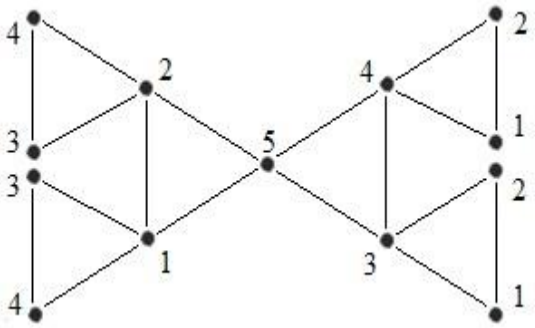

Рисунок 6 – Наихудший случай раскраски в 5 цветов

4) Из внешнего вида окрестности следует, что цвет 5 может мигрировать в любом направлении, например, поменяться местами с 4, как показано на Рисунке 7.

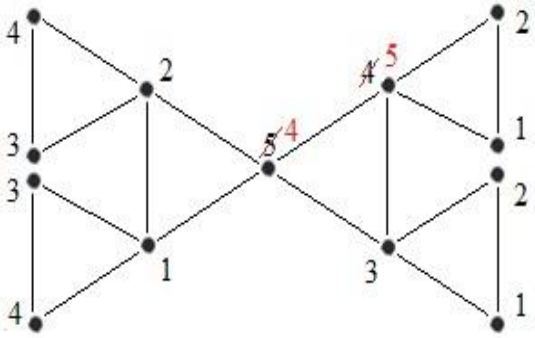

Рисунок 7 – Пример замены цвета

5) С помощью описанного механизма миграции цвета 5 можно отогнать все такие цвета на границу графа (где степень вершин меньше 4) и перекрасить жадным способом. Так как мы рассматриваем наихудший сценарий, то будем считать, что границ у графа нет.

6) Если в графе присутствует более одной вершины цвета 5, то с помощью механизма миграции приблизим их друг к другу в следующую конфигурацию, как показано на Рисунке 8.

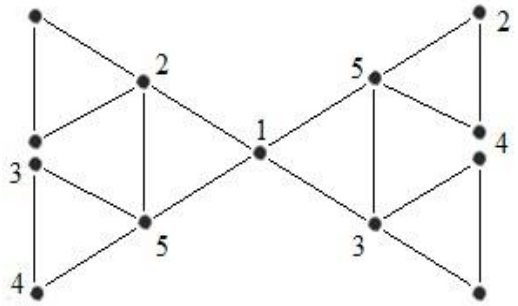

Рисунок 8 – Результат миграции

после приближения двух вершин цвета 5 на расстояние 2 друг к другу, мы может в данном случае перекрасить их в цвет 1, а единичку перекрасить в 5, уменьшив таким образом количество вершин цвета 5. Так как мы рассматриваем наихудший вариант, то будем считать, что у нас в графе осталась одна вершина цвета 5.

7) Предположим, что в графе без границы осталась одна вершина цвета 5, и при любых действиях по миграции данной вершины перекрасить ее не получается. Рассмотрим один из больших циклов, в который входит данная вершина (A), как показано на Рисунке 9.

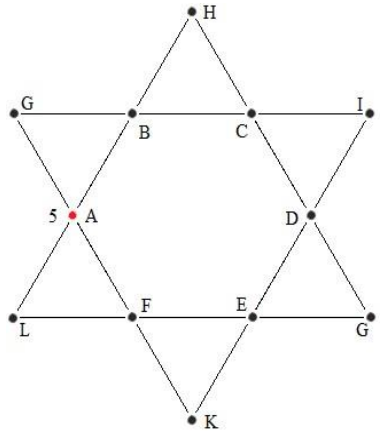

Рисунок 9 – Пример цикла четной длины

Пусть этот цикл четной длины. Для определенности нарисуем цикл длины 6, как показано на рисунке. Так как вершину A перекрасить нельзя, то без ограничения общности будем считать, что вершины G и B покрашены в цвета 1 и 2 (какая из них покрашена в цвет 1, а какая в цвет 2 - не имеет значения), а вершины L и F покрашены в цвета 3 и 4. Мы условились считать, что после миграции цвета 5 в вершину B, эту вершину B перекрасить не удастся. Так как после миграции цвета 5 в вершину B вершины AG окажутся окрашенными в цвета 12, то ребро HC обязано быть окрашенным в цвета 34 (обозначим HC ~ 34).

Аналогичными рассуждениями получим DI ~ 12, EG ~ 34, FK ~ 12. Однако FK ~ 12 противоречит LF ~ 34. А это значит, что в какой-то момент во время миграции цвета 5 по четному циклу мы смогли бы перекрасить вершину. Это можно увидеть на Рисунке 10.

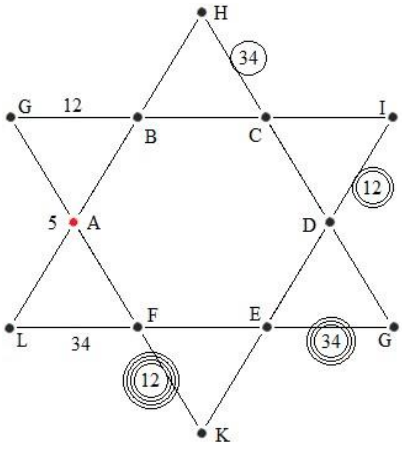

Рисунок 10 – Пример противоречия

Рассмотрим аналогично случай, когда большой цикл нечетной длины. Для определенности нарисуем цикл длины 5, как показано ниже на Рисунке 11.

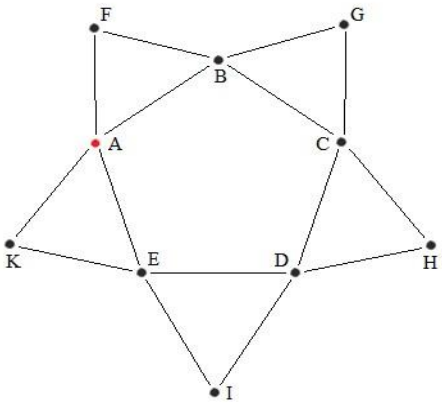

Рисунок 11 – Пример цикла нечетной длины

И снова начинаем рассуждения с того, что так как вершину A перекрасить нельзя, то без ограничения общности будем считать, что вершины F и B покрашены в цвета 1 и 2 (какая из них покрашена в цвет 1, а какая в цвет 2 - не имеет значения), а вершины K и E покрашены в цвета 3 и 4. Мы условились считать, что после миграции цвета 5 в вершину B, эту вершину B перекрасить не удастся. Так как после миграции цвета 5 в вершину B вершины AF окажутся окрашенными в цвета 12, то ребро GC обязано быть окрашенным в цвета 34 (обозначим GC ~ 34). Аналогичными рассуждениями получим DH ~ 12, EI ~ 34, AK ~ 12, что противоречит тому, что KE покрашены в цвета 3 и 4. А это значит, что в какой-то момент во время миграции цвета 5 по нечетному циклу мы смогли бы перекрасить вершину. Результат можно увидеть на Рисунке 12.

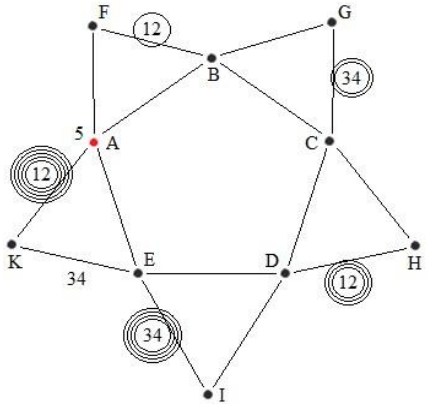

Рисунок 12 – Пример противоречия

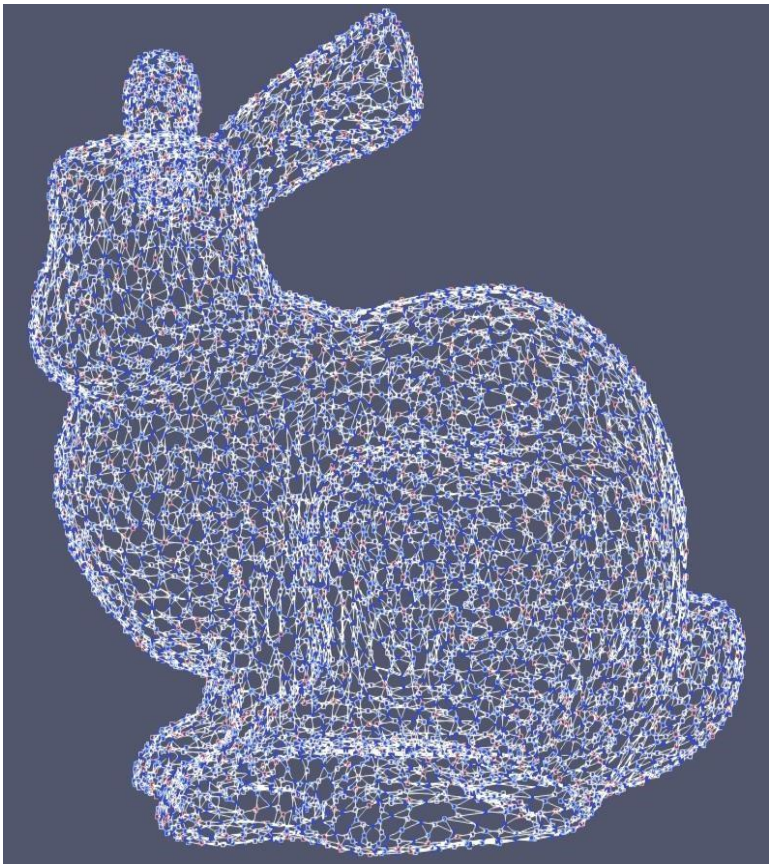

Рисунок 13 – Результат оптимизации жадного алгоритма на графе «Кролик»

Результат описанных выше алгоритмов представлен на Рисунке 13, где можно убедиться, что представленные графы можно разукрасить не в 5 цветов, а в 4. Вершины, которые имеют два цвета и есть те вершины, которые были перекрашены, у них внешний ободок – это цвет который выдал жадный алгоритм, в центральный цвет – это цвет, в который их можно перекрасить для реализации более оптимальной раскраски.

Выполнение запусков на суперкомпьютере и анализ полученных результатов

Для измерения масштабируемости вычислений на неструктурированной поверхностной расчетной сетке использовалась тестовая поверхность обтекаемого трехмерного тела, содержащая около 10^5 узлов и 10^5 ячеек. В ячейках были выполнены расчеты, связанные с моделированием течения пленки жидкости, решением уравнений

теплового баланса на поверхности, а также с перестройкой и сглаживанием поверхности.

Расчёты выполнялись на одном вычислительном узле. Основной целью запусков было измерение сильной масштабируемости вычислений с задействование различного количества потоков внутри вычислительного узла.

То есть для всех запусков использовалась одна и та же поверхность (которая обеспечивалась разным количеством потоков).

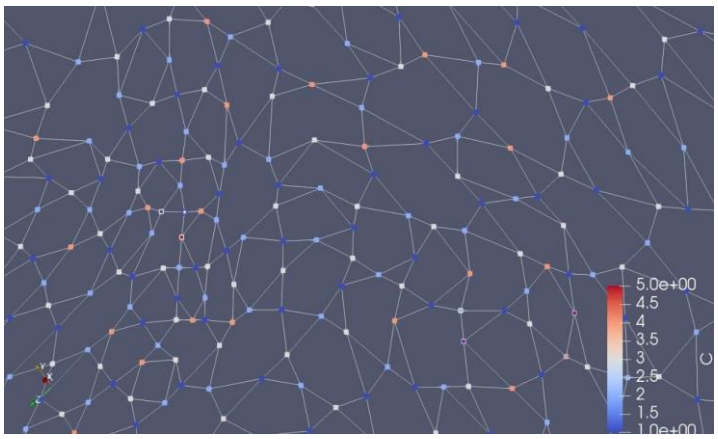

Рисунок 14 – Результат оптимизации жадного алгоритма на графе «Кролик» вблизи

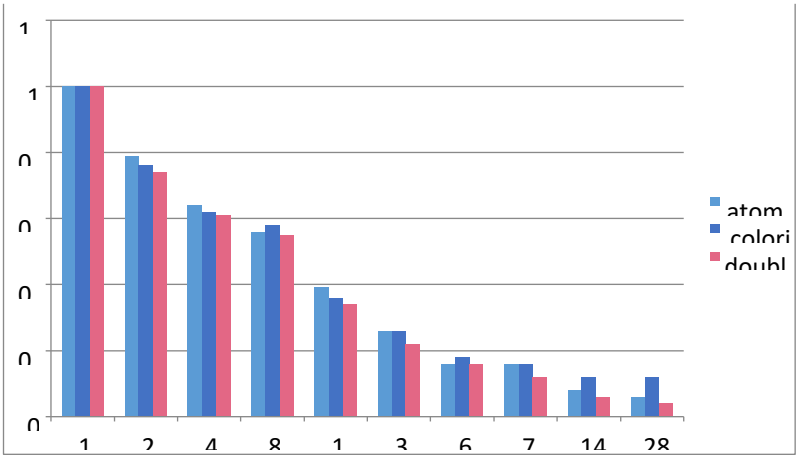

Рисунок 15 – Эффективность масштабирования вычислений на одном вычислительном узле на базе микропроцессора микропроцессора Intel Xeon Phi KNL при увеличении числа потоков.

На Рисунке 15 представлена диаграмма эффективности масштабирования вычислений для различных методов распараллеливания вычислений потоков с увеличением количества потоков в рамках одного узла.

Таблица 1 – Показатели эффективности масштабирования вычислений для различных методов распараллеливания вычислений потоков с увеличением количества используемых вычислительных узлов

Количество потоков	atomic	coloring	double
1	1	1	1
2	0,79	0,76	0,74
4	0,64	0,62	0,61
8	0,56	0,58	0,55
16	0,39	0,36	0,34
32	0,26	0,26	0,22
64	0,16	0,18	0,16
72	0,16	0,16	0,12
144	0,08	0,12	0,06
288	0,06	0,12	0,04

Таким образом можно сделать вывод, что при увеличении числа вычислительных потоков наиболее эффективным является способ разбиения рёбер (границ между расчетными ячейками) на независимые множества (раскраска графа) и их обработка по очереди.

Заключение

В заключении необходимо отметить, что упрощение процесса вычислений на основании конечно-объемных методов при решении задачи формирования адаптируемой расчетной сетки посредством распараллеливания вычислений является обоснованным решением. Важно лишь осуществить грамотное распараллеливание программы, и обеспечение процедур по организации синхронизации данных, с целью обеспечения максимально корректных результатов вычисления.

В результате было предложено три способа параллельной организации вычисления протекания потоков консервативных величин через границы расчетных ячеек: использование блокировок с помощью OpenMP, использование промежуточного сохранения величин потоков на границах между ячейками, разбиение множества границ протекания потоков на неконфликтующие множества с использованием раскраски графов. Проведено теоретическое обоснование, практическая реализация и сравнение эффективности всех предложенных способов.

Список литературы

1. Дж. Мавриплис, С. Пирзаде: Крупномасштабные параллельные неструктурированные расчеты сетки для трехмерного анализа высокого подъема. Документ AIAA 99-0537, 1999.
2. Воробьев, Е.С. Численные методы и математическое моделирование. Основы численных методов и приемы построения математических моделей на их основе и эти решения в различных пакетах / Е.С. Воробьев, В.Е. Воробьева – Учебное пособие – Казань, 2017 г.
3. Бабичев, С. Л. Распределенные системы: учебное пособие для вузов / С. Л. Бабичев, К. А. Коньков. – Москва: Издательство Юрайт, 2022. – 507 с.
4. Корт, Б.: Комбинаторная оптимизация. Теория и алгоритмы, 2015.
5. Куштанова, Г.Г. Математическое моделирование геофизических процессов, Казань – 2015 г.
6. Метод конечных элементов [Электронный ресурс] – Режим доступа: https://clck.ru/CG2mX (Дата обращения 27.12.2022)
7. Рыбаков. А. Внутреннее представление и механизм межпроцессного обмена для блочной сетки для суперкомпьютерных вычислений // Программные продукты и системы. Алгоритмия 8 (1), 121–134 (2017)
8. Рыбаков. А. Распределение вычислительной нагрузки между узлами гетерогенного вычислительного кластера // Прогр. Продукты Сист. Алгоритмы 1, 1–7 (2018).

References

1. J. Mavriplis, S. Pirzadeh: Large-Scale Parallel Unstructured Mesh Computations for High Lift 3D Analysis. AIAA Document 99-0537, 1999.
2. Vorobyov, E.S. Numerical methods and mathematical modeling. Fundamentals of numerical methods and techniques for constructing mathematical models based on them and these solutions in various packages / E.S. Vorobyov, V.E. Vorobieva - Textbook - Kazan, 2017.
3. Babichev, S. L. Distributed systems: textbook for universities / S. L. Babichev, K. A. Konkov. - Moscow: Yurayt Publishing House, 2022. - 507 p.
4. Court, B.: Combinatorial optimization. Theory and Algorithms, 2015.
5. Kushtanova, G.G. Mathematical modeling of geophysical processes, Kazan - 2015
6. Finite element method [Electronic resource] - Access mode: https://clck.ru/CG2mX (Accessed 12/27/2022)
7. Rybakov. A. Internal representation and mechanism of interprocess exchange for a block grid for supercomputing // Software products and systems. Algorithmia 8 (1), 121–134 (2017)
8. Rybakov. A. Distribution of the computational load between the nodes of a heterogeneous computing cluster // Progr. Products Syst. Algorithms 1, 1–7 (2018).