1. Тема работы «Организация и оптимизация суперкомпьютерных вычислений в задаче моделирования обледенения поверхности». Несущие аэродинамические поверхности летательных аппаратов и поверхности управления воздушным судном подвержены воздействию метеорологических условий обледенения, что может привести к недопустимому ухудшению аэродинамических характеристик летательного аппарата. Для проектирования современных летательных аппаратов и обеспечения безопасности полетов необходимо выполнять компьютерное моделирование процесса обледенения.
2. Процесс обледенения является комплексным, он включает в себя расчет аэродинамического поля вокруг обтекаемого тела, расчет движения капель в аэродинамическом поле, расчет количества накопленного льда на поверхности, а также перестроение поверхности в результате накопления льда. Расчет процесса обледенения разбивается на отдельные макроитерации с фиксированной продолжительностью моделируемого времени. В начале каждой макроитерации происходит расчет аэродинамического поля и расчет движения капель в нем. Затем происходит расчет количества накопленного льда на поверхности. На последнем этапе выполняется перестроение поверхности, которое необходимо для расчета аэродинамического поля на следующей макроитерации, так как на него существенно влияет геометрия обтекаемого тела. Расчет аэродинамического поля выполняется на объемной расчетной сетке – мы будем рассматривать использование объемной блочно-структурированной расчетной сетки. Расчет количества накопленного льда выполняется на поверхностной неструктурированной расчетной сетке с треугольными ячейками для возможности описания произвольной поверхности в пространстве.
3. Процесс расчета количества накопленного льда на поверхности – в каждой ячейке поверхностной неструктурированной расчетной сетки – также является комплексным процессом и выполняется с помощью конечно-объемных численных методов, каждая итерация расчетов выполняется с определенным шагом по времени. При этом в каждой ячейке расчетной сетки предусматривается наличие нескольких взаимодействующим друг с другом слоев – элементы системы обогрева, поверхность твердого тела, слой жидкости подо льдом, слой твердого льда, слой мокрого льда, слой текущей по льду жидкости, слой окружающего воздуха – могут выделяться и другие слои. Названные слои взаимодействуют друг с другом посредством переноса потоков вещества, импульса и энергии между ними, а также между соседними ячейками. Расчет количества накопленного льда включает в себя моделирование множества взаимосвязанных процессов и аспектов – выпадение влаги на поверхность, нахождение массового и теплового баланса, теплопроводность через различные слои, расчет течения жидкости, определение шероховатости поверхности, расчет срывов влаги и повторного выпадения и многие другие. Комплексность вычислений, выполняющихся на каждой итерации расчета количества накопленного льда, предъявляет высокие требования к вычислительным ресурсам. Расчет количества накопленного льда на поверхностных сетках, содержащих несколько сотен тысяч ячеек, с характерным временем моделирования в тысячу секунд и с шагом по времени 10 в -4-ой степени секунды уже требует распараллеливания вычислений на несколько высокопроизводительных вычислительных узлов.
4. Актуальность работы. Дефицит суперкомпьютерных ресурсов, расширение сфер применения высокопроизводительных вычислений и усложнение моделей компьютерного моделирования отражает актуальность исследований, направленных на повышение эффективности использования суперкомпьютерных ресурсов. Исследование процесса обледеления имеет важное научно-практическое значение для проектирования летательных аппаратов и обеспечения безопасности полетов, что говорит о необходимости разработки методов и средств для моделирования процесса обледенения. Таким образом, организация и оптимизации суперкомпьютерных вычислений в применении к задаче моделирования обледенения поверхности, проводимых на поверхностных неструктурированных расчетных сетках, описывающих поверхность в трехмерном пространстве, а также на объемных блочно-структурированных расчетных сетках, описывающих область пространства вокруг этой поверхности, обладает актуальностью и практической значимостью.
5. Целью работы является повышение эффективности использования суперкомпьютерных ресурсов в применении к задаче моделирования обледенения поверхности при проведении вычислений на поверхностных неструктурированных расчетных сетках, описывающих поверхность в трехмерном пространстве, а также на объемных блочно-структурированных расчетных сетках, описывающих область пространства вокруг этой поверхности.
6. Для достижения поставленной цели были сформулированы следующие задачи, решаемые в работе. Разработать и реализовать архитектуру поверхностной неструктурированной расчетной сетки с поддержкой декомпозиции для распараллеливания вычислений и перестроения для моделирования эволюции поверхности в процессе ледообразования. Разработать и реализовать архитектуру объемной блочно-структурированной расчетной сетки с поддержкой декомпозиции для распараллеливания вычислений. Разработать и реализовать механизм сопряжения поверхностной неструктурированной расчетной сетки с объемной блочно-структурированной расчетной сеткой для расчета процесса обтекания поверхности в условиях изменения геометрии. Разработать и реализовать методы декомпозиции и распределения вычислительной нагрузки для распараллеливания вычислений на поверхностной неструктурированной расчетной сетке и объемной блочно-структурированной расчетной сетке в модели распараллеливания с передачей сообщений. Разработать и реализовать методы распараллеливания вычислений на поверхностной неструктурированной расчетной сетке и объемной блочно-структурированной расчетной сетке в модели распараллеливания на общей памяти. Разработать и реализовать методы низкоуровневого распараллеливания вычислений с помощью векторизации программного кода.
7. Работа содержит четыре главы. Геометрические методы организации работы с поверхностной неструтурированной расчетной сеткой. Методы распаралеливания вычислений с передачей сообщений. Методы распараллеливания вычислений на общей памяти. Методы векторизации программного кода для повышения его производительности.
8. Описанные в работе методы и алгоритмы апробированы на суперкомпьютерах Межведомственного суперкомпьютерного центра Российской академии наук и Национального центра «Курчатовский институт» и реализованы в рамках инструментов суперкомпьютерного моделирования, на которые оформлены 7 свидетельств о государственной регистрации программы для ЭВМ. В частности, разработанные в рамках диссертации методы перестроения поверхностной неструктурированной расчетной сетки, а также методы повышения эффективности суперкомпьютерных расчетов на ней нашли свое отражение в программном модуле компьютерного моделирования процесса обледенения элементов авиационных силовых установок «Кристалл». Также свидетельства были оформлены: на инструменты подготовки поверхностной неструктурированной расчетной сетки и объемной блочно-структурированной расчетной сетки для суперкомпьютерных вычислений; на библиотеку организации вычислений на поверхностной неструктурированной расчетной сетке; а также на программные коды для векторизации высокопроизводительных вычислений.
9. Основные положения, выносимые на защиту. Предложен метод перестроения поверхностной неструктурированной расчетной сетки с использованием окрестностей ячеек для задачи ледообразования. Предложен метод устранения самопересечений поверхностной неструктурированной расчетной сетки. Предложен алгоритм распределения вычислительной нагрузки при расчетах на объемной блочно-структурированной расчетной сетке с дроблением блоков. Предложен алгоритм сглаживания границ доменов при декомпозиции поверхностной неструктурированной расчетной сетки. Проведен анализ распараллеливания плотных вычислений на общей памяти и обоснован выбор вычислительной системы с точки зрения эффективности распараллеливания. Введено понятие плоского цикла, определены основные его характеристики и обоснован подход к векторизации, основанный на создании программного кода с помощью композиции плоских циклов.
10. Материалы диссертации были представлены на 10 конференциях.
11. Глава 1. Вначале будем рассматривать задачу перестроения поверхностной сетки в двумерном случае, когда каждая ячейка представлена отрезком (слева вверху). Постановка задачи выглядит следующим образом. Пусть в каждой ячейке накоплен известный объем льда. Требуется найти новые положения узлов расчетной сетки, чтобы для каждой ячейки объем, заметаемый ей при движении инцидентных узлов был наиболее близок к объему накопленному в ячейке льда. Задача разбивается на два этапа – нахождение направлений смещений узлов и нахождение величин смещения узлов. Будем рассматривать направления смещения узлов, совпадающие с суммой единичных нормалей к инцидентным ячейками. Это общепринятый подход, называемый также методом биссектрис (так как в этом случае направление движение узла совпадает с биссектрисой угла, образованного двумя инцидентными ячейками). Для решения поставленной задачи будем пользоваться тем фактом, что площадь заметаемая при движении отдельной ячейки может быть найдена непосредственно при известных направлениях и величинах смещения инцидентных вершин (сверху справа).Тогда можно решить поставленную задачу методом градиентного спуска, в процессе которого будем искать величины смещения для каждого узла. Запишем выражение заметаемой площади для каждой ячейки в отдельности. Зная целевое значение количества льда в каждой ячейке, можно определить отклонение, его квадрат и суммарную квадратичную ошибку. Производная суммарной квадратичной ошибки по величине смещения каждого отдельного узла может быть вычислена непосредственно. В общем задача может быть решена методом градиентного спуска, где нахождение градиента наискорейшего убывания ошибки находится из решения системы линейных алгебраических уравнений. Такой метод является слишком трудозатратным, а также имеет склонность застревать в локальных минимумах.
12. Чтобы избежать озвученных проблем используются приближенные методы перестроения поверхности, основанные на приближении количества накопленного льда в ячейке в виде геометрических примитивов. Первый метод – метод прямоугольников, в котором объем накопленного льда в ячейке представлен в виде прямоугольника сторонами которого является ячейка и высота накопленного льда. Величина смещения узла определяется как среднее арифметическое высот льда в инцидентных ячейках. Второй метод – метод трапеций. В нем объем накопленного в ячейке льда представляется в виде трапеции, где ячейка является одним из оснований, а боковые стороны направлены вдоль направления смещения инцидентных вершин (высота трапеции находится из решения квадратного уравнения). После построения трапеций у каждого внутреннего узла сетки появляется два образа – по левой и правой трапеции. Финальный образ определяется как средняя точка между левым и правим образом.
13. Рассмотрим оценку погрешности методов прямоугольников и трапеций. Для этого будем рассматривать модельную задачу. Пусть все ячейки сетки имеет одну и ту же длину l. Пусть также угол наклона каждой следующей ячейки к предыдущей постоянен (если он положителен, то рассматривается выпуклая сетка, если он отрицателен, то вогнутая). Пусть также высота накопленного льда в ячейках изменяется линейно слева направо (на рисунках). В этом случае заметаемую площадь для рассматриваемой ячейки AB при движении ее узлов, можно вычислить явно, через длину ячейки, угол альфа и изменение высоты льда.
14. На рисунке слева представлен график относительного отклонения полученной заметаемой площади от целевого объема льда при постоянной высоте льда в сетке (дельта H равно нулю). Видно, что в описанных условиях метод трапеций является абсолютно точным, что следует из его определения. Точность метода прямоугольников варьируется в зависимости от угла. Рассмотрим одно из значений угла – 0,7 – это значение, при котором относительное отклонение метода прямоугольников положительно и максимально. Для этого значения угла альфа построим зависимость относительного отклонения от относительного изменения высоты льда (дельна H деленное на l). Из построенного графика справа видно, что метод трапеций более точен, чем метод прямоугольников.
15. Перейдем к перестроению поверхностной сетки в трехмерном случае. Для этого будем накладывать на сетку следующие условия. Сетка должна быть целостная (не содержать изолированных и висячих узлов), связная (каждое ребро должно иметь две инцидентные вершины и грани, каждая грань должна иметь три инцидентные вершины и ребра), односторонняя (у каждой ячейке должна быть определена ее внешняя нормаль – направление роста льда), замкнутая (не должно быть краевых ребер). Направление движения узла определяется как и в двумерном случае и совпадает с суммой единичных нормалей инцидентных ячеек. Для нахождения оптимальных величин смещений узлов в трехмерном случае градиентный метод неприменим, так как для выполнения его шага нужно решать уже систему нелинейных уравнений относительно величин смещений отдельных узлов. В трехмерном случае применимы методы представления объема накопленного льда в ячейке в виде геометрических примитивов, аналогично двумерному случаю. Метод призм является аналогом метода прямоугольников, в нем величина смешения узла определяется как среднее арифметическое высот льда во всех инцидентных ячейках. Метод пирамид является аналогом метода трапеций, в нем объем льда аппроксимируется объемом призматоида основанием которого является ячейка, а боковые ребра направлена по направлениям смещения инцидентных ячеек (такая фигура в общем случае не явлется пирамидой, это призматоид, но чтобы не путать это с методом призм, будем называть его методом пирамид). Высота таких призматоидов находится из решения кубических уравнений. Величина смещения узла – это средняя точка среди образом узла относительно всех призматоидов, построенных в инцидентный ячейках.
16. Известны методы повышения точности приближенного перестроения поверхностной сетки. Многослойное перестроение позволяет разделить объем накопленного льда поровну на k шагов и выполнить их последовательно. При этом после каждого шага направления смещения вершин будут меняться, обеспечивая более точное описание объема накопленного льда криволинейными фигурами. Многослойное перестроение с коррекцией предусматривает корректировку оставшегося в ячейке количества льда в соответствии с фактически заметенным объемом при движении ячейки (в этом случае точность всего перестроения совпадает с точностью перестроения на последнем шаге с высотой льда деленной на k). Основной проблемой при перестроении поверхности является потенциальное пересечение направлений смещения вершин, что означает конфликт в моделировании роста льда. С физической точки зрения это нормальная ситуация, но для геометрического перестроения это является проблемой. В работе Tong предложено итерационное сглаживание поля нормалей (направлений роста льда в ячейках и узлах), после которого пересечения направлений роста льда либо предотвращаются, либо откладываются. Также в работе Tong предложен алгоритм сглаживания поверхности, направленного на перераспределение узлов, выполняемое с целью на уменьшение разницы в площадях соседних ячеек.
17. Предлагаемый метод окрестностей перестроения расчетной сетки. Метод основан на принципе Гюйгенса-Френеля распространения волны, который предлагается использовать для распространения фронта роста льда. Пусть мы ходим перестроить поверхность, в которой накоплен объем льда за некоторый промежуток времени дельта t. Пусть мы знаем высоту накопленного льда в каждой ячейке. Тогда, как и в методе призм, будем считать, что высота накопленного льда в узле сетки равна среднему арифметическому высот инцидентных ячеек, но в отличие от метода прямоугольников мы не будем сразу продвигать узел, а будем считать, что фронтом продвижения льда от этого узла является сфера с центром в узле и радиусом, равным величине смещения. Фронтом ячейки продвижения льда будем называть окрестность ячейки, являющуюся выпуклой оболочкой фронтов продвижения льда трех инцидентных вершин (определение окрестности ячейки приведено сверху слева, иллюстрация – сверху справа). Окрестность сетки представляет собой объединение окрестностей всех ее ячеек. Тогда новое положение узла сетки определяется как точка пересечения направления движения узла и границы окрестности сетки. Поиск точек пересечения направления и окрестности ячейки (являющейся пучком сфер) представляет собой параметрическую задачу, кратко записанную на рисунке слева.
18. Основным достоинством предложенного метода является сглаживание дефектов сетки, а именно затягивание впадин и сглаживание острых пиков, представленное на рисунке. Сверху показаны схематическое выравнивание впадины и сглаживание пика в двумерном случае. Внизу показана трехмерная иллюстрация сглаживания дефектов по сравнению с методом пирамид перестроения поверхности (темно-синий цвет – дополнительно нарощенный лед для сглаживания впадин, светло серый – лишний лед на острых пиках).
19. Проведем оценку сглаживания пиков и впадин для двумерного случая по сравнению с методами прямоугольников и трапеций. На рисунке слева представлена модель острого пика (вся сетка является плоской, но пара ячеек образует пик с углом 2 альфа). На рисунке справа представлена модель впадины (вся сетка является плоской, но пара ячеек образует впадину с углом 2 альфа). Для представленных моделей пика и впадины для известных величин смещения узлов hA и hB получены формулы значения сглаженного угла альфа с крышечкой (для пика) и альфа с галочкой (для впадины), а также выведены ограничения на область применения формулы при условии отсутствия конфликтов (самопересечения сетки). Видно, что для случая пика ограничение области применения отсутствует, тогда как для впадины возможно возникновение самопересечений.
20. Проведена оценка сглаживания угла при остром пике для рассмотренных методов прямоугольников, трапеций и окрестностей при одинаковой высоте накопленного льда в каждой ячейке, равной l делить на 4. Метод окрестностей показал наилучший результат (лучшее сглаживание угла). Метод трапеций показал худший результат, особенно при малых значениях угла альфа, приводящих к росту величины hA.
21. Проведена оценка сглаживания угла при впадине для рассмотренных методов прямоугольников, трапеций и окрестностей при одинаковой высоте накопленного льда в каждой ячейке, равной l делить на 4. Можно отметить, что метод прямоугольников вообще демонстрирует отрицательное сглаживание, то есть угол при впадине становится еще острее. Метод трапеций демонстрирует наилучшее сглаживание, однако с более ограниченной областью применения. Заметим, что ограничение области применения для метода трапеций критично, так как при пересечении направлений роста льда невозможно построить трапецию с требуемой заметаемой площадью. Из приведенных оценок можно сделать вывод, что метод окрестностей наиболее пригоден для использования в перестроении поверхности. Точность перестроения этого метода близка к точности метода прямоугольников, но возможности сглаживания дефектов лучше по сравнению с методами прямоугольников и трапеций.
22. До этого момента мы не касались самой важной проблемы, возникающей при перестроении поверхности, а именно возникновения самопересечений сетки. Возникновение самопересечения это серьезный дефект, препятствующий дальнейшему перестроения, поэтому самопересечения должны быть обработаны специальным образом. Прежде чем переходит рассмотрению самопересечений рассмотрим две простые над сеткой, которые при этом понадобятся. Первая операция – измельчение ячейки, которое может быть выполнена по произвольному набору точек внутри ячейки (с помощью триангуляции Делоне) и по серединам сторон. А также сгягивание короткого ребра, с помощью которого можно удалить из сетки слишком близко расположенные друг к другу узлы.
23. Первой фазой устранения самопересечения поверхностной неструктурированной сетки с треугольными ячейками является поиск пар пересекающихся ячеек. Чтобы найти все пары пересекающихся ячеек будем хранить все ячейки в специальной древовидной структуре – двоичное дерево облаков треугольников. Для этого будем использовать понятие охватывающего прямоугольного параллелепипеда, или контейнера множества точек (контейнер – это наименьший прямоугольный параллелепипед со сторонами параллельными осям координат, который содержит все точки). Итак, узлом рассматриваемого дерева является контейнер некоторого множества треугольников. Корнем дерева является контейнер всех треугольников, листом – контейнер одного треугольника. Для каждого нелистового узла два дочерние узла являются контейнерами для множества треугольников родительского узла, которые разделены пополам одной из плоскостей OXY, OXZ, OYZ (если треугольник пересекается плоскостью, то он попадает в оба потомка). Построив такую рекурсивную структуру можно быстро получить следующие данные: по дереву A и дереву B найти множество пар контейнеров треугольников, где первый треугольник из пары входит в дерево A, а второй – в дерево B. Если контейнеры деревьев A и B не пересекаются, то такое множество пусто. А если контейнеры деревьев A и B пересекаются, то такое множество формируется как объединение четырех таких же попарных запросов для двух потомком A и двух потомков B. Подав в запрос в качестве деревьев A и B дерево для всех треугольников сетки, получим множество пар пересекающихся контейнеров отдельных треугольников. Так как пересечение контейнеров треугольников еще не означает пересечение самих треугольников (смю рисунок внизу слева), то для такой потенциальной пары треугольников нужно провести полный анализ на пересечение. Два треугольника пересекаются, если сторона одного треугольника пересекает второй треугольник или наоборот. Таким образом для определения факта пересечения двух треугольников необходимо решить 6 экземпляров задачи о поиске точки пересечения прямой с плоскостью при заданных линеных ограничениях. Результатом пересечения двух треугольников может быть пустое множество, либо выпуклая плоская фигура с количеством точек от 1 до 6.
24. После того как найдены все пары пересекающихся треугольников и все точки пересечения, выполняется классификация ячеек. Ячейки, которые ни с кем не пересекаются, и должны остаться в сетке – статические ячейки. Ячейки – которые ни с кем не пересекаются, но должны быть удалены – скрытые ячейки (внутренность впадины, которая заросла в процессе наращивания льда). Ячейки пересечения – все остальные. Первым шагом является дробление ячеек по всем точкам пересечения (может быть выполнено с помощью триангуляции Делоне, но с фиксацией некоторых ребер). Во время дробления осуществляется локальная коррекция – если оказывается, что точка одного треугольника лежит в плоскости другого треугольника, то эта точка смещается в направлении нормали этого треугольника, внутри которого она лежит. После такой коррекции два треугольника могут пересекаться только по отрезку. После выполнения дроблений треугольников по точкам пересечения сетка приобретает новое качество, в ней появляются ребра, имеющие 4 инцидентные ячейки – из них две должны быть удалены и две остаться. Для удаления лишних ячеек выполняется обход сетки начиная с любой статической ячейки. Встретив ребро с четырьмя инцидентными ячейками, для дальнейшего обхода выбирается ячейка расположенная в том же полупространстве, в которое направлена нормаль текущей ячейки. После завершения обхода все неучтенные ячейки могут быть удалены.
25. Предложенный алгоритм устранения самопересечений продемонстрировал свою пригодность для удаления заросших впадин на различных сетках в процессе тестирования моделирования процесса ледообразования в составе программного модуля Кристалл.
26. В начале каждой макроитерации процесса моделирования ледообразования необходим пересчет аэродинамического поля вокруг тела со сложной геометрией. Одним из подходов, применяемых при тестировании программного модуля Кристалл стало использование декартовой структурированной расчетной сетки для газодинамического решателя и метода погруженной границы для аппроксимации граничных условий вблизи сложной границы. Для этого были проанализированы работы Tseng, Ferziger, Винников, Ревизников и обобщены на трехмерный случай использования фиктивных ячеек. Основной идеей метода является разделение множества ячеек объемной сетки на типы: обычные ячейки – ячейки, в которых рассчитываются аэродинамические характеристики (целиков лежат вне тела, либо лежат внутри тела малой частью), фиктивные ячейки – большей частью лежат внутри тела. Так как фиктивные ячейки большей частью лежат внутри тела, то для них не рассчитываются газодинамические величины, однако их соседями являются обычные ячейки, поэтому газодинамические величины в них нужны для расчета потоков между ячейками. Выходом является аппроксимация газодинамических величин фиктивных ячеек через данные близлежащих обычных ячеек и некоторую точку поверхности тела (с помощью которой задается условие непротекания). Определенные таким образом газодинамические величины фиктивных ячеек в дальнейшем на этой итерации расчетов участвуют в определении газодинамических величин обычных ячеек. На рисунке справа показана различная конфигурация обычных и фиктивных ячеек при формировании расчетной сетке разной грубости вокруг тела со сложной геометрией.
27. Глава 2. Распараллеливание вычислений с передачей сообщений. При использовании нескольких вычислительных узлов в проведении расчетов область данных задачи должна быть разбита на подобласти, обрабатываемые разными узлами, а на границе этих подобластей должна осуществляться синхронизация данных с помощью обмена сообщениями. Ускорение и эффективность распараллеливания при этом определяется естественным образом. Будем считать, что один шаг обработки данных чередуется с одним шагом выполнения информационных обменов. Для вычислений, выполняемых на расчетных сетках, распределение вычислений между вычислительными узлами связано с декомпозицией расчетной сетки. Для декомпозиции расчетной сетки будем использовать следующие показатели качества. Неравномерность распределения – отклонение от среднего (оптимального значения). Чем выше показатель неравномерности, тем неэффективнее проведение вычислений, так как скорость обработки всей области будет определяться скоростью обработки наиболее крупного домена. Максимальная длина границы между доменами – в случае одновременного выполнения информационных обменов на границах между парами доменов длина максимальной границы также определяет скорость обменов на одной итерации расчетов. Дополнительный параметр качества декомпозиции – суммарная длина границ между доменами – определяет суммарный объем данных, пересылаемый во время информационных обменов.
28. Рассмотрим блочно-структурированную расчетную сетку, используемую для расчета аэродинамического поля вокруг поверхности. Блоки сетки являются трехмерными массивами ячеек, доступ к которым осуществляется по индексам. Блоки могут касаться друг друга, область касания называется интерфейсом, интерфейс представлен двумерным массивом граней ячеек. Если блоки обрабатываются в разных процессах, то информационные обмены осуществляются через интерфейс. Ячейки блока разделяются на внутренние ячейки (вся вычислительная окрестность, или вычислительная молекула которых) находится в том же блоке, и граничные. Процесс вычислений внутри блока может быть разбит таким образом, чтобы скрыть информационные обмена за полезными вычислениями – сначала обрабатываются граничные ячейки, затем запускаются асинхронные обмены и вместе с ними вычисления во внутренних ячейках.
29. Задача распределения блоков блочно структурированной расчетной сетки между вычислителями гетерогенного вычислительного кластера может быть сформулирована в общем виде. Если H – граф вычислительной системы, f – функция скорость обработки данных на вычислителе, l – скорость передачи данных по каналу связи. G – граф задачи, w – количество данных в узле, i – количество пересылаемых данных между частями задачи, время одной итерации расчетов, одной итерации межпроцессных обменов могут быть выражены представленными формулами. Нижняя формула – суммарное время итерации при постоянных значениях f и l.
30. Жадный алгоритм распределения нагрузки между узлами вычислительного кластера (распределение вершин графа задачи на вершина графа вычислительной системы) может быть сформулирован следующим образом. Пусть выбраны вершины графа задачи (инициирующие вершины), образующие начальные партиции, распределенные на вычислители. На каждом шаге распределения вычислительной нагрузки выбирается наименее загруженная партиция. К выбранной партиции из ее окрестности добавляется вершина графа G с максимальным показателем q. Первое слагаемое означает время обработки данных из v на вычислителе текущей партиции. Второе слагаемое означает оценку потенциального поглощения межпроцессных обменов (эти межпроцессные обмены пропадают, если вершина v попадает в партицию). Для описанного жадного алгоритма был смоделирован процесс распределения вычислительной нагрузки для гомогенного и гетерогенного вычислительного кластера. По аналогии с показателем t-max был рассчитан показатель t-min, относительная разница которых представлена на графике. Можно заметить, что равномерность распределение вычислительной нагрузки для гетерогенного кластера сильно хуже, чем для гомогенного кластера.
31. Полученные оценки по распределению вычислительной нагрузки с помощью жадного алгоритма показали, что качество распределения оставляет желать лучшего. Для гомогенного вычислительного кластера и в условии пренебрежения межпроцессными обменами можно привести простую оценку неравномерности распределения нагрузки. Она оценивается максимальным значением остаточного члена упорядоченного массива весов узлов. Уже из этой оценки видно, что наличие больших весов, после которых остальные веса сильно меньше приводят к неэффективному распределению. Поэтому для блочно-структурированной расчетной сетки был внедрен механизм автоматического дробления блоков, при котором рекурсивно дробятся все инцидентные объекты сетки (интерфейсы, граничные условия и т.д.).
32. Так как мы может оценить неравномерность распределения вычислительной нагрузки по вычислителям за линейное время (от количества блоков), то можно выполнять дробление максимального блока пополам до тех пор, пока не будет достигнута требуемая точность распределения. На приведенных гистограммах распределения показано, как дробление крупных блоков влияет на равномерность распределения. Исходя из приведенных гистограмм можно отметить, что количество выполняемых разрезов достаточно велико.
33. В качестве альтернативы дроблению максимального блока (UG) предлагается механизм распределения нагрузки с дроблением блоков с минимизацией количества дроблений. Алгоритм может быть сформулирован следующим образом. Определяется значение mid – среднее количество ячеек на процесс, max – текущее максимально допустимое количество ячеек на процесс, изначально равно mid. Алгоритм продолжает работу пока есть нераспределены блоки. Пока из нераспределенных блоков может быть взят блок или его часть и распределены на некоторый процесс так, чтобы суммарное количество ячеек в нем не превосходило max – делаем это. Если так сделать нельзя, то взять блок (или его часть) w и распределить его на процесс W таким образом, чтобы минимизировать значение (W + w) – max. Далее увеличить значение max = W + w.
34. Проведенные эксперименты по сравнению стратегий дробления UG и MCC показали, что с помощью MCC можно добиться показателей неравномерности распределения не хуже чем UG, но с меньшим количеством дроблений блоков (а количество дроблений увеличивает объемы данных, пересылаемых при межпроцессных обменах).
35. Применение алгоритмов распределения вычислительной нагрузки с дроблением блоков показало сверхлинейную масштабируемость вычислений на модельной задаче расчета на входном устройстве воздухозаборника с использованием RANS/ILES метода высокого разрешения.
36. Был проведен эксперимент по подготовке расчетной сетки для выполнения на разном количестве узлов и тестированию эффективности распараллеливания на расчетных кодах RANS/ILES. Дробление выполнялось с помощью алгоритма UG. Целевое количество процессов использовалось 16, 32 и 64. Допустимая неравномерность распределения бралась 10% и 1%. На неподготовленной сетке видим остановку распараллеливания после 5 процессов, что говорит, о том, что вычисления уперлись в крупный блок. При подготовке для 16 процессов распараллеливание остановилось на 16 процессов, что говорит о том же. При подготовке для 32 и 64 процессов результаты масштабирования до 32 процессов практически одинаковые. А вот при снижении порога неравномерности распределения наблюдается резкая просадка производительности. Это связано с большим количеством дроблений и возрастанием объема межпроцессных обменов.
37. Перейдем к методам декомпозиции поверхностной неструктурированной расчетной сетки. Для организации вычислений по определению наколенного льда в ячейках сетки требуется декомпозиция с низкими показателями неравномерности, максимальной границы и суммарной границы. Рассмотрим в сравнении 4 метода: случайное распределение (заведомо плохой метод для сравнения), линейное распределение (может оказаться приемлемым при использовании априорной информации о сетке), наращивание доменов (пузырькового роста), иерархическое деление пополам вдоль наиболее протяженной координаты. Из графиков показателей качества декомпозиции на этих простых алгоритмах можно сделать следующие выводы. Случайный алгоритм плох из-за межпроцессных обменов. Алгорим наращивания доменов плох из-за неудачного выбора инициирующих вершин из-за которых распределение неравномерное, алгоритм линейного распределения проигрывает по длине границы. Проведен более детально анализ алгоритма иерархического дробления. У него хороший показатель равномерности распределения.
38. Если посмотреть поближе на результат работы алгоритма иерархического дробления, то можно увидеть протяженные пилообразные границы, которые очевидно могут быть уменьшены без ущерба показателю равномерности распределения. Рассмотрим как это можно сделать на примере представленных шаблонов сглаживания границ между доменами. Мы может применить один или несколько из приведенных шаблонов сглаживания границы, каждый из которых сокращает длину границы и изменяет баланс по количеству ячеек между соседними доменами. При рассмотрении конкретной границы требуется найти такое множество шаблонов, которые могут быть использованы одновременно, не нарушают баланса ячеек и максимально сокращают длину границы.
39. Рассмотрим алгоритм сглаживания границы между доменами, реализуемый с помощью метода динамического программирования. Пусть имеется граница (на рисунке представлена цепью 0-16). На ней отмечены некоторые шаблоны, соседние шаблоны могут быть независимы (использованы одновременно оба), либо зависимы (тогда из них может быть использован только один). Определим функцию B(t, u, x) – которая означает изменение длины границы при обработке всех шаблонов начиная с t, при балансе ячеек между доменами u, а x означает булевый признак использования шаблона t. Инициализация начинается с последнего шаблона. Он может быть использован или не использован. При обработке k-го шаблона мы всегда можем получить ответ для x = 0 (минимальный показатель для k + 1-го шаблона). Для x = 1 есть два варианта. Если шаблоны k и k+1 не конфликтуют, то k-й шаблон можно использовать вне зависимости от того, был использован шаблон k+1 или нет.. Если конфликт есть, то k-й шаблон исользуем только если не использован k+1-й. В результате получаем алгоритм, графическое изображение которого приведено на рис справа (при этом мы можем найти решение для любого допустимого значения баланса по количеству ячеек между доменами). Приведена оценка сложности алгоритма.
40. На иллюстрациях приведены результаты работа алгоритма сглаживания границ между доменами. Длина границы сокращается примерно на 10 процентов.
41. Отметим один момент по поводу алгоритма пузырькового наращивания доменов. Сам алгоритм является довольно полезным, так как он формирует достаточно гладкие границы в результате использования обхода дуального графа расчетной сетки в ширину. Вариант Фархата наращивания доменов лучше по показателю равномерности распределения, однако границы формирует хуже. Для алгоритма пузырькового роста можно было бы найти лучшее расположение инициирующих вершин, что было сделано с помощью генетического алгоритма. Существуют подходы к декомпозиции графа с помощью генетических алгоритмов. Особью в таких алгоритмах является декомпозитованный граф. Но в известных приведенных алгоритмах генотипом является кодирование той же особи, что противоречит принципу работы генетических алгоритмов, так как генетический алгоритм подразумевает, что генотип является неким кодом – инструкций – для изготовления особи, а не ее точным чертежом. В результате мы сталкиваемся с перегруженным генотипом, и очень долгой работой для достижения какого-то результата, который выглядит в конечном итоге как просто перебор. Предлагается использование в качестве генотипа набора инициирующих вершин для доменов, а получение особи из генотипа – с помощью быстрого алгоритма пузырькового роста. Мутацией является смещение инициирующей вершины по ребру. И хоть предложенный алгоритм все еще проигрывает по показателю равномерности распределения алгоритму иерархического дробления, он позволяет довольно быстро повысить качество декомпозиции с помощью пузырькового роста.
42. Был поставлен эксперимент по масштабированию вычислений на поверхностной неструктурированной расчетной сетке в задаче ледообразования. Распределение вычислительной нагрузки было выполнено с помощью алгоритма иерархической декомпозиции с использованием сглаживания границ между доменами, также использовался механизм скрытия издержек на межпроцессные обмены за полезными вычислениями. Эксперимент проведен на вычислительных системах на базе вычислительных узлов с микропроцессорами Intel различного поколения. Наилучший результат по масштабированию продемонстрировала система на базе микропроцессоров Cascade Lake.
43. Глава 3. Так как современные микропроцессоры являются многоядерными и поддерживают многопоточное исполнение, то естественно желание использовать многопоточные вычисления в суперкомпьютерных приложениях. При выполнении расчетов на объемных блочно-структурированных и поверхностных неструктурированных расчетных сетках обычно происходит обработка объектов одного типа в циклах. Например, обработка всех ячеек расчетной сетки, или обработка всех узлов расчетной сетки. В конечно-объемных методах обработка одной ячейки не зависит от другой и они могут выполняться параллельно. Поэтому обработка в цикле всех ячеек домена может быть легко выполнена в автоматическом режиме. Сложнее дело обстоит когда приходится выполнять пересчет перетекания вещества, импульса, энергии между соседними ячейками (потоки через грань для объемной сетки и через ребро для поверхностной сетки). Одновременная обработка перетекания вещества, импульса, энергии между ячейками может привести к конфликту по данным. Предотвратить такие конфликты можно разными способами. Самым простым является использование директивы openmp, запрещающей одновременный доступ нескольких потоков к одному и тому же коду. Однако такой подход может затормозить исполнение, так как в таких точках будут возникать блокировки потоков, ожидающих выполнения одного потока. Хорошей практикой является разделение потенциально конфликтующих объектов на подмножества без конфликтов и дальнейшая их обработка в параллельном режиме.
44. Рассмотрим такой подход для поверхностной неструктурированной расчетной сетки. Для простоты будем считать, что наша расчетная сетка описывает односвязную поверхность, тогда ее дуальный граф является плоским кубическим графом, а задача разделения множества ребер расчетной сетки на подмножества без конфликтов сводится к задаче реберной раскраски этого дуального графа. Проанализируем, в какое количество цветов может быть раскрашен такой граф. Задача раскраски плоского кубического графа в 5 цветов является тривиальной и выполняется за один проход по ребрам. Раскраска в 4 цвета также возможна согласно теореме Визинга, однако теорема визинга не использует свойство планарности и сложность такой раскраски выше линейной. Приведем краткое доказательство возможности раскраски в 4 цвета. Доказательство можно провести по индукции по количеству вершин графа. Главным местом является индуктивный переход. Нетрудно видеть, что в нашем планарном графе всегда найдется грань размера 3, 4 или 5. Для каждого из этих случаев можно выполнить стягивание графа по этой грани с получением графа меньшего порядка, для которго требуемая раскраска существует по предположению индукции. Тогда по этой раскраске можно явно построить раскраску исходного графа, совершить тем самым индуктивный переход и получив линейный алгоритм раскраски в 4 цвета.
45. Раскраска в 3 цвета является гораздо более серьезной проблемой. Такая раскраска называется раскраской Тейта и она возможна для плоских кубических графов без мостов в силу теоремы о четырех красках. Доказательство аналогичное раскраске в 4 цвета не проходит для стягивания грани размера 5. Для построения раскраски Тейта был использован цикл работ Курапова, Давидовского, Толока, в которых описан алгоритма построения раскраски Тейта, реконструкцию которого я приведу без доказательства. Алгоритм заключается в последовательном редуцировании графа по ребру, где на каждом этапе редуцирования получается граф меньшего порядка.
46. После получения тривиального кубического графа происходит процесс восстановления графа с корректировкой его раскраски. Корректировка раскраски графа выполняется с помощью базовых операций – перекраски двухцветного цикла и добавления нового ребра на двухцветный цикл. Для доказательства корректности работы алгоритма необходимо утверждение, что при восстановлении редуцированного ребра на каждом шаге алгоритма в графе найдется двухцветный цикл, разбиваемый этим ребром. Сложность описанного алгоритма в худшем случае является квадратичной, однако на небольших сетках он вполне применим.
47. Был проведен эксперимент по сравнению двух подходов к устранению зависимостей по данным с помощью директивы openmp и реберной раскраски. Эксперимент проводится на микропроцессоре Intel Xeon Phi KNL, допускающем выполнение до 288 потоков. В эксперименте замерялось время исключительно перетекания вещества и энергии через ребра расчетной сетки, однако видно, что на большом количестве потоков реберная раскраска демонстрирует значительное улучшение по сравнению с использованием директивы openmp. В заключение можно отметить, что количество цветов, в которые выполняется раскраска дуального графа не сказывается на производительности, так как для простоты можно использовать тривиальную раскраску в 5 цветов.
48. Также был поставлен эксперимент по выбору стратегии распараллеливания вычислений на Intel Xeon Phi KNL с помощью openmp. Из трех вариантов распараллеливания на общей памяти (разделение массива данных на последовательные участки, разделение на участки в шахматном порядке, динамическое выделение следующей порции данных для обработки) наиболее эффективным вариантом оказался вариант распределения данных в шахматном порядке. При нем было достигнуто ускорение более чем в 50 раз при расчете на примерно 120 потоках при решении задачи газовой динамики на объемной блочно-структурированной сетке.
49. Для выбранной стратегии распараллеливания вычислений было проведено сравнение распараллеливания газодинамического решателя (с использованием метода годунова) вычислительных узла на базе микропроцессоров линейки Intel. Сравнение проводилось для скалярной и векторизованной векрсий газодинамического решателя. Для обеих версий программного кода наилучшую эффективности распараллеливания продемонстрировал микропроцессор Intel Xeon Phi.
50. Глава 4. Последним уровнем распараллеливания, рассматриваемым в работе является векторизация программного кода. Векторизация допускает кратное увеличение производительности при правильном использовании. Векторизация представлена во всех современных архитектурах, включая x86, ARM, Power, «Эльбрус», LoongArch, Sunway и других. Наиболее перспективный набор векторных инструкций – AVX-512 – поддержка выборочной обработки элементов векторов с помощью векторных масок, поддержка множественного обращения в память с произвольными смещениями, поддержка вычисления экспоненты и обратных значений, поддержка комбинированных операций и другие особенности. Будем использовать понятия ширины векторизации, ускорения от векторизации и эффективности векторизации для сравнения векторизованного кода.
51. Исходя из смысла векторизации можно констатировать, что для успешного выполнения векторизации должны быть выделены однотипные операции, которые могут быть объединены в векторный код. Рассмотрим пример операций по умножению матрицы на вектор и перемножения матриц. Был поставлен эксперимент по выделению однотипных операций в этом коде. По сути обе этих операции представляют собой выполнение большого количества операций скалярного произведения пар векторов. И если для поэлементного перемножения двух векторов существует конкретная операция, то операции суммирования элементов должны быть оптимизированы дополнительно.
52. Для массового вычисления сумм элементов векторов была реализована схема, состоящая из последовательных фаз операций перестановки элементов и суммирования векторов.
53. Также для сравнения была векторизована операция получения обратной матрицы, выполненная по алгоритму Гаусса-Жордана в терминах операций над строками. Достигнутая эффективность векторизации в районе 0,075 тысячных до 0,2, продемонстрировала, что требуются меры борьбы с операциями множественного обращения в память, а также требуется использование комбинированных операций.
54. Были рассмотрены операции перемножения малоразмерных матриц, которые массово используются в реализации газодинамического решателя RANS/ILES (а именно 5 на 5). Причем для обеспечения выровненности в памяти эти матрицы находятся внутри бОльших матриц 8 на 8. Была рассмотрена схема вычисления двух строк результирующей матрицы с использованием комбинированных операций.
55. Это позволило достичь эффективности векторизации в диапазоне 0,15 – 0,4 для разного размера матриц.
56. Сильная зависимость эффективности векторизации от способа выделения однотипных операций привела к мысли, что необходима унификация подхода к векторизации программного кода. Для попытки такой унификации введено понятие плоского цикла, удовлетворяющего следующим требованиям. Это цикл for с индуктивной переменной i, где i меняется от 0 до w – 1 (w – ширина векторизации). На i-ой итерации доступ к данным на запись имеет вид a[i], а доступ к данным на чтение имеет вид a[i] либо чтение скаляра. Все массивы данных выровнены в памяти на размер вектора. Отсутствуют другие межитерационные зависимости. В этом случае итерации плоского цикла могут быть выполнены вместе, а значит и параллельно.
57. Можно заметить, что семантика многих векторных инструкций по сути и является плоским циклом. А векторная маска, используемая в векторной инструкции является векторизованным условием внутри плоского цикла. Верно и обратное, зачастую плоский цикл, содержащий простые вычисления, может быть векторизован автоматически с использованием оптимизирующего компилятора. Основным препятствием к векторизации плоского цикла является наличие сложного управления внутри него.
58. Был поставлен эксперимент по векторизации газодинамического решателя на блочно-структурированной расчетной сетке. Для этого было сделано два преобразования. Данные газодинамических величин вместо схемы хранения «массив структур» были изменены на «набор массивов». После этого в простых функциях обработки данных операции чтения и записи элементов данных преобразовались в чтение и запись векторов, а арифметика заменилась на векторные аналоги. Вторым преобразованием стало расщепление цикла по константному условию, что позволило избавиться от лишних условий внутри плоского цикла.
59. Этот подход был проверен на газодинамическом решателе, с использованием схемы Стегера-Уорминга расщепления потоков, а также с использованием метода погруженной границы для расчета аэродинамического поля. С помощью автоматической векторизации удалось достичь эффективности векторизации до 0,6 на программном коде с простыми вычислениями.
60. Программный код без условий или с константным условием это скорее исключение. Рассмотрим пример избавления от маловероятного кода внутри плоского цикла. Пусть внутри цикла блок block\_true выполняется с вероятностью, близкой к 100%, а блок block\_false – с вероятностью близкой к нулю. Пусть переход на тот или иной блок осуществляется по условию cond. Если цикл плоский, то он может быть расщеплен на три разных цикла с сохранением условия во временную переменную (временный массив). Тогда если блок block\_true содержит относительно простые вычисления, то цикл с получением условия и цикл с блоком block\_true может быть векторизован, а последний цикл – маловероятный и его можно не оптимизировать.
61. В общем случае передача управления в плоском цикле может быть векторизована с помощью записи кода в предикатном виде и последующей векторизацией условий в виде векторной маски. Пусть блок A выполняется с вероятностью p, блок B выполняется с вероятностью 1-p, блок A в альфа раз длиннее B и суммарно их длина равна единице. Тогда можно явно вычислить эффективность векторизации при условии, что сами блоки A и B векторизуются идеально. Из графика видно, что при a = 1 эффективность равна 0,5 и постоянна. При других альфа эффективность тем выше, чем вероятнее наиболее длинный блок. Также из графика можно сделать вывод, что наличие вложенных условий в блоках уменьшает эффективность векторизации экспоненциально, поэтому безусловное слияние путей исполнения по условию – не лучший способ векториазции.
62. Так как векторная маска является по сути целым числом, то перед выполнением маловероятного блока ее можно проверить на пустоту. Это позволяет повысить эффективность векторизации при значениях p близких к 0 или 1.
63. Эффективность векторизации все еще низка если векторная маска содержит пару выставленных биток. Проверка на пустоту в этом случае ничего не дает и векторный код с низкой плотностью векторных масок исполняется, отрицательно влияя на производительность. Рассмотрим подход к векторизации такого кода – объединение векторных масок. Пусть у нас есть два последовательный блока кода, выполняемых под непересекающимися масками. Тогда данные этих блоков можно объединить и выполнить блок один раз, используя объединенную маску. При этом возникают накладные расходу связанные с объединением входных данных и разделением выходных данных, но экономится время на исполнения самого блока кода. Теоретически можно объединять более двух копий блока, однако эксперименты показали, что в этом случае накладные расходы уже превышают потенциальную пользу.
64. Аналогичный подход можно применить к исполнению двух блоков под пересекающимися масками, в случае если суммарное количество битов в них не превышает полную маску. В этом случае над одной или обеими масками сначала нужно выполнить обратимое преобразование, чтобы мы получили снова две непересекающиеся маски, к которым можно применить метод объединения масок. Такой подход выглядит слишком трудоемким и его эффективность не проверялась на практике.
65. Был поставлен эксперимент по векторизации одной из функции газодинамического римановского решателя, содержащей одно условие. При выводе теоретических оценок я полагал, что переход это в определенном смысле случайная величина. Однако при выполнении физических расчетов это не совсем верно. При объединении вычислений в плоские циклы входные данные от итерации к итерации меняются медленно (например, значение температуры при переходе от одной ячейке к другой). А значит и условия перехода меняются медленно. Но так как условие это дискретная величина, то условие часто не меняется от итерации к итерации, что приводит к появлению большого количества пустых и полупустых масок – это делает выгодным проверку масок на пустоту. На наличие почти пустых или почти полных масок делает выгодным применение векторизации с объединением масок. На гистограммах сверху показано теоретическое и реальное распределения плотности масок. На гистограммах снизу показаны результаты эксперимента по применению проверки масок и объединения масок – из данных видно, что предложенные оптимизации обеспечивают прирост эффективности векторизации.
66. Кратко отметим, что для успешной векторизации тело плоского цикла не обязано состоять только из простой арифметики. В рассматриваемом примере показана реализация определения пересечения треугольника и прямоугольного параллелепипеда из реализации метода погруженной границы. Тело плоского цикла само по себе содержит цикл, гнездо из двух циклов, два выхода из цикла, вызов функции. Однако, несмотря на это, может быть достаточно эффективно векторизовано. Главным тут является то, что условия выхода из вложенных циклов являются константными (постоянное количество итераций), а значит это условие не меняется в векторной версии кода. Дополнительно отметим, что векторизация вызова функции под предикатом транформируется в векторный аналог с передачей аргумента-маски внутрь функции.
67. Хуже дело обстоит когда внутри плоского цикла есть цикл с непостоянным условием выхода, то есть с непостоянным количеством итераций, как это наблюдается в центральной функции точного римановского решателя. В векторизованной версии этой функции внутренний цикл выполняется до тех пор, пока хотя бы одно из условий соответствующих скалярных циклов выполняется (пока векторизованное условие не обнулится). Это сразу сказывается на производительности.
68. Самым плохим вариантом является наличие внутри плоского цикла гнезда циклов, количество итераций в котором непостоянно и никак не связано при переходе от одной итерации к другой. Эталонным примером неэффективной векторизации можно считать реализацию сортировки Шелла, ядро которой состоит из трех циклов: k – цикл по шагу сортировки, два других цикла – сортировка всех срезов массива с шагом k. Даже если не обращать внимания, что при k < 16 эффективной векторизации будет мешать зависимость по доступу в память. При векторизации цикла по индуктивной переменной i проявляется проблема – количество итераций вложенного цикла предсказать невозможно.
69. Для разных последовательностей сортировки Шелла были вычислены теоретическое ускорение (выраженное в сокращении количества итераций внутреннего цикла) и замерено ускорение на машине и результат оказался крайне низким.
70. На рисунке проиллюстрирована упаковка итераций вложенного цикла при векторизации шириной 8 и 16. Видно, что при этом падает плотность масок выполнения векторных операций, что приводит к деградации производительности.
71. В заключение кратко приведу пример векторизации другого дискретного кода. Во второй главе был рассмотрен генетический алгоритм декомпозиции графа. Для этого алгоритма была выполнена векторизация быстрого алгоритма пузырькового наращивания доменов, основой которого является обход графа в ширину. Этот программный код содержит полный набор недостатков для векторизация – дискретные структуры данных, приводящие к дополнительной косвенности и операциям множественного обращения в память, а также те же циклы с нерегулярным количеством итераций. Как итог – эффективность векторизации в районе 0,1.
72. На последнем слайде собрана карта эффективности векторизации разного программного кода, рассмотренного в главе. Наилучшую эффективность векторизации показал простой код с арифметикой, либо код с условиями из физических расчетов с использованием проверки масок на пустоту. Наихудшим оказался код с неполными матрицами и гнезда циклов с нерегулярным количеством итераций.