1. X
2. X
3. X
4. X
5. X
6. **П**ервые три главы посвящены задаче разработки метода перестроения поверхностной неструктурированной расчетной сетки для повышения стабильности вычислений. **В** первой главе предлагается и анализируется метод перестроения сетки в двумерном случае. **В**о второй главе предлагается его обобщение на трехмерный случай. **В** третьей главе рассматриваются вопросы пересечения расчетных сеток – предлагаются методы удаления самопересечений поверхностной расчетной сетки и рассматривается пересечение поверхностной сетки с объемной расчетной сеткой для выполнения газодинамических расчетов вокруг поверхности.
7. **П**ервая глава – перестроение поверхностной сетки в двумерном случае. **Д**ля рассмотрения перестроения поверхностной расчетной сетки в двумерном случае под поверхностной расчетной сеткой понимается ломаная на плоскости, не содержащая самопересечений. **Л**оманая состоит из n ячеек-звеньев с длинами l. **Д**ля каждой ячейки определена внешняя нормаль, через которые вычисляются нормали узлов. **Н**ормали определяют направление изменения геометрии сетки. **В** каждой ячейке определено значение H большое, характеризующее смещение ячейки в направлении ее нормали такое, что в процессе перестроения сетки площадь, заметаемая ячейкой, соответствует целевой площади T большое. **В** задаче ледообразования величины H и T соответствуют толщине и объему накопленного в ячейке льда. **Т**ак как ячейки не могут смещаться произвольным независимым друг от друга образом, то рассматривается движением узлов сетки, в результате чего фактическая заметаемая площадь S отличается от целевой. **В**еличина дельта большое характеризует абсолютное отклонение фактической площади от целевой, а дельта маленькое – относительное. **Ц**елью перестроения сетки является нахождение таких новых положений узлов, которые приводят к наименьшим значениям дельта. **В** этой главе рассматривается постановка задачи при фиксированных направлениях нормалей узлов (требуется найти только величины смещений вдоль нормалей).
8. **Т**ак как решение задачи нахождения смещений узлов вдоль нормалей в общей виде с помощью градиентного спуска связано с большими вычислительными затратами и приводит к попаданию в локальные минимумы, то рассматриваются приближенные методы перестроения, встречающиеся в литературе, основанные на представлении целевой площади в виде геометрических примитивов. **В** методе прямоугольников целевая площадь представлена прямоугольником со сторонами l и H, в этом случае величина смещения узла берется как среднее арифметическое величин смещения инцидентных ячеек. **В** методе трапеций целевая площадь ячейки представлена трапецией, в основании которой лежит ячейка, а боковые стороны направлены вдоль нормалей инцидентных узлов. **П**остроенная для ячейки трапеция определяет новые положения инцидентных узлов. **Т**аким образом, в качестве нового положения конкретного узла берется среднее из положений, построенных по трапециям двух инцидентных узлу ячеек.
9. **П**редлагается новый метод перестроения расчетной сетки – метод окрестностей. **Д**ля произвольного множества точек, на котором задана функция радиуса R, можно рассмотреть окрестность этого множества – множество точек, попадающих хотя бы в один шар с центром в точке C и радиусом R от C. **Е**сли для каждого узла определить радиус как среднее арифметическое величин H большое инцидентных ячеек, а для внутренних точек ячейки определить линейное изменение радиуса, то окрестностью ячейки будет являться выпуклая оболочка двух шаров, построенных на инцидентных узлах. **В** методе окрестностей в качестве нового положения узла примем точку пересечения направления нормали узла с границей окрестности всех инцидентных ячеек этого узла.
10. **Д**ля проведения оценок точности методов прямоугольников, трапеций и окрестностей перестроения сетки рассмотрены выпуклая и вогнутая сетки, на которых велична H большое изменяется линейно, а параметр альфа характеризует кривизну сетки. **П**олучена формула S большое в общем случае, а также также формулы для методов прямоугольников и трапеций, где H большое с индексом t – высота трапеции, построения для конкретной ячейки.
11. **П**олучены явные формулы вычисления S большое для метода окрестностей для случаев выпуклой и вогнутой сеток.
12. **Н**а основании формул для S большое построены графики поверхностей дельта малое от альфа и отношения дельта H к длине ячейки l (рисунок слева). **Н**а рисунках справа приведены срезы поверхностей для фиксированных значений альфа (положительное значение для выпуклой сетки и отрицательное – для вогнутой). **Н**а основании проведенного анализа можно отметить, что при нулевом дельта H большое метод трапеций демонстрирует абсолютную точность. **В** целом метод трапеций наиболее точен, а точность методов прямоугольников и окрестностей близки.
13. **Д**ля методов прямоугольников, трапеций и окрестностей проведена оценка сглаживания дефектов сетки. **Д**ля этого рассматривается плоская сетка, за исключением двух ячеек (слева – две ячейки образуют пик с углом два альфа, справа – две ячейки образуют впадину с углом два альфа). **П**олучены формулы сглаживания углов при пике и впадине в общем случае для произвольных смещений узла дефекта и одного из смежных с ним узлов.
14. **Д**ля методов прямоугольников, трапеций и окрестностей получены формулы сглаженного угла при пике и впадине и построены графики сглаженных углов и эффективности сглаживания этих углов, представленные на рисунках внизу (0 – сглаживание отсутствует, 1 – полное сглаживание до угла пи пополам). **И**з зависимостей сглаживания пиков можно отметить: метод окрестностей демонстрирует наилучшие результаты, а метод трапеций не сглаживает пик при малых углах и приводит к неконтролируемому росту высоты пика). **И**з зависимостей сглаживания впадин можно отметить: метод прямоугольников не сглаживает впадины (отрицательная эффективность сглаживания), а метод трапеций обладает меньшей областью применимости (раньше наступает самопересечение сетки).
15. **В** выводах к главе отмечается: для предложенного метода перестроения поверхностной сетки точность перестроения близка к точности метода прямоугольников. **М**етод окрестностей, в отличие от методов прямоугольников и трапеций, позволяет устранять дефекты расчетной сетки – сглаживать как пики, так и впадины.
16. **В**торая глава – перестроение поверхностной сетки в трехмерном случае. **Р**ассматривается обобщение двумерной постановки. **П**оверхностная неструктурированная расчетная сетка с треугольными ячейками. **С**етка описывает двустороннюю поверхность, для каждой ячейки определена внешняя нормаль, для каждого узла вычислена нормаль через нормали инцидентных ячеек. **С**етка является целостной и замкнутой. **Т**акже необходимо найти новые положения узлов, чтобы объем заметаемый ячейкой при движении был близок к целевому объему V равному S большое умноженное на H большое.
17. **А**налогично двумерному случаю рассматриваются приближенные методы перестроения. **М**етод призм – аналог метода прямоугольников (представление целевого объема в виде призмы) и метод пирамид (представление целевого объема в виде призматоида).
18. **Н**а данном этапе рассматриваются дополнительные аспекты, встречающиеся в литературе, и применяемые при перестроении расчетной сетки в трехмерном случае. **М**ногослойное перестроение, позволяющее повысить точность за счет разделения целевого объема на несколько более мелких частей. **С**глаживание нормалей, позволяющее изменить направления смещения узлов для предотвращения раннего схлопывания ячеек. **С**глаживание поля высот (значения H большое) для устранения поверхностного шума. **С**глаживание сетки по размеру ячеек для устранения областей со слишком сгущенным или слишком разреженным расположением узлов.
19. **П**остановка метода окрестностей аналогична двумерному случаю. **Т**реугольник описывается в виде геометрического места точек относительно одной из вершин. **П**о этому описанию вводится функция радиуса по трем радиусам в узлах ячейки. **П**о введенной функции радиуса определяется окрестность ячейки. **Н**овое положение узла определяется как точка пересечения траектории движения узла с границей окрестности инцидентных узлу ячеек. **Д**ля поиска точки пересечения траектории движения узла с окрестностью ячейки рассматривается максимального значения альфа (где альфа – один из корней данного уравнения) при ограничениях на бета и гамма.
20. **З**адача поиска максимального значения альфа распадается на два случая. **П**ри бета равном нулю, гамма равном нулю или сумме бета и гамма равной единице задача сводится к поиску экстремума функции одной переменной на отрезке. **Д**ля других значений параметров бета и гамма задача сводится к поиску точки пересечения траектории движения узла и общей касательной плоскости к трем сферам.
21. **К**ак и в двумерном случае метод окрестностей позволяет сглаживать впадины. **Н**а рисунке слева продемонстрирована схема стягивания впадины, на рисунке справа – сравнение с методом прямоугольников (темно-синим цветом показан эффект от сглаживания впадины на модельной сетке).
22. **Т**акже метод окрестностей позволяет сглаживать острые складки сетки. **Н**а рисунках продемонстрировано перестроение расчетной сетки с помощью FENSAP-ICE в сравнении с реализованным методом окрестностей в модуле Кристалл.
23. **В** выводах к главе отмечается. **П**редложен метод окрестностей перестроения поверхностной расчетной сетки в трехмерном случае как обобщение двумерного метода. **М**етод позволяет сглаживать дефекты расчетной сетки (пики и впадины), имеет линейную по количеству узлов сложность и реализован в программном модуле Кристалл.
24. **В**не зависимости от используемого метода перестроения расчетной сетки при достаточно продолжительном времени моделирования при больших значениях H большое неизбежно возникновение самопересечений сетки – критического дефекта, препятствующего дальнейшему проведению вычислений. **В** задаче моделирования обледенения возникновение самопересечений является неизбежным при длительном времени моделирования. **С**амопересечения сетки должны быть удалены для возможности проведения расчетов.
25. **Д**ля задачи удаления самопересечений были проанализированы известные алгоритмы поиска пересечений фигур, ограниченных поверхностными сетками, которые используются в частности в приложениях компьютерной графики и CAD приложениях. **К** особенностям этих алгоритмов можно отнести поиск пересечений в виде ломаных, обход линий пересечения с помощью обхода соседей пересекающихся треугольников, локальное дробление ячейки по ломаной, коррекцию расчетной сетки для избавления от треугольников с околонулевой площадью и наложения ячеек, компенсацию ошибок с плавающей арифметикой. **В** применении к задаче удаления самопересечений отметим несколько особенностей этой задачи. **В**о-первых, не требуется поиск всех самопересечений – после обработки расчетной сетки в корректном состоянии должна остаться внешняя поверхность сетки, а все внутренние области могут быть удалены без детального анализа. **В**о-вторых в задачах эволюции расчетной сетки в результате перестроения нельзя гарантировать отсутствие ячеек с околонулевой площадью, а также наложение ячеек.
26. **В** предлагаемом подходе удаления самопересечений можно выделить три основные фазы. **П**оиск потенциально пересекающихся треугольников. **О**бход внешней поверхности расчетной сетки (которая должна сохраниться после удаления самопересечений), во время обхода возможно дробление ячеек на более мелкие. **У**даление лишних ячеек – всех ячеек, которые не были обработаны во время выполнения обхода внешней поверхности.
27. **П**оиск потенциально пересекающихся треугольников. **Д**ва треугольника называются потенциально пересекающимися, если пересекаются охватывающие их прямоугольные параллелепипеды. **Д**ля поиска потенциально пересекающихся треугольников используются BHV-деревья (bounding volume hierarchy tree). **С** их помощью все пары потенциально пересекающихся треугольников могут быть найдены со сложностью n log n. **Т**акие треугольники будем также называть конфликтными.
28. **О**бход внешней поверхности. **О**бход выполняется, пока следующая ячейка обхода не окажется конфликтной. **В** случае наличия конфликта конфликтная ячейка и все пересекающиеся с ней ячейки дробятся на более мелкие по пересечению с другими ячейками. **П**родолжить обход невозможно, если следующее ребро, через которое нужно перейти на соседнюю ячейку, имеет более двух инцидентных ячеек. **Е**сли рассматриваемая сетка является простой (нет наложений ячеек, точки и ребра не лежат в неинцидентных им ячейках, ребра сетки не пересекаются), то таких ячеек ровно 4 (как это представлено на рисунке слева). **В** общем случае таких ячеек произвольное количество (рисунок справа). **В** обоих случаях для каждой ячейке – претендента на следующую ячейку в обходе определяется функция F, и переход выполняется на ячейку с минимальным значением этой функции.
29. **П**оиск пересечения с другими ячейками. **О**бласть пересечения двух ячеек является выпуклым многоугольником с количеством вершин от 1 до 6. **Д**ля избежания потери точности поиск пересечений выполняется в рациональных координатах, в которых задача поиска пересечений геометрических примитивов – точек, прямых, отрезков, плоскостей, треугольников – решается точно. **Р**езультат пересечения представлен в виде точки, либо набора отрезков.
30. **М**ножество, по которому требуется разбить треугольник на более мелкие является произвольным набором точек и отрезков. **С**начала ищутся все точки пересечения отрезков, а затем решается задача триангуляции, в которой все отрезки входят в итоговую триангуляцию.
31. **П**редложенный метод позволяет удалить самопересечения произвольной замкнутой сетки. **В** областях самопересечения возможно возникновение множества ячеек неправильной формы или с околонулевой площадью, которые могут быть устранены с помощью локальной коррекции.
32. **Ч**асто исследования проводятся на псевдотрехмерных профилях – сетка шириной в одну ячейку, представляющая собой ленту в пространстве. **Д**ля таких сеток предложенный метод в общем случае неприменим. **И**з-за точности вычислений граничные ребра псевдотрехмерного профиля могут не пересечься, что вызовет проблемы при обходе сетки. **П**роблема может быть решена, если передняя и задняя плоскости псевдотрехмерного профиля параллельны одной из плоскостей OXY, OYZ, OXZ.
33. **О**днако даже в этом случае появляется другая проблема – профиль перестает быть псевдотрехмерным – после удаления самопересечений могут появиться узлы сетки, не являющиеся граничными.
34. **Д**ля такого случая предложен грубый метод удаления самопересечений. **В** пространстве строится сеточная подложка – расчетная сетка с мелкими прямоугольными ячейками. **С** помощью обхода подложки находится внешняя поверхность расчетной сетки. **Д**алее все ячейки, не относящиеся к внешней поверхности, а также все конфликтные ячейки назначаются помеченными и последовательно выполняется стягивание сетки по граничным ребрам, инцидентным помеченным ячейкам. **Ч**ерез несколько шагов выполнения этой операции получается огрубленная расчетная сетка без самопересечений, являющаяся псевдотрехмерным профилем.
35. **Н**а рисунке приведено сравнение результатов работы удаления самопересечений псевдотрехмерного профиля с использованием подложки (слева) и с использованием дробления ячеек (справа). **И**спользование подложки сохраняет псевдотрехмерный профиль, но результат получается грубее.
36. **П**оиск пересечения треугольника с ячейкой подложки выполняется путем решения системы линейных неравенств с двумя переменными путем свертывания этой системы по методу Черникова. **Д**ля поиска пересечений треугольника со всеми ячейками подложки используется ограничение на диапазон, существенно сокращающее количество анализируемых ячеек подложки.
37. **Р**ассматривается применение сеточной подложки для реализации численных схем решения уравнений газовой динамики с помощью метода погруженной границы с использованием фиктивных ячеек. **Д**ля этого на сложной границе создается слой фиктивных ячеек, газодинамические параметры в которых получаются путем аппроксимации значений газодинамических параметров соседних обычных ячеек, а также направления внешней нормали, проведенной к точке поверхности.
38. **М**етод погруженной границы применим для решения задач обтекания тел со сложной геометрией. **Н**а слайде представлена реализации работы решателя с использованием схемы Стегера-Уорминга и метода погруженной границы. **И**терация расчетов состоит из аппроксимации газодинамических величин в фиктивных ячейках, вычислении значений векторов F, G, H во всех ячейках сетки и пересчете поток между ячейками.
39. **П**риведены выводы из главы. **Р**ассмотрены вопросы, касающиеся пересечения расчетных сеток. **П**редложены методы удаления самопересечений поверхностных расчетных сеток в общем виде и для псевдотрехмерных профилей, основанные на обходе внешней поверхности расчетной сетки. **Р**ассмотренные методы применимы для моделирования обледенения поверхности. **Р**ассмотрен метод поиска пересечения поверхностной расчетной сетки с подложкой, который находит свое применение в частности в реализации метода погруженной границы для газодинамических расчетов для тел со сложной геометрией.
40. **П**ерейдем к четвертой главе и методам повышения производительности при распараллеливании вычислений на поверхностных и объемных расчетных сетках в модели распараллеливания с передачей сообщений. **В** этой модели область задачи разделяется на k отдельных частей, каждая из которых обрабатывается в своем вычислительном процессе. **Д**ля синхронизации процессы обмениваются сообщениями, в которых содержатся данные о состоянии расчетной области на границах подобластей. **О**тсюда логически следуют используемые для оценки эффективности распараллеливания показатели: ускорение при распараллеливании, эффективность распараллеливания, а также показатели качества декомпозиции расчетной области на подобласти – D – неравномерность распределения (влияет на простой вычислительных ресурсов), L – длина максимальной границы (определяет время, затрачиваемое на межпроцессные обмены) и дополнительный показатель I – общая длина границ между доменами.
41. **С**начала рассмотрим вопросы распределения вычислений для блочно-структурированных расчетных сеток. **Б**лочно-структурированная расчетная сетка состоит из блоков – массивов ячеек, обращение к которым осуществляется по индексам. **О**бработка таких сеток выполняется быстрее, однако они более накладные с точки зрения создания и управления.
42. **В**ажной научно-практической задачей является отображение информационного графа задачи (отражающего подобласти задачи и потоки данных между ними) на граф вычислительной системы (отражающий разные вычислители и каналы связи между ними). **П**оиску приближенных эвристических алгоритмов решения этой задачи в мире посвящено множество исследований.
43. **Б**удем рассматривать задачу распределения блоков расчетной сетки по вычислительным процессам аналогично известной задаче о наилучшем разбиении множества, или задаче «о куче камней». **В** этой задаче требуется разделить множество X на k подмножеств, чтобы вес наибольшего подмножества был минимален. **Э**то соответствует достижению наилучшего показателя декомпозиции задачи D. **И**звестен алгоритм решения этой задачи с помощью метода ветвей и границ, однако он имеет экспоненциальную сложность.
44. **С**начала рассмотрим простой жадный алгоритм распределения множества блоков по партициям. Будем считать, что исходный массив отсортирован по убывания. **Н**а каждом шаге будем брать самый крупный блок (по количеству ячеек) и определять его в наиболее легкую партицию. **Д**ля этого алгоритма приведена его сложность и оценка точности через остаточные члены.
45. **В**виду экспоненциальной сложности точного алгоритма сравнение жадного алгоритма с точным решением проводилось на ограниченном количестве процессов. **И**з результатов сравнения видно, что жадный алгоритм сильно проигрывает точному решению. **Т**акже из оценки точности можно отметить, что высокие значения остаточных членов являются индикаторами наличия в сетке крупных блоков, которые мешают равномерно распределять вычислительную нагрузку между процессами.
46. **Д**ля достижения лучших показателей при распределении блоков расчетной сетки между процессами для блочно-структурированных сеток был реализован механизм дробления блоков – разделение блока на два более мелких блока по одному из измерений.
47. **Д**алее рассматривается задача определения минимального количества разрезов, которые необходимо выполнить для извлечения из блока части заданного размера – функция P большое. **З**начение этой функции может быть вычислено явно рекурсивно, однако связано с большими вычислительными затратами.
48. **Ф**ункция P большое обладает свойствами симметричности и всегда не больше 5, что несколько снижает затраты на ее вычисление, но все равно не делает их приемлемыми для применения. **Т**еоретически блочно-структурированная расчетная сетка с количеством ячеек, кратным k, может быть распределена между k процессами абсолютно равномерно и использованием не более 5(k – 1) разрезов блоков, но такой подход неприменим на практике. **Т**ребуются приближенные алгоритмы решения поставленной задачи.
49. **Р**ассмотрим приближенный алгоритм, основанный на жадном распределении. **В** этом алгоритме применяется дробление максимального блока пополам до тех пор, пока жадный алгоритм не приведет к приемлемому показателю неравномерности распределения, заданному параметром эпсилон. **Д**робления блоков расчетной сетки, выполняемые для достижения требуемого показателя неравномерности распределения будем называть подготовкой сетки.
50. **П**оставлен численный эксперимент влияния степи подготовки расчетной сетки для разного количества партиций на масштабируемость вычислений на 1-32 вычислительных процессах (в эксперименте каждый процесс запускался на отдельном вычислительном узле). **Ч**ерный и серый графики демонстрируют наличие крупных блоков расчетной сетки, из-за которых масштабирование остановилось. **С**иний и зеленый графики демонстрируют схожее поведение, что говорит об излишней подготовке сетки для 64 партиций. **К**расный график соответствует подготовке сетки на 64 партиций с низким требуемым показателем неравномерности распределения (1%). **Н**а графиках наблюдается явный провал по производительности, вызванный большим количеством дроблений блоков, что привело к возрастанию межпроцессных обменов. **Т**аким образом, требуются алгоритмы, уменьшающие количество дроблений блоков расчетной сетки.
51. **Р**ассмотрим приближенный алгоритм распределения блоков по партициям с уменьшением количества дроблений. **В** качестве дополнительных ограничений использовался запрет на разрезы, слишком близкие к границе блока, и создание слишком мелких блоков. **Д**ля работы алгоритма используются 4 вспомогательные операции: заполнение партиции полным блоком снизу не превышая значения тау, а также аналогичные операции для заполнения частью блока снизу, полным блоком сверху и частью блока сверху.
52. **Т**огда используя эти вспомогательные операции получим семейство алгоритмов, представленное на слайде. **П**ри этом для гарантированного завершения обе операции заполнения полными блоками должны использоваться. **А** также должна быть использована хотя бы одна операция заполнения частями блоков. **П**орядок применения операций продиктован сдерживанием значения тау и уменьшением количества разрезов.
53. **Э**ти три алгоритма сравнивались на наборах случайных данных по показателю неравномерности распределения и по количеству выполненных разрезов. **П**о результатам анализа выбран алгоритм, использующий заполнение полными блоками сверху и снизу, а также заполнение частями блоков сверху (наименьшее количество дроблений и не худший показатель неравномерности распределения).
54. **П**о рассмотренной задачей распределения блоков блочно-структурированной расчетной сетки можно отметить. **П**редложены алгоритмы распределения блоков расчетной сетки по партициям с дроблением блоков. **П**роведен анализ алгоритмов и рассмотрено семейство алгоритмов, позволяющих уменьшить количество разрезов блоков. **И**з этого семейства выбран наилучший алгоритм по показателю неравномерности распределения и количеству разрезов.
55. **Д**ля поверхностных неструктурированных расчетных сеток для распараллеливания вычислений в модели распараллеливания с передачей сообщений были рассмотрены различные алгоритмы декомпозиции их дуального графа. **С**реди них выделено два алгоритма, позволяющие добиться низкого значения показателя качества декомпозиции D. **А**лгоритм иерархического деления доменов пополам. **А**лгоритм пузырькового роста доменов с коррекцией инициирующих вершин с помощью генетического подхода.
56. **П**ри выполнении декомпозиции расчетной сетки возможно появление протяженных пилообразных границ между доменами, как это показано на рисунке слева. **П**редлагается алгоритм сглаживания границ между доменами, позволяющий с помощью локальных шаблонов преобразования границы добиться уменьшения длины границы без изменения баланса ячеек между доменами. **Н**апример на рисунке справа показаны шаблоны уменьшения длины границы на 1 с переходом одной ячейки из верхнего домена в нижний и наоборот, на рисунке справа внизу показаны шаблоны уменьшения длины границы на 1 с переходом трех ячеек из верхнего домена в нижний и наоборот.
57. **Н**екоторые шаблоны могут конфликтовать друг с другом. **З**адача сглаживания границы между парой доменов сводится к выбору максимального подмножества неконфликтующих друг с другом шаблонов, таких, чтобы суммарно они не нарушали баланс ячеек. **Н**а рисунке черным показана исходная граница, зеленым отмечены участки без изменения, а красным – случаи применения шаблонов. **З**адача решается точно с помощью динамического программирования. **В** качестве минимизируемой функции рассматривается B от t, u, x – решение на всем множестве шаблоном начиная с t, u – изменение количества ячеек в верхнем шаблоне, x – признак использования шаблона t.
58. **Р**ешение задачи. **С**начала выполняется инициализация решения для последнего шаблона. **П**ри переходе от шаблона с номером k + 1 к шаблону с номером k рассматриваются отдельно случаи игнорирования шаблона и его обработке при отсутствии конфликта и при его наличии. **В** результате получаем все возможные варианты решения задачи для всех доступных значений параметра u, в том числе и для u = 0. **С**ложность алгоритма квадратичная по количеству шаблонов при использовании шаблонов ограниченной длины (как в нашем случае – шаблон имеет длину 2 или 3 ребра).
59. **Н**а рисунках продемонстрированы результаты работы алгоритма сглаживания границ между доменами после применения алгоритмов декомпозиции иерархического деления доменов пополам и алгоритма Фархата. **А**нализ данных показал, что в результате применения алгоритма длина границ между доменами уменьшается примерно на 10 процентов.
60. **П**редложенный алгоритм обеспечивает точное решение с учетом вида используемых шаблонов. **Т**акже алгоритм может быть расширен на шаблоны произвольного вида, а также на трехмерный случай для неструктурированных расчетных сеток в пространстве.
61. **Б**ыл поставлен численный эксперимент по масштабированию вычислений по моделированию обледенения на поверхностной расчетной сетке. **В** эксперименте для декомпозиции использовался алгоритм иерерхического деления домена пополам со сглаживанием границ между доменами. **П**родемонстрированы результаты масштабирования с эффективностью распараллеливания в районе 0,8 вплоть до использования 64 вычислительных узлов.
62. **Д**ля выполнения вычислений на поверхностной неструктурированной расчетной сетке по моделированию обледенения с использованием конечно-объемного численного метода рассматривались методы разрешения конфликтов по доступу к данным при распараллеливании на общей памяти. **К**онфликты по данным возникают при одновременной обработке ребер расчетной сетки, через которые перетекают потоки вещества. **К**онфликты могут быть разрешены с помощью использования директив OpenMP, запрещающих одновременный доступ двух потоков к одним данным. **Д**ругим способом является разбиение множества ребер на неконфликтующие подмножества.
63. **З**адача разбиения множества ребер на неконфликтующие подмножества сводится к построению реберной раскраски дуального графа расчетной сетки. **Д**ля поверхностной неструктурированной расчетной сетки дуальный граф является кубическим графом без мостов (а в случае односвязной поверхности еще и плоским), что допускает реберные раскраски в 5, 4 и 3 цвета (раскраска Тейта). **П**оставлен численный эксперимент по сравнению двух методов разрешения конфликтов по доступу к данным на микропроцессоре Intel Xeon Phi KNL (допускающим максимальное количество потоков – 288). **П**ри использовании большого количества потоков метод реберной раскраски оказывается эффективнее (хотя количество цветов раскраски на эффективность не влияет).
64. **П**еред тем как перейти к последней главе, посвященной векторизации, приведем еще один численный эксперимент. **Н**а разных узлах суперкомпьютера МВС-10П были проведены замеры масштабируемости одного и того же расчетного кода, представленного в двух версиях – скалярный код и векторизованный код – газодинамический решатель основанный на точном решении задаче о распаде разрыва. **Б**ыло отмечено, что эффективность распараллеливания на общей памяти для векторного кода хуже и эта разница возрастает при увеличении количества потоков. **Г**рафик снизу показывает тот минимально допустимый порог, ниже которого нет смысла проводить векторизацию.
65. **П**ятая глава посвящена вопросам векторизации вычислений – низкоуровневой оптимизации, с помощью которой можно добиться кратного ускорения программного кода. **В**екторизация заключается в объединении однотипных операций, работающих со скалярными данными в векторные операции над векторами, содержащими несколько элементов скалярных данных. **П**ри изучении векторизации используются такие понятия как ширина векторизации, ускорение от векторизации, эффективность векторизации. **Т**акже используется понятие логического ускорения, выраженного через отношение количества скалярных инструкций к количеству соответствующих им векторных инструкций, и понятие логического ускорения. **В**екторные инструкции поддержаны во всех современных архитектурах: x86, ARM, Power, Эльбрус. **Н**аиболее перспективным можно признать набор инструкций AVX-512, основной особенностью которого является использование векторных масочных регистров, позволяющих выборочно обрабатывать элементы векторов.
66. **П**ри векторизации кода важен способ объединения однотипных инструкций в векторные команды. **В** этой главе делается попытка определить некоторый общий вид программного контекста, к которому могут быть применены оптимизации векторизации. **Р**асчетные коды и в частности вычисления на расчетных сетках содержат как правило обработку массивов элементов – вершин, ребер, ячеек – это происходит в цикле. **Д**ля общих целей векторизации определим удобный для векторизации программный контекст в следующем виде. **П**лоским циклом назовем цикл for с индуктивной переменной, изменяющейся от 0 до w – 1, где w – ширина векторизации, удовлетворяющий следующим требованиям. **Н**а i-ой итерации цикла все обращения к данным на запись имеют вид a[i]. **Н**а i-ой итерации цикла все обращения к данным на чтение имеют вид a[i] или являются чтением скаляров. **В**се массивы данных, к которым идет обращение в виде a[i] выровнены в памяти на размер вектора. **В** цикле отсутствуют межитерационные зависимости, все итерации являются независимыми. **Т**акие требования могут показаться слишком жесткими, однако огромная доля расчетных кодов может быть представлена в виде таких циклов. **Т**ребование на количество итераций цикла вводится для удобства, так как любой цикл с большим числом итераций может быть разщеплен на несколько последовательных циклов с w итераций или меньше. **Б**удем исследовать методы векторизации плоских циклов с телом произвольного вида.
67. **В** расчетных кодах часто речь идет о вычислениях с вещественными числами. **В** таблице на слайде приведена таблица с основными векторными командами для работы с упакованными fp32 – арифметические операции с одним, двумя и тремя операндами, операции сравнения и слияние по маске. **С**емантика работы этих инструкций явно описывается в виде плоских циклов. **О**братное также часто оказывается верным – плоский цикл с телом без сложного управления может быть векторизован путем замены скалярных инструкций векторными аналогами.
68. **О**дной из причин потери производительности при векторизации является организация обрабатываемых данных в виде «массива структур», что естественно в программировании. **Р**еорганизация в виде «набора массивов» позволит объединить обращения в память в векторного чтения и записи векторов.
69. **Д**ругой причиной, препятствующей векторизации, является наличие в теле цикла большого количества условий. **В** приведенном на слайде примере продемонстрировано разбиение цикла по константному условию.
70. **В** некоторых случаях следования этим и похожим рекомендациям позволяет компилятору выполнить автоматическую векторизацию с достаточно высокой эффективностью. **П**оставлен численный эксперимент по автоматической векторизации газодинамического решателя, использующего схему Стерега-Уорминга и метод погруженных границ из главы 3. **Р**езультаты показаны на рисунке справа. **Ц**иклы, полностью удовлетворяющие требованиям, предъявляемым к плоским циклам, показали эффективность векторизации 0,5 – 0,6, тогда как циклы с нарушением некоторых требований (псевдоплоские) продемонстрировали более низкую эффективность векторизации.
71. **О**птимизация выноса маловероятного региона из плоского цикла. **П**рименима в том случае, когда в конце итерации присутствует блок тяжелого кода, выполняющийся по маловероятному условию. **Н**ахождение этого блока внутри цикла препятствует его векторизации. **С**охранение условия входа в маловероятный регион и вынос этого региона из цикла позволяет векторизовать остальной вероятный код внутри цикла.
72. **Б**ыл поставлен численный эксперимент по выносу маловероятного региона из цикла. **В** практической задаче определения конфликтов траектории собственного ЛА с опасными зонами движения других ЛА с последующей обработкой конфликтов является подходящим контекстом для применения векторизации. **П**роверка на наличие конфликта является хорошо векторизуемой операцией, а обработка конфликта – крайне редкой и медленной. **В**ынос кода обработки конфликта из плоского цикла позволил добиться ускорения кода более чем в 5 раз на fp32.
73. **О**сновной причиной потери производительности при векторизации является наличие условий и операций передачи управления внутри плоского цикла. **Т**ак как операции передачи управления не векторизуются, то общим способом векторизовать такой код является оптимизация слияния ветвей исполнения по условию (известная также как ifconversion, и реализованная в оптимизирующем компиляторе для архитектуры Эльбрус). **О**сновной идеей оптимизации является постановка всех операций под предикаты достижимости от начала итерации. **В** архитектуре x86 предикаты для скалярных операций отсутствуют, однако при векторизации кода они соответствуют как раз векторным маскам, что позволяет переводит код из предикатного представления в векторную форму. **Н**а слайде слева приведена схема слияния двух блоков по условию и построены графики эффективности векторизации от вероятности перехода p. **С**права показано то же преобразование, но с проверкой маски на пустоту перед выполнением векторного блока.
74. **Б**ыл проведен мониторинг векторных масок на расчетных приложениях. **Р**асчеты физических процессов характерны тем, что при переходе от одного объекта расчетной сетки к соседнему физические величины меняются достаточно медленно. **А** значит медленно меняются и условия, полученные на их основе. **Н**о так как условие – дискретная величина, это приводит к появлению большого количества пустых и полным масок. **Н**а слайде сверху сравнение профиля векторной маски на случайных данных и на реальном приложении. **Э**то говорит о целесообразности проверки масок на пустоту. **Д**ругой оптимизаций, связанных с плотностью масок является объединение выполнения двух соседних векторных блоков, выполняющихся под разными масками. **Е**сли эти маски не пересекаются, то вместо выполнения обоих блоков можно выполнить векторный блок один раз, объединив входные данные под соответствующими масками. **Б**олее сложность является оптимизация объединения вычисления двух векторных блоков с пересекающимися масками, но она не проверялась на практике.
75. **Т**ело плоского цикла может содержать другие циклы, характер которых также влияет на производительность результирующего кода. **Р**ассмотрим структуру «плоский цикл / внутренний цикл». **Н**а рисунке сверху показана схема его векторизации. **С**калярное условие продолжения выполнения внутреннего цикла превращается в векторную маску, и внутренний цикла в векторной форме заканчивает свое выполнение в случае истощения этой маски. **Т**аким образом количество итераций внутреннего цикла в векторной версии равно максимальному количеству итераций внутреннего цикла в скалярной версии для всех i от 0 до w – 1. **И**сходя из этого можно вывести ограничение сверху на логическую эффективность векторизации через величину e I vec, которую можно условно назвать эффективность векторизации в терминах количества итераций. **П**араметр эпсилон можно назвать характеристикой нерегулярности количества итераций внутреннего цикла.
76. **Б**ыли рассмотрены разные виды программного контекста, содержащие структуру «плоский цикл / внутренний цикл». **В**нутренний цикл с постоянным количеством итераций (эпсилон равен нулю, e I vec равен единице). **Т**акой тип контекста встретился в задаче поиска пересечений поверхностной сетки с подложкой из главы 3. **Д**ля поиска пересечения треугольника и прямоугольного параллелепипеда всегда решалась одна и та же система неравенств, и количество итераций внутреннего цикла было постоянно. **Н**а данном контексте было достигнуто ускорение в 6,7 раз. **В**нутренний цикл с непостоянным количеством итераций (e I vec близко к единице). **Т**акой тип контекста встретился в точном римановском решателе в функции решения нелинейного уравнения, непостоянное количество итераций внутреннего цикла позволило достичь ускорения для функции starpu всего лишь 4,4 раза. **Н**ерегулярное количество итераций. **Т**акой тип контекста характерен для дискретных задах. **Н**апример он был обнаружен на задаче обхода графа в ширину при декомпозиции расчетной сетки с помощью алгоритма пузырькового роста из главы 4 (эпсилон сравним с количеством итераций внутреннего цикла в векторной версии, e I vec достаточно мал). **Д**ля такого программного контекста ускорение от векторизации не превысило двух раз.
77. **Н**а заключительном слайде представлена карта программного контекста, на котором ставились эксперименты по векторизации. **Н**аиболее высокую эффективность векторизации продемонстрировали плоские циклы, содержащие арифметику и простое управление с применением проверки масок на пустоту и объединением масок соседних векторных блоков. **Н**аихудшие результаты продемонстрировали квазиплоские циклы, а также структуры с внутренними циклами с непостоянным и нерегулярным количеством итераций.
78. **О**тметим выводы к главе. **В**екторизация позволяет кратно увеличить производительность кода. **П**лоский цикл является удобным программным контекстом для векторизации. **С** его помощью код может быть записан в предикатной форме и переведен в векторное представления путем замены скалярных операций на векторные аналоги. **Х**отя векторизован может быть плоский цикл с практически произвольным телом, наиболее подходящим контекстом для векторизации является тело, содержащее арифметические операции и простое управление. **Т**акже эффективность векторизации может быть повышена с помощью проверок масок на пустоту и объединения масок соседних векторных блоков.
79. **Н**а слайде приведен список конференций и семинаров, на которых были представлены результаты работы.
80. **Н**а слайде приведены основные публикации по теме работы.
81. **Н**а слайде приведены зарегистрированные программы для ЭВМ, в которых были использованы результаты работы.
82. **С**пасибо за внимание. **Г**отов ответить на ваши вопросы.