Казахский национальный университет имени аль–Фараби

УДК 004.4

На правах рукописи

САКЫПБЕКОВА МЕРУЕРТ ЖУМАБЕКОВНА

Оптимальные раскладки памяти и коммуникационные шаблоны для параллельных неструктурированных CFD-кодов

6D075100 – Информатика, вычислительная техника и управление

Диссертация на соискание степени
доктора философии (PhD)

Научные консультанты:
PhD, доцент Исахов А.А.,
PhD, Dr.-Ing. Andreas Lintermann
(Германия)

Республика Казахстан,
Алматы, 2022

СОДЕРЖАНИЕ

НОРМАТИВНЫЕ ССЫЛКИ

В диссертации были приведены ссылки на следующие стандарты:

ГОСО РК 5.04.034-2011 «Государственный общеобязательный стандарт образования Республики Казахстан. Послевузовское образование. Докторантура». Основные правила заверены Министерством Образования и Науки РК. «17» июнь 2011ж. №261. Астана 2011.

«Инструкция по оформлению диссертаций и авторефератов», Министерство образования и культуры РК, ВАК, Алматы, 2004. МЭСТ 7.1-2003. Библиографическая запись.

ОПРЕДЕЛЕНИЯ

CFD – computational fluid dynamics (вычислительная гидродинамика);

HPC – High Performance Computing (высокопроизводительные вычисления);

FLOPS – FLoating-point Operations Per Second (операция с плавающей запятой в секунду);

CPU – Central processing unit (центральный процессор);

GPU – Graphics processing unit (графический процессор);

CUDA – Compute Unified Device Architecture (программно-аппаратная архитектура параллельных вычислений);

SPMD – Single Program Multiple Data (единая программа, множество данных);

SM – Stream Multiprocessor (потоковый мультипроцессор);

SP – Shader processor (шейдерный процессор);

FVM – Finite Volume Method (метод конечных объемов);

SFC – Space-filling curve (заполнения пространства кривой Гильберта).

GPGPU – General Purpose computing on Graphics processing unit (Вычисления общего назначения на графическом процессоре)

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Кривая Гильберта – это непрерывная кривая, построенная с использованием рекурсии, которая фрактально заполняет квадрат или куб.

Методы конечных объемов (FVM), часто называемые методами контрольного объема, формулируются на основе внутреннего произведения основных дифференциальных уравнений в частных производных с единичной функцией.

ВВЕДЕНИЕ

Диссертация посвящена оптимальной раскладке памяти и коммуникационным шаблонам для параллельных неструктурированных CFD-кодов.

Актуальность темы исследования. Высокоточное гидродинамическое моделирование, как правило, связано с большими вычислительными требованиями, острыми с каждым новым поколением суперкомпьютеров. Тем не менее, в настоящее время необходимы значительные исследования, чтобы раскрыть вычислительную мощность передовых систем, называемых до экафлопсовыми системами, основанными на более совершенных архитектурах. На вычислительной машине распараллеливание кода собственной разработки для численных вычислений становится обычным явлением. По мере развития численных решателей и сложности задач растут и возможности параллельных вычислений. Различные предложенные схемы используются для распараллеливания с помощью центральных процессоров и графических процессоров.

Ранее высокопроизводительные вычисления производились в больших кластерах компьютеров, каждый из которых мог выполнять небольшое количество параллельных потоков. Однако в последнее десятилетие графические процессоры (GPU) общего назначения продемонстрировали большой рост производительности. Каждый графический процессор может одновременно выполнять тысячи потоков с более низкими издержками. Хотя этот тип производительности изначально был разработан для поддержки видеоприложений, они стали необходимы для научных вычислений и ускорения работы алгоритмов машинного обучения.

В настоящее время в связи с развитием высокопроизводительной вычислительной техники учеными, осуществившими исследовательские работы на графическом процессоре и показавшие эффективность полученных результатов являются Weng, Y. (2021), Rostami, S. R. и др. (2019), Mohammadi, S. и др. (2020), Ayyad, M. и др. (2020), Afzal, A. (2021), Кривов М.А. (МГУ), Притула М.Н (ИПМ). Отечественные ученые - Урмашев Б. А., Ахмед-Заки Д.Ж., Дарибаев Б. С., Лебедев Д. В., Иманкулов Т.С., Алтыбай А. и др.

На сегодняшний день вычислительная гидродинамика (CFD) - распространенная методика решения сложных задач во многих областях техники. Заслугой CFD является разработка новых и усовершенствованных устройств и системных структур, а также оптимизация существующего оборудования за счет компьютерного моделирования, что приводит к повышению эффективности и снижению эксплуатационных расходов. С быстрым развитием компьютерных технологий вычислительная гидродинамика (CFD) играет важную роль в анализе аэродинамических характеристик, эффективном проектировании и исследовании сложного механизма потока.

Применение неструктурированных сеток используется в получении точных численных решений при моделировании течений в автомобилях, самолетах, поездах, в химической инженерии и в биомедицине.

Отмечается высокопроизводительность вычисления на графических процессорах (GPU) общего назначения со сложными связанными симуляциями с нетривиальными разложениями в области с помощью заполнения пространства кривой Гильберта (SFC) при использовании неструктурированных вычислительных сеток.

Цель диссертационной работы: Целью настоящей работы является применение схемы заполнения пространства кривой Гильберта (SFC) при использовании неструктурированной вычислительной сетки для различных ресурсоемких физико-технических задач на графических процессорах общего назначения. Предложенная схема позволяет повысить эффективность комплексного моделирования на основе измерений производительности во время моделирования. Данный подход позволяет автоматизировать распределение вычислительной сетки на одномерные массивы, которые можно использовать для эффективного распределения рабочей нагрузки.

Задачи исследования, реализующие цель диссертационной работы:

− численное исследование эффективности высокопроизводительных вычислений на графических процессорах общего назначения для циркуляционного несжимаемого вязкого течения в каверне;

− численное исследование эффективности высокопроизводительных вычислений на графических процессорах общего назначения для задач несжимаемого вязкого течения за обратным уступом при использовании структурированных вычислительных сеток;

− численное исследование эффективности высокопроизводительных вычислений на графических процессорах общего назначения для задач несжимаемого вязкого течения за обратным уступом с помощью заполнения пространства кривой Гильберта (SFC) при использовании неструктурированных вычислительных сеток.

Объектом исследования являются высокопроизводительные вычисления на графических процессорах общего назначения со схемой заполнения пространства кривой Гильберта (SFC) для неструктурированных вычислительных сеток для различных ресурсоемких физических задач.

Методы исследования. В данной диссертационной работе предлагаются методы, которые являются новым инструментом в исследовании распределения нагрузки на различный процессор при использовании неструктурированных вычислительных сеток. Схема заполнения пространства кривой Гильберта (SFC) позволяет повысить эффективность при разложении вычислительной сетки на домены.

Для численного расчета в работе применяются параллельные численные алгоритмы на графических процессорах общего назначения, и полученные результаты сравниваются с расчетными и экспериментальными данными других известных ученных.

Предметом исследования являются эффективные высокопроизводительные вычисления на графических процессорах общего назначения с применением технологии CUDA для численного решений уравнения Пуассона, циркуляционного несжимаемого вязкого течения в каверне и задач несжимаемого вязкого течения за обратным уступом.

Теоретическая и практическая ценность. Результаты, представленные в настоящей работе, могут быть широко применены при решении важных прикладных задач, которые связанны с численным моделированием на графических процессорах общего назначения с применением технологии CUDA. Разработанные схемы и численные алгоритмы, предназначенные для численного моделирования на графических процессорах общего назначения, (GPU) вносят непосредственный вклад в развитие науки и сферы информационных технологий страны.

Практическая ценность данной диссертационной работы состоит в том, что используемая схема заполнения пространства кривой Гильберта (SFC) для неструктурированных вычислительных сеток на графических процессорах общего назначения не только позволяет получать существенно «быстрые», по сравнению с последовательными вычислениями результат, но и значительно расширяет возможности реализации трудоемких методов и алгоритмов для решения важных прикладных и фундаментальных задач.

Научная новизна. В работе разработана оптимальная раскладка памяти и коммуникационные шаблоны распараллеливания для неструктурированных вычислительных сеток на графических процессорах общего назначения чтобы повысить эффективность производительности массивно-параллельных вычислений. Это подход применяется для различных ресурсоемких физических задач при использовании неструктурированной вычислительной сетки с помощью заполнения пространства кривой Гильберта (SFC).

С помощью построенного параллельного численного алгоритма заполнения пространства кривой Гильберта (SFC) на графических процессорах общего назначения (GPU) были выполнены:

− численные исследования эффективности высокопроизводительных вычислений на графических процессорах общего назначения для уравнения Пуассона;

− численные исследования эффективности высокопроизводительных вычислений на графических процессорах общего назначения для циркуляционного несжимаемого вязкого течения в каверне;

− численные исследования эффективности высокопроизводительных вычислений на графических процессорах общего назначения для задач несжимаемого вязкого течения за обратным уступом при использовании структурированных вычислительных сеток;

− численные исследования эффективности высокопроизводительных вычислений на графических процессорах общего назначения для задач несжимаемого вязкого течения за обратным уступом с помощью заполнения

пространства кривой Гильберта (SFC) при использовании неструктурированных вычислительных сеток;

− проведено сравнение полученных результатов моделирования с численными данными, полученными с помощью структурированных и неструктурированных вычислительных сеток;

− проведено сравнение полученных результатов моделирования с численными данными и экспериментальные данными других авторов;

− для структурированной вычислительной сетки и параллельного численного вычисления для неструктурированной вычислительной сетки с использованием схемы заполнения пространства кривой Гильберта (SFC) проведен анализ полученных результатов параллельного численного вычисления.

Положения, выносимые на защиту:

− результаты численного исследования эффективности высокопроизводительных вычислений на графических процессорах общего назначения для различных ресурсоемких физических задач;

− обоснование использования различных раскладов памяти и коммуникационных шаблонов на графических процессорах общего назначения для различных ресурсоемких физических задач;

− эффективное использование схемы высокопроизводительного вычисления на графических процессорах общего назначения со сложными связанными симуляциями с нетривиальными разложениями в области с помощью заполнения пространства кривой Гильберта (SFC) при использовании неструктурированных вычислительных сеток.

Публикации и апробации результатов. Результаты диссертации были опубликованы в 11 научных работах, из них 1 опубликована в журнале, входящая в базу данных Web of Science и 2 опубликованы в журналах из базы данных SCOPUS, 6 статей – из списка, рекомендованного Комитетом по обеспечению качества в сфере образования и науки Министерства образования и науки Республики Казахстан, 2 работы – в материалах международных и республиканских конференций.

Объем и структура работы. Общий объем работы – 90 страниц. Диссертационная работа состоит из введения, 5 разделов, заключения, списка используемых источников из 84 наименований, 2 приложения, включая 46 рисунков и 7 таблиц.

Основное содержание работы. Данная работа представлена в следующем порядке.

Во введении обсуждаются актуальность выбранной темы диссертационный работы, цель, объект, предмет и задачи исследования. Описаны полученные результаты проведенных исследований, их научная новизна и практическая значимость.

Первый раздел посвящен анализу современных систем высокопроизводительных вычислений для прикладных задач. Описаны расчеты с помощью многоядерных CPU и основные принципы расчетов на

GPU. В конце этой главы мы представляем анализ и оценку эффективности высокопроизводительных вычислении на CPU и GPU.

Во втором разделе представлена математическая модель для моделирования. В этой главе обобщаются основные уравнения гидродинамики: уравнение неразрывности, уравнение движения сплошной среды. Описаны обезразмеривание и дискретизация уравнения Навье-Стокса.

В третьем разделе рассматривалось решение уравнения Пуассона в некоторой прямоугольной области, охватывающего широкий класс прикладных задач. Применяя данное уравнение, был представлен численный расчет на CPU и GPU, на основании данного расчета был сделан сравнительный анализ, который показал эффективность параллельного численного алгоритма.

В четвертом разделе рассматривается задача несжимаемого вязкого течения в каверне. На основе данной задачи получен результат эффективности на GPU с применением разных размеров блока. Сделан сравнительный анализ времени процессора и графического процессора производительности, показавший значительное увеличение скорости графического процессора. Данное сравнение вычислительного времени показывает преимущество технологии GPU в решении инженерных задач, требующих интенсивных численных вычислений. Анализ числа потоков в блоке, возможно, наиболее важного параметра распараллеливания в CUDA, показывает наличие оптимального значения. Этот результат является простым, но мощным методом оптимизации CUDA, который значительно влияет на общее время обработки.

В пятом разделе представлены результаты смешанно-конвективного теплопереноса в процессе ламинарного двумерного течения в вертикальном канале в структурированных и неструктурированных сетках. Численно исследуется широкий диапазон условий течения на входе и температур стенки, охватывающий область от чисто вынужденного конвективного течения. В данном разделе показан результат использования задач несжимаемого вязкого течения за обратным уступом на структурированных и неструктурированных сетках, на основе применения распараллеливания в технологии CUDA с использованием схемы заполнения пространства кривой Гильберта (SFC). Результаты данного исследования, которые применялись для неструктурированной сетки впервые, показали совпадения на структурированных и неструктурированных сетках.

В заключении представлены выводы данной диссертационной работы.

1 АНАЛИЗ СОВРЕМЕННЫХ СИСТЕМ ДЛЯ ПРИКЛАДНЫХ ЗАДАЧ

1.1 Обзор систем высокопроизводительных вычислений для параллельных неструктурированных CFD-кодов

Сегодня вычислительная гидродинамика (CFD) становится мощным и широко используемым инструментом во многих отраслях промышленности и биомедицины, в котором каждое решение представляет собой богатый набор математической физики, численных методов, пользовательских интерфейсов и современных методов визуализации. Вычислительная гидродинамика (CFD) – это одной из особенно главных областей использования высокопроизводительных вычислений, задающей темп улучшения научных вычислений. При одновременном увеличении количества решателей CFD и вычислительных возможностей машин даже самая мощная вычислительная машина не может давать аналогичные результаты в численных расчетах в короткий период, поскольку размер и сложность численных задач возрастают [1-4]. Требования CFD для решения все более трудных проблемных задач, возникающих в инженерных областях на системном уровне, в особенности в аэрокосмической и автомобильной промышленности, архитектуре, в электронной и химической инженерии, в сочетании с возникновением новых, более сильных поколений параллельных компьютеров, обычным образом привели к работе с параллельными вычислениями [5].

CFD становится актуально значимым компонентом в исследовании индустриальных товаров и систем. CFD коды структурированы численным алгоритмом, рассматривающим проблемы жидкости потока. В работе Lee рассмотрены три основных компонента CFD коды для предоставления полезной информации: предварительная обработка, решатель, постобработка [6]. На рисунке 1.1 приводится блок-схема подхода, которая показывает общие требования, предъявляемые к выполнению исследования CFD.

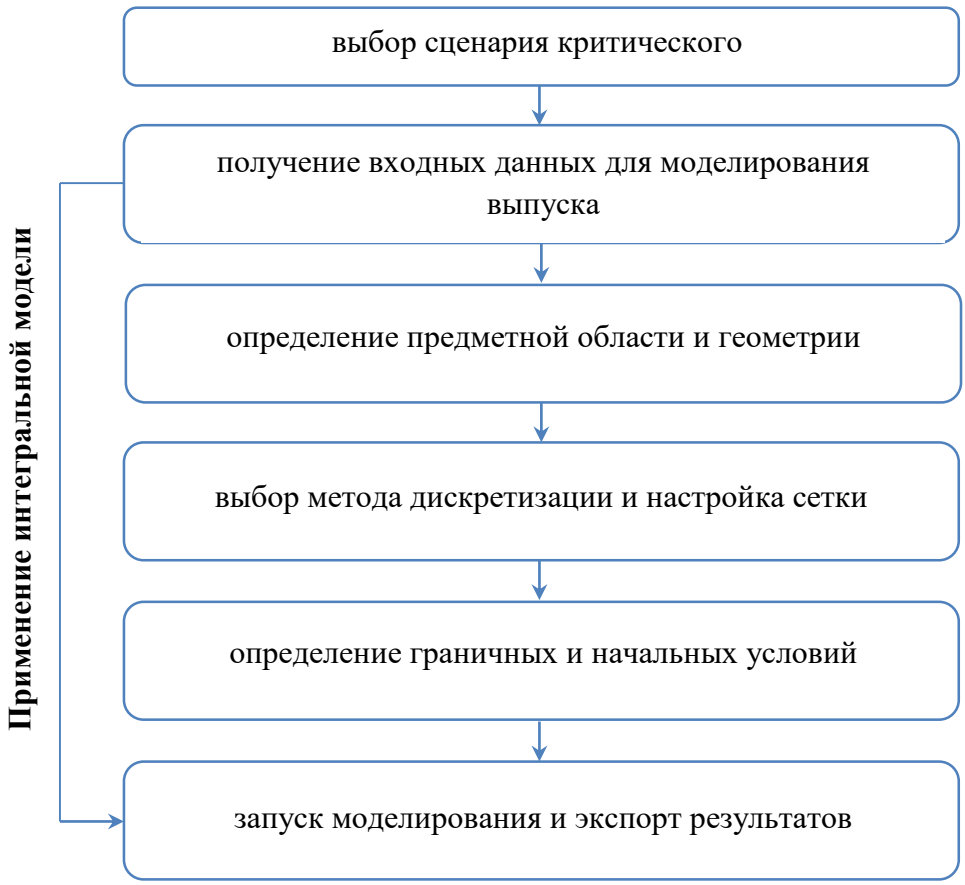


Рисунок 1.1 – Блок-схема подхода и общие требования для выполнения исследования CFD

В зависимости от сложности задачи CFD для представления и решения используются структурированные или неструктурированные сетки. Алгоритмы эффективно отображают коды структурированной сетки на параллельные машины, которые были разработаны за прежнее десятилетие и получили широкое признание. Моделирование гидродинамики вычислительной техники CFD сегодня широко используется в качестве инструмента для разработки предприятий промышленности, оптимизации процесса. Благодаря постоянному увеличению мощности компьютеров инженеры-технологи смогли моделировать реагирующие многофазные потоки в реалистичной геометрии с хорошим разрешением сетки [7].

В области суперкомпьютерных архитектур высокопроизводительные вычисления (HPC) не обладает строгим определением, но, как правило, относятся к (массово) параллельной обработке, предназначенной для быстрых выполнений прикладных программ. Это означает, что вычислительная система выполняет множество операций с плавающей запятой в секунду (FLOPS). Хотя HPC можно исполнять на одной системе, действительно сильные потенциалы обнаруживаются при употреблении кластеров из нескольких высокопроизводительных вычислительных узлов, также называемых суперкомпьютерами. По последней информации сайта https://www.top500.org/

по состоянию на июнь 2021 года 10 групп представили списки суперкомпьютеров [8]:

Первое место занимает японский суперкомпьютер Fugaku. Система Fugaku, разработанная Riken и Fujitsu, имеет результат теста HPL (High-Performance Linpack) 442 Pflop/s, у него 7630848 ядро. Суперкомпьютер основан на процессоре ARM A64FX от Fujitsu.

На втором месте в мире можно определить систему Summit с производительностью 148,8 Pflop/s, созданная IBM в национальной лаборатории Oak Ridge (ORNL) в США.

На третьем месте - система Sierra состоящая из 4320 узлов с двумя процессорами Power 9 и четырьмя графическими процессорами NVIDIA Tesla V100, имеющая результат теста HPL 94,6 Pflop/s, которая была разработана в США в Ливерморской национальной лаборатории Лоуренса.

Система Sunway TaihuLight занимает четвертое место с результатом теста HPL 93 Pflop/s, которая была разработана Национальным исследовательским центром параллельной вычислительной техники и технологий Китая.

Perlmutter с показателем 64,6 Pflop/s на пятом месте. Perlmutter основан на платформе HPE Cray «Shasta», гетерогенной системе с узлами 1536 с ускорением NVIDIA A100 и с узлами на базе AMD EPYC.

Система под названием Selene, установленная на предприятия NVIDIA в США, занимает шестое место. Selene разработана NVIDIA DGX A100 SuperPOD, основана на процессоре AMD EPYC с NVIDIA A100. Система набрала 63,4 Pflop/s.

Tianhe-2A (Milky Way-2A) на седьмом месте с производительностью 61,4 Pflop/s. Она разработана в Национальном университете оборонных технологий Китая.

"JUWELS Booster Module" или BullSequana под номером восемь, установлена в Forschungszentrum Juelich (FZJ) в Германии и создана Atos. Для ускорения данной системы применяется процессор AMD EPYC с NVIDIA A100, а также Mellanox HDR InfiniBand для сети, аналогичной системе Selene. BullSequana набрал 44,1 Pflop/s и является самой мощной в Европе.

Система PowerEdge, которая была создана Dell и установлена итальянской компанией Eni Spa занимает девятое место, обеспечивающая производительность 35,5 Pflop/s за счет применения NVIDIA Tesla V100 в качестве ускорителей, также Mellanox HDR InfiniBand в качестве сети.

На десятом месте с результатом 23,5 Pflop/s с использованием 448 448 ядер Intel Xeon находится американская система Frontera (Dell C6420).

Анализируя 5 лучших суперкомпьютеров в мировом рейтинге топ-500 как видно в таблице 1.1, суперкомпьютеры имеют больше графических процессоров, чем центральные процессоры.

Таблица 1.1 - Список вычислительных систем из топ-500 мирового рейтинга суперкомпьютеров

Рейтинг	Суперкомпьютеры	CPU	GPU
1	FRONTIER	9,400 (AMD Epyc)	37,632 (AMD Radeon Instinct)
3	LUMI	2,560 (AMD Epyc)	10,240 (AMD Radeon Instinct)
4	SUMMIT	9,216 (IBM POWER9)	27,648 (NVIDIA Tesla V100)
5	SIERRA	(IBM POWER9)	4320 (NVIDIA Tesla V100)
6	Sunway TaihuLight	40960 SW26010	3750 (NVIDIA GPU or Intel Xeon Phi

Суперкомпьютеры из этого списка помогают быстро получить численное решение прикладных задач.

1.2 Анализ систем параллельных вычислений для прикладных задач

Современные системы реального времени нуждаются в эффективном решении прикладных задач. Достаточная дискретизация требуется для получения желаемых результатов численного моделирования и исследования области потока жидкости. Для получения результатов с достаточной точностью для численного моделирования и исследования состояния потока жидкости проблемная область должна быть достаточно дискретизирована. Основными уравнениями движения жидкости являются уравнения в частных производных, которые трудно решить аналитическим путем. Поэтому часто используются численные методы. Моделирование CFD обычно требует значительных вычислительных ресурсов для точных решений, особенно для сложных сценариев моделирования, таких как околозвуковые или турбулентные потоки. Высокопроизводительные вычислительные платформы (HPC) позволяют создавать быстрые и лучшие решения. Поэтому распараллеливание численного моделирования гидродинамики с самого начала является привлекательной темой исследования в этой области [31]. Большинство современных компьютерных систем имеют несколько ядер, обращающихся к общей памяти. В исследовательских вычислениях обычно объединяют несколько архитектур с общей памятью в кластер. Кластеры могут представлять собой важный ресурс для тех исследователей, которые могут в полной мере использовать их. Для достижения целей одним из наиболее распространенных методов является параллельное программирование [9]. Чтобы получить численное решение, время вычислений очень велико, и параллельные вычисления применяются, чтобы сократить время вычисления. При параллельных вычислениях для одновременного решения прикладных задач программа разбивается на несколько частей, и каждая часть должна выполняться одновременно на разных

процессорах. Параллельное вычисление является способом организации компьютерного вычисления, в котором программы разработаны как набор взаимосвязанных процессов вычисления, работающих параллельно. Для распараллеливания программного обеспечения используются центральные процессоры (CPU) и графические процессоры (GPU) [9]. Чтобы распараллелить программное обеспечение используются разные технологии OpenMP, MPI, OpenCL, OpenHMPP, CUDA, OpenACC, Apache Hadoop, Apache Spark, Apache Flink, Cilk Plus, FastFlow, Erlang. OpenMP, MPI, CUDA наиболее широко используемые технологии для распараллеливания программного обеспечения CFD. И еще существует различные коммерческие программные пакеты CFD, где можно моделировать: ANSYS FLUENT, OPENFOAM, CFX, STARCCM и т.д.

В последние годы для ускорения вычислительного кода CPU использовали гетерогенные кластеры с многоядерными процессорами и графические процессоры. Гетерогенные вычисления являются аппаратным ускорителем и одним из быстро развивающихся ведущих подходов, которые приводят к повышению вычислительной мощности и эффективности современных систем высокопроизводительных вычислений. Как известно, распространенная форма аппаратных ускорителей — это графические процессоры. В последнее время для увеличения своих вычислительных возможностей использования ускорителей, таких как графические процессоры NVIDIA для кластеров HPC становится все более популярным [11].

Распределенная память и общая память - распространенные методы, чтобы получить параллельные вычисления, а эти два метода используются в гибридных вычислениях. Существует несколько исследований, которые были посвящены внедрению гетерогенных вычислений с использованием параллельных алгоритмов MPI+OpenMP/Cuda, OpenMP+Cuda, MPI+OpenACC. Например, Song et al. [11] продемонстрировали для решения двумерного уравнения переноса нейтронов методом характеристики (MOC), Altybay и др. [12] разработали код для двумерного уравнения акустической волны, Gimenez et al. [13] разработали код для метода эффективности BSC, Polyakov и др. [14] провели распараллеливание нового численного подхода при решении многомасштабных задач технической газовой динамики и т.д.

1.2.1. Расчет с помощью многоядерных CPU (Central Processing Units)

Многоядерные процессоры имеют два или более процессора в одном интегрированном чипе, который содержит несколько блоков обработки ядра. Чтобы использовать многоядерный процессор на полную мощность, приложения, выполняемые в системе, должны быть многопоточными, чтобы добиться всех преимуществ и эффективности, необходимо научиться писать параллельные приложения. Методы параллельного программирования получают выгоду из использования нескольких ядер, а программистам распараллеливание помогает эффективно решать вычислительные задачи. Существующие технологии параллельного программирования OpenMP, MPI,

OpenHMPP, Click Plus, FastFlow, Erlang могут использоваться на многоядерных платформах. Каждый инструмент подходит лишь определенной параллельной модели и ни один инструмент не подходит для распараллеливания всех видов приложений [15].

OpenMP [16] является стандартом для распараллеливания программ на языках Fortran и C/C++ для архитектур с общей памятью, где параллельная программа строится из своего последовательного аналога путем добавления прагм компилятора. В OpenMP есть способность из любого запущенного потока создавать собственную команду дочерних потоков, и это является вложенным параллелизмом. Неправильное использование вложенного параллелизма может привести к превышению лимита ресурсов процессора и к значительному снижению производительности. Например, Ouro и др. [17] распараллеливали масштабируемость эйлерово-лагранжевого решателя для моделирования больших вихрей, Hückelheim и др. [18] представляет алгоритмическое дифференцирование в обратном режиме, Shan и др. [19] исследовали новые реализации эффективных приближений функции Грина и показали численную эффективность. На рисунке 1.2 показана модель разветления и соединения с помощью OpenMP для параллельного выполнения различных задач.

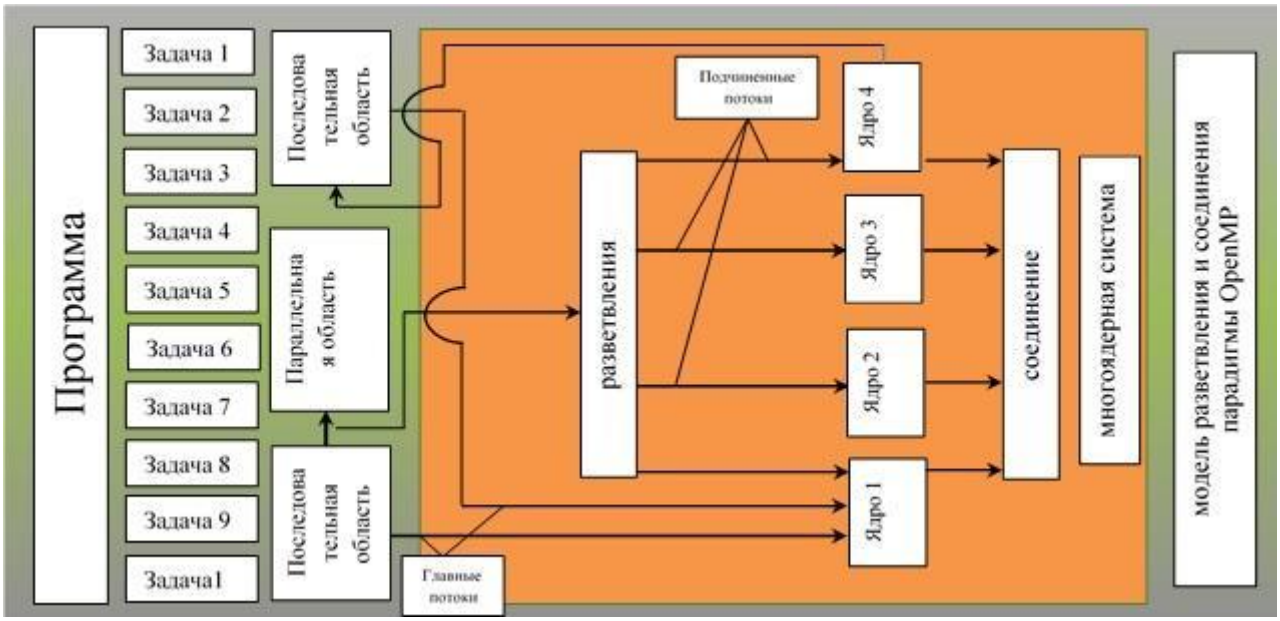


Рисунок 1.2 – Парадигма OpenMP

MPI [20] — это эффективная стандартная модель программирования для распределенных вычислений, которая предоставляет явный метод передачи сообщений методом программирования. Реализация MPI предназначена для работы в различных параллельных средах и поддерживания классических коммуникаций. На рисунке 1.3 показано как передается сообщение на разные процессоры с помощью MPI для параллельного выполнения различных задач в параллельной области программы. Каждый процессор обменивается данными с

помощью сообщений по сети для передачи данных друг другу. В программном интерфейсе MPI (API) есть множество типов связи, таких как коллективная связь и связь точка-точка. Коллективные вызовы MPI приводят к обмену данными между всеми вычислительными ядрами в группе. Эти связи могут быть получены с помощью блокирующих или неблокирующих протоколов. В протоколе блокировки выполнение программы откладывается до тех пор, пока буфер сообщения не станет безопасным для использования, а в протоколах без блокировки выполнение программы не ожидается для того, чтобы убедиться в безопасности использования буфера связи. Вызовы MPI "точка-точка" связаны с обменом данными между двумя конкретными процессами. Неблокирующая связь позволяет выполнять вычисления сразу после вызова связи MPI [21].

Во многих научных работах реализовали параллельные алгоритмы, используя библиотеку MPI, которая включает архитектуру параллельных компьютеров как с общей, так и распределенной памятью: для точного описания ультракоротких импульсов в оптических волокнах [2], в явных методах конечной разницы для PDE в многоядерных и мультиядерных системах с общей памятью [23], параллельное создание неэрмитовой матрицы, вычисленных по заданным спектрам для сравнения решателей [24], PAMPAR - набор тестовых программ для оценки производительности и энергопотребления, поддерживающий интерфейсы параллельного программирование[25].

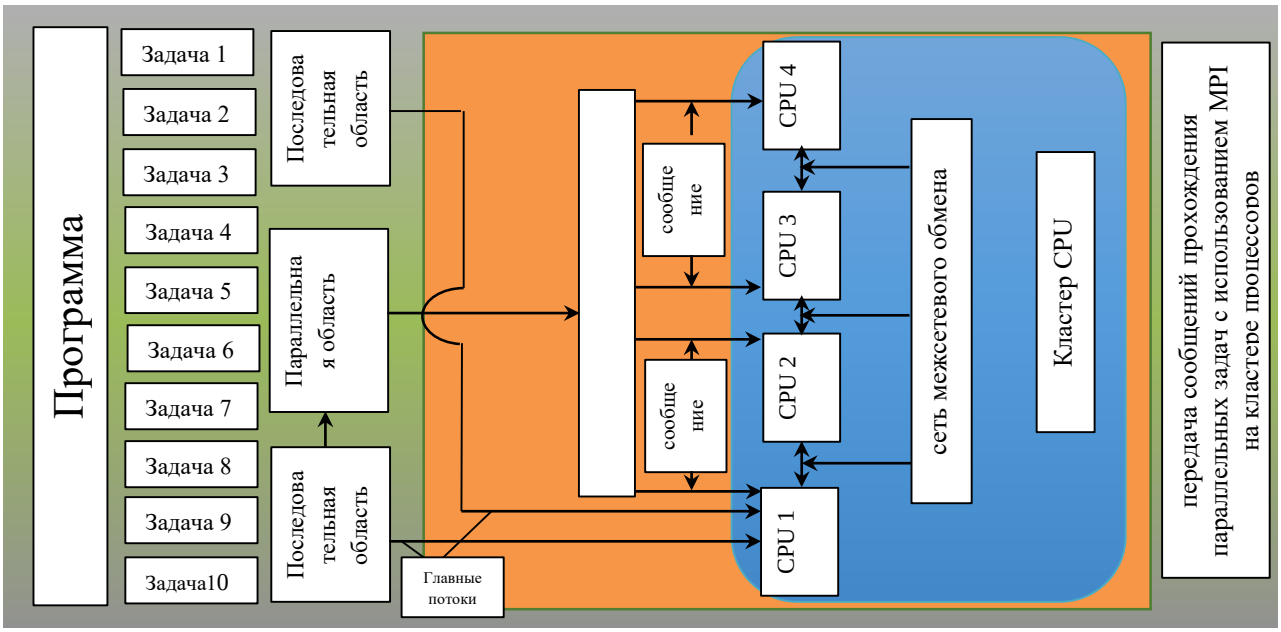


Рисунок 1.3 – Парадигма MPI

В вычислениях CFD на основе MPI сетка делится на множество внутренних слоев и распределяется между различными процессорами. Данные передаются между соседними внутренними рядами интерфейса конфигурации CFD.

1.2.2. Основные принципы расчетов на GPU, технология Compute Unified Device Architecture (CUDA)

GPU является мощной моделью параллельного программирования, в которой видеокарты используются в качестве исполнительного устройства, включающего большое количество высокопроизводительных процессоров. Базовая архитектура GPU состоит из набора многопоточных процессоров (SM), каждый из которых имеет несколько потоковых процессоров (SP). Все SP (тонкие ядра) внутри SM (толстые ядра) следуют одним и тем же инструкциям, но они генерируются различными экземплярами данных (одна инструкция, несколько данных) в соответствии с моделью SIMD (Single Program Multiple Data). Количество SP и SM на GPU отличается от одной видеокарты к другой. GPU поддерживает тысячи легких параллельных потоков, и, в отличие от потоков CPU, затраты на создание и переключение потоков незначительны. Потоки на каждом SM организованы в группы потоков, которые совместно используют регистры. Группа потоков делится на несколько блоков таблиц, называемых динамически запланированными деформациями в SM. Все нити из одного блока выполняются на одном мультипроцессоре (SM). В зависимости от характера SIMD модулей SP выполняются последовательно, а не параллельно, если потоки в табличном модуле выполняют различные действия, такие как замена ветвей. Кроме того, если поток перестает работать с памятью, вся деформация прекращается до окончания доступа к памяти. В этом случае SM выбирает другую готовую базу и переходит к ней. Объем глобальной памяти GPU обычно измеряется в гигабайтах. Это внешняя память с высокой пропускной способностью и высокой задержкой доступа. Чтобы компенсировать высокую задержку этой памяти, важно иметь больше потоков, чем количество SPS, и иметь доступ к последовательным адресам памяти, которые можно легко интегрировать в warp.

Чтобы понять модель памяти и иерархию потоков GPU, рассмотрим рисунок 1.4, потоки организованы в блоки, и все блоки принадлежат сетке. Графический процессор имеет различные области памяти: общую, глобальную, постоянную, текстурную и локальную. У каждого есть свои преимущества и недостатки. Общая память кэшируется, поэтому она имеет более быстрое время доступа, но она меньше (16 КБ) и доступна только для блока. Общая память используется для загрузки части задачи из глобальной памяти в общую память, проведения многочисленных вычислений данных из общей памяти и последующей записи результата обратно в глобальную память. Глобальная память имеет большой размер (4 ГБ) и доступна для всех потоков всех блоков, но не кэшируется, что приводит к медленному времени доступа. Постоянная память кэшируется, но только для чтения и имеет небольшой размер (64 КБ); это память полезна, если есть справочные таблицы.

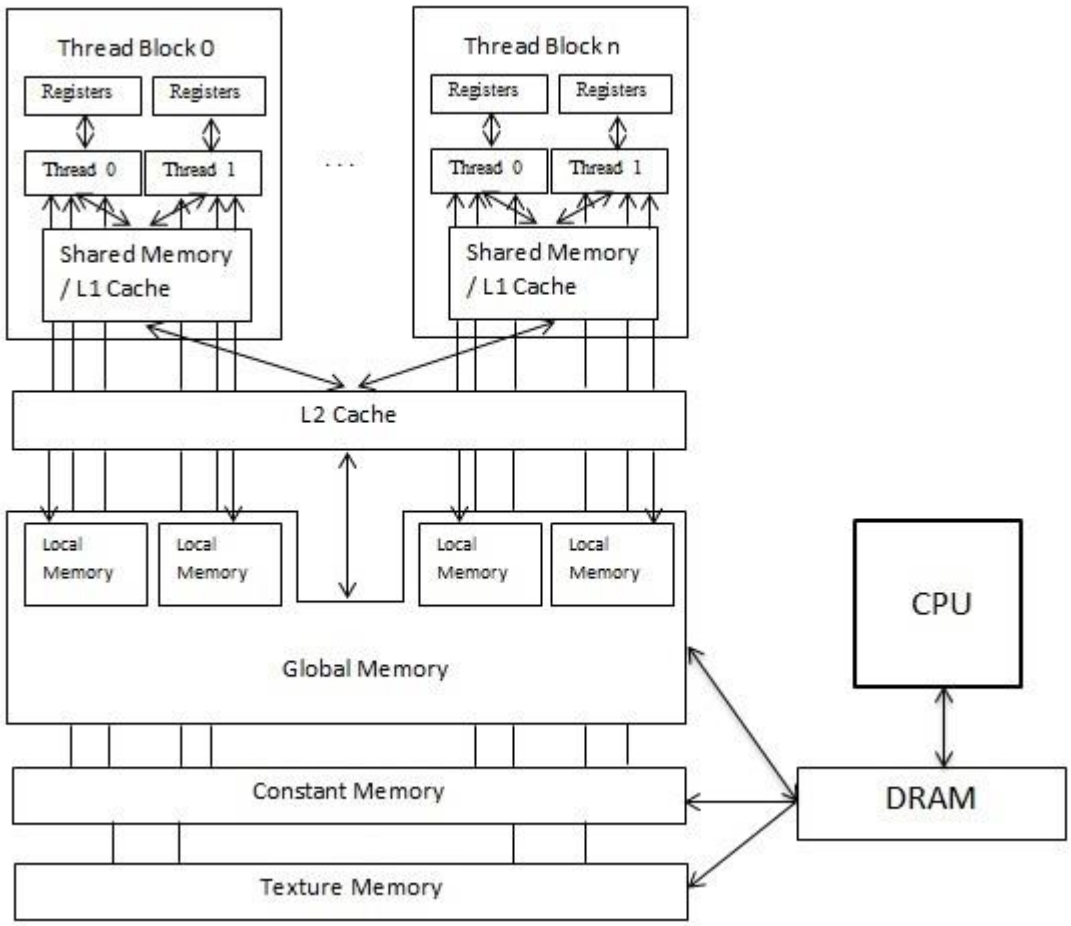


Рисунок 1.4 – Модель памяти и иерархия потоков графического процессора

Глобальная память устройства CUDA реализована с использованием DRAM. Каждый раз при доступе к ячейке DRAM фактически осуществляется доступ к диапазону последовательных ячеек, включая запрошенную ячейку. Каждый чип DRAM имеет множество датчиков, которые работают параллельно. Каждый из них определяет содержимое бита в этих последовательных местах. После обнаружения датчиками данные из всех этих последовательных мест могут быть переданы на процессор с очень высокой скоростью. Доступ к этим последовательным расположениям и передача данных называются пакетами DRAM. Спецификатор __global__ используется для функций, определяющих ядро, и им передается несколько конкретных переменных.

Локальная память расположена в глобальной памяти и может работать в 150 раз медленнее, чем регистровая или общая память. Доступна только для потока. Имеет время жизни потока.

Разделяемая память встроена в чип, она намного быстрее, чем локальная и глобальная память. Фактически, задержка в разделяемой памяти примерно в 100 раз меньше, чем задержка в глобальной памяти без кэширования (при условии отсутствия конфликтов между потоками, о которых мы поговорим позже в этом посте). Потоки могут получать доступ к данным в разделяемой

памяти, загруженным из глобальной памяти другими потоками в том же блоке потоков. Эта возможность (в сочетании с синхронизацией потоков) имеет ряд применений, таких как управляемые пользователем кэши данных, высокопроизводительные параллельные алгоритмы (такие как параллельные сокращения) и для облегчения глобального объединения памяти в случаях, когда в противном случае это было бы невозможно. Для размещения в разделяемой памяти используется спецификатор __shared__.

Константная память. Данные хранятся в глобальной памяти устройства, и их можно прочитать через постоянный кэш мультипроцессоров. Каждый мультипроцессор оснащен 64 КБ константной памяти и 8 КБ кэша. Данные отправляются всем потокам в группе. Таким образом, если все потоки в группе запрашивают одно и то же значение, оно будет передано за один цикл.

Текстурная память - это еще один тип памяти, доступной только для чтения. С появлением CUDA сложная текстурная память графического процессора также может использоваться для вычислений общего назначения. Хотя текстурная память изначально была разработана для рендеринга в OpenGL и DirectX, она обладает свойствами, которые делают ее очень полезной для вычислительных целей. Как и память, доступная только для чтения, текстурная память кэшируется внутри чипа, поэтому она может обеспечить более высокую эффективную пропускную способность, чем внекристальная DRAM. В частности, текстурные кэши предназначены для схем доступа к памяти, характеризующихся высокой пространственной локальностью.

Графический процессор также обеспечивает быструю кристаллическую память, доступную для всех SP SM. Этот объем памяти невелик, но он имеет задержку и может использоваться в качестве кэша, управляемого программным обеспечением. Передача данных от CPU к GPU и наоборот осуществляется через соединение PCI Express. Программа на GPU показывает параллельность с помощью функции ядра SPMD (Single Program Multiple Data) для параллельных данных. Программист может настроить количество используемых потоков. Каждый поток имеет идентификационный номер, с помощью которого программист может сопоставить его с GPU с помощью ядра. Потоки вместе с данными проводят параллельные вычисления ядра и в дальнейшем организуются в группы (блоки потока), организованные в сеточную структуру. При запуске ядра блоки в сетке делятся на пустые SM, а потоки соответствуют SP. Потоки в блоке потоков выполняются SP одного SM и могут обмениваться данными через общую память SM. Кроме того, каждый поток внутри блока имеет свои собственные регистры (частная локальная память) и использует глобальный индекс блока потока и индекс локального потока в блоке потока, чтобы однозначно идентифицировать свои данные. Потоки, принадлежащие к разным блокам, не могут четко взаимодействовать и полагаться на глобальную память для обмена результатами [26-27]. Данное направление было исследовано такими учеными как R. Gaioso, J.D. Owens,V. Gil-Costa, D. Luebke, H.Guardia, N. Govindaraju, A.E. Lefohn, S.R.M. Rostami, M. Harris, M.Ghaffari, J. Krüger.

Базовые алгоритмы различных кодов CFD и их реализация на графическом процессоре требуют различных методов для оптимизации производительности. Ряд исследований посвящен реализации CFD-кодов на GPU. Их можно разделить на два раздела: исследования с использованием CFD без сетки [29, 30], и исследования с использованием CFD на основе сетки [4, 10, 17, 28, 31, 32, 83]. В работе Mintu и др. [33] авторы применили CFD-код без сетки, чтобы оценить производительность вычисления. Традиционная реализация CFD на основе сетки на графическом процессоре в работах других исследователей показала значительное сокращение времени вычислений [31].

GPU появился в игровой индустрии, в настоящее время широко используется в качестве программируемых механизмов с использованием инструментов программирования Compute Unified Device Architecture (CUDA), Open Accelerator (OpenAcc), Open Computing Language (OpenCL), в частности, CUDA. CUDA необходима для связи с GPU и использует стандартный язык C для выполнения алгоритмов на GPU. Возможность многопоточности делает GPU очень эффективным в вычислительном отношении. CUDA-популярный инструмент параллельных вычислений NVIDIA, который позволяет создавать большие параллельные приложения [46].

Для решения физико-технических задач используется типичный шаблон:

- делится на подзадачи;
- входные данные разбиваются на блоки для размещения в разделяемой памяти;
- из глобальной памяти подблок загружается в разделяемую память;
- соответствующие вычисления выполняются над данными в разделяемой памяти;
- результаты копируется из разделяемой памяти в глобальную память.

Для использования GPU в CUDA сначала задаются размеры блоков и сетки, а затем вызывается ядро (код, выполняемый на GPU) из основной программы. Код в ядре выполняется каждым потоком во всех блоках. Таким образом, одни и те же инструкции выполняются несколькими потоками, где каждый поток соответствует различным частям данных. Ядро определяется глобальным спецификатором уведомлений, и количество потоков CUDA для каждого вызова отображается с помощью нового синтаксиса <<<...>>>. Чтобы получить сетку в графическом процессоре, показанную на рисунке 1.4, где переменные block и grid, содержат размеры блока и сетки по следующей структуре, показанной на рисунке 1.5:

```
dim3 block(2,1);
dim3 grid(2,1);
kernel<<<grid, block>>>(var1, var2, ...);
```

Рисунок 1.5 – Размеры блока и сетки

Как показано на рисунке 1.6, модель реализации CUDA имеет следующие пять уровней: выделение памяти на хосте и устройстве; передача данных с хоста на устройство; выполнение ядра (вычисление устройства); передача результата с устройства на хост; освобождение памяти [28].

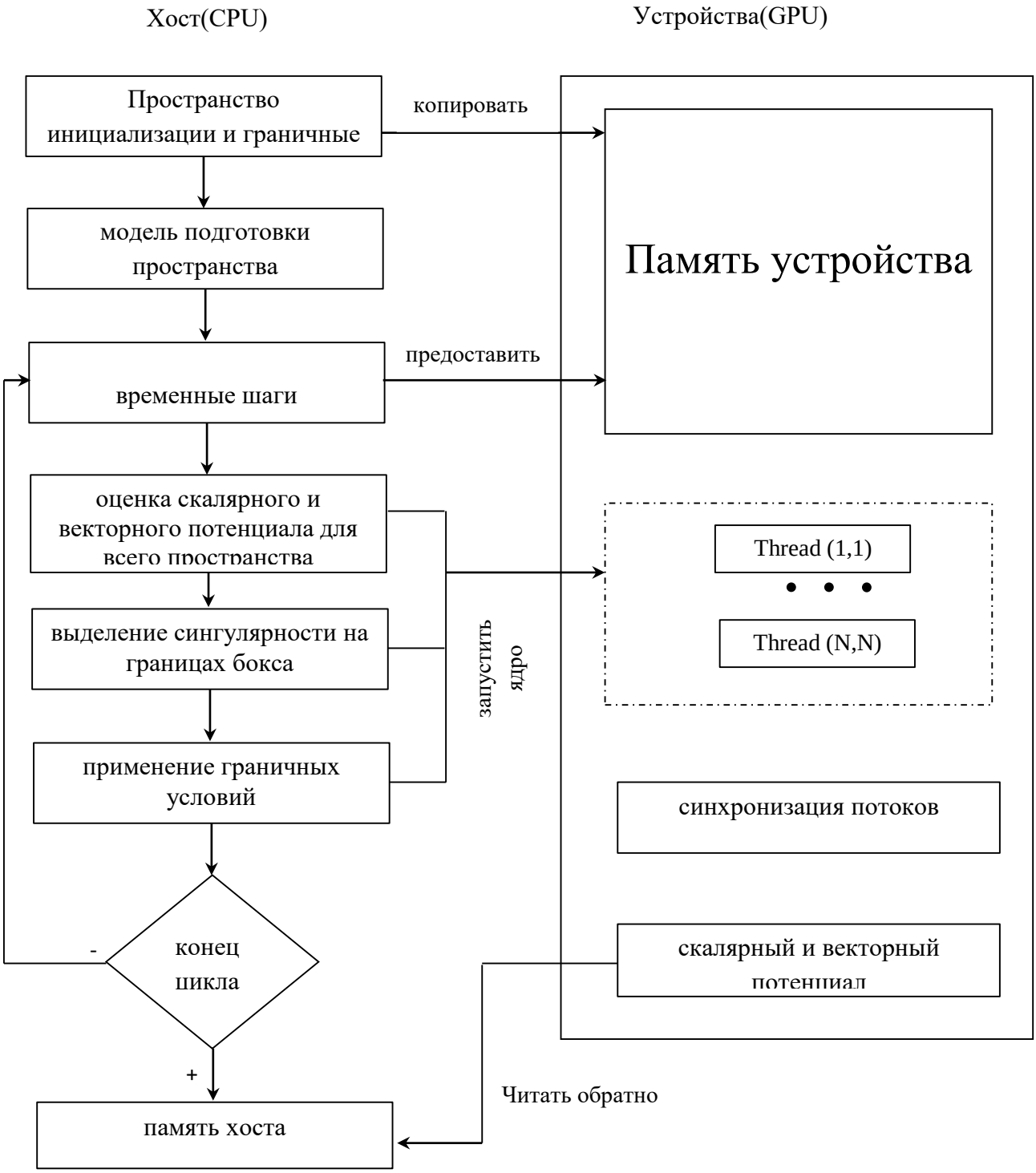


Рисунок 1.6 – Алгоритм реализации CUDA

При нахождении максимума очень больших массивов с использованием графических процессоров используется параллельная редукция, которые

объединяют массив элементов, в результате чего получается одно значение. Параллельная редукция имеет несколько операций, то есть операцию сложения, вычитания и умножения. Параллельная редукция является одним из видов алгоритмов параллельной машины с произвольным доступом (PRAM). Параллельная машина с произвольным доступом (PRAM) представляет абстрактную машину с общей памятью, которая используется разработчиками параллельных алгоритмов для оценки производительности алгоритма. В оптимизации ядро сокращения CUDA из глубокого погружения Mark Harris, идея состоит в том, чтобы использовать несколько блоков потоков в графическом процессоре для небольшой части массива. Внутри каждого блока потока используется сокращение на основе дерева, а общая память используется для обеспечения связи между потоками одного блока и синхронизации между ними (см. рисунок 1.7).

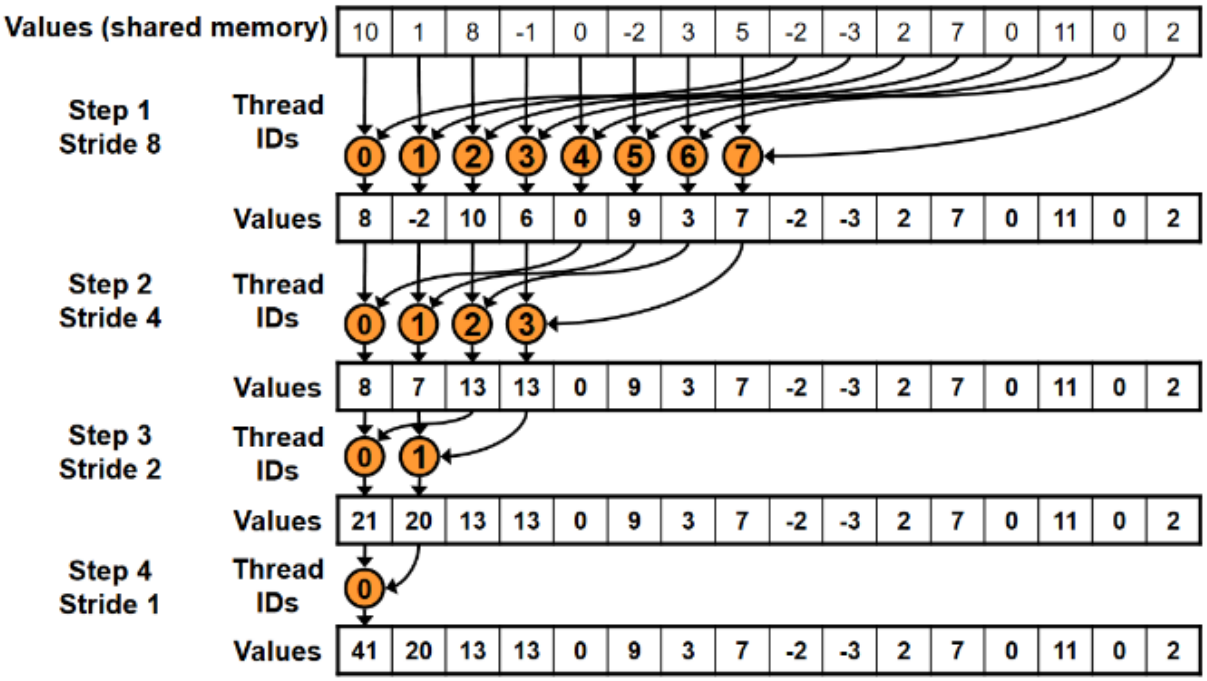


Рисунок 1.7 – Параллельная редукция: адресация с чередованием

1.3 Анализ и оценка эффективности высокопроизводительных вычислений на CPU и GPU

Большинство практических CFD-моделирований требует больших вычислительных ресурсов и время моделирования можно сократить двумя способами. Наиболее распространенным подходом является использование высокопроизводительных вычислений (HPC) в суперкомпьютерах, состоящих из нескольких ядер процессора. Многие инженерные фирмы либо используют отдельный кластер из 8–32 ядер или арендуют массивный кластер с помощью облачных вычислений. Общая стоимость владения (TCO) кластера

относительно высока, а расходы на обслуживание являются еще одним бременем. Облачные вычисления на общедоступном сервере немного дешевле, но защита конфиденциальной информации с помощью IP (интеллектуальной собственности) небезопасна. В качестве альтернативы моделирование можно дешево ускорить за счет использования новых вычислительных архитектур, таких как универсальные вычисления на графических процессорах (GPGPU), называемые GPU. Еще одна привлекательная особенность GPU - более низкая стоимость и простота обслуживания по сравнению с большими многоядерными системами HPC. Увеличение вычислительной мощности и более низкая стоимость GPU растут быстрее, чем у традиционных CPU [33].

Современные графические процессоры могут обеспечить пропускную способность памяти и производительность операций с плавающей запятой, что на несколько порядков выше, чем у стандартного центрального процессора. На рисунке 1.8 сравниваем производительность CPU и GPU.

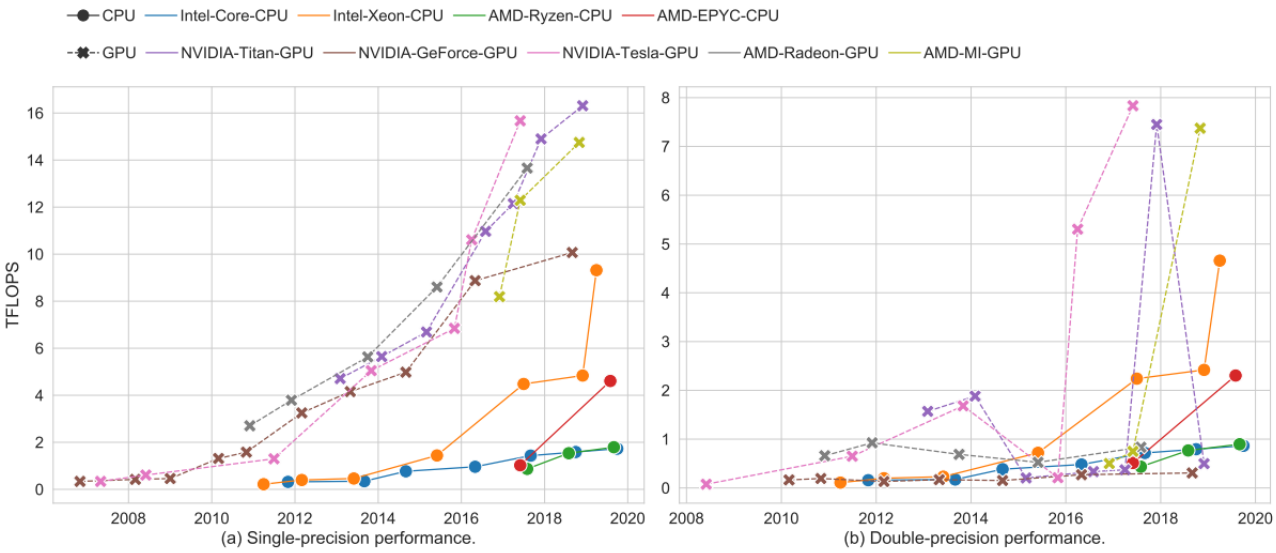


Рисунок 1.8 – Сравнение производительности процессоров и графических процессоров с одной и двойной точностью

В данном анализе рассматриваются только теоретические вычислительные возможности. Вычислительные возможности включают вычислительную мощность, которую могут предоставить устройства при условии использования всех вычислительных ресурсов. Однако не всегда все ресурсы могут быть использованы в реальных приложениях, и использование ресурсов зависит от аппаратного и программного обеспечения. В целом, GPU сохраняет значительные преимущества с точки зрения производительности с одной точностью по сравнению с CPU потребительского уровня.

Некоторые исследователи реализовали свои задачи на графических процессорах и центральных процессорах, включая (но не ограничиваясь) метод конечных разностей во временной области (FDTD) [28, 34-36], метод конечных

элементов (FEM) [37, 38], методы на основе интегральных уравнений [39 - 42], конечно-разностные явные методы [23] и метода конечных объемов [10, 43, 44].

В таблице 1.1 показано ускорение предложенных моделей параллельного программирования при реализации научных задач ученых в научно-исследовательской работе.

Таблица 1.2 - Ускорение при реализации научных задач исследователей

Авторы, год	Ускорение CPU (с оптимизацией)	Ускорение GPU
Rostami, S. R. и др., 2019	6.14x	96.53x
Mohammadi, S. и др., 2020	2,43	21x
Rodrigues, M. J. и др., 2018	-	65x
Xiong, L. и др., 2018	-	26x
Ayyad, M. и др., 2020	3x	33x
Zhang, W. и др., 2021	-	11x
Laqua, H; 2021	-	7x
Yamazaki, I., 2018	-	4x
Yoshikawa, T., 2019	-	53x
Xu, J., 2019	-	93.9x, 70.4x
Weng, Y., 2021	1x	21,7x
Afzal, A., 2021	1,35	8x
Wang, S. и др., 2020	6x	64x

Любой компьютер с картой GPU с поддержкой CUDA может использоваться как устройство высокопроизводительных вычислений (HPC). Кроме того, портативный графический процессор GPU можно подключить к ноутбуку, чтобы он работал так же хорошо, как рабочая станция. Более того, совокупная стоимость владения больших кластеров очень высока, а графические процессоры являются очень дешевой альтернативой, которая также потребляет меньше энергии. Несмотря на то, что в отрасли есть много достижений, доступ к мощным возможностям моделирования CFD по разумной цене остается общей проблемой. Вычисления на графическом процессоре могут значительно сэкономить на моделировании и получении численного решения.

Это, вероятно, дешевле облачных вычислений, учитывая стоимость хранения и передачи данных, а также стоимость программного обеспечения [33]. Была применена оптимизация кода, и, в конце концов, ясно показано, что реализация CUDA на сегодняшний день приводит к лучшему ускорению. Как правило, графические процессоры обладают значительным потенциалом в качестве устройств для параллельных вычислений.

Заключение по 1 разделу.

В данном разделе показан анализ современных систем высокопроизводительных вычислений для прикладных задач. Описаны расчеты с помощью многоядерных CPU и основные принципы расчетов на GPU. В конце этой главы мы представляем анализ и оценку эффективности высокопроизводительных вычислений на CPU и GPU.

2 ОСНОВНЫЕ УРАВНЕНИЯ ГИДРОДИНАМИКИ

2.1 Уравнение неразрывности

В динамике дискретной системы материальных точек динамические характеристики, в том числе и инерционные, приписывались отдельным точкам системы. В случае абсолютно твердого тела рассматривались как суммарные, но относящиеся к резко очерченным областям твердого тела характеристики, так и сосредоточенные в конкретных точках твердого тела силы, моменты сил. В динамике сплошных сред, газов, жидкостей, упругих и других твердых деформируемых тел, такой подход заменяется заданием непрерывных распределений динамических и физические величины по сплошной среде, характеризующемуся плотностью распределения этих величин.

Примером такой является плотность распределения массы в виде предела отношения массы Δm малого объема $\Delta\tau$, содержащего данную точку М, к объему $\Delta\tau$, когда последний стремится к нулю, стягиваясь к точке М. Отношение $\frac{\Delta m}{\Delta\tau}$ называется средней плотностью и объемом $\Delta\tau$, а предел этого отношения при $\Delta\tau \to \infty$ есть плотность среды в данной точке М и обозначается буквой ρ, что

$$\rho = \lim_{\Delta\tau \to 0} \frac{\Delta m}{\Delta\tau} \tag{2.1}$$

Стягиваем к точке М и получаем:

$$\rho = \lim_{\Delta\tau \to 0} \frac{\Delta m}{\Delta\tau} = \frac{\delta m}{\delta\tau} \tag{2.2}$$

Для данного уравнения (2.2) следуют выражения массы элементарного объема через плотность ρ:

$$\delta m = \rho \delta\tau \tag{2.3}$$

и массы m выделенного в среде конечного объема τ:

$$m = \int_{\tau} \rho \delta\tau$$

Дифференцируем уравнение (2.3) по времени и получаем:

$$\frac{d}{dt}(\delta m) = \frac{d}{dt}(\rho \delta\tau) = \frac{d\rho}{dt}\delta\tau + \rho\frac{d}{dt}(\delta\tau) = 0$$

Учитывая, что $\frac{d}{dt}(\delta m) = 0$ (масса не меняется по времени), воспользуемся определением:

$$\frac{d}{dt}(\delta\tau) = div\, V \cdot \delta\tau$$

Учитывая вышестоящее выражение, можно будет переписать уравнение в виде:

$$\frac{d\rho}{dt}\delta\tau + \rho div\, V \cdot \delta\tau = 0$$

В конце можно будет записать в виде:

$$\left(\frac{d\rho}{dt} + \rho div\, V\right)\delta\tau = 0 \Rightarrow \frac{d\rho}{dt} + \rho div\, V = 0$$

Раскрывая уравнения с учетом полной производной, можно будет записать в виде:

$$\frac{d\rho}{dt} = \frac{\partial\rho}{\partial t} + V\ grad\,\rho = \frac{\partial\rho}{\partial t} + u\frac{\partial\rho}{\partial x} + v\frac{\partial\rho}{\partial y} + w\frac{\partial\rho}{\partial z}$$

Тогда уравнение неразрывности можно будет записать в виде:

$$\frac{\partial\rho}{\partial t} + V\, grad\,\rho + \rho\, div\, V = 0$$

Теперь пользуемся данными свойствами:

$$div(\rho V) = V\, grad\,\rho + \rho\, div V$$

В конечном виде уравнение неразрывности для сжимаемого случая будет выглядеть в виде:

$$\frac{d\rho}{dt} + div\,(\rho V) = 0$$

Для несжимаемого случая, пользуясь свойством $\frac{\partial\rho}{\partial t} = 0$. уравнение неразрывности можно будет переписать в виде:

$$div\,\rho V = 0 \text{ или } div\,V = 0$$

В конечном случае уравнение неразрывности для несжимаемого случая будет в следующем виде:

$$div\,V = \frac{\partial u}{\partial x} + \frac{\partial \nu}{\partial y} + \frac{\partial w}{\partial z} = 0.$$

2.2 Уравнения движения сплошной среды

Теорема о количестве движения: Индивидуальная производная о главном векторе количества движения жидкого объема и поверхностных сил, действующих на частицы, находящиеся в объеме и на ограничивающей поверхности соответственно:

$$\delta k = \rho V \delta\tau,$$

δk - количество движение.

Главный вектор количества движения во всем объеме τ равен сумме или интегралу элементарных количеств движения:

$$k = \int_{\tau} \rho V \delta\tau$$

Главные векторы внешних объемных и поверхностных сил будут соответственно равны:

$$\int_{\tau} \rho F \delta\tau \text{ и } \int_{\sigma} p_n \delta\sigma$$

Таким образом, только по теореме об изменении количеств движения приводит к равенству:

$$\int_{\tau} \rho V \delta\tau = \int_{\tau} \rho F \delta\tau + \int_{\sigma} p_n \delta\sigma$$

Дифференцируем по времени первое слагаемое слева уравнение и получаем:

$$\frac{d}{dt}\int_{\tau} \rho V \delta\tau = \int_{\tau} \rho F \delta\tau + \int_{\sigma} p_n \delta\sigma \qquad (2.4)$$

В левой части уравнения (2.4) выполним преобразование:

$$\frac{d}{dt}\int_{\tau}\rho V\delta\tau=\int_{\tau}\frac{d}{dt}(\rho V\delta\tau)=\int_{\tau}\rho\frac{dV}{dt}(\delta\tau)+\int_{\tau}V\frac{d}{dt}(\rho\delta\tau)$$

Пользуясь свойством $\frac{d}{dt}(\rho\delta\tau)=\frac{d}{dt}(\delta m)=0$ для последнего слагаемого, перепишем вышестоящее уравнение в таком виде:

$$\frac{d}{dt}\int_{\tau}\rho V\delta\tau=\int_{\tau}\rho\frac{d}{dt}\delta\tau$$

В последнем слагаемом правой части уравнения (2.4) произведем замену и согласно теореме Гаусса – Остроградского будем иметь эквивалентные друг другу результаты:

$$\int_{\sigma}p_n\delta\sigma=\int_{\sigma}(n_1p_1+n_2p_2+n_3p_3)\delta\sigma$$

Воспользуемся теоремой Гаусса – Остроградского и получим:

$$\int_{\sigma}(n_1p_1+n_2p_2+n_3p_3)\delta\sigma=\int_{\tau}\left(\frac{\partial p_1}{\partial x_1}+\frac{\partial p_2}{\partial x_2}+\frac{\partial p_3}{\partial x_3}\right)\delta\tau$$

Или вышестоящее уравнение можно будет записать в таком виде:

$$\int_{\sigma}p_n\delta\sigma=\int_{\sigma}nP\delta\sigma=\int_{\tau}divP\delta\tau$$

Подставляем полученное в уравнение (2.4) и получим:

$$\int_{\tau}\rho\frac{dV}{dt}\delta\tau=\int_{\tau}\rho F\delta\tau+\int_{\tau}divP\delta\tau$$

2.3 Обезразмеривание уравнение Навье-Стокса

Рассмотрим условия подобия двух изометрических потоков ньютоновских вязких несжимаемых жидкостей с разными непостоянными плотностями и вязкостями. Приведем уравнения Навье-Стокса к безразмерному виду, выбрав в качестве масштабов времени, длины (в частности, координат), скорости, давления и объемных сил соответственно некоторых характерные для потока постоянные величины T, L, U_0, P[47].

Обозначая верхним подчеркиванием безразмерные величины времени, координат положения, скорости, сил и давления, поставим (здесь более удобнее пользоваться буквенной индексацией: x, y, z, u, v), что

$$\bar{u} = \frac{u}{U_0},\ \bar{v} = \frac{v}{U_0},\quad \bar{p} = \frac{p}{P},$$
$$\bar{x} = \frac{x}{L},\ \bar{y} = \frac{y}{L},\quad \bar{t} = \frac{t}{T}$$

Преобразуем указанные выражения к виду:

$$u = \bar{u}U_0,\ v = \bar{v}U_0,$$
$$x = \bar{x}L,\ y = \bar{y}L, \qquad (2.5)$$
$$p = \bar{p}P,\ t = \bar{t}T,$$

где

$$T = \frac{L}{U_0}$$

Рассмотрим уравнения Навье-Стокса:

$$1)\ \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial P}{\partial x} + v\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \qquad (2.6)$$

$$2)\ \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial P}{\partial y} + v\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \qquad (2.7)$$

$$3)\ \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \qquad (2.8)$$

Поставим (2.5) в уравнения (2.6), (2.7), (2.8) и для простоты восприятия опустим верхние подчеркивание и напишем в виде:

$$1)\quad \frac{U_0^2}{L}\frac{\partial u}{\partial t} + \frac{U_0^2}{L}\left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) = -\frac{P}{\rho L}\frac{\partial P}{\partial x} + \frac{U_0}{L^2}v\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

$$2)\quad \frac{U_0^2}{L}\frac{\partial v}{\partial t} + \frac{U_0^2}{L}\left(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) = -\frac{P}{\rho L}\frac{\partial P}{\partial y} + \frac{U_0}{L^2}v\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)$$

$$3)\quad \frac{U_0}{L}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0$$

Сокращая полученные уравнения движение на $\frac{U_0^2}{L}$, а уравнение неразрывности на $\frac{U_0}{L}$, получим:

1) $\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-\frac{P}{\rho U_0^2}\frac{\partial P}{\partial x}+\frac{v}{LU_0}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)$

2) $\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-\frac{P}{\rho U_0^2}\frac{\partial P}{\partial y}+\frac{v}{LU_0}\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)$

3) $\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0$

Далее представлены следующие безразмерные одночленные комплексы, которые называются «числами подобия»:

$$Eu=\frac{P}{\rho U_0^2},\ \mathrm{Re}=\frac{LU_0}{v}$$

Eu - число Эйлера, Re - число Рейнольдса.

И используя данные безразмерные числа, перепишем уравнения движения и неразрывности в виде:

1) $\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-Eu\frac{\partial P}{\partial x}+\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)$ (2.9)

2) $\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-Eu\frac{\partial P}{\partial y}+\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)$ (2.10)

3) $\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0$ (2.11)

Первое уравнение описывает скорость в направлении x. Второе уравнение описывает скорость в направлении y. Третье уравнение описывает уравнение неразрывности.

Здесь u, v - компоненты скорости, p - давление.

2.4 Дискретизация уравнения Навье-Стокса

Для дискретизации уравнения (2.9), (2.10) и (2.11) воспользуемся идеей конечно-разностной аппроксимации частных производных. Большинство встречающихся в гидродинамике и теплопередаче уравнении в частных производных используют не боле трех узлов разносной сетки. Рассмотрим уравнение (2.9):

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-Eu\frac{\partial P}{\partial x}+\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)\Rightarrow$$

$$\frac{\partial u}{\partial t}=-u\frac{\partial u}{\partial x}-v\frac{\partial u}{\partial y}-Eu\frac{\partial P}{\partial x}+\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)$$

Запишем итерационный метод Якоби и расчетные формулы в соответствии с методом будут иметь вид (верхний индекс, показывает номер итерации):

$$\frac{u_{ij}^{n+1}-u_{ij}^{n}}{\Delta t}=-u_{ij}^{n}\frac{u_{ij}^{n}-u_{i-1j}^{n}}{\Delta x}-v_{ij}^{n}\frac{u_{ij}^{n}-u_{ij-1}^{n}}{\Delta y}-Eu\frac{P_{ij}^{n}-P_{i-1j}^{n}}{\Delta x}+$$
$$+\frac{1}{\mathrm{Re}}\left(\frac{u_{i+1j}^{n}-2u_{ij}^{n}+u_{i-1j}^{n}}{\Delta x^{2}}+\frac{u_{ij+1}^{n}-2u_{ij}^{n}+u_{ij-1}^{n}}{\Delta y^{2}}\right)$$

Методам расщепления по физическим параметрам данный алгоритм расписывается так:

$$\frac{u_{ij}^{n+1}-u_{ij}^{n}+u_{ij}^{*}-u_{ij}^{*}}{\Delta t}=-u_{ij}^{n}\frac{u_{ij}^{n}-u_{i-1j}^{n}}{\Delta x}-v_{ijk}^{n}\frac{u_{ij}^{n}-u_{ij-1}^{n}}{\Delta y}+$$
$$+\frac{1}{\mathrm{Re}}\left(\frac{u_{i+1j}^{n}-2u_{ij}^{n}+u_{i-1j}^{n}}{\Delta x^{2}}+\frac{u_{ij+1}^{n}-2u_{ij}^{n}+u_{ij-1}^{n}}{\Delta y^{2}}\right)-Eu\frac{P_{ij}^{n}-P_{i-1j}^{n}}{\Delta x}$$

Уравнение разделяем на два уравнения:

1) $$\frac{u_{ij}^{*}-u_{ij}^{n}}{\Delta t}=-u_{ij}^{n}\frac{u_{ij}^{n}-u_{i-1j}^{n}}{\Delta x}-v_{ij}^{n}\frac{u_{ij}^{n}-u_{ij-1}^{n}}{\Delta y}+$$
$$+\frac{1}{\mathrm{Re}}\left(\frac{u_{i+1j}^{n}-2u_{ij}^{n}+u_{i-1j}^{n}}{\Delta x^{2}}+\frac{u_{ij+1}^{n}-2u_{ij}^{n}+u_{ij-1}^{n}}{\Delta y^{2}}\right) \quad (2.12)$$

2) $$\frac{u_{ij}^{n+1}-u_{ij}^{*}}{\Delta t}=-Eu\frac{P_{ij}^{n}-P_{i-1j}^{n}} \quad (2.13)$$

На основе (2.12) и (2.13) построим итерационный процесс:

1) $$u_{ij}^{*}=\Delta t(-u_{ij}^{n}\frac{u_{ij}^{n}-u_{i-1j}^{n}}{\Delta x}-v_{ijk}^{n}\frac{u_{ij}^{n}-u_{ij-1}^{n}}{\Delta y}+$$
$$+\frac{1}{\mathrm{Re}}(\frac{u_{i+1j}^{n}-2u_{ij}^{n}+u_{i-1j}^{n}}{\Delta x^{2}}+\frac{u_{ij+1}^{n}-2u_{ij}^{n}+u_{ij-1}^{n}}{\Delta y^{2}}))+u_{ij}^{n} \quad (2.14)$$

2) $$u_{ij}^{n+1}=-Eu\Delta t\frac{P_{ij}^{n}-P_{i-1j}^{n}}{\Delta x}+u_{ij}^{*} \quad (2.15)$$

Рассмотрим (2.10) второе уравнение Навье-Стокса.

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-Eu\frac{\partial P}{\partial y}+\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)$$

Отсюда находим,

$$\frac{\partial v}{\partial t}=-u\frac{\partial v}{\partial x}-v\frac{\partial v}{\partial y}-Eu\frac{\partial P}{\partial y}+\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right).$$

Запишем уравнение в дискретном виде:

$$\frac{v_{ij}^{n+1}-v_{ij}^{n}}{\Delta t}=-u_{ij}^{n}\frac{v_{ij}^{n}-v_{i-1j}^{n}}{\Delta x}-v_{ij}^{n}\frac{v_{ij}^{n}-v_{ij-1}^{n}}{\Delta y}-Eu\frac{P_{ij}^{n}-P_{i-1j}^{n}}{\Delta x}+$$
$$+\frac{1}{\mathrm{Re}}\left(\frac{v_{i+1j}^{n}-2v_{ij}^{n}+v_{i-1j}^{n}}{\Delta x^2}+\frac{v_{ij+1}^{n}-2v_{ij}^{n}+v_{ij-1}^{n}}{\Delta y^2}\right)$$

Методам расщепления по физическим параметрам данный алгоритм расписывается так:

$$\frac{v_{ij}^{n+1}-v_{ij}^{n}+v_{ij}^{*}-v_{ij}^{*}}{\Delta t}=-u_{ij}^{n}\frac{v_{ij}^{n}-v_{i-1j}^{n}}{\Delta x}-v_{ijk}^{n}\frac{v_{ij}^{n}-v_{ij-1}^{n}}{\Delta y}+$$
$$+\frac{1}{\mathrm{Re}}\left(\frac{v_{i+1j}^{n}-2v_{ij}^{n}+v_{i-1j}^{n}}{\Delta x^2}+\frac{v_{ij+1}^{n}-2v_{ij}^{n}+v_{ij-1}^{n}}{\Delta y^2}\right)-Eu\frac{P_{ij}^{n}-P_{ij-1}^{n}}{\Delta y}$$

Уравнение разделяем на два уравнения:

1) $$\frac{v_{ij}^{*}-v_{ij}^{n}}{\Delta t}=-u_{ij}^{n}\frac{v_{ij}^{n}-v_{i-1j}^{n}}{\Delta x}-v_{ij}^{n}\frac{v_{ij}^{n}-v_{ij-1}^{n}}{\Delta y}+$$
$$+\frac{1}{\mathrm{Re}}\left(\frac{v_{i+1j}^{n}-2v_{ij}^{n}+v_{i-1j}^{n}}{\Delta x^2}+\frac{v_{ij+1}^{n}-2v_{ij}^{n}+v_{ij-1}^{n}}{\Delta y^2}\right) \quad (2.16)$$

2) $$\frac{v_{ij}^{n+1}-v_{ij}^{*}}{\Delta t}=-Eu\frac{P_{ij}^{n}-P_{ij-1}^{n}}{\Delta y} \quad (2.17)$$

На основе (2.16) и (2.17) построим итерационный процесс:

1) $$v_{ij}^{*}=\Delta t(-u_{ij}^{n}\frac{v_{ij}^{n}-v_{i-1j}^{n}}{\Delta x}-v_{ij}^{n}\frac{v_{ij}^{n}-v_{ij-1}^{n}}{\Delta y}+$$
$$+\frac{1}{\mathrm{Re}}(\frac{v_{i+1j}^{n}-2v_{ij}^{n}+v_{i-1j}^{n}}{\Delta x^2}+\frac{v_{ij+1}^{n}-2v_{ij}^{n}+v_{ij-1}^{n}}{\Delta y^2}))+v_{ij}^{n} \quad (2.18)$$

$$2)\ v_{ij}^{n+1} = -Eu\Delta t\frac{P_{ij}^{n} - P_{i-1j}^{n}}{\Delta y} + v_{ij}^{*} \qquad (2.19)$$

Далее рассмотрим число Эйлера $Eu = 1.$

Рассмотрим (2.11) уравнение неразрывности:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

В уравнение неразрывности подставляем уравнения (2.15) и (2.19) и получаем уравнение Пуассона:

$$\frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial y^2} = \frac{1}{\Delta t}\left(\frac{\partial u^*}{\partial x} + \frac{\partial v^*}{\partial y}\right)$$

Пятиточечная схема, предложенная Рунге в 1908 году, наиболее часто используется для построения конечномерного разностного аналога двумерного уравнения Пуассона. Далее для решения задачи воспользуемся методом Якоби. Приближение производных конечными разностями будет иметь вид:

$$\frac{P_{i+1j}^{n} - 2P_{ij}^{n+1} + P_{i-1j}^{n}}{\Delta x^2} + \frac{P_{ij+1}^{n} - 2P_{ij}^{n+1} + P_{ij-1}^{n}}{\Delta y^2} = \frac{1}{\Delta t}\left(\frac{u_{ij}^{*} - u_{i-1j}^{*}}{\Delta x} + \frac{v_{ij}^{*} - v_{ij-1}^{*}}{\Delta y}\right) \qquad (2.20)$$

На основе (2.20) построим итерационный процесс:

$$P_{ij}^{n+1}\left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}\right) = \frac{P_{i+1j}^{n} + P_{i-1j}^{n}}{\Delta x^2} + \frac{P_{ij+1}^{n} + P_{ij-1}^{n}}{\Delta y^2} - \frac{1}{\Delta t}\left(\frac{u_{ij}^{*} - u_{i-1j}^{*}}{\Delta x} + \frac{v_{ij}^{*} - v_{ij-1}^{*}}{\Delta y}\right)$$

$$P_{ij}^{n+1} = \frac{1}{2}\left(\frac{P_{i+1j}^{n} + P_{i-1j}^{n}}{\Delta x^2} + \frac{P_{ij+1}^{n} + P_{ij-1}^{n}}{\Delta y^2} - \frac{1}{\Delta t}\left(\frac{u_{ij}^{*} - u_{i-1j}^{*}}{\Delta x} + \frac{v_{ij}^{*} - v_{ij-1}^{*}}{\Delta y}\right)\right) \Big/ \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)$$

(2.21)

Решение уравнения Пуассона для давления, создаваемого при численном решении уравнений Навье-Стокса на несжимаемую жидкость (INSE), в настоящее время является наиболее дорогостоящей частью вычислительной процедуры и часто является основным ограничивающим фактором для параллельной реализации [48].

Заключение по 2 разделу

В данном разделе рассматривается основное уравнение гидродинамики, обезразмеривание и дискретизация уравнения Навье-Стокса.

3 РАЗРАБОТАТЬ ЭФФЕКТИВНОГО ВЫСОКОПРОИЗВОДИТЕЛЬНОГО ВЫЧИСЛЕНИЯ ДЛЯ РЕШЕНИЯ УРАВНЕНИЯ ПУАССОНА

Рассматривается решение модельного двумерного эллиптического уравнения, охватывающего широкий класс прикладных задач. Для решения уравнений гидродинамики несжимаемой жидкости решение уравнения Пуассона требует значительных вычислительных усилий. Уравнение Пуассона является уравнением эллиптического типа в частных производных, присутствует во многих геофизических моделях. При моделировании задач используется уравнение Пуассона: в уравнении Навье-Стокса несжимаемой жидкости для определения давления получают уравнение Пуассона (Johnston and Liu, 2012, [19]; Rannacher, 1993, [20]); зажигания дуги переменного тока на холодных электродах в аргоне атмосферного давления (Santos и др., 2021, [12]); начальный слой и квазинейтральный предел трехмерных моделей электродиффузии (Wang and Jiang, 2021, [13]): обратные задачи оценки момента в размерности 2 (Leblond and Pozzi, 2021, [14]); физико-совместимые методы дискретизации (Palha и др., 2014, [15]); моделирование ветра (Bozorgmehr, 2021, [16]); моделирование пространственного заряда (Qiang, 2017, [17]).

Эллиптические уравнения с частными производными, связанные с краевыми задачами (не зависящими от времени), а не с проблемами эволюции, довольно распространены во многих областях физики, таких как тепловой поток, электростатический и гравитационный потенциал, потенциальный поток, статическая упругость и квантовая механика.

3.1 Постановка задачи

Рассмотрим уравнение Пуассона в некоторой прямоугольной области:

$$\nabla^2 U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = -f(x, y),\ (x, y) \in [0; l] \times [0; l], \tag{3.1}$$

граничные условия Дирихле:

$$u(x, 0) = 0, u(x, l) = 0,\ x \in (0; l), \tag{3.2}$$

$$u(0, y) = 1, u(l, y) = 1,\ y \in (0; l), \tag{3.3}$$

3.2 Дискретизация и численный алгоритм

Для решения задачи используем метод Якоби. Теперь вводится прямоугольная сетка с шагами h_1 и h_2 по переменным x и y соответственно:

$$\Omega_{h_1,h_2} = \{x_i = ih_1,\ i = \overline{0,\ N};\ y_i = jh_2,\ j = \overline{0,\ M}\}, \quad (3.4)$$

где $h_1 = l/(N-1),\ h_2 = l/(M-1)$.

Методом конечных разностей аппроксимируем выражения (3.1)-(3.3). По каждому направлению для простоты положим $N := N = M$ и обозначим $h := h_1 = h_2$.

$$\frac{u^n_{i+1,j} - 2u^{n+1}_{i,j} + u^n_{i-1,j}}{h^2} + \frac{u^n_{i,j+1} - 2u^{n+1}_{i,j} + u^n_{i,j-1}}{h^2} = -f_{i,j}. \quad (3.5)$$

На основе (3.5) построим итерационный процесс:

$$u^{n+1}_{ij} = \frac{1}{4}(u^n_{i+1j} + u^n_{i-1j} + u^n_{ij+1} + u^n_{ij-1} + f_{ij}h^2),\ 0 < i < N,\quad 0 < j < N \quad (3.6)$$

Граничные значения получены из (3.2), (3.3):

$$u^0_{ij} = 0,\ 0 \le i \le N,\quad 0 \le j \le N,$$
$$u_{i0} = 0,\ 0 \le i \le N,$$
$$u_{iN} = 0,\ 0 \le i \le N,$$
$$u_{0j} = 1,\ 0 \le j \le N,$$
$$u_{Nj} = 0,\ 0 \le j \le N.$$

При решении этого уравнения численный метод определяется для прямоугольной области. В результате находится линейная система уравнений. Завершаем процесс итерации, если отклонение между значениями двух последних итераций меньше заданного параметра ε,

$$\max_{ij}\left|u^{n+1}_{ij} - u^n_{ij}\right| < \varepsilon.$$

3.3 Создание параллельных схем в системе CUDA. Проблемы, пути решения

Доступны методы численного решения, такие как методы конечных разностей, конечных элементов, конечных объемов и спектральные методы. У каждого есть свой численный метод, но в основе решателя лежит аппроксимация неизвестных переменных потока с помощью простых функций, дискретизация путем подстановки приближений в основной поток и алгебраическое решение. Если пользователь использует технику решения, затраченное время зависит от вычислительной мощности компьютера.

Сегодня высокопроизводительные вычисления невозможно представить без распараллеливания, так как самые мощные вычислительные системы имеют работающих одновременно в тесном взаимодействии сотни и тысячи процессоров, то есть параллельно. При параллельных вычислениях программа

делится на множество подпрограмм, поэтому все они выполняются параллельно для вычисления требуемых значений.

Подход CUDA обеспечивают прямую реализацию параллельных алгоритмов и предоставляет небольшой набор расширений для стандартных языков программирования, таких как C. Когда приложения используют CPU и GPU, CUDA, поддерживаются гетерогенные вычисления (Fuentes-Alventosa и др., 2014, [21]). Программа CUDA вызывает параллельные функции. Параллельные функции также называются ядрами, которые выполняются почти во всех параллельных потоках. Разработчик либо компилятор соединяет эти потоки в блоки потоков и сетки блоков потоков. Применяемый для сброса регистров, вызовов функций и автоматических переменных массивов C в модификации параллельного программирования, CUDA имеет отдельное пространство памяти. Параллелизм данных достигается путем последовательной избыточной релаксации и оптимизированной итеративной процедуры драйвера с помощью классического метода Якоби. Параллелизм задач улучшается за счет минимизации обмена данными между графическими процессорами по мере выполнения итераций для снижения коммуникационных издержек. В алгоритме 3.1 показан алгоритм решения задачи (3.1)-(3.3).

Алгоритм 3.1: Реализация двумерного уравнения с использованием CPU

compute initial function matrix u_prev
from initial condition (2.2) we get u ← u_prev
do
call function P2D (u_prev, u, f, N, h)
calculate matrix u
swap (u_prev, u)
while (k < nIterations)

Для начала необходимо выполнение расчетов размера блока в строгом соответствии с размером матрицы и номерами шагов прямого и обратного решения. В алгоритме 3.2 показан алгоритм реализации двумерного уравнения с использованием GPU.

Алгоритм 3.2: Реализация двумерного уравнения с использованием GPU

```
dim3 dimBlock(BLOCK_SIZE_X, BLOCK_SIZE_Y);
dim3 dimGrid(N / dimBlock.x + ((N % dimBlock.x) ? 1 : 0), N / dimBlock.y + ((N % dimBlock.y) ? 1 : 0));
do {
P2D << <dimGrid, dimBlock >> > (d_u_prev, d_u, dev_f, N, h);
    float* pingPong ← d_t_prev;
    d_t_prev ← d_t;
    d_t ← pingPong;
    k++;
```

} while (k < nIterations);

Таким образом, рассчитываем массив d_u и расчетные данные d_u с устроиства на хост копируем с помощью *cudaMemcpy(d_u, u, sizeof(float) * N * N, cudaMemcpyHostToDevice);*

3.4 Результаты численного расчета

Значения параметров моделирования даны в виде: $\Delta x = \Delta y = 1/N$, а относительная погрешность 10^{-6}. В результате использования явного метода (3.5) была получена сеточная функция $u_{i,j}^{n+1}$. Для получения более реалистичных данных были протестированы семь случаев с разной площадью рассматриваемой области: 32 × 32, 64 × 64, 128 × 128, 256 × 256, 512 × 512, 1024 × 1024, 2048 × 2048. Результаты тестирования численного решения краевых задач для уравнения Пуассона, который был предложен нами является эффективным, что показывает решение с относительной погрешностью 10^{-6} в минимальных затратах машинного времени. Затем система уравнений реализуется на параллельном алгоритме CUDA как задача численной линейной алгебры. Выполняется программа CUDA и анализируются результаты. В таблице 3.1 указывается время выполнения для последовательного подхода (в ЦП), CUDA (в графическом процессоре) для задач (3.6).

Таблица 3.1– Время выполнение (мсек)

Mesh size	CPU	GPU	Speed-up
32x32	0.035	0.2160385	0,162008
64x64	0.497	1.0914834	0,455344
128x128	6.871	7.5760875	0,906933
256x256	91.917	72.6604385	1,265021
512x512	1292.31	842.4293213	1,534028
1024x1024	16771.3	10528.823242	1,592894
2048x2048	202957	125857.4296875	1,612595

3.5 Оценка эффективности параллельного численного алгоритма

Параллельные вычисления на GPU реализованы на основе Compute Unified Device Architecture (CUDA), инструментария, разработанного NVIDIA для выполнения параллельных вычислений с использованием своих видеокарт. В данном разделе продемонстрированы результаты, полученные на настольном компьютере с конфигурацией 640 ядра GeForce GTX 1050, NVIDIA GPU и с CPU Intel(R) Core(TM) i5-7500, 3.40GHz, RAM 16Gb.

Результаты тестов программ приведены в таблице 3.1 и при увеличении размера сетки вычисление на графическом процессоре быстрее, чем на центральном процессоре. Это видно на рисунке 3.1 и 3.2.

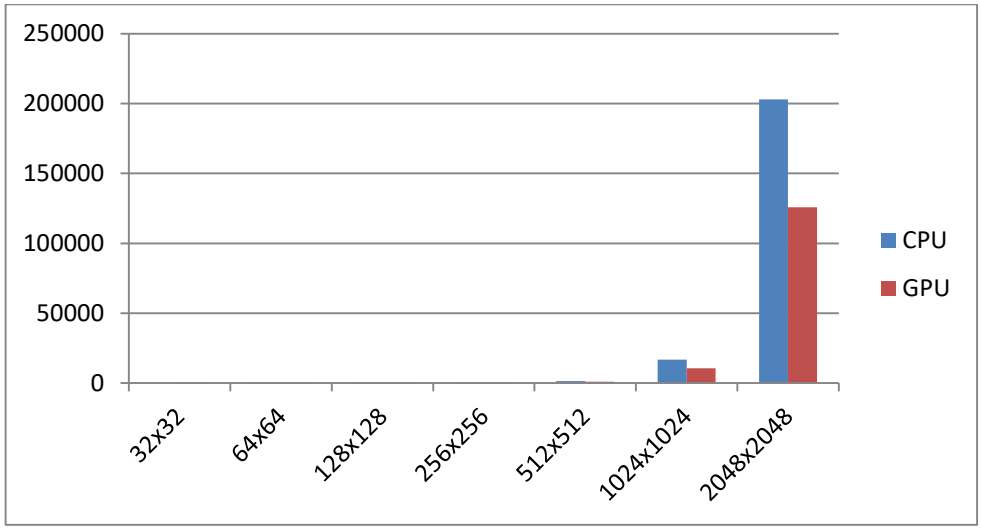


Рисунок 3.1 – Время выполнения программ

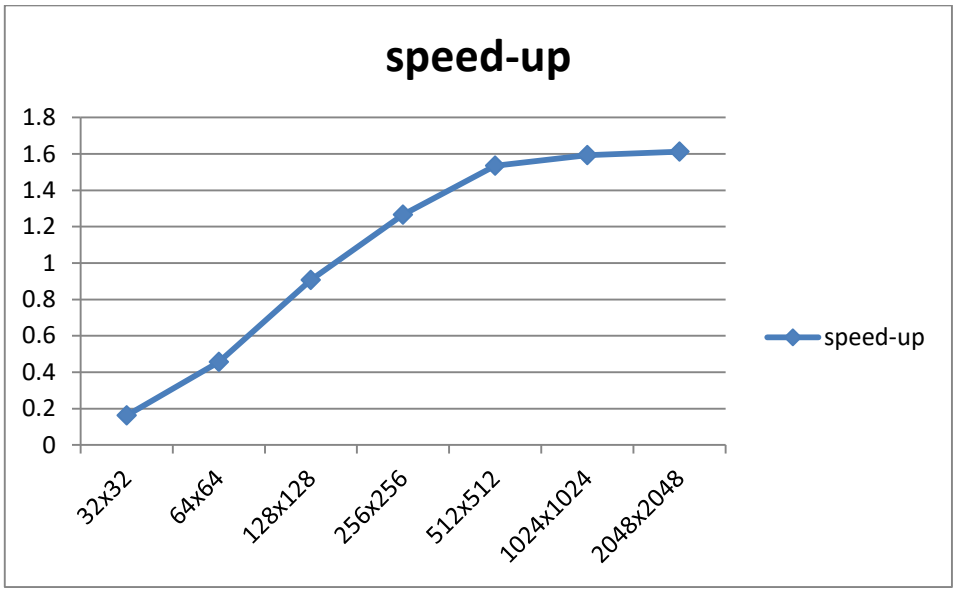


Рисунок 3.2 – Ускорение программ

Замечено, что для большего количества данных и количества пространственных узлов в пространстве решений достигаются более высокие значения ускорения. Кроме того, следует отметить, что из-за процесса распараллеливания как на CPU, так и на GPU потери точности не произошло.

Решение уравнения Пуассона является наиболее решающей частью с точки зрения общей производительности. Это показывает важность использования эффективных алгоритмов линейной алгебры в процессе решения. Дискретизированное уравнение Пуассона обычно дает разреженную линейную систему, называемую проблемой ограниченной пропускной способности. Например, наш процессор 640 ядра GeForce GTX 1050, NVIDIA GPU и с CPU Intel(R) Core(TM) i5-7500, 3.40GHz, RAM 16Gb. Изучив эти цифры, можно сказать, что для задачи с ограниченной пропускной

способностью ускорение должно быть в 6 раз больше. Таким образом, мы можем провести несколько сравнений производительности CPU и GPU для решения уравнения Пуассона, чтобы оправдать дополнительные усилия при использовании GPU [49].

Заключение по 3 разделу

В данном разделе рассматривалось решение уравнения Пуассона в некоторой прямоугольной области, охватывающее широкий класс прикладных задач. Применяя данное уравнение, был представлен численный расчет на CPU и GPU, на основании данного расчета был сделан сравнительный анализ, который показал эффективность параллельного численного алгоритма.

4 РАЗРАБОТАТЬ ЭФФЕКТИВНОГО ВЫСОКОПРОИЗВОДИТЕЛЬНОГО ВЫЧИСЛЕНИЯ ДЛЯ РЕШЕНИЙ ЦИРКУЛЯЦИОННЫХ НЕСЖИМАЕМЫХ ВЯЗКИХ ТЕЧЕНИЙ В КАВЕРНЕ

4.1 Математическая постановка задачи несжимаемого вязкого течения в каверне

Для несжимаемой жидкости уравнения Навье-Стокса моделировали течение несжимаемой и вязкой жидкости. В несжимаемом потоке изменение плотности из-за давления игнорируется, поэтому плотность можно считать постоянной. Этот поток жидкости образуется в потоке воды, масла и воздуха с низкой скоростью. Поток несжимаемой жидкости представляется системой нелинейных уравнений в частных производных и состоит из уравнений сохранения массы и сохранения количества движения. Это уравнение трудно решить численно, потому что нет уравнения состояния, которое связывает поле давления с переменной скоростью. Для преодоления вышеуказанной трудности существует несколько подходов к численному решению, первый из которых - удаление поля давления с помощью функции тока и завихренности [50, 55]. Второй подход основан на примитивных переменных, этот подход включает в себя: метод маркеров и ячеек (MAC) [51], метод дробного шага [52], полунеявный метод для уравнений давления (SIMPLE) [53] и метод искусственной сжимаемости [54]. Методы маркеров и ячеек, метод дробного шага и полунеявный метод для уравнений давления получают поле давления путем решения уравнения Пуассона. Метод искусственного сжатия добавляет производную давления по времени в уравнение сохранения массы, поэтому давление может быть получено непосредственно.

Стабилизированный метод решения несжимаемых уравнений Навье-Стокса (N–S) используется в важной области изучения конечных элементов. Стабилизация конечного элемента на основе методов решения состоит из двух компонентов: стабилизации пространственных колебаний, вызванных отбором членов конвекции в больших числах Рейнольдса, и стабилизации давления. Для обхода пространственных колебаний, вызванных стандартной центростремительной дискретизацией члена конвекции, были предприняты значительные усилия по разработке нестандартного метода конечного элемента Галеркина (МКЭ). Некоторые из популярных и общих методов стабилизации включают Петрова-Галеркина (Pranowo и др., 2018), метод конечных элементов наименьших квадратов (LSFEM) (Shui и др., 2018; Castelo и др., 2021) и разделение по характеристикам (CBS)(Shui, 2020).

В этом разделе мы представляем численные реализации двумерного циркуляционного несжимаемого вязкого течения в каверне. Эта эталонная задача широко использовалась для оценки CFD-решателей, поскольку подробно известно решение задачи в зависимости от числа Рейнольдса, размера сетки и необходимых временных шагов, решение задачи требует значительных вычислительных ресурсов. При решении задачи течения в каверне,

управляемое крышкой (LDCF), большинство авторов использовали формулировку функции скорости–течения, так как неизвестное только два, а уравнение неразрывности выполняется автоматически. Несжимаемые уравнения Навье-Стокса в двумерной декартовой системе используются для моделирования интересующей проблемы LDCF. Это набор нелинейных, зависящих от времени дифференциальных уравнений в частных производных:

$$1)\ \frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-\frac{\partial P}{\partial x}+v\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right) \tag{4.1}$$

$$2)\ \frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-\frac{\partial P}{\partial y}+v\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right) \tag{4.2}$$

$$3)\ \frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0 \tag{4.3}$$

где p — давление, (x, y) — координаты, t — время, (u, v) — компоненты скорости, Re — число Рейнольдса.

Для корректной постановки задачи необходимо поставить начальные и граничные условия:
начальная условия:

$$u(x,y,0)=0, \tag{4.4}$$

$$v(x,y,0)=0, \tag{4.5}$$

граничные условия:

$$\begin{aligned} &u(0,y,t)=0, && u(1,y,t)=0,\\ &u(x,0,t)=0, && u(x,1,t)=1,\\ &v(0,y,t)=0, && v(1,y,t)=0,\\ &v(x,0,t)=0, && v(x,1,t)=0, \end{aligned} \tag{4.6}$$

4.2 Дискретизация и численный алгоритм

Обычными методами дискретизации, используемыми для численных решений, являются методы конечных разностей, конечного объема и конечных элементов. Эти методы основаны на сетке и требуют сетки для численной дискретизации. Построение топологий связности элементов и узлов становится относительно сложной задачей. В качестве альтернативы предлагается разработать бессеточный метод. Этот метод не требует сетки, а требует только узлы распределения, ряд бессеточных методов получения приближенных решений уравнений Навье-Стокса [57, 58].

Для решения уравнений используется метод дробного шага [52] для варианта коррекции давления. Для первого шага вычислите промежуточные компоненты скорости (u*, v*) неявным образом, опустив члены давления в уравнении (4.1) и (4.2):

$$u_{ij}^{*} = \Delta t(-u_{ij}^{n}\frac{u_{ij}^{n}-u_{i-1j}^{n}}{\Delta x} - v_{ijk}^{n}\frac{u_{ij}^{n}-u_{ij-1}^{n}}{\Delta y} + \\ + \frac{1}{\mathrm{Re}}(\frac{u_{i+1j}^{n}-2u_{ij}^{n}+u_{i-1j}^{n}}{\Delta x^{2}} + \frac{u_{ij+1}^{n}-2u_{ij}^{n}+u_{ij-1}^{n}}{\Delta y^{2}})) + u_{ij}^{n} \tag{4.7}$$

$$v_{ij}^{*} = \Delta t(-u_{ij}^{n}\frac{v_{ij}^{n}-v_{i-1j}^{n}}{\Delta x} - v_{ij}^{n}\frac{v_{ij}^{n}-v_{ij-1}^{n}}{\Delta y} + \\ + \frac{1}{\mathrm{Re}}(\frac{v_{i+1j}^{n}-2v_{ij}^{n}+v_{i-1j}^{n}}{\Delta x^{2}} + \frac{v_{ij+1}^{n}-2v_{ij}^{n}+v_{ij-1}^{n}}{\Delta y^{2}})) + v_{ij}^{n} \tag{4.8}$$

Промежуточные компоненты скорости не удовлетворяют уравнению неразрывности, т.е. уравнению (4.3). Поэтому, чтобы получить правильные компоненты скорости u_{ij}^{n+1}, v_{ij}^{n+1}, их необходимо корректировать следующим образом:

$$u_{ij}^{n+1} = -\Delta t\frac{P_{ij}^{n}-P_{i-1j}^{n}}{\Delta x} + u_{ij}^{*} \tag{4.9}$$

$$v_{ij}^{n+1} = -\Delta t\frac{P_{ij}^{n}-P_{i-1j}^{n}}{\Delta y} + v_{ij}^{*} \tag{4.10}$$

Связь скорости с давлением приводит к замене уравнение неразрывности следующим уравнением Пуассона для давления:

$$\frac{\partial^{2}P}{\partial x^{2}} + \frac{\partial^{2}P}{\partial y^{2}} = \frac{1}{\Delta t}\left(\frac{\partial u^{*}}{\partial x} + \frac{\partial v^{*}}{\partial y}\right) \tag{4.11}$$

Пятиточечная схема часто используется для построения конечно-разностного аналога двумерного уравнения Пуассона. Для решения задачи воспользуемся методом Якоби. Приближение производных конечными разностями будет иметь вид:

$$P_{ij}^{n+1} = \frac{1}{2}\left(\frac{P_{i+1j}^{n}+P_{i-1j}^{n}}{\Delta x^{2}} + \frac{P_{ij+1}^{n}+P_{ij-1}^{n}}{\Delta y^{2}} - \frac{1}{\Delta t}\left(\frac{u_{ij}^{*}-u_{i-1j}^{*}}{\Delta x} + \frac{v_{ij}^{*}-v_{ij-1}^{*}}{\Delta y}\right)\right)/\left(\frac{1}{\Delta x^{2}} + \frac{1}{\Delta y^{2}}\right) \tag{4.12}$$

Проделываем показанную процедуру до выполнения условия:

$$\max_{ij}\left|P_{ij}^{n+1} - P_{ij}^{n}\right| < \varepsilon$$

4.3 Создание параллельных схем в системе CUDA. Проблемы, пути решения

Приведен алгоритм распараллеливания на GPU. Мы уделяем особое внимание повышению производительности параллельных алгоритмов. Замечено, что коды, работающие на платформе CUDA, дают ожидаемые результаты. Сравнивая наши тесты на графическом процессоре с тестами, полученными при запуске последовательного кода той же симуляции на ЦП, оказывается, что симуляции на графическом процессоре выполняются намного быстрее, чем на ЦП. GPU выполняет все вычисления, а процессор отвечает только за управление программным обеспечением GPU, ввод-вывод файлов и передачу данных. Этот код написан с использованием C++ и CUDA C/C++.

Используемая явная схема конечных разностей позволяет проводить независимые вычисления в каждом узле, поэтому существуют разные возможности схемы распараллеливания. Группа узлов может быть обработана блоком потоков, в котором каждый поток оценивает метод конечной разности на соответствующем узле. Потоки блоков обрабатываются одним и тем же потоковым процессором (SM), а распараллеливание зависит от количества ядер в SM. Архитектура GPU имеет ограничение на количество потоков в блоке. В этой работе карта NVIDIA GeForce GTX 1050 использовалась до 1024 потоков на блок. Тем не менее, большее количество потоков на число блоков, как правило, не означает более высокую производительность. Разные блоки могут обрабатываться в разных SM одновременно, тогда распараллеливание блоков зависит от количества SM в GPU. Число потоков на блок (потоки) было зафиксировано на значении 32. Затем число блоков было равно Blocks = Nodes/Threads. Решение было реализовано только с использованием глобальной памяти. В этом подходе каждый поток из более медленной глобальной памяти должен считывать значения, необходимые для оценки шаблона конечных разностей. Значения вокруг каждого узла должны быть загружены соответствующим потоком. В результате каждый узел читается несколько раз, что влияет на производительность. Решением этой проблемы является подход, называемый повторным использованием данных, при котором набор данных загружается в более быструю общую память и оценивается группа узлов. Чем больше набор данных, тем меньше происходит многократных считываний, и, таким образом, повышается производительность.

Алгоритм 4.1 показывает алгоритм распараллеливания на GPU. CUDA код состоит из передачи данных в глобальную память GPU, запуска ядра CUDA и передачи памяти из GPU в память CPU. Для решения этой задачи используем блоки размером 16x16 и размер сетки определяется по формуле:

dim3 dimGrid((N - 1) / dimBlock.x + 1, (N - 1) / dimBlock.y + 1).

Алгоритм 4.1: Реализация двумерного несжимаемого вязкого течения в каверне

```
compute initial function matrix u,v
from initial condition (4.4), (4.5) we get un ← u, vn←v
do {
        call function u(d_un, d_u, d_v, N, h, dt);
        d_un←d_u;
        call function v(d_vn, d_v, d_u, N, h, dt);
        d_vn←d_v;
        call function f(d_u, d_v, d_f, N, h, dt);
        call function P(d_Pn, d_p, d_f, N, h, d_max);
        d_Pn←d_P
        k++;
} while (k < nIterations);
```

Для отклонения между значениями двух последних итераций используется параллельная редукция, как показано на алгоритме 4.2.

Алгоритм 4.2: отклонения между значениями двух последних итераций

```
unsigned int s ← blockDim.x / 2;
            while (s != 0) {
                    if (tid < s) {
                            if (shArrr[tid + s] > shArrr[tid])
                                    shArrr[tid] ← shArrr[tid + s];
                    }
                    __syncthreads();
                    s /← 2;
            }
            if (tid == 0) {
                    d_max[index] ← shArrr[0];
            }
    sData_P[d_ti][d_tj] ← output_Pn[index];
```

После этого мы копируем рассчитанные данные с устройства на хост с помощью cudaMemcpy(resultOnHost, nb, cudaMemcpyDeviceToHost).

4.4 Результаты численного расчета

Течение в каверне, управляемое крышкой, является одной из самых популярных задач в области вычислительной гидродинамики. Многие исследования, начиная с ранней работы Burggraf (1966), изучали динамику течения внутри управляемой каверне. Несмотря на то, что геометрия простая, эта модель позволяет лучше понять более сложные режимы течения в каверне. С помощью настоящей модели получен ряд численных решений двумерного течения в каверне, управляемой крышкой, с индексом h, установленным равным 1/(N-1). Как показано на рисунке 4.1 безразмерный размер полости равен 1 × 1. Безразмерные скорости $u = 1$ и $v = 0$ задаются на верхней стенке, а

остальные три стороны являются сплошными стенками, на которых заданы граничные условия $u=0$ и $v=0$.

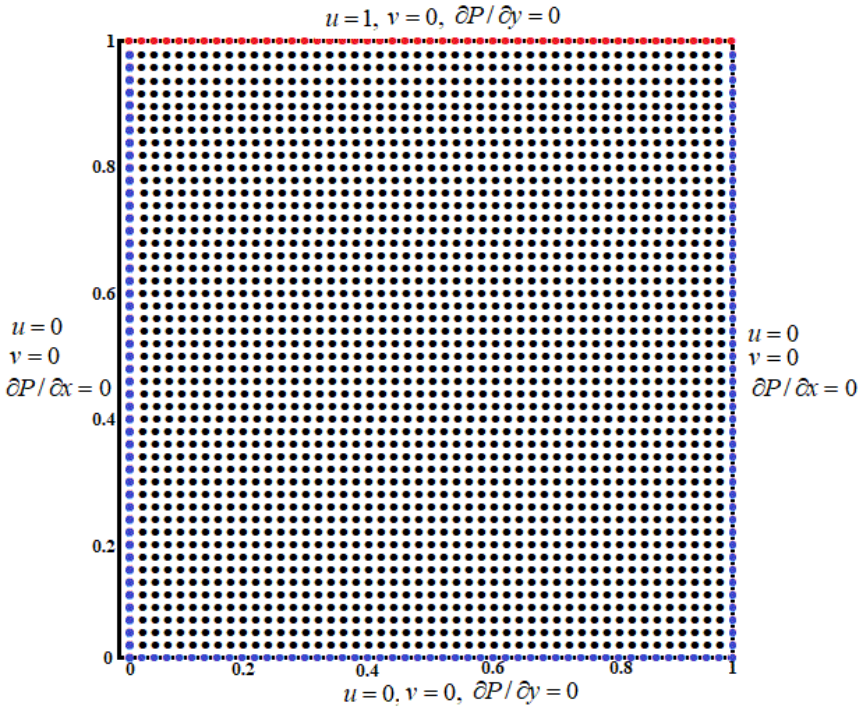


Рисунок 4.1 – Задача несжимаемого вязкого течения в каверне

Мы используем однородную сетку, состоящую из 129x129 сеток, которая показана на рисунке 4.2. Выбранный размер сетки является оптимальным на основе независимого теста сетки.

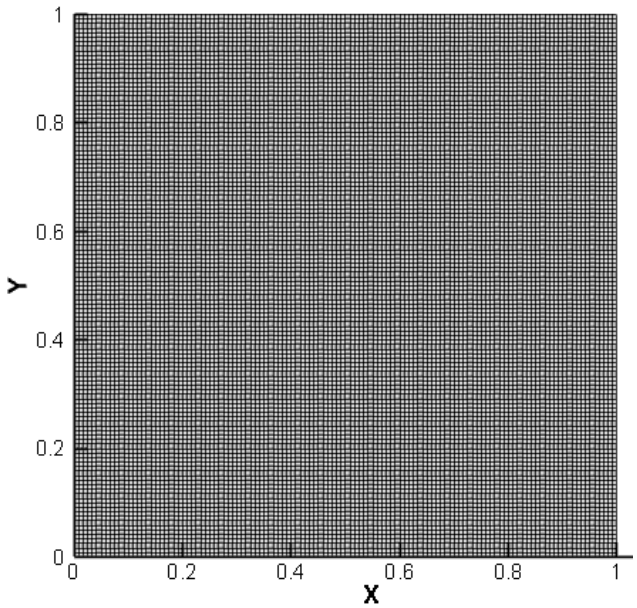


Рисунок 4.2 – Схематическая диаграмма, представляющая размер сетки 129x129

На рисунке 4.3 показан результат моделирования каверны, управляемой крышкой. На рисунке 4.4 показан первичный вихрь, возникающий вблизи центра каверны, с большим акцентом на вторичные вихри, малые локальные циркуляции, возникающие в углах, которые обозначены интенсивными линиями потока.

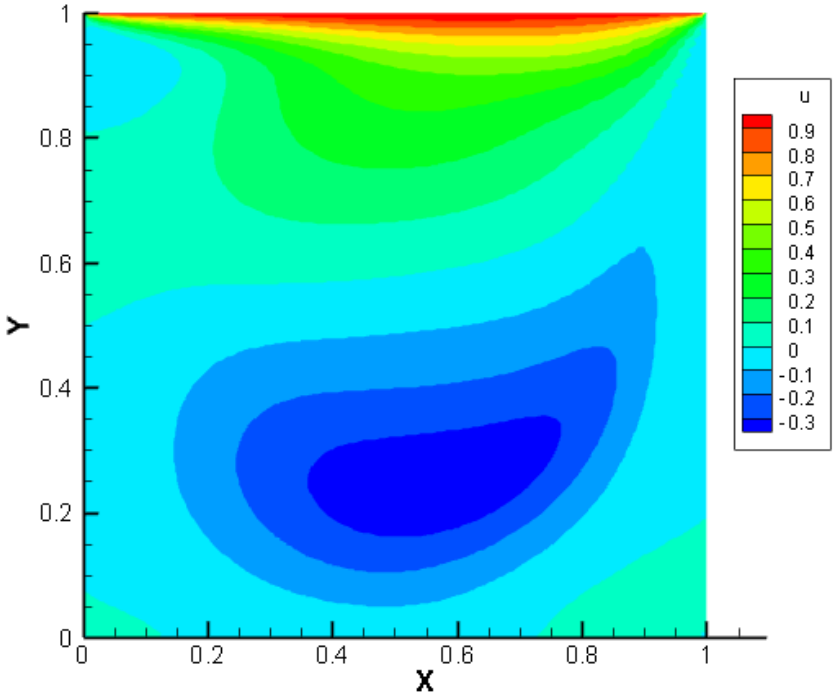


Рисунок 4.3 – Контуры u скорости

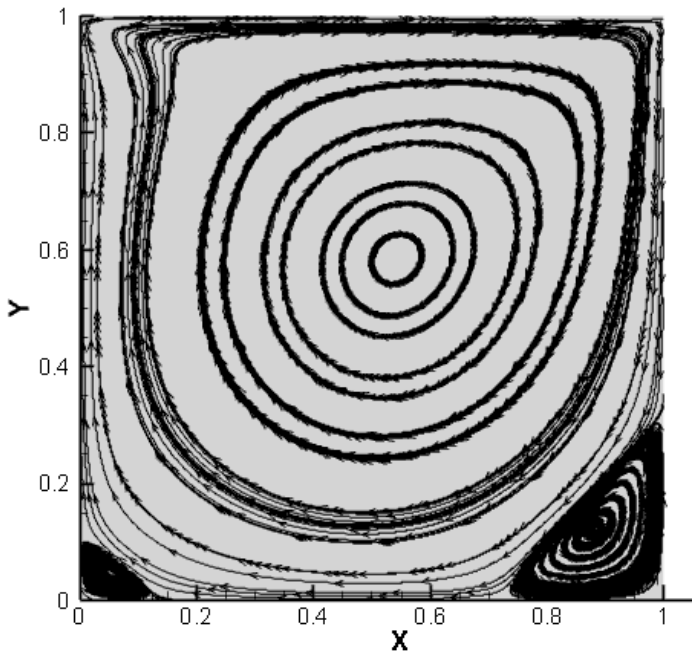


Рисунок 4.4 – Линии потока для каверны с односторонним приводом от крышки

Результаты вычислений при различных числах Рейнольдса. На рисунке 4.5 показаны распределения u-скорости по средней вертикальной линии и v-

скорости по горизонтальной средней линии для Re = 100 и 400. Видно, что результаты настоящей модели хорошо согласуются с результатами Ghia и др. (1982), Shui (2020), Huang & Lim (2020). Количество сеток составляет 64×64, 128×128, 512×512 при Re=100 и Re=400. Это указывает на то, что поток, движущийся с крышкой, имеет максимальную скорость в крышке, которая постепенно уменьшается к дну и переходит к отрицательной скорости в нижней области за счет циркуляции потока внутри полости. Из-за состояния прилипания к поверхности, максимальный отрицательный пик появляется не у стенки, а ближе у нее. Кроме того, максимальная горизонтальная скорость u значительно меняется в зависимости от числа Re. С увеличением Re значение максимальной горизонтальной скорости увеличивается, а точка максимального пика смещается в сторону стенки. Как показано на рисунке 4.5(c, д), при увеличении числа Re пик максимальной горизонтальной скорости также увеличивается, а пиковая точка также смещается к стенке. Как видно из результатов, моделирование соответствует результатам существующих исследований [56, 59, 60].

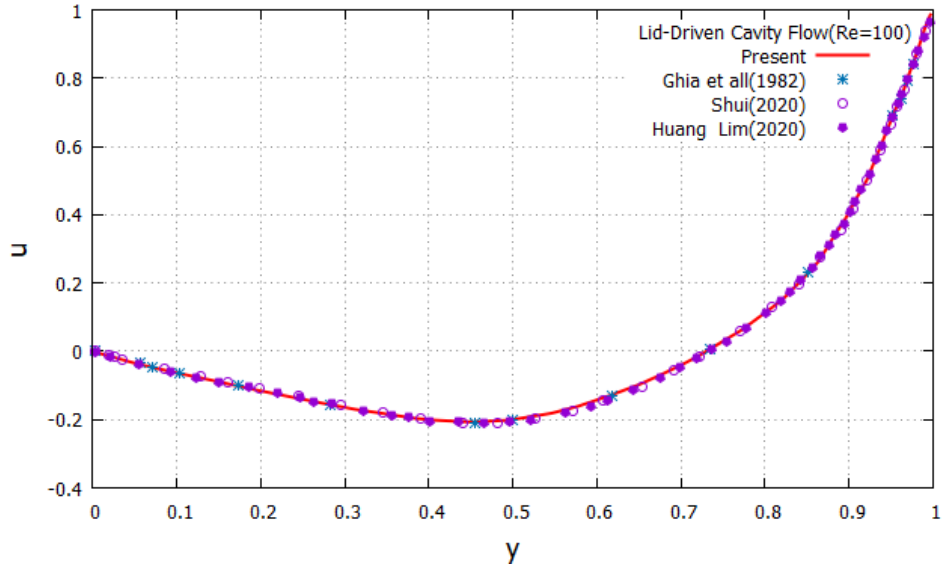


Рисунок 4.5 (а) – Скорости вдоль линий, проходящих через геометрический центр, профиль горизонтальной скорости, Re=100

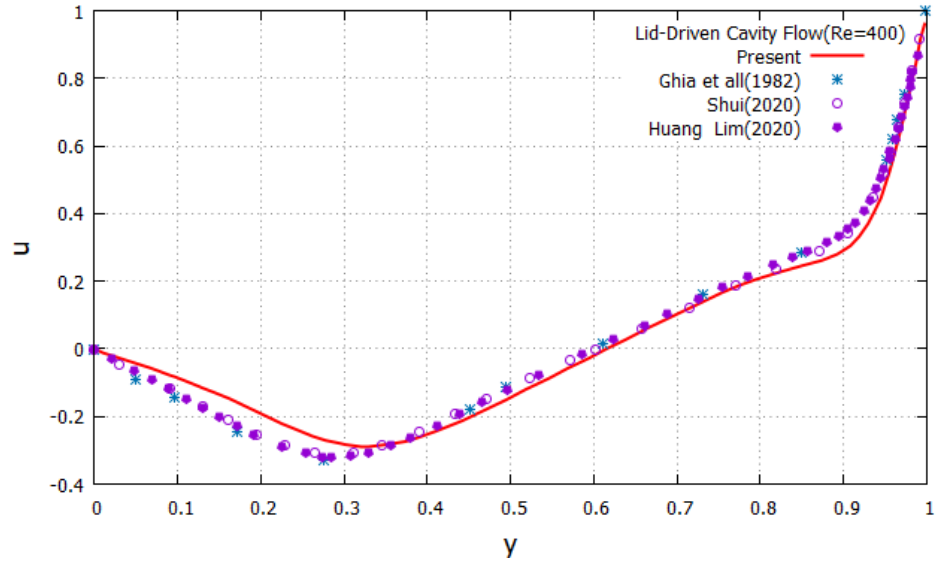


Рисунок 4.5 (б) – Скорости вдоль линий, проходящих через геометрический центр, профиль горизонтальной скорости, Re=400

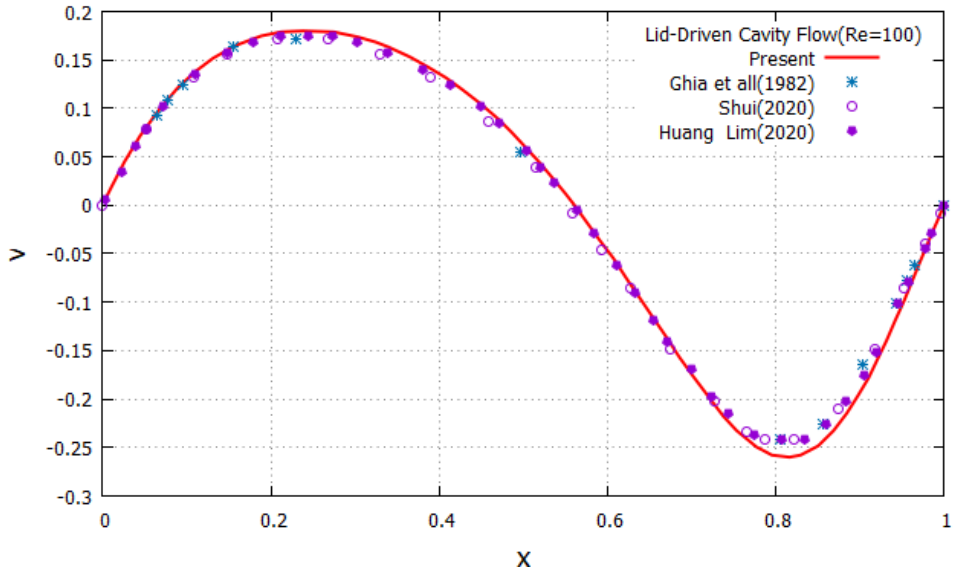


Рисунок 4.5 (с) – Скорости вдоль линий, проходящих через геометрический центр, профиль вертикальной скорости, Re=100

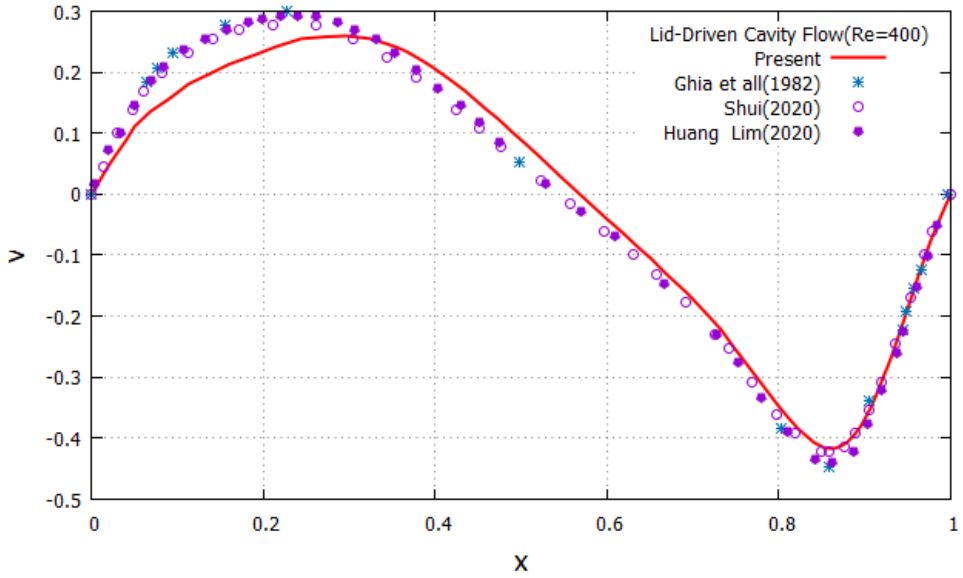


Рисунок 4.5(д) – Скорости вдоль линий, проходящих через геометрический центр, профиль вертикальной скорости, Re=400

4.5 Оценка эффективности параллельного численного алгоритма

На картинке 4.6 показано вертикальное распределение горизонтальных скоростей вдоль осевых линий для точного сравнения компонентов скорости параллельного численного решения на осевых линиях. На каждом рисунке координаты x и y обозначают соответствующие оси, а u и v представляют компоненты горизонтальной и вертикальной скоростей при x = 0,5 и y = 0,5 соответственно. О двумерной задаче с квадратными полостями, управляемыми крышкой, при различных числах Рейнольдса, получены сравнительные результаты Ghia и др. (1982), Shui (2020), Huang & Lim (2020).

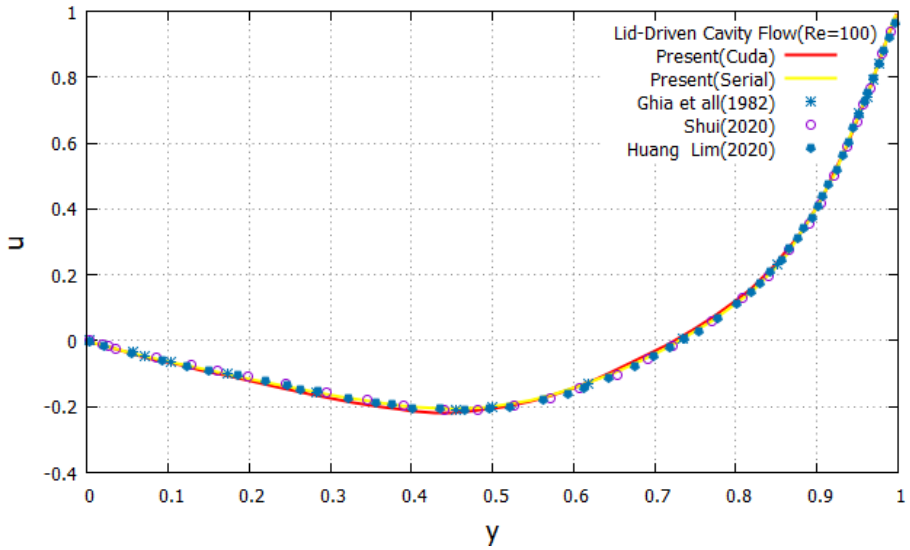


Рисунок 4.6 (а) – Скорости вдоль линий, проходящих через геометрический центр, по горизонтальной скорости, Re=100

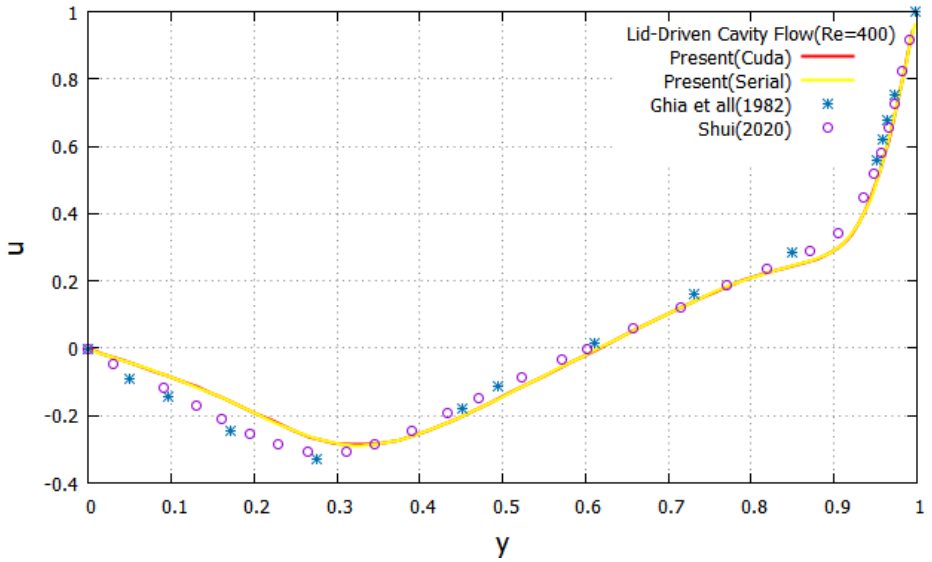


Рисунок 4.6 (б) – Скорости вдоль линий, проходящих через геометрический центр, по горизонтальной скорости, Re=400

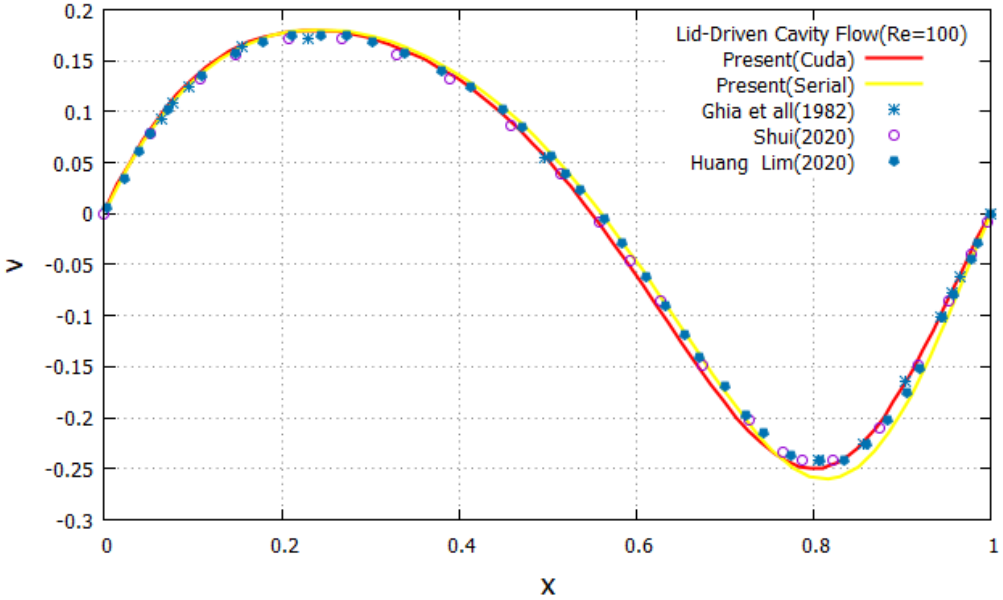


Рисунок 4.6 (с) – Скорости вдоль линий, проходящих через геометрический центр, по вертикальной скорости, Re=100

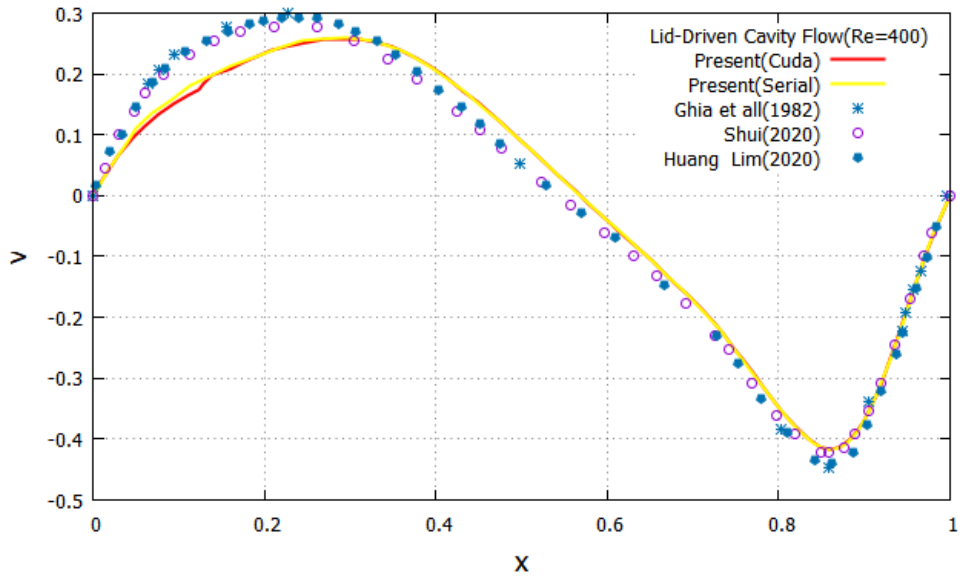


Рисунок 4.6 (д) – Скорости вдоль линий, проходящих через геометрический центр, по вертикальной скорости, Re=400

В таблице 4.1 показано сравнение времени процессора и графического процессора, что сделало возможным проанализировать производительность, показывающее значительное увеличение скорости, расчеты с использованием графического процессора. Сравнение вычислительной времени показывает преимущество технологии GPU в решении инженерных задач, требующих интенсивных численных вычислений. Анализ числа потоков в блоке, возможно, наиболее важного параметра распараллеливания в CUDA, показывает наличие оптимального значения. Этот результат является простым, но мощным методом оптимизации CUDA, который значительно влияет на общее время обработки (таблица 4.2).

Таблица 4.1 – Увеличение скорости вычислений

Время выполнения(мсек)					
Размер сетки	CPU (C)	GPU (shared memory)	GPU (global memory)	u-velocity (shared memory) v-velocity (global memory)	v-velocity (shared memory) u-velocity (global memory)
Re=100					
64x64	1436,53	1082,4	1345,5	1235,8	1237,6
128x128	3663,23	1266,15	2538,4	1756,4	1698,9
512x512	54101,6	13163,4	30012,8	2583,4	2678,9
Re=400					
64x64	1598,01	1122,6	1485,9	1356,8	1345,7

128x128	3831,09	1590,4	2945,6	2236,6	2348,3
512x512	56450,5	14685,8	37604,8	30696,3	29456,4

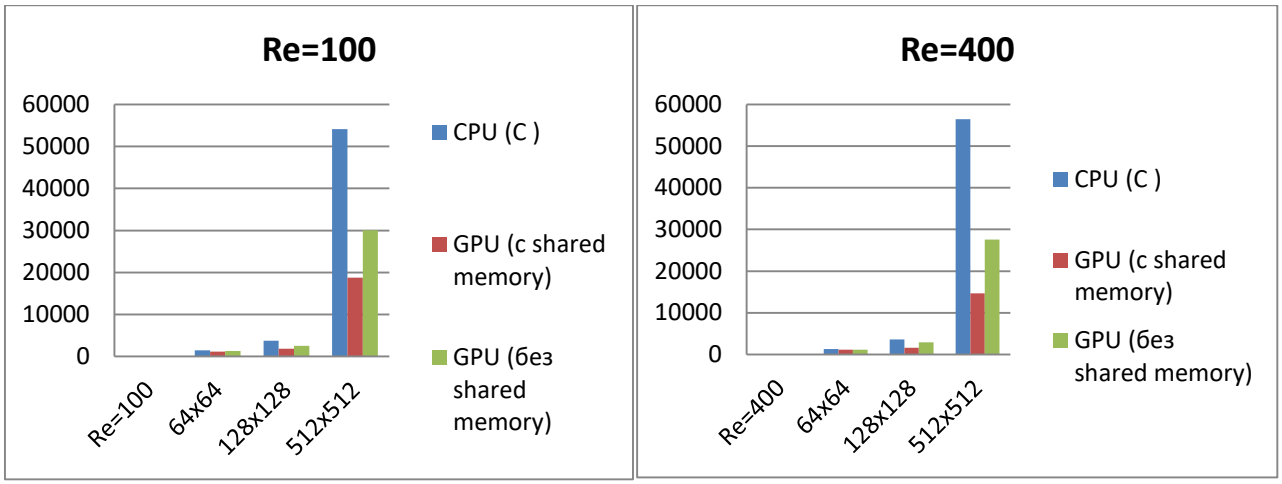


Рисунок 4.7 – Время выполнения программ при Re=100 и Re=400

Таблица 4.2 – Время вычисления с разным размером блока

Время вычисления(мсек)				
Размер сетки / Размер блока	64x64	128x128	512x512	1024x1024
4x4	1338,2	2434,38	23585,1	95280,6
8x8	1082,4	1266,15	13163,4	54372,6
16x16	1122,6	1590,4	14685,8	60232,1
8x32	1212,69	1852,65	21644,0	95649,8
32x8	1189,15	1651.02	14396.9	93072,6

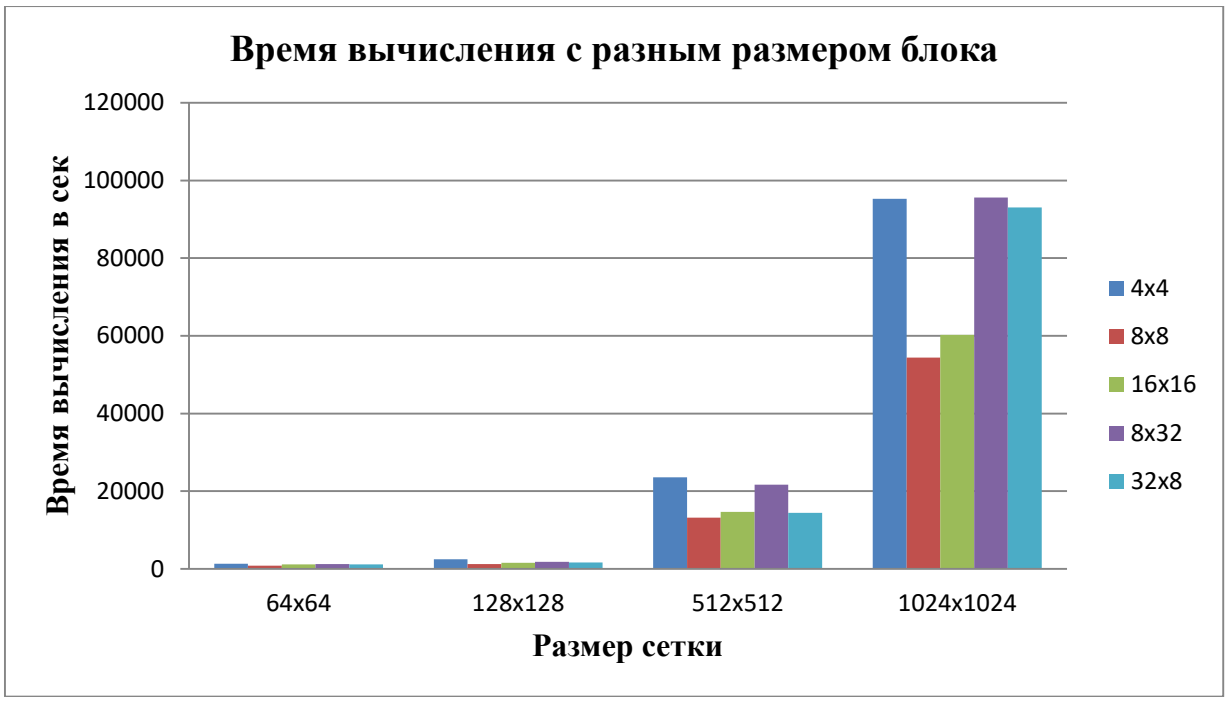


Рисунок 4.8 – Время вычисления с разным размером блока

Рисунок 4.9 – ускорение блоков с разными размерами

Времена вычислений полученного результата в разных блоках рисунков 4.8 и 4.9 были получены на 640-ядерном процессоре GeForce GTX 1050, графическом процессоре NVIDIA в микропроцессоре Pascal и процессоре Intel(R) Core (TM) i5-7500, 3,40 ГГц, ОЗУ 16 Гб.

Вычислительные возможности GTX 1050 составляют 6.1, что может удовлетворить потребности в крупномасштабных научных вычислениях. Большой объем памяти и мощные вычислительные возможности делают GTX 1050 привлекательным для решения циркуляционных несжимаемых вязких

течений в каверне. В таблице 4.3 приведены основные параметры работы GPU. Все характеристики влияют на производительность вычисления программы.

Таблица 4.3 - Основные параметры видеокарты NVIDIA GTX 1050

Параметры GPU	GTX 1050
Память устройства	2Gb
Вычислительные возможности	6.1
Количество потоковых процессоров	640
Пропускная способность	112Gb/sec
Одинарной с плавающей запятой	1.8 TFLOPs

Относительно недорогой графический процессор может значительно ускорить вычисления CFD. Массовое производство графических процессоров сократило расходы для конечного потребителя, а исследования и инновации производителей резко увеличили вычислительную мощность и улучшили использование энергии. Важным свойством является масштабируемость, поскольку несколько устройств могут быть объединены для увеличения вычислительной мощности. Кроме того, программное обеспечение постоянно совершенствуется, включая новые возможности языка и библиотеки для конкретных задач. Соответственно, технология GPU-интересный и доступный вариант ускорения инженерных вычислений.

Заключение по 4 разделу

В данном разделе рассматривается задача несжимаемого вязкого течения в каверне. На основе данной задачи получен результат эффективности на GPU с применением разных размеров блока. Сделан сравнительный анализ времени процессора и графического процессора производительности, показавший значительное увеличение скорости графического процессора. Данное сравнение вычислительного времени показывает преимущество технологии GPU в решении инженерных задач, требующих интенсивных численных вычислений. Анализ числа потоков в блоке, возможно, наиболее важного параметра распараллеливания в CUDA, показывает наличие оптимального значения. Этот результат является простым, но мощным методом оптимизации CUDA, который значительно влияет на общее время обработки.

5 РАЗРАБОТАТЬ ЭФФЕКТИВНОГО ВЫСОКОПРОИЗВОДИТЕЛЬНОГО ВЫЧИСЛЕНИЯ ДЛЯ ЗАДАЧ НЕСЖИМАЕМОГО ВЯЗКОГО ТЕЧЕНИЯ ЗА ОБРАТНЫМ УСТУПОМ ДЛЯ СТРУКТУРИРОВАННОЙ И НЕСТРУКТУРИРОВАННОЙ СЕТКИ

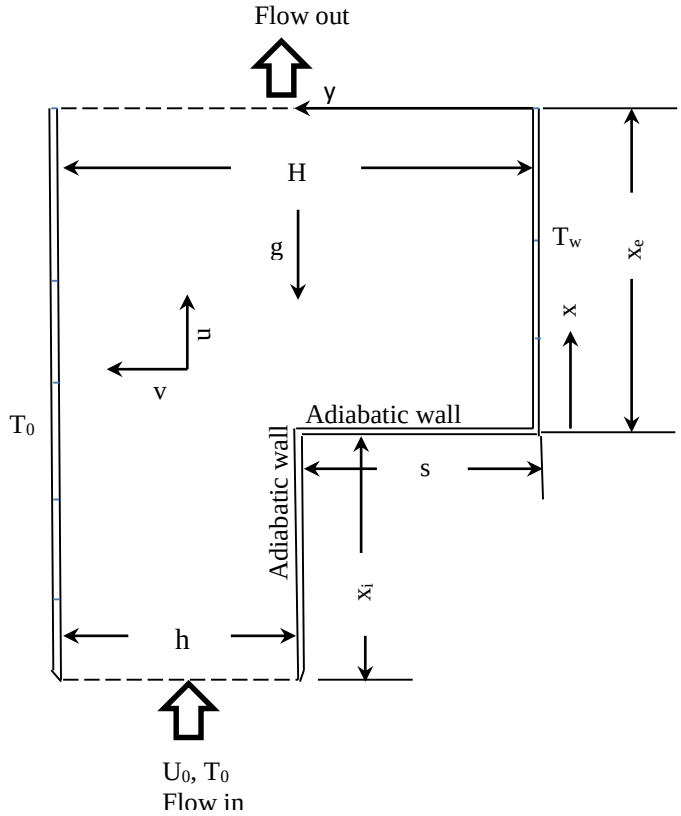
5.1 Математическая постановка задачи несжимаемого вязкого течения за обратным уступом

Через возвратную ступень течение жидкости и теплопередача представляют собой интересные проблемы прикладных исследований. Это решение представляет собой решение различных реальных промышленных проблем и в строительстве крупных городов, чтобы сохранить естественную аэродинамику между зданиями. Теоретически задачи этого класса могут быть использованы в различных промышленных устройствах, а также в бытовой технике. Некоторые из них относятся к производственным процессам, распределительному потоку за транспортными средствами, потоку двигателя, поступающему в туннель, теплообменникам конденсатора/камеры сгорания или электронным устройствам и микросхемам. Во многих потоках, представляющих практический интерес, часто встречается явление разделения потока, связанное с внезапным расширением геометрии и дальнейшим соединением. Наличие отрыва потока и наличие зоны рециркуляции оказывают существенное влияние на работу теплообменных устройств, таких как электронное охлаждающее оборудование, охлаждающие каналы лопаток турбин, камеры сгорания и многие другие поверхности теплообменников, встречающиеся в технических конструкциях [77, 84]. Таким образом, во многих работах, посвященных этим явлениям, можно найти множество численных, экспериментальных и аналитических исследований.

Резкие изменения ориентации стенок в проточных каналах значительно оказывает большое влияние на характеристики течения вблизи стенки, и иногда вызывают разрушение и повторное присоединение. Типичная ситуация протекает через уступ, обращенный назад, где уступ, обращенный вперед, образует узкий поток, а уступ, направленный назад, приводит к внезапному расширению потока. Многочисленные исследования показали, что изменение коэффициента теплопередачи и коэффициента трения позволяет оптимизировать некоторые теплопередающие устройства, такие как электронные компоненты и системы охлаждения реакторов, каналы охлаждения, камеры сгорания и многие другие поверхности теплообмена. [61–66]. Исследования обратного ступенчатого течения в основном направлено на распределение скорости, нестабильности течения, структуру зоны рециркуляции и влияние коэффициента расширения и теплопередачи на нижнюю стенку. Обратный уступ, который образует внезапное расширение в канале течения, вызывает отрыв течения и создает зону рециркуляции за пределами смещения. Наличие рециркуляционной зоны обеспечивает высокую теплоотдачу ламинарному потоку, как описано в работах [67–69, 73]. В работе [68] исследована длина присоединения и расположение максимальной скорости

теплообмена при ламинарном течении в канале за обратным уступом. В работе [70] анализируются угловые вихревые и первичные рециркуляционные вихревые образования за обратным отступом для различных коэффициентов расширения в малых и средних числах Рейнольдса. Кроме того, в работе [73] исследовали влияние высоты смещения на турбулентный расщепленный поток с помощью численного моделирования. Они обнаружили, что размер области первичной рециркуляции увеличивается с высотой ступеней, а высота ступеней также оказывает сильное влияние на коэффициент трения, турбулентную кинетическую энергию и объемную температуру. Oztop и Abu-Nada [27] численно исследовали естественную конвекцию в прямоугольных оболочках, частично обогреваемых от боковой стенки методом конечных объемов. В рассматриваемом исследовании показано влияние выталкивающих сил на характеристики течения и теплообмена в отдельных течениях. Численные решения для ламинарного смешанно-конвективного течения воздуха (Pr = 0,7) в двумерном вертикальном канале с обратным уступом высотой s для сохранения эффекта плавучести представлены на рисунке 5.1. Представляющие интерес численные результаты, такие как распределения скоростей и температур, длины повторного связывания и коэффициенты трения, представлены с целью иллюстрации влияния сил плавучести на эти параметры[71].

Рисунок 5.1 – Эскиз конфигурации потока через ступеньку, обращенную назад



Конфигурация, исследуемая в настоящей работе, представляет собой обратную ступеньку, направленной вертикально с высотой ступени s как видно

на рисунке 5.1. Прямая стенка канала поддерживается при постоянной температуре, равной температуре воздуха на входе T_0. При обратной стенке ступени нижняя часть канала нагревается до равномерной температуры, которую можно регулировать до любого желаемого значения T_w. Верхняя часть ступенчатой стенки и задняя часть облицовочного уступа устанавливаются как адиабатические поверхности. Входная длина канала x_i и выходная нижняя длина канала x_e имеют соответствующие размеры. Эти длины предполагаются бесконечными, но область расчета ограничена длиной $L_e = x_e + x_i$. Меньший участок канала перед выступом имеет высоту h, а большой участок под сценой имеет высоту H = h + s. Воздух движется вверх по каналу со средней скоростью u_0 и однородной температурой T_0. Ускорение свободного падения g в этой задаче считается направленным вертикально вниз. Для описания этой физической задачи использовались предположения о постоянных свойствах и приближение Буссинеска. Эту систему уравнений в необъятном виде можно записать в виде:

$$\frac{\partial U}{\partial X} + \frac{\partial V}{\partial Y} = 0 \tag{5.1}$$

$$\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial X} + V\frac{\partial U}{\partial Y} = -\frac{\partial P}{\partial X} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 U}{\partial X^2} + \frac{\partial^2 U}{\partial Y^2}\right) + \frac{Gr}{\mathrm{Re}^2}\theta \tag{5.2}$$

$$\frac{\partial V}{\partial t} + U\frac{\partial V}{\partial X} + V\frac{\partial V}{\partial Y} = -\frac{\partial P}{\partial Y} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 V}{\partial X^2} + \frac{\partial^2 V}{\partial Y^2}\right) \tag{5.3}$$

$$\frac{\partial \theta}{\partial t} + U\frac{\partial \theta}{\partial X} + V\frac{\partial \theta}{\partial Y} = \frac{1}{\mathrm{Pr}\,\mathrm{Re}}\left(\frac{\partial^2 \theta}{\partial X^2} + \frac{\partial^2 \theta}{\partial Y^2}\right) \tag{5.4}$$

Безразмерные параметры в приведенных выше уравнениях определяются по формуле:

$$\begin{gathered} U = \frac{u}{u_0},\ V = \frac{v}{u_0},\ X = \frac{x}{s},\ Y = \frac{y}{s}, \\ \theta = \frac{(T - T_0)}{(T_w - T_0)},\ P = \frac{p}{\rho_0 u_0^2},\ \mathrm{Pr} = \frac{\nu}{\alpha}, \\ \mathrm{Re} = \frac{u_0 s}{\nu},\ Gr = \frac{g\beta\,(T_w - T_0)s^3}{\nu^2}. \end{gathered} \tag{5.5}$$

где α – температурная диффузия, ν – кинематическая вязкость и β – коэффициент теплового расширения оцениваются при температуре пленки $T_f = (T_0 + T_w)/2$ (см. рисунок 5.2).

Граничные условия:

(a) Условия на входе: В точке $X=-X_i$ и $1\le Y\le H/s$: $U=u_i/u_0$, $V=0$, $\theta=0$, $\dfrac{\partial p}{\partial x}=-\dfrac{Gr}{\mathrm{Re}^2}\theta$, где u_i— локальное распределение скоростей на входе, которое предполагается имеющим параболический профиль, а $\dfrac{u_i}{u_0}$— средняя скорость на входе, т. е. определяемая формулой:

$$\frac{u_i}{u_0}=6\left[-y^2+(H+s)y+Hs\right]/(H-s)^2.$$

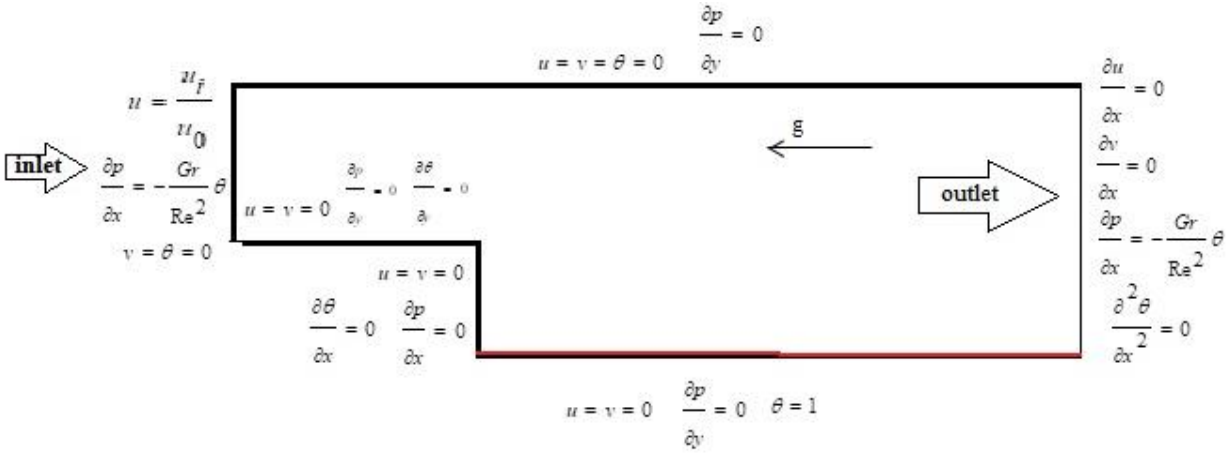


Рисунок 5.2 – Граничные условия

(b) Условия выхода: В точке $X=X_e$ и $0\le Y\le H/s$: $\dfrac{\partial U}{\partial X}=0$, $\dfrac{\partial^2\theta}{\partial X^2}=0$, $\dfrac{\partial V}{\partial X}=0$ $\dfrac{\partial p}{\partial x}=-\dfrac{Gr}{\mathrm{Re}^2}\theta$.

(c) на верхней стене: В точке $Y=H/s$ и $-X_i\le X\le X_e$: $U=0$, $V=0$, $\theta=0$, $\dfrac{\partial p}{\partial y}=0$.

(d) на стене верхней ступени: В точке $Y=1$ и $-X_i\le X<0$: $U=0$, $V=0$, $\dfrac{\partial\theta}{\partial Y}=0$, $\dfrac{\partial p}{\partial y}=0$.

(e) на стене нижней ступени: В точке $X=0$ и $0\le Y\le 1$: $U=0$, $V=0$, $\dfrac{\partial\theta}{\partial X}=0$, $\dfrac{\partial p}{\partial x}=0$.

(f) на стене под ступени: В точке $Y=0$ и $0\le X\le X_e$: $U=0$, $V=0$, $\theta=1$, $\dfrac{\partial p}{\partial y}=0$.

Последний член в правой части уравнения (5.2) является вкладом выталкивающей силы. Длина потока вниз по потоку от области моделирования была выбрана равной 70 шагам (Xe = 70). Верхняя длина расчетной области была выбрана равной 5 шагам (т. е. Xi = 5), а профиль скорости на входе был задан как параболический профиль, например, $u_i / u_0 = 6\left[- y^2 + (H + s)y - Hs\right]/(H - s)^2$, а температура выбиралась равномерная T_0 [73].

5.2 Дискретизация и численный алгоритм

5.2.1 Дискретизация и численный алгоритм для структурированной сетки

Для численного решения системы уравнений (5.1) - (5.4) применялся проекционный метод [74-75]. На первом этапе предполагается, что передача импульса осуществлялась только за счет конвекции и диффузии, а промежуточное поле скоростей решалось методом Рунге-Кутты 4-го порядка. На втором этапе по найденному промежуточному полю скорости находилось поле давления. Двухмерное уравнение Пуассона, которое показано во 2 разделе использовалось для поля давления с помощью метода Якоби. На третьем этапе перенос осуществлялся только за счет градиента давления. На четвертом этапе находилось уравнение теплообмена методом Рунге-Кутты 4-го порядка [66]. Разобьем равномерной сеткой по каждому направлению, $\Delta x = 2/(N-1)$, $\Delta y = 75/(M-1)$.

I. $$\frac{\vec{u}^* - \vec{u}^n}{\Delta t} = -\left(\vec{u}^n \vec{u}^* - \frac{1}{\mathrm{Re}}\nabla\vec{u}^*\right) n_i - \frac{Gr}{\mathrm{Re}^2}\theta$$

II. $$\nabla p = \frac{\nabla \vec{u}^*}{\Delta t},$$

III. $$\frac{\vec{u}^{n+1} - \vec{u}^*}{\Delta t} = -\nabla p,$$

IV. $$\frac{\theta^* - \theta^n}{\Delta t} = -\left(u^n\theta^* - \frac{1}{\mathrm{Re}\,\mathrm{Pr}}\nabla\theta^*\right) n_i.$$

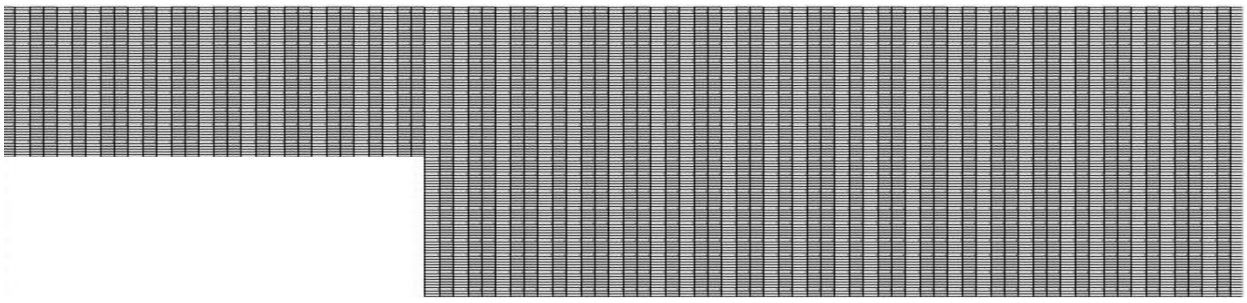


Рисунок 5.3 – Структурированная сетка

5.2.2 Дискретизация и численный алгоритм для неструктурированной сетки

Для численного решения задачи несжимаемого вязкого течения за обратным уступом на неструктурированных сетках (на рисунке 5.4) применялся алгоритм кривая Гильберта. Чтобы сгенерировать сетку, мы используем настраиваемое уточнение сетки без необходимости повторного построения сетки во время моделирования, так как геометрические особенности остаются неизменными. Нахождение всех ближайших соседей используется кривой Гильберта. Алгоритм построения кривой Гильберта показаны на рисунке 5.5.

Современные компьютерные системы характеризуются глубокой иерархией памяти, состоящей из основной памяти и нескольких уровней кэш-памяти, других специально подготовленных типов памяти. К этой иерархии в параллельных и распределенных системах добавляются дополнительные уровни памяти. Это связано с эффективным использованием иерархической памяти с точки зрения времени выполнения для достижения лучшей производительности приложения вычислительной науки. А неэффективное использование иерархической памяти может привести к перемещению данных. Это верно в отношении GPU, где методы обеспечения устойчивости к задержкам, основанные на планировании потоков, применяются для маскировки несоответствия между пропускной способностью глобальной памяти и пиковой скоростью выполнения GPU [81].

Из-за отсутствия полного упорядочения пространственной близости между пространственными объектами подход с использованием кривой Гильберта был предложен для сохранения пространственной локальности.

Локальность данных является основным фактором эффективного использования иерархической памяти. Когда один элемент перемещается из нижнего уровня памяти в верхний, другие элементы, находящиеся рядом в памяти, также перемещаются вместе с ним. Большинство вычислений и явлений являются локальными, поэтому, если элементы хранятся в памяти в зависимости от местоположения, когда один элемент перемещается на более высокий уровень памяти, другие элементы, связанные с его обработкой, также перемещаются, таким образом, используя локальность пространственных данных и повышая производительность [81].

Для первой проблемы легко понять, что процесс иерархической декомпозиции должен быть прекращен, когда каждая ячейка сетки включает не более одного пространственного объекта. Количество ячеек сетки должно быть больше количества пространственных объектов. Согласно структуре кривой Гильберта, кривая Гильберта порядка M имеет ячейки сетки$2^M \times 2^M$. Таким образом, M и n должны соответствовать условию $n < 2^{2M}$, т. е. $M > \frac{1}{2} log_2 n$. Здесь пусть окончательный порядок $M = \left[\frac{1}{2} log_2 n\right] + 1$. Теоретически, начальный порядок m_0 может быть любым положительным целым числом, для

которого нужно только условие $m_0 < M$, однако, если m_0 слишком мало, это приведет к переполной иерархической декомпозиции, тогда как, если m_0 слишком велико, это может быть ненужным. Таким образом, исходя из практического опыта, мы предлагаем значение начального порядка $m_0 = [M/2]$[82].

Данный алгоритм применялся в научных исследованиях для улучшенного использования кэша и показания его эффективности [79], для быстрой и точной операционной модели цунами в реальном времени с расчетом по всему океану [80], для реализации решателя конечного объема на сетке с несколькими разрешениями multi-GPU [81], для моделирования молекулярной динамики [82].

Заполнения пространства кривая Гильберта (рисунок 5.5) использовалась для расчета несжимаемых вязких течений от обратной дуги на неструктурированных расчетных сетках (рисунок 5.4).

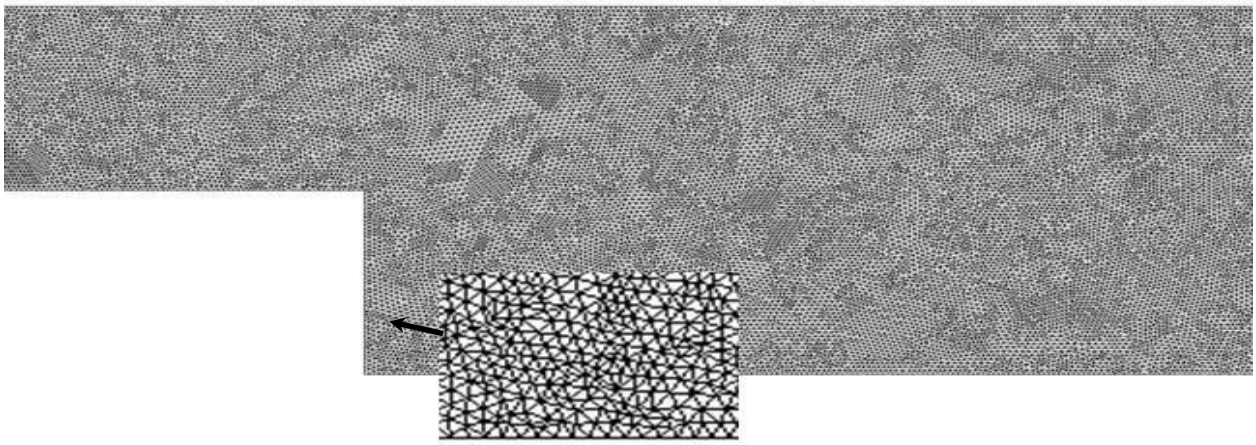


Рисунок 5.4 – Неструктурированная сетка

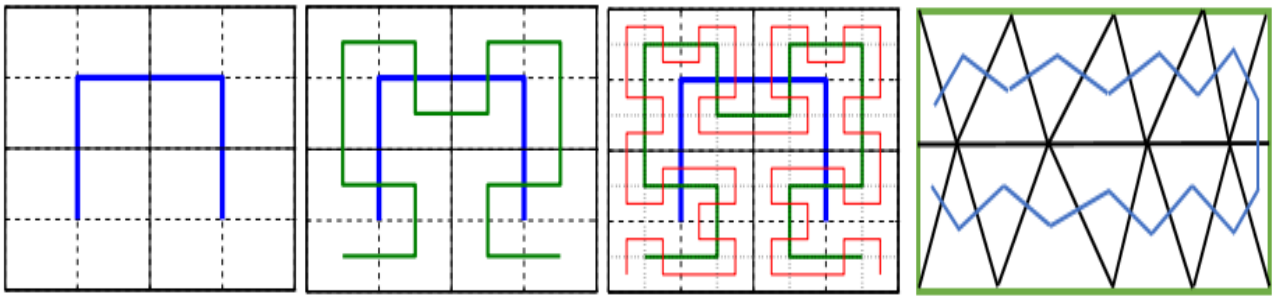


Рисунок 5.5 – Алгоритм построения кривой Гильберта

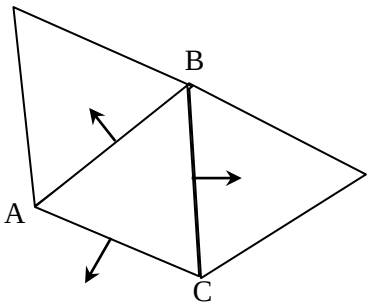


Рисунок 5.6 - контрольный объем по ячейке

Для неструктурированных сеток, содержащих миллиарды элементов, решение уравнений в частных производных с использованием FVM представляет собой очень сложную вычислительную задачу. Параллельная реализация может дать решение в течение разумного времени, но они страдают от меньшего использования кэша из-за неструктурированных шаблонов доступа к данным. В этой задаче мы изменяем способ хранения вершин и элементов сетки в памяти, используя заполнения пространства кривой Гильберта для улучшения использования кэша в FVM для неструктурированных сеток. Этот метод переупорядочивания перечисляет элементы сетки таким образом, чтобы параллельные потоки могли получать доступ к общим вершинам в течение разных периодов времени, что уменьшает время, затрачиваемое на ожидание получения блокировок, охраняющих атомарные области [78].

Рассмотрим формулировки FVM для системы уравнение (5.1)-(5.4). Уравнение конечного объёма становиться

I. $$\int \frac{\Delta u}{\Delta t}\, d\Omega = \int \left(-uun_1 - uvn_2\right) d\Gamma + \frac{1}{\mathrm{Re}} \int \left(\frac{\Delta u}{\Delta x} n_1 + \frac{\Delta u}{\Delta y} n_2 \right) d\Gamma + \int \frac{Gr}{\mathrm{Re}^2}\, T\, d\Omega$$

$$\int \frac{\Delta v}{\Delta t}\, d\Omega = \int \left(-vun_1 - vvn_2\right) d\Gamma + \frac{1}{\mathrm{Re}} \int \left(\frac{\Delta v}{\Delta x} n_1 + \frac{\Delta v}{\Delta y} n_2 \right) d\Gamma$$

II. $$\int \left(\frac{\Delta P}{\Delta x} n_1 + \frac{\Delta P}{\Delta y} n_2 \right) d\Gamma = \int \frac{1}{\Delta t} \left(un_1 + vn_2 \right) d\Gamma$$

III. $$\int \frac{\Delta u}{\Delta t}\, d\Omega = \int Pn_1\, d\Gamma$$

$$\int \frac{\Delta v}{\Delta t}\, d\Omega = \int Pn_2\, d\Gamma$$

IV. $$\int \frac{\Delta T}{\Delta t}\, d\Omega = \int \left(-uTn_1 - vTn_2\right) d\Gamma + \frac{1}{\mathrm{Pr}\,\mathrm{Re}} \int \left(\frac{\Delta T}{\Delta x} n_1 + \frac{\Delta T}{\Delta y} n_2 \right) d\Gamma$$

или

I. $$\sum_{CV} \frac{\Delta u}{\Delta t} \Delta\Omega = \sum_{CS}^{A,B,C} \left(-uun_1 - uvn_2\right) \Delta\Gamma + \frac{1}{\mathrm{Re}} \sum_{CS}^{A,B,C} \left(\frac{\Delta u}{\Delta x} n_1 + \frac{\Delta u}{\Delta y} n_2 \right) \Delta\Gamma + \frac{Gr}{\mathrm{Re}^2} \sum_{CV} T \Delta\Omega$$

$$\sum_{CV}\frac{\Delta v}{\Delta t}\Delta\Omega=\sum_{CS}^{A,B,C}\left(-vun_1-vvn_2\right)\Delta\Gamma+\frac{1}{\mathrm{Re}}\sum_{CS}^{A,B,C}\left(\frac{\Delta v}{\Delta x}n_1+\frac{\Delta v}{\Delta y}n_2\right)\Delta\Gamma$$

II. $$\sum_{CS}^{A,B,C}\left(\frac{\Delta P}{\Delta x}n_1+\frac{\Delta P}{\Delta y}n_2\right)\Delta\Gamma=\sum_{CS}^{A,B,C}\frac{1}{\Delta t}\left(un_1+vn_2\right)\Delta\Gamma$$

III. $$\sum_{CV}\frac{\Delta u}{\Delta t}\Delta\Omega=\sum_{CS}^{A,B,C}Pn_1\Delta\Gamma$$

$$\sum_{CV}\frac{\Delta v}{\Delta t}\Delta\Omega=\sum_{CS}^{A,B,C}Pn_2\Delta\Gamma$$

IV. $$\sum_{CV}\frac{\Delta T}{\Delta t}\Delta\Omega=\sum_{CS}^{A,B,C}\left(-uTn_1-vTn_2\right)\Delta\Gamma+\frac{1}{\mathrm{Pr\,Re}}\sum_{CS}^{A,B,C}\left(\frac{\Delta T}{\Delta x}n_1+\frac{\Delta T}{\Delta y}n_2\right)\Delta\Gamma$$

Для схемы с центром в ячейке это можно записать в виде:

$$\frac{\Delta u}{\Delta t}\Delta\Omega=-uun_1\Delta\Gamma_{AB}-uvn_2\Delta\Gamma_{AB}-uun_1\Delta\Gamma_{BC}-uvn_2\Delta\Gamma_{BC}-uun_1\Delta\Gamma_{CA}-uvn_2\Delta\Gamma_{CA}+$$

$$+\frac{1}{\mathrm{Re}}\left(\frac{\Delta u}{\Delta x}n_1\Delta\Gamma_{AB}+\frac{\Delta u}{\Delta y}n_2\Delta\Gamma_{AB}+\frac{\Delta u}{\Delta x}n_1\Delta\Gamma_{BC}+\frac{\Delta u}{\Delta y}n_2\Delta\Gamma_{BC}+\frac{\Delta u}{\Delta x}n_1\Delta\Gamma_{CA}+\frac{\Delta u}{\Delta y}n_2\Delta\Gamma_{CA}\right)+$$

$$+\frac{Gr}{\mathrm{Re}^2}T\Delta\Omega,$$

$$\frac{\Delta v}{\Delta t}\Delta\Omega=-vun_1\Delta\Gamma_{AB}-vvn_2\Delta\Gamma_{AB}-vun_1\Delta\Gamma_{BC}-vvn_2\Delta\Gamma_{BC}-vun_1\Delta\Gamma_{CA}-vvn_2\Delta\Gamma_{CA}+$$

$$+\frac{1}{\mathrm{Re}}\left(\frac{\Delta v}{\Delta x}n_1\Delta\Gamma_{AB}+\frac{\Delta v}{\Delta y}n_2\Delta\Gamma_{AB}+\frac{\Delta v}{\Delta x}n_1\Delta\Gamma_{BC}+\frac{\Delta v}{\Delta y}n_2\Delta\Gamma_{BC}+\frac{\Delta v}{\Delta x}n_1\Delta\Gamma_{CA}+\frac{\Delta v}{\Delta y}n_2\Delta\Gamma_{CA}\right),$$

$$\frac{\Delta P}{\Delta x}n_1\Delta\Gamma_{AB}+\frac{\Delta P}{\Delta y}n_2\Delta\Gamma_{AB}+\frac{\Delta P}{\Delta x}n_1\Delta\Gamma_{BC}+\frac{\Delta P}{\Delta y}n_2\Delta\Gamma_{BC}+\frac{\Delta P}{\Delta x}n_1\Delta\Gamma_{CA}+\frac{\Delta P}{\Delta y}n_2\Delta\Gamma_{CA}=$$

$$=\frac{1}{\Delta t}(un_1\Delta\Gamma_{AB}+vn_2\Delta\Gamma_{AB}+un_1\Delta\Gamma_{BC}+vn_2\Delta\Gamma_{BC}+un_1\Delta\Gamma_{CA}+vn_2\Delta\Gamma_{CA}),$$

$$\frac{\Delta u}{\Delta t}\Delta\Omega=Pn_1\Delta\Gamma_{AB}+Pn_1\Delta\Gamma_{BC}+Pn_1\Delta\Gamma_{CA},$$

$$\frac{\Delta v}{\Delta t}\Delta\Omega=Pn_2\Delta\Gamma_{AB}+Pn_2\Delta\Gamma_{BC}+Pn_2\Delta\Gamma_{CA},$$

$$\frac{\Delta T}{\Delta t}\Delta\Omega=-uTn_1\Delta\Gamma_{AB}-vTn_2\Delta\Gamma_{AB}-uTn_1\Delta\Gamma_{BC}-vTn_2\Delta\Gamma_{BC}-uTn_1\Delta\Gamma_{CA}-vTn_2\Delta\Gamma_{CB}+$$

$$+\frac{1}{\mathrm{PrRe}}(\frac{\Delta T}{\Delta x}n_1\Delta\Gamma_{AB}+\frac{\Delta T}{\Delta y}n_2\Delta\Gamma_{AB}+\frac{\Delta T}{\Delta x}n_1\Delta\Gamma_{BC}+\frac{\Delta T}{\Delta y}n_2\Delta\Gamma_{BC}+\frac{\Delta T}{\Delta x}n_1\Delta\Gamma_{CA}+\frac{\Delta T}{\Delta y}n_2\Delta\Gamma_{CA}.$$

Данный алгоритм для задачи несжимаемого вязкого течения за обратным уступом на неструктурированных сетках показал высокий результат. Графический процессор используется для дальнейшего ускорения вычислений,

а измеренная заполнения кривая Гильберт-пространства используется для создания сбалансированной рабочей нагрузки.

5.3 Создание параллельных схем в системе CUDA. Проблемы, пути решения

Приведен алгоритм распараллеливания на GPU. Мы уделяем особое внимание повышению производительности параллельных алгоритмов. Замечено, что коды, работающие на платформе CUDA, дают ожидаемые результаты. Сравнивая наши тесты на графическом процессоре с тестами, полученными при запуске последовательного кода той же симуляции на ЦП, оказывается, что симуляции на графическом процессоре выполняются намного быстрее, чем на ЦП. GPU выполняет все вычисления, а процессор отвечает только за управление программным обеспечением GPU, ввод-вывод файлов и передачу данных. Этот код написан с использованием C++ и CUDA C/C++. В алгоритме 5.1 показан решение задачи (5.1)-(5.4).

Алгоритм 5.1: Алгоритм распараллеливания CFD-кода CUDA

```
1.  void initializeCPU()            // выделение памяти на хосте
2.  void unInitializeCPU()   //очистить выделенную память на хосте
3.  void initializeGPU()            // выделение памяти на устройстве
4.  void unInitializeGPU()   //очистить выделенную память на устройстве
5.  __global__ void uCalculation()
6.  __global__ void vCalculation()
7.  __global__ void fnCalculation()
8.  __global__ void PnCalculation()
9.  __global__ void unp1Calculation()
10. __global__ void vnp1Calculation()
11. __global__ void TCalculation()
12. Input                   // параметрические значения и параметры
13. Initialization          // инициализация переменных
14. for i=1 to imax do
15.     update boundary grid points
16. end for
17. cudaMemcpy()  //копировать данные с хоста на устройство
18. while k<nIterations do
19.    uCalculation << <dimGrid, dimBlock >> >()  //запуск ядро
20.    vCalculation << <dimGrid, dimBlock >> >()  //запуск ядро
21.    fnCalculation << <dimGrid, dimBlock >> > ()//запуск ядро
22.    PnCalculation << <dimGrid, dimBlock >> > ()//запуск ядро
23.    unp1Calculation << <dimGrid, dimBlock >> >()  //запуск ядро
24.    vnp1Calculation << <dimGrid, dimBlock >> > ()//запуск ядро
25.    TCalculation << <dimGrid, dimBlock >> > ()//запуск ядро
26.    k++
27. end while
28. cudaMemcpy()  //копировать данные с устройство на хоста
29. print U, V, P and T values
```

Как показано на рисунке 5.7, модель реализации задачи несжимаемого вязкого течения за обратным уступом.

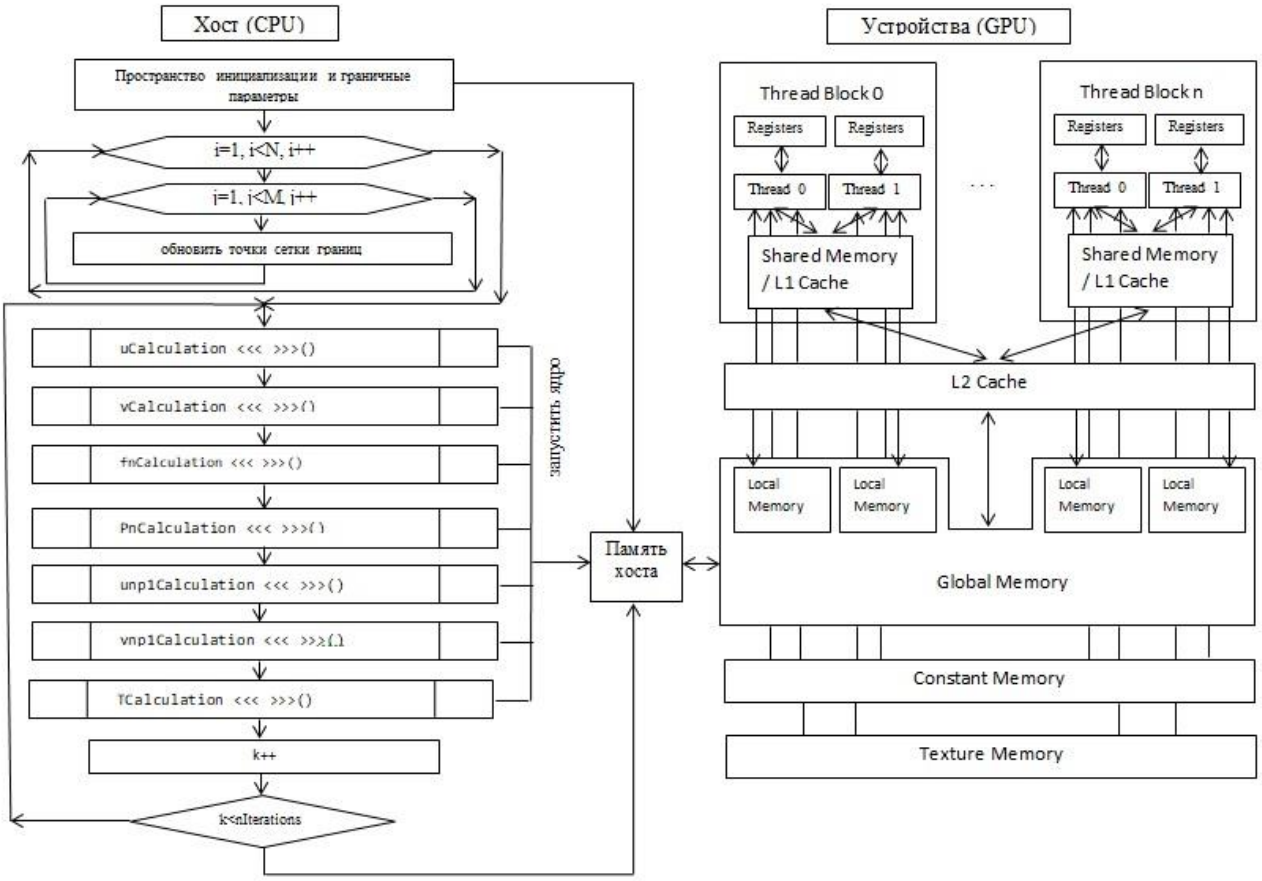


Рисунок 5.7 - Модель реализации задачи несжимаемого вязкого течения за обратным уступом

5.4 Результаты численного расчета

Для решения задачи применялись уравнения Навье-Стокса с силами плавучести и уравнение теплопереноса. Эта система решалась численно проекционным методом. Геометрические параметры указаны на рис. 5.1: высота канала H = 2s, длина канала L = 75, высота ступеньки s = 1. Численные результаты получены для безразмерных чисел Pr = 0,7, Re = 50 и Gr = 15,7.

В списке литературы приведены результаты ученных в статьях, с которыми были сравнены полученные нами экспериментальные и численные результаты ламинарного течения за обратным уступом.

Численные результаты были получены с плотностью сетки для узлов скорости, а именно 1423 x 41. На рисунке 5.8 и 5.9 показаны результаты скорости и распределения температуры для безразмерных параметров Re = 50, Pr = 0,7 и Gr = 15,7.

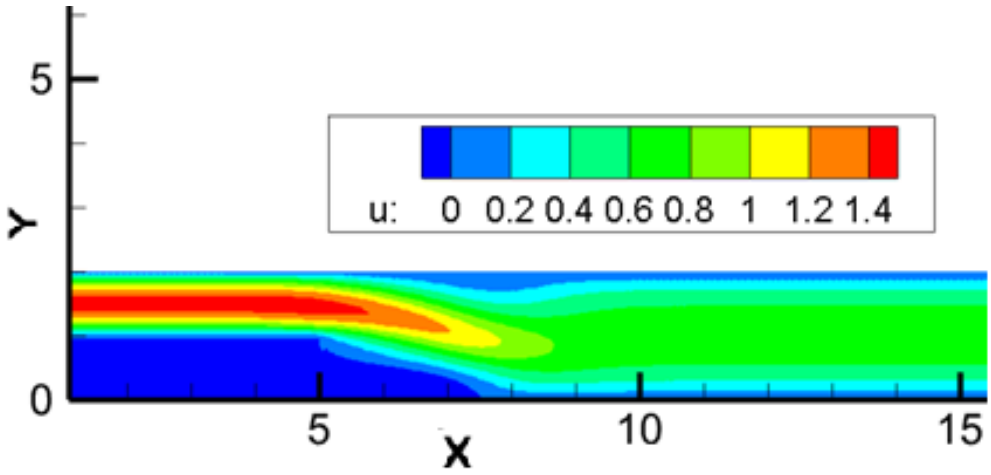


Рисунок 5.8 – Результаты моделирования линии тока скорости над каналом с обратными уступами

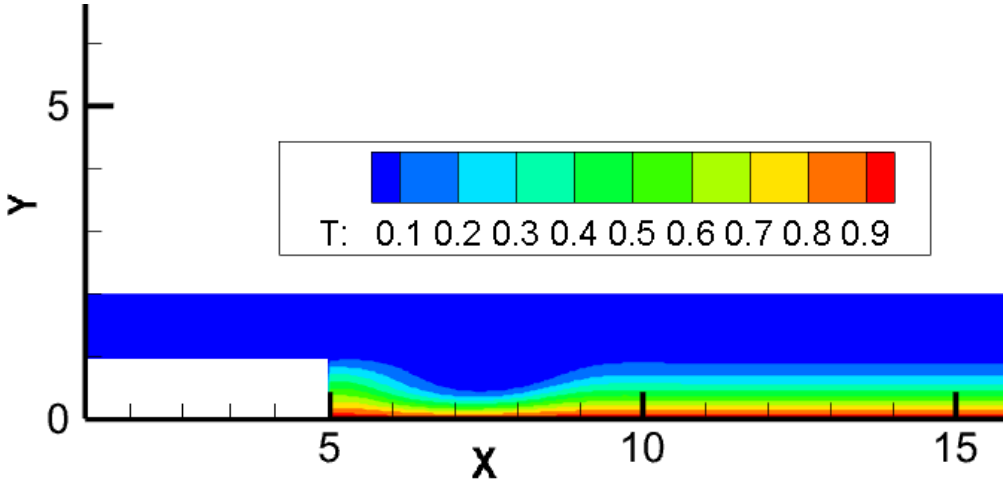


Рисунок 5.9 – Результаты распределения температуры над каналом с обратными уступами

На рисунке 5.10(а,б,с) и 5.11(а,б,с) показаны профили скорости и температуры для параметров Pr = 0,7, Re = 50 и Gr = 15,7 для x/x_f={0,5; 1; 2}. Результаты совпадает сравнительным результатом Lin и др.

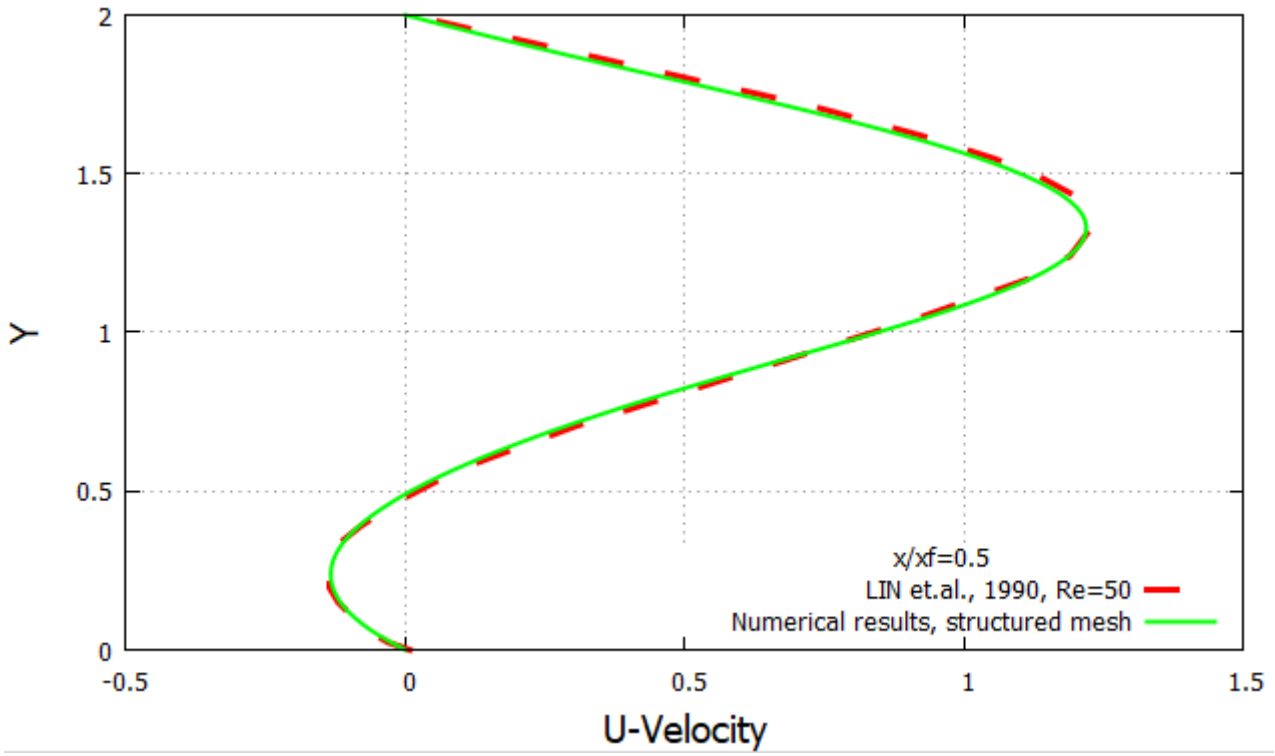


Рисунок 5.10(а) – Влияние плавучести на распределение скорости для Re=50 при $\Delta T = 1°C$, поперечная сечения $x / x_f = 0.5$

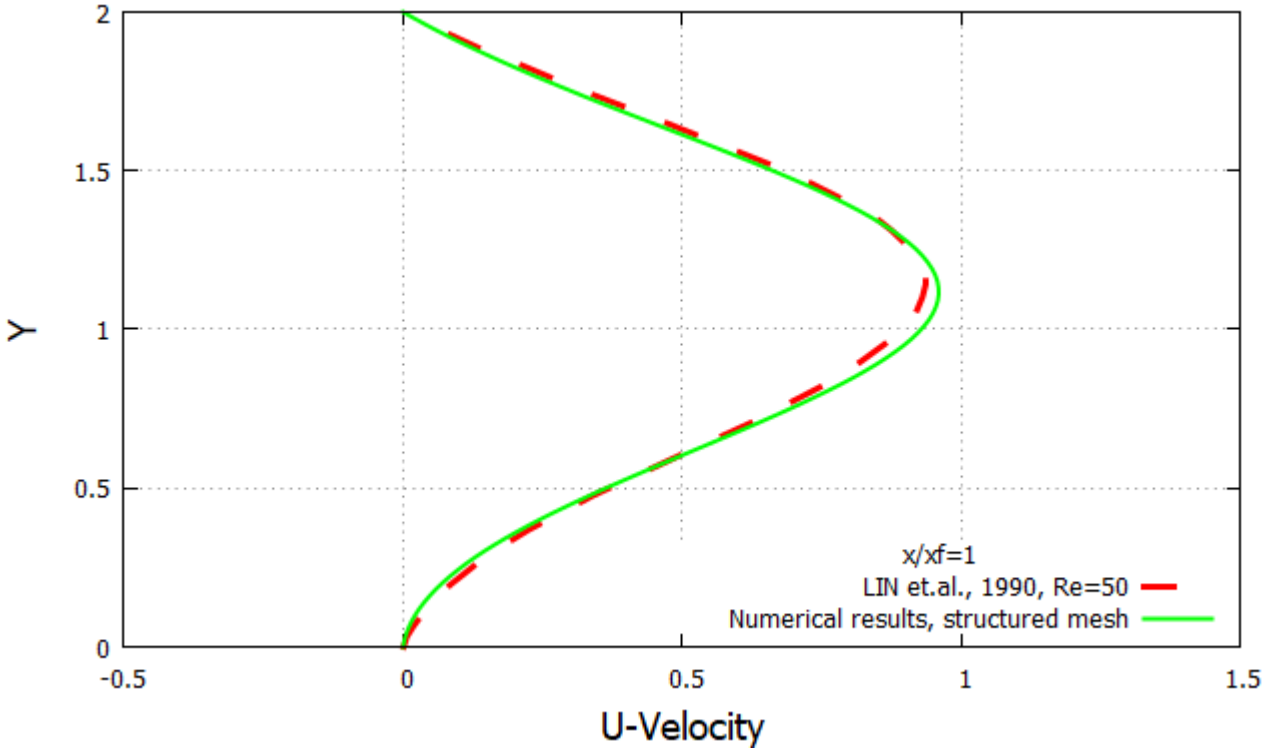


Рисунок 5.10(б) – Влияние плавучести на распределение скорости для Re=50 при $\Delta T = 1°C$, поперечная сечения $x / x_f = 1$

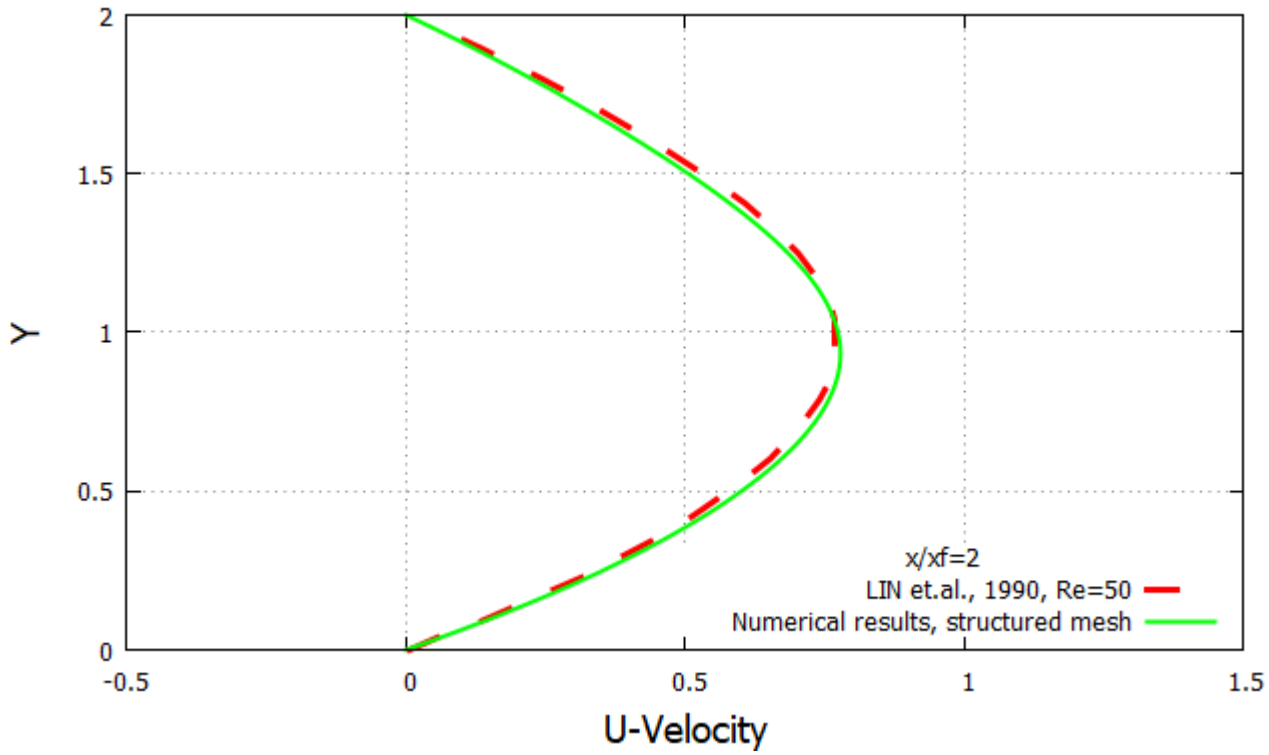


Рисунок 5.10(c) – Влияние плавучести на распределение скорости для Re=50 при $\Delta T = 1°C$, поперечная сечения $x/x_f = 2$

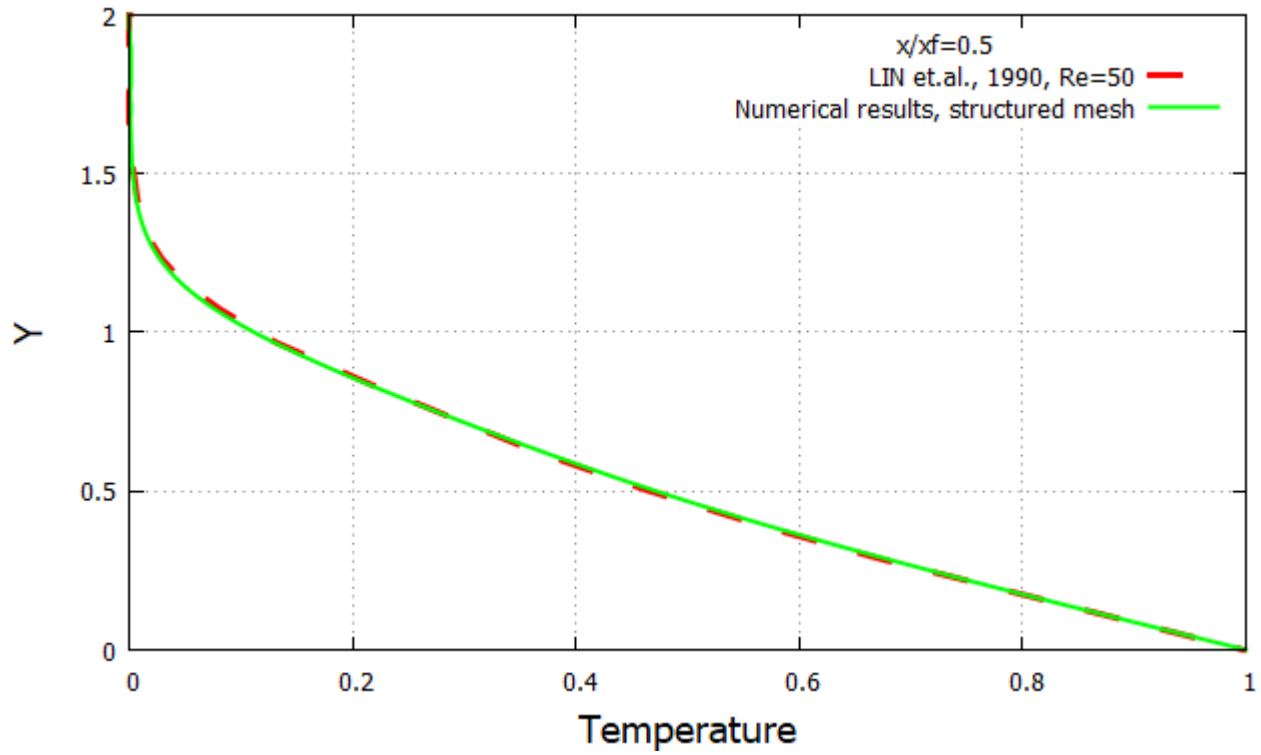


Рисунок 5.11(a) – Влияние плавучести на распределение температуры для Re=50 при $\Delta T = 1°C$, поперечная сечения $x/x_f = 0.5$

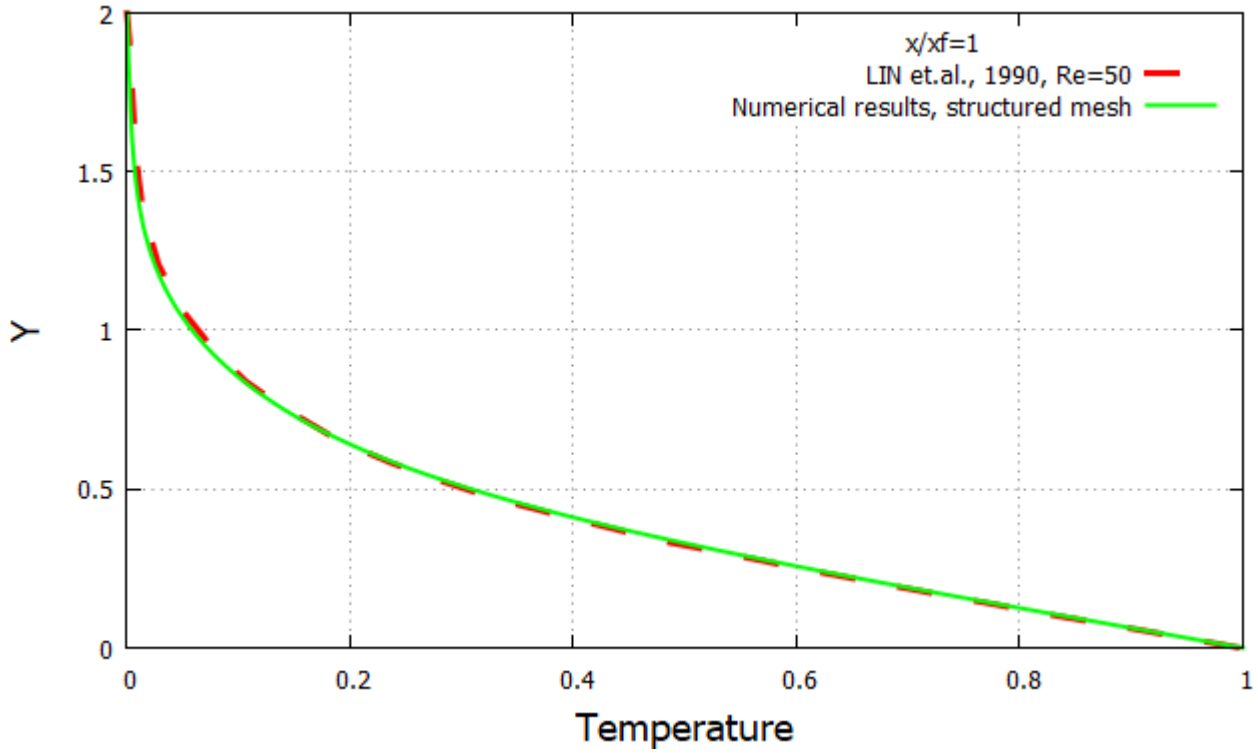


Рисунок 5.11(б) – Влияние плавучести на распределение температуры для Re=50 при $\Delta T = 1°C$, поперечная сечения $x / x_f = 1$

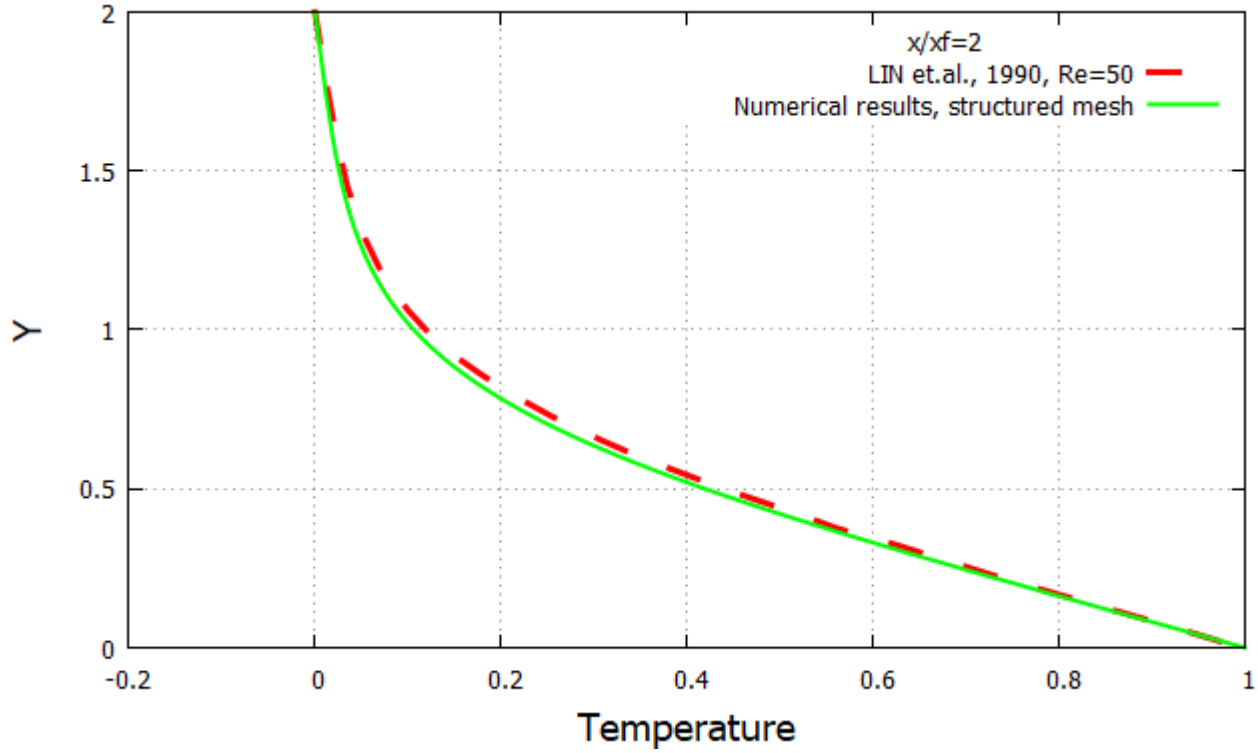


Рисунок 5.11(с) – Влияние плавучести на распределение температуры для Re=50 при $\Delta T = 1°C$, поперечная сечения $x / x_f = 2$.

5.5 Оценка эффективности параллельного численного алгоритма

С использованием CUDA технологию на графическом процессоре, полученные результаты расчетов в структурированных и неструктурированных сетках сравнивали с расчетными значениями и измерительными данными Lin и др. [77]. Численные результаты, полученные по результатам сравнительного анализа, показали удовлетворительные результаты.

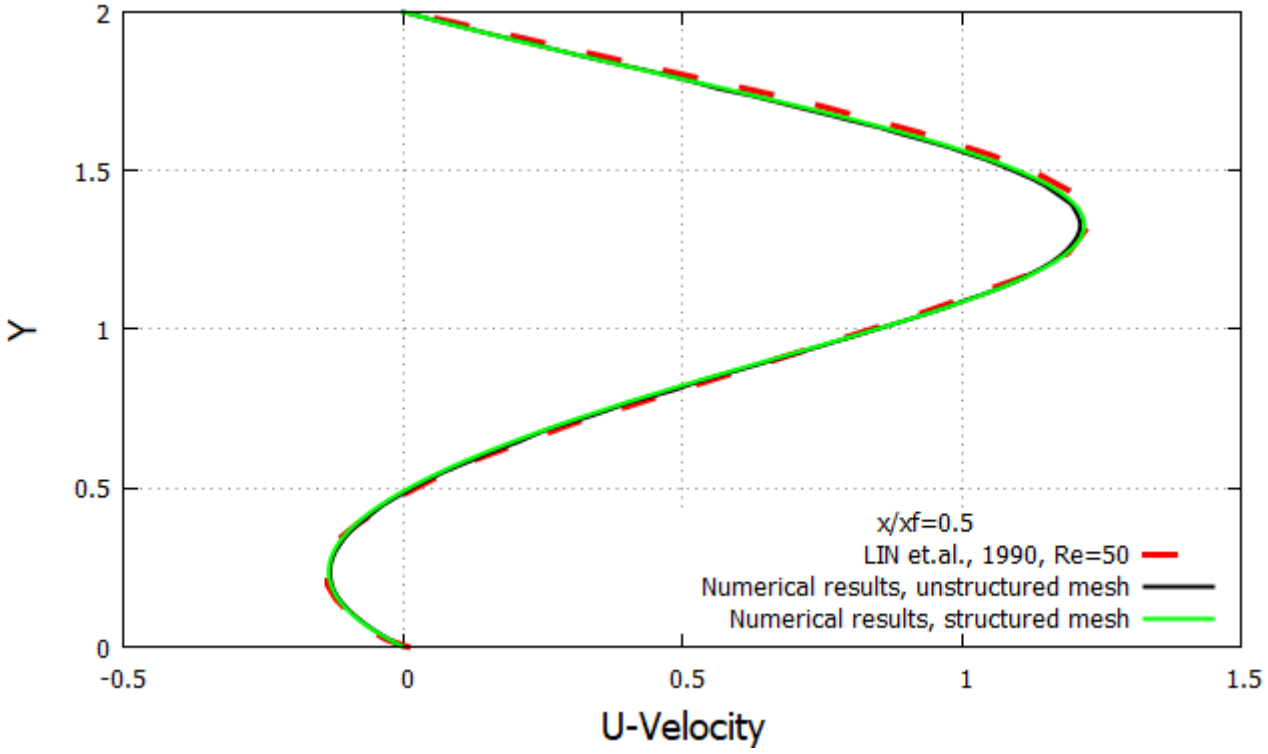


Рисунок 5.12(а) – Влияние плавучести на распределение скорости для Re=50 при $\Delta T = 1°C$ в поперечных сечениях $x/x_f = 0.5$

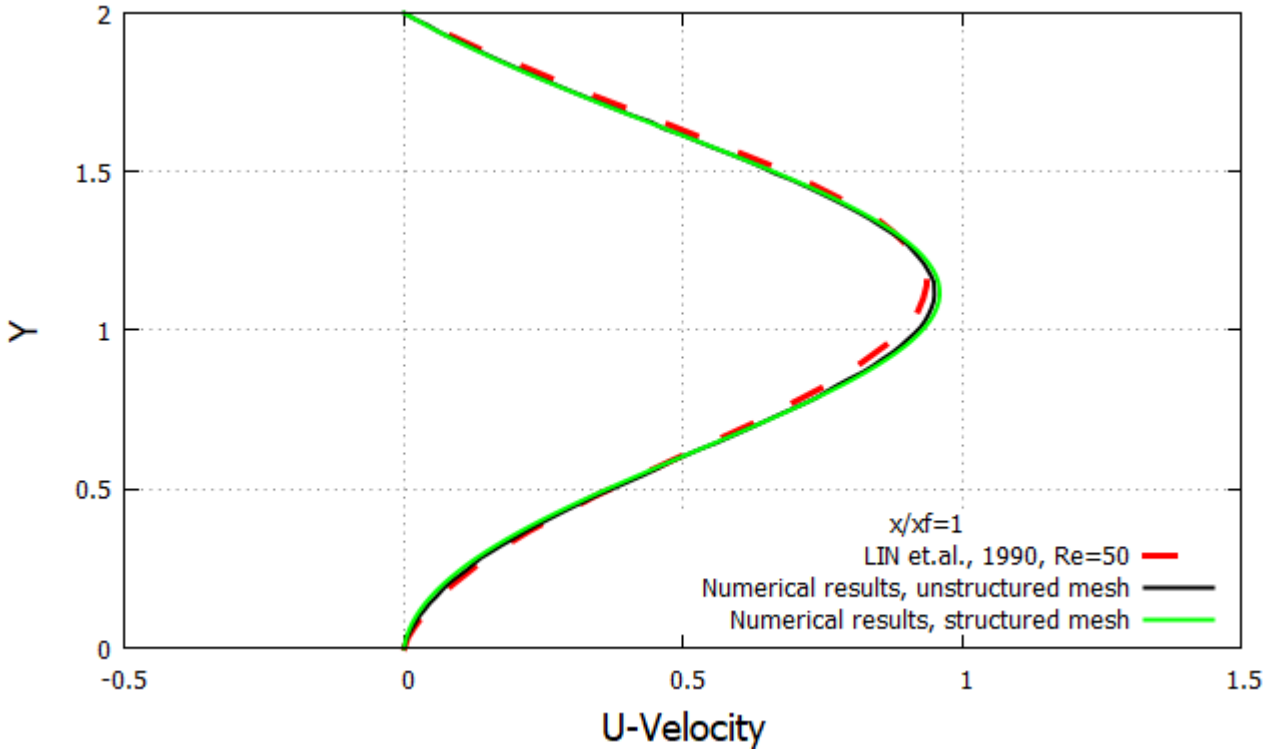


Рисунок 5.12(б) – Влияние плавучести на распределение скорости для Re=50 при $\Delta T = 1°C$ в поперечных сечениях $x/x_f = 1$

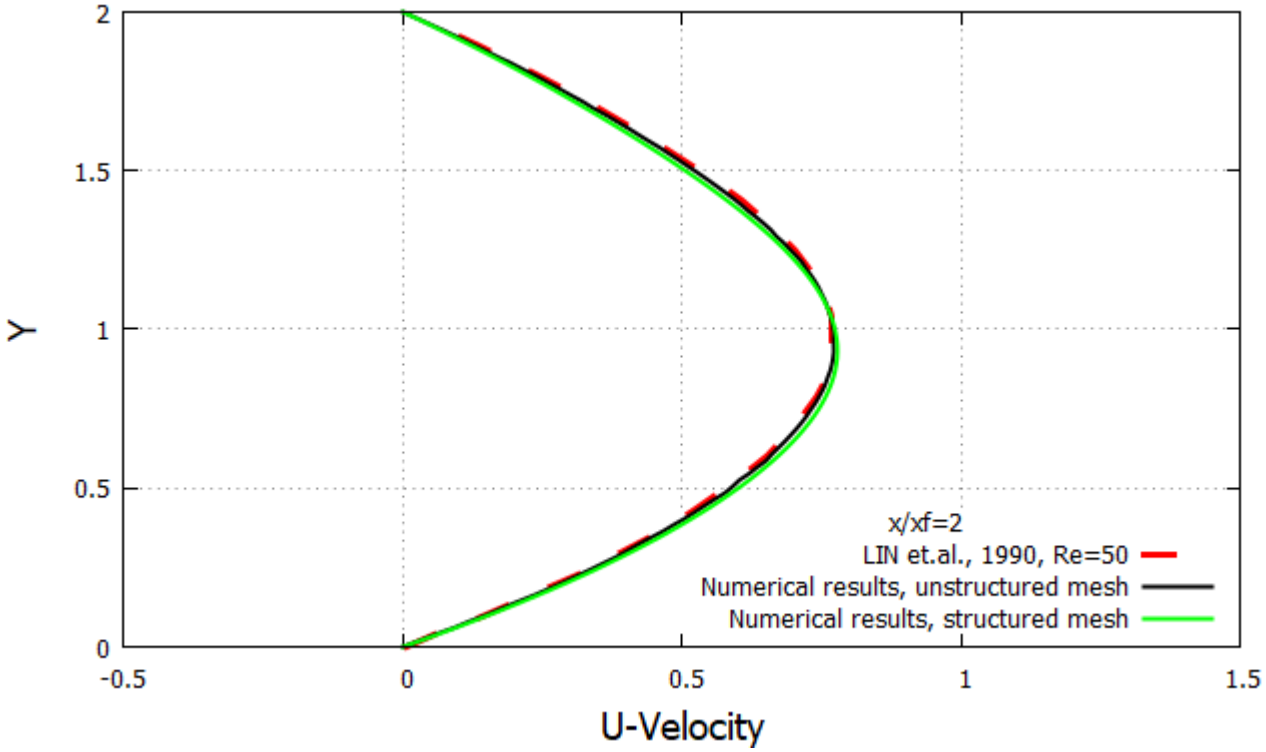


Рисунок 5.12(с) – Влияние плавучести на распределение скорости для Re=50 при $\Delta T = 1°C$ в поперечных сечениях $x/x_f = 2$

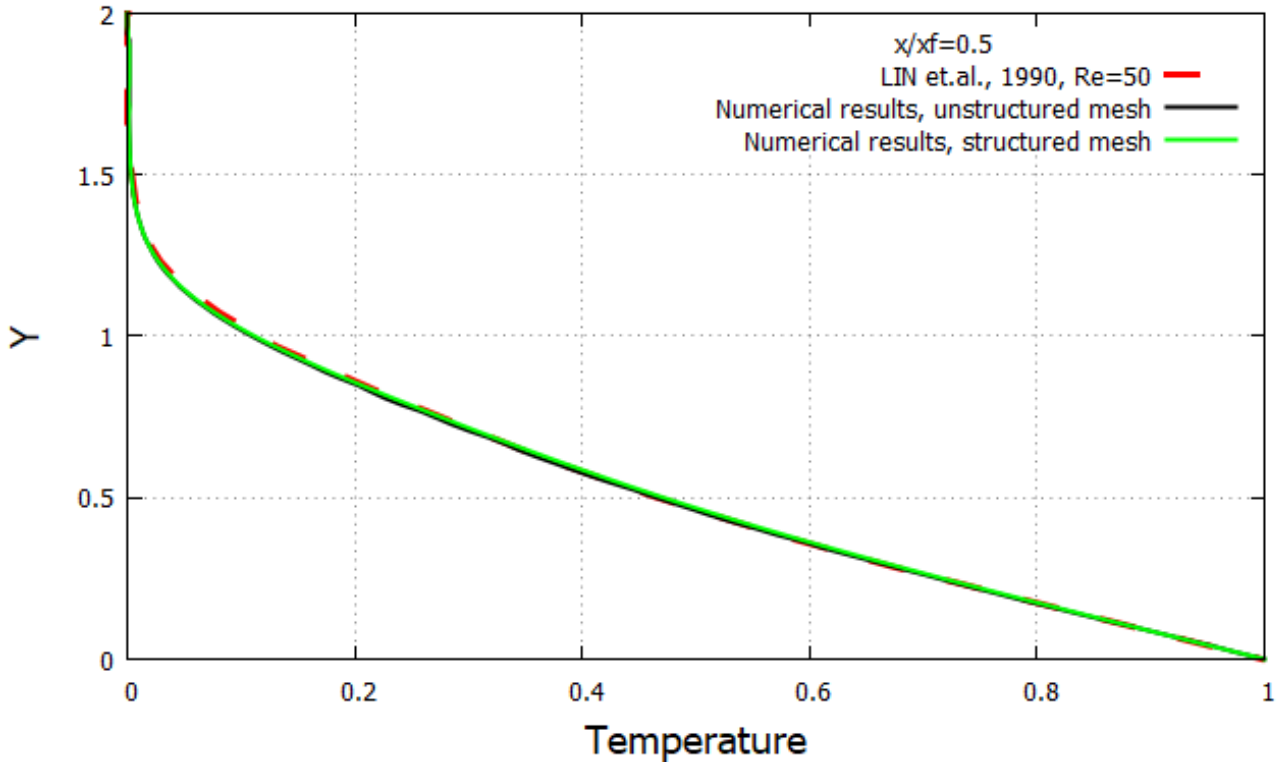


Рисунок 5.13(а) – Влияние плавучести на распределение температуры для Re=50 при $\Delta T = 1°C$ в поперечных сечениях $x / x_f = 0.5$

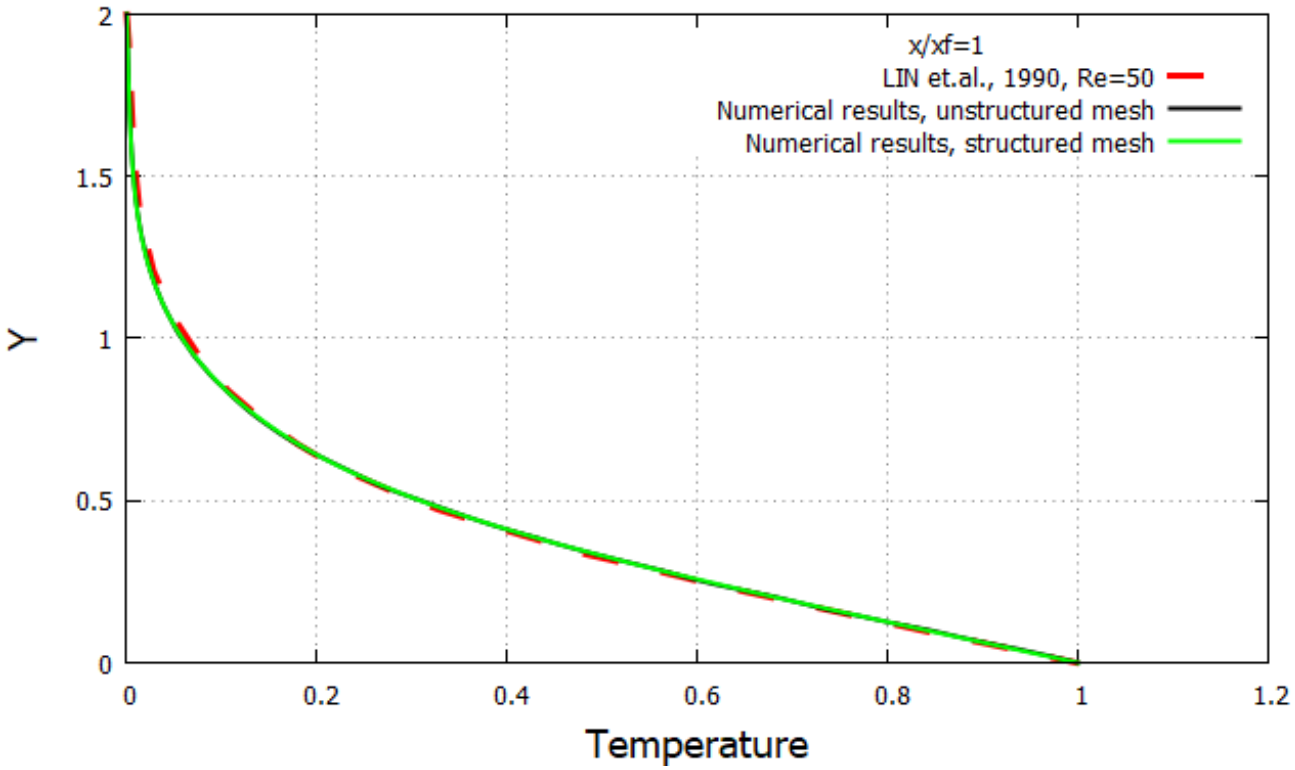


Рисунок 5.13(б) – Влияние плавучести на распределение температуры для Re=50 при $\Delta T = 1°C$ в поперечных сечениях $x / x_f = 1$

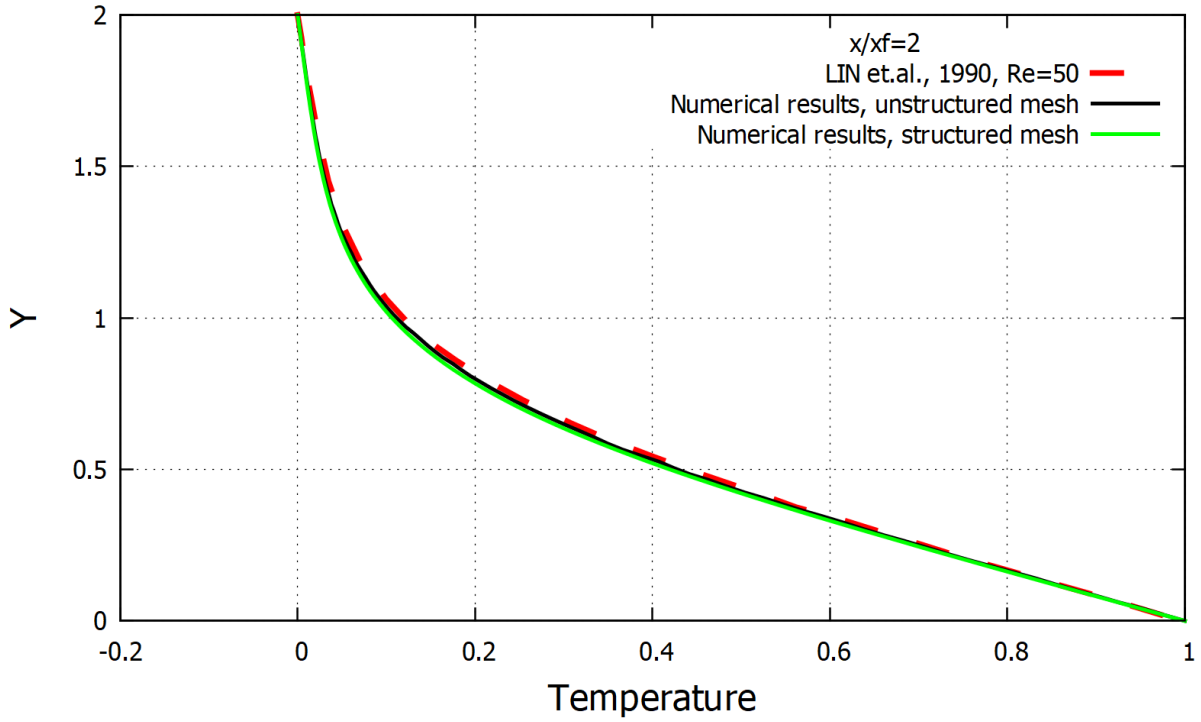


Рисунок 5.13(c) – Влияние плавучести на распределение температуры для Re=50 при $\Delta T = 1°C$ в поперечных сечениях $x / x_f = 2$

В таблице 5.1 показано время вычислений на CPU и GPU, сравнение времени позволило провести анализ производительности, отражающий значительное увеличение скорости, вычисления с помощью GPU.

Таблица 5.1 – Время вычислений

	Размер сетки	Время выполнения (мсек)	Speedup
CPU	1423x41	193985,145	
GPU (без shared memory)		4001,780	48x
GPU (с shared memory)		2971,760	65x

Сравнение вычислительной времени показывает преимущество технологии GPU с использованием разделяемой памяти в решении инженерных задач, требующих интенсивных численных вычислений. Анализ числа потоков в блоке, возможно, наиболее важного параметра распараллеливания в CUDA, показывает наличие оптимального значения. Этот результат является простым, но мощным методом оптимизации CUDA, который значительно влияет на общее время обработки.

Заключение по 5 разделу

В данном разделе показан результат использования задач несжимаемого вязкого течения за обратным уступом на структурированных и неструктурированных сетках, на основе применения распараллеливания в технологии CUDA с использованием схемы заполнения пространства кривой Гильберта (SFC). Результаты данного исследования показали совпадения на структурированных и неструктурированных сетках, который применялся для неструктурированной сетки впервые.

ЗАКЛЮЧЕНИЕ

В работе разработана эффективная раскладка памяти и коммуникационные шаблоны распараллеливания для неструктурированных вычислительных сеток на графических процессорах общего назначения (GPU), чтобы повысить эффективность производительности массивно-параллельных вычислений. Это подход применяется для различных ресурсоемких физических задач при использовании неструктурированной вычислительной сетки с помощью заполнения пространства кривой Гильберта (SFC).

Выводы по результатам диссертационного исследования:

– проведено численное исследование эффективности высокопроизводительных вычислений на графических процессорах общего назначения для уравнения Пуассона.

– проведено численное исследование эффективности высокопроизводительных вычислений на графических процессорах общего назначения для циркуляционного несжимаемого вязкого течения в каверне.

– проведено численное исследование эффективности высокопроизводительных вычислений на графических процессорах общего назначения для задач несжимаемого вязкого течения за обратным уступом при использовании структурированных вычислительных сеток.

– проведено численное исследование эффективности высокопроизводительных вычислений на графических процессорах общего назначения для задач несжимаемого вязкого течения за обратным уступом с помощью заполнения пространства кривой Гильберта (SFC) при использовании неструктурированных вычислительных сеток.

– проведено сравнение полученных результатов моделирования с численными данными, полученными с помощью структурированных и неструктурированных вычислительных сеток.

– проведено сравнение полученных результатов моделирования с численными данными и экспериментальные данными других авторов;

– для структурированной вычислительной сетки и параллельного численного вычисления для неструктурированной вычислительной сетки с использованием схемы заполнения пространства кривой Гильберта (SFC) проведен анализ полученных результатов параллельного численного вычисления и получено ускорения в 65х раза.

Оценка полноты решения поставленных задач. В работе показана эффективность использования схемы высокопроизводительного вычисления на графических процессорах общего назначения со сложными связанными симуляциями с нетривиальными разложениями в области с помощью заполнения пространства кривой Гильберта (SFC) при использовании неструктурированных вычислительных сеток.

Разработка рекомендаций и исходных данных для конкретного использования результатов. Полученные в данной работе результаты рекомендуется использовать для реализации проектов, связанных с решением задач в области информационных технологий.

Получено авторское свидетельство: Авторское свидетельство о внесении сведений в государственный реестр прав на объекты, охраняемые авторским правом РК, № 25762 от 4 мая 2022г., Разработка эффективного высокопроизводительного вычисления для задач несжимаемого вязкого течения за обратным уступом.

Сравнивали с лучшими достижениями в изучаемой области для оценки научного уровня выполненной работы. Диссертационная работа содержит новые научно - обоснованные результаты, решающие важную научную задачу, а именно использование схемы высокопроизводительного вычисления на графических процессорах общего назначения (GPU) со сложными связанными симуляциями с нетривиальными разложениями в области с помощью заполнения пространства кривой Гильберта (SFC) при использовании неструктурированных вычислительных сеток.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Vittal Shenoy D and Safdari Shadloo M, Peixinho J, Hadjadj A. Direct numerical simulations of laminar and transitional flows in diverging pipes // Int J Numer Methods Heat Fluid Flow. – 2019. –Vol. 30. – P. 75–92.

2 Ebrahimpour M., Shafagha, R., and Alamian R., Safdari Shadloo M. Numerical Investigation of the Savonius Vertical Axis Wind Turbine and Evaluation of the Effect of the Overlap Parameter in Both Horizontal and Vertical Directions on Its Performance. // Symmetry. – 2019. – Vol. 11, Issue 6. – P. 821.

3 Goodarzi M., Safaei M. R., Karimipour A., Hooman K., Dahari M., Kazi S. N. and Sadeghinezhad E. Comparison of the Finite Volume and Lattice Boltzmann Methods for Solving Natural Convection Heat Transfer Problems inside Cavities and Enclosures // Abstract and Applied Analysis. – 2014. – Vol. 2014. – P. 1–15.

4 Piquet A., Zebiri B., Hadjadj A., and Safdari Shadloo M. A parallel high-order compressible flows solver with domain decomposition method in the generalized curvilinear coordinates system // International Journal of Numerical Methods for Heat & Fluid Flow. – 2019.

5 Tai C. H., Zhao Y., and Liew K. M. Parallel computation of unsteady three-dimensional incompressible viscous flow using an unstructured multigrid method // Computers & Structures, – 2004. – Vol. 82, №28. – P. 2425-2436.

6 Lee B.-K. Computational Fluid Dynamics in Cardiovascular Disease // Korean Circulation Journal. – 2011. – Vol. 41, №8. – P. 423.

7 Stopford P. J. Recent applications of CFD modelling in the power generation and combustion industries // Applied Mathematical Modelling. – 2002. – Vol. 26, №2. – P. 351-374.

8 https://www.top500.org/lists/top500/2021/06/

9 González-Álvarez D. L., Vega-Rodríguez M. A., and Rubio-Largo Á. Searching for common patterns on protein sequences by means of a parallel hybrid honey-bee mating optimization algorithm // Parallel Computing. – 2018. – Vol.76. – P. 1–17.

10 Afzal A., Saleel C. A., Prashantha K., Bhattacharyya S., & Sadhikh M. Parallel finite volume method-based fluid flow computations using OpenMP and CUDA applying different schemes // Journal of Thermal Analysis and Calorimetry. – 2021. – Vol. 145, Issue 4. – P. 1891–1909.

11 Song P., Zhang, Z. Zhang, Q., Liang L., & Zhao, Q. Implementation of the CPU/GPU hybrid parallel method of characteristics neutron transport calculation using the heterogeneous cluster with dynamic workload assignment // Annals of Nuclear Energy. – 2020. – Vol. 135. – 106957.

12 Altybay A., Ruzhansky M. and Tokmagambetov N. A parallel hybrid implementation of the 2D acoustic wave equation // International Journal of Nonlinear Sciences and Numerical Simulation. – 2020. – Vol. 21, Issue 7-8. – P. 821–827.

13 Gimenez J., Mercadal E., Llort G. and Mendez S. Analyzing the Efficiency of Hybrid Codes // 2020 19th International Symposium on Parallel and Distributed Computing (ISPDC). – 2020.

14 Polyakov S. V., Podryga V. O. and Puzyrkov D. V. High Performance Computing in Multiscale Problems of Gas Dynamics // Lobachevskii Journal of Mathematics. – 2018. – Vol. 39, Issue 9. – P. 1239–1250.

15 Mathews M. and Abraham J. P. Automatic Code Parallelization with OpenMP task constructs // 2016 International Conference on Information Science (ICIS). – 2016.

16 Official home page – OpenMP [online]. Available: www.openmp.org

17 Ouro P., Fraga, B., Lopez-Novoa U., and Stoesser T. Scalability of an Eulerian-Lagrangian large-eddy simulation solver with hybrid MPI/OpenMP parallelization // Computers & Fluids. – 2018.

18 Hückelheim J. C., Hovland P. D., Strout M. M. and Müller J.-D. Parallelizable adjoint stencil computations using transposed forward-mode algorithmic differentiation // Optimization Methods and Software. – 2018. – Vol. 33, Issue 4-6. – P. 672-693.

19 Shan P., Zhu R., Wang F. and Wu J. Efficient approximation of free-surface Green function and OpenMP parallelization in frequency-domain wave–body interactions // Journal of Marine Science and Technology. – 2018.

20 "Message passing interface forum." [Online]. Available: https: //www.mpi-forum.org/

21 Afzal A., Ansari Z., Faizabadi A. R. and Ramis M. K. Parallelization Strategies for Computational Fluid Dynamics Software: State of the Art Review // Archives of Computational Methods in Engineering. – 2016. –Vol. 24, Issue 2. – P. 337–363.

22 Amiranashvili S., Radziunas M., Bandelow U. and Čiegis R. Numerical methods for accurate description of ultrashort pulses in optical fibers // Communications in Nonlinear Science and Numerical Simulation. – 2019. – Vol. 67. – P. 391–402.

23 Cabral F. L., Gonzaga de Oliveira S. L., Osthoff C., Costa G. P., Brandão D. N. and Kischinhevsky M. An evaluation of MPI and OpenMP paradigms in finite-difference explicit methods for PDEs on shared-memory multi- and manycore systems // Concurrency and Computation: Practice and Experience. – 2019. – Vol. 32.

24 Wu X., Petiton S. G. and Lu Y. A parallel generator of non-Hermitian matrices computed from given spectra // Concurrency and Computation: Practice and Experience. – 2020.

25 Marques Garcia A., Schepke C. and Girardi A. PAMPAR: A new parallel benchmark for performance and energy consumption evaluation // Concurrency and Computation: Practice and Experience. – 2019.

26 Gaioso R., Gil-Costa V., Guardia H. and Senger H. Performance evaluation of single vs. batch of queries on GPUs // Concurrency and Computation: Practice and Experience. – 2019. – Vol. 32, Issue 20.

27 Owens J.D., Luebke D., Govindaraju N., Harris M., Krüger J., Lefohn A.E., Purcell T.J. A survey of general-purpose computation on graphics hardware, Compu // Graph. Forum. – 2007. – Vol. 26, Issue 1. – P. 80–113.

28 Rostami S. R. M. and Ghaffari-Miab M. Finite Difference Generated Transient Potentials of Open-Layered Media by Parallel Computing Using OpenMP, MPI, OpenACC, and CUDA // IEEE Transactions on Antennas and Propagation. – 2019. – Vol. 67. – P. 6541-6550.

29 Crespo A. C., Dominguez J. M., Barreiro A., Gómez-Gesteira M. and Rogers B. D. GPUs, a New Tool of Acceleration in CFD: Efficiency and Reliability on Smoothed Particle Hydrodynamics Methods // PLoS ONE. – 2011. – Vol. 6, Issue 6.

30 Alawneh Shadi, Thijssen William and Richard Martin. Accelerating Numerical Ice Engineering Tools Using GPGPU // Offshore Technology Conference, OTC 27386. – 2016. – October 24-28.

31 Liu X., Zhong Z. and Xu K. A hybrid solution method for CFD applications on GPU-accelerated hybrid HPC platforms // Future Generation Computer Systems. – 2016. – Vol. 56. – P. 759–765.

32 Tutkun B. and Edis F. O. An implementation of the direct-forcing immersed boundary method using GPU power // Engineering Applications of Computational Fluid Mechanics. – 2016. – Vol. 11, Issue 1. – P. 15–29.

33 Mintu S. A., Molyneux D. Application of GPGPU to Accelerate CFD Simulation // CFD and FSI. – 2018. –Vol. 2.

34 Mohammadi S., Karami H., Azadifar M. and Rachidi F. On the Efficiency of OpenACC-aided GPU-Based FDTD Approach: Application to Lightning Electromagnetic Fields // Applied Sciences. – 2020. – Vol. 10, Issue 7. – P. 2359.

35 Rodrigues M. J., Fernandes D. E., Silveirinha M. G. and Falcão G. Simulating electron wave dynamics in graphene superlattices exploiting parallel processing advantages // Computer Physics Communications. – 2018. – Vol. 222. – P. 240–249.

36 Xiong L., Wang X., Liu S., Peng Z. and Zhong S. The electromagnetic waves propagation in unmagnetized plasma media using parallelized finite-difference time-domain method // Optik. – 2018. – Vol. 166. – P. 8–14.

37 Ayyad M., Guaily A. and Hassanein M. A. Stabilized variational formulation of an oldroyd-B fluid flow equations on a Graphic Processing Unit (GPU) architecture // Computer Physics Communications. – 2020. – Vol. 258. – P. 1-7.

38 Zhang W., Zhong Z., Peng C., Yuan W. and Wu W. GPU-accelerated smoothed particle finite element method for large deformation analysis in geomechanics // Computers and Geotechnics. – 2021. – Vol. 129. – P. 103856.

39 Laqua H; Kussmann J. and Ochsenfeld C. Accelerating seminumerical Fock-exchange calculations using mixed single- and double-precision arithmethic // Journal of chemical physics. – 2021. – Vol. 154. – P. 214116.

40 Kuzmina K. S., Marchevsky I. K. and Ryatina E. P. The VM2D Open Source Code for Incompressible Flow Simulation by Using Meshless Lagrangian

Vortex Methods on CPU and GPU // 2019 Ivannikov Memorial Workshop (IVMEM). – 2019.

41 Yamazaki I., Abdelfattah A., Ida A., Ohshima S., Tomov S., Yokota R. and Dongarra J. Performance of Hierarchical-matrix BiCGStab Solver on GPU Clusters // 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS). – 2018.

42 Yoshikawa T., Komoto N., Nishimura Y., Nakai H. GPU-Accelerated Large-Scale Excited-State Simulation Based on Divide-and-Conquer Time-Dependent Density-Functional Tight-Binding // Journal of Computational Chemistry. – 2019.

43 Xu J., Fu H., Luk W., Gan L., Shi W., Xue W., Yang G., Jiang Y., Yang Ch., He C. Optimizing Finite Volume Method Solvers on Nvidia GPUs // IEEE Transactions on Parallel and Distributed Systems. – 2019. – Vol. 32. – P. 2790-2805.

44 Weng Y., Zhang X., Guo X., Zhang X., Lu Y., Liu Y. Effects of mesh loop modes on performance of unstructured finite volume GPU simulations // Advances in Aerodynamics. – 2021. – Vol. 21.

45 Сакыпбекова М.Ж. Эллиптикалық типтегі теңдеуді сандық және компьютерлік шешу // ҚазҰТЗУ хабаршысы. – 2017. – Т. 3, №121. – Б. 455-458.

46 Исахов А.А., Сакыпбекова М.Ж. Құрылымды емес торды қолдануда есептеу гидродинамикасындағы параллельді технологиялардың теориялық негізі // «Көліктегі инновациялық технологиялар: білім, ғылым, тәжиірбе» атты XLI Халықаралық ғылыми-практикалық конференцияның материалдары. – 2017. – Т. 1. – 3-4 сәуір. – С. 100-102.

47 Исахов А.А. Математическое и компьютерное моделирование физических процессов: учебник. – Алматы: Қазақ университеті, 2018. – 324 с.

48 Zolfaghari H., Obrist D. A high-throughput hybrid task and data parallel Poisson solver for large-scale simulations of incompressible turbulent flows on distributed GPUs // Journal of Computational Physics. – 2021. – Vol. 437. – P. 110329.

49 Сакыпбекова М.Ж. Разработка эффективного высокопроизводительного вычислении для решения уравнения Пуассона // Вестник ЕНУ, Серия техниические науки и технологии. – 2022. – Т. 139, №2. – С. 24-29.

50 Kamel A. G., Haraz E. H., Hanna S. N. Numerical simulation of three-sided lid-driven square cavity // Engineering Reports. – 2020. –Vol. 2, Issue 4.

51 Shobha A., Lakshmi C. V., Venkatadri K., Prasad V. R. Comparative numerical simulation of lid-driven cavity flow problem with pressure term handling methods // International conference on mathematical sciences and applications (ICMSA-2019). – 2020.

52 Moon H., Hong S. and You D. Application of the parallel diagonal dominant algorithm for the incompressible Navier-Stokes equations // Journal of Computational Physics. – 2020. – Vol. 423.

53 He W., Qin G., Wang Y. and Bao Z. A segregated spectral element method for the 2D transient incompressible Navier-Stokes equations // Computers & Fluids. – 2020. – Vol. 216. – 104643.

54 Shi X., Agrawal T., Lin C.-A., Hwang F.-N. and Chiu T.-H. A parallel nonlinear multigrid solver for unsteady incompressible flow simulation on multi-GPU cluster // Journal of Computational Physics. – 2020. – Vol. 414. – 109447.

55 Сакыпбекова М.Ж. Екі өлшемді ағынды модельдеуде гидродинамиканың негізгі теңдеуін сандық шешу // ҚазҰТЗУ Хабаршысы. – 2018. – №3. – С. 482-486.

56 Shui Q. Penalty and characteristic-based operator splitting with multistep scheme finite element method for unsteady incompressible viscous flows // Progress in Computational Fluid Dynamics, an International Journal. – 2020. – Vol. 20, №3. – P. 125-142.

57 Zhan N., Chen R. and You Y. Meshfree method based on discrete gas-kinetic scheme to simulate incompressible/compressible flows // Physics of Fluids. – 2021. – Vol. 33, №1. – 017112.

58 Zamolo R., Nobile E. Numerical analysis of advection-diffusion problems on 2D general-shaped domains by means of a RBF Collocation Meshless Method // Journal of Physics: Conference Series. – 2019. – Vol. 1224.

59 Ghia U., Ghia K.N., Shin C.T. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method // J. Comput. Phys. – 1982. – Vol. 48. – P. 387–411.

60 Huang T., Lim H.-C. Simulation of Lid-Driven Cavity Flow with Internal Circular Obstacles // Applied Sciences. – 2020. – Vol. 10, №13. – P. 4583.

61 Armaly B.F., Durst F., Pereira J.C.F., Schounung B. Experimental and theoretical investigation of backward facing step flow // J. Fluid Mech. – 1983. – Vol. 127. – P. 473-496.

62 Erturk E. Numerical solutions of 2-D steady incompressible flow over a backward-facing step, Part I: high Reynolds number solutions // Comput. Fluid. – 2008. – Vol. 37. – P. 633-655.

63 Biswas G., Breuer M., Durst F. Backward-facing step flows for various expansion ratios at low and moderate Reynolds numbers // Trans. ASME J. Fluids Eng. – 2004. – Vol. 126. – P. 362–374.

64 Terhaar S., Velazquez A., Arias J.R., Sanchez-Sanz M. Experimental study on the unsteady laminar heat transfer downstream of a backwards facing step // Int. Commun. Mass Transf. – 2010. – Vol. 37. – P. 457–462.

65 Ghasemi J., Razavi S.E. On the finite-volume Lattice Boltzmann modeling of thermo-hydrodynamics // Comput. Math. Appl. – 2010. – Vol. 60. – P. 1135–1144.

66 Issakhov A., Zhandaulet Y., Abylkassyomova A., Sakypbekova M. and Issakhov A. Mixed convection in a channel with buoyancy force over backward and forward facing steps: The effects of inclination and geometry // Case Studies in Thermal Engineering. – 2021. – Vol. 26. 101152.

67 Aung W. An experimental study of laminar heat transfers downstream of backsteps // J. Heat Tran. – 1983. – Vol. 105. – P. 823–829.

68 Sparrow E.M., Kang S.S., Chuck W. Relation between the points of flow reattachment and maximum heat transfer for regions of flow separation // Int. J. Heat Mass Tran. – 1987. – Vol. 30. – P. 1237–1246.

69 Sparrow E.M., Chuck W. PC solutions for transfer and fluid flow downstream of an abrupt, asymmetric enlargement in a channel // Numerical heat transfer. – 1987. – Vol. 12. – P. 19-40.

70 Biswas G., Breuer M., Durst F. Backward-Facing Step Flows for Various Expansion Ratios at Low and Moderate Reynolds Numbers // Journal of Fluids Engineering. – 2004. – Vol. 126, №3. – P. 362-374.

71 Issakhov A., Abylkassyomova A., Sakypbekova M. Applications of parallel computing technologies for modeling the mixed convection in backward-facing step flows with the vertical buoyancy forces // International Journal of Mathematics and Physics. – 2017. – Vol. 8. – №2. – P.43-50.

72 Chen Y. T., Nie J. H., Armaly B. F. and Hsieh H. T. Turbulent separated convection flow adjacent to backward-facing step—effects of step height // International Journal of Heat and Mass Transfer. – 2006. – Vol. 49, №19-20 – P. 3670-3680.

73 Issakhov A., Abylkassyomova A., Sakypbekova M. Applications of parallel computing technologies for modeling the flow separation process behind the backward facing step channel with the buoyancy forces // Communications in Computer and Information Science. – 2019. – Vol. 998. – P. 97-113.

74 Issakhov A., Abylkassyomova A., Sakypbekova M. Applications of parallel computing technologies for modeling of the wind flow around the architectural obstacles with the vertical buoyancy forces // News of the National academy of sciences of the Republic of Kazakhstan-series physico-mathematical. – 2018. – Vol. 4, № 320. – P. 48-57.

75 Chorin A.J. Numerical solution of the Navier-Stokes equations // Math. Comp. –1968. – Vol. 22. - P. 745-762.

76 Oztop H. F., Abu-Nada, E. Numerical study of natural convection in partially heated rectangular enclosures filled with nanofluids // International Journal of Heat and Fluid Flow. – 2008. – Vol. 29, №5. – P. 1326-336.

77 Lin J. T., Armaly B. F. and Chen T. S. // International Journal of Heat and Mass Transfer. – 1990. – Vol. 33, №10. – P. 2121–2132.

78 Sastry S. P., Kultursay E., Shontz S. M. and Kandemir M. T. Improved cache utilization and preconditioner efficiency through use of a space-filling curve mesh element- and vertex-reordering technique // Engineering with Computers. – 2014. – Vol. 30, №4. – P. 535–547.

79 Arce Acuña M., Aoki T. Tree-based mesh-refinement GPU-accelerated tsunami simulator for real-time operation // Natural Hazards and Earth System Sciences. – 2018. – Vol. 18, №9. – P. 2561–2602.

80 Turchetto M., Dal Palu A., Vacondio R. A general design for a scalable MPI-GPU multi-resolution 2D numerical solver // IEEE Transactions on Parallel and Distributed Systems. – 2019. – Vol 31.

81 Al-Kharusi I., Walker D. W. Locality properties of 3D data orderings with application to parallel molecular dynamics simulations // The International Journal of High Performance Computing Applications. – 2019. – Vol. 33.

82 Zhou Y., Zhu Q., Zhang Y. GIS spatial data partitioning method for distributed data processing // MIPPR 2007: Remote Sensing and GIS Data Processing and Applications; and Innovative Multispectral Technology and Applications. – 2007.

83 Wang S., Wang W., Che Y. GPU Acceleration of a High-Order CFD Program // Proceedings of the 2020 4th International Conference on High Performance Compilation, Computing and Communications. – 2020. – P. 123-128.

84 Исахов А.А., Абылкасымова А.Б., Сакыпбекова М.Ж. Применение параллельных вычислительных технологий для моделирования процесса отрыва течения за обратным уступом в канале с учетом сил плавучести// Вестник КазНУ имени аль-Фараби. Серия математика, механика, информатика. – 2018. – Т.97, № 1 – С.143-158.

ПРИЛОЖЕНИЕ А – Авторское свидетельство

Авторское свидетельство о внесении сведений в государственный реестр прав на объекты, охраняемые авторским правом РК, № 25762 от 4 мая 2022г., Разработка эффективного высокопроизводительного вычисления для задач несжимаемого вязкого течения за обратным уступом.

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ

РЕСПУБЛИКА КАЗАХСТАН

СВИДЕТЕЛЬСТВО

О ВНЕСЕНИИ СВЕДЕНИЙ В ГОСУДАРСТВЕННЫЙ РЕЕСТР ПРАВ НА ОБЪЕКТЫ, ОХРАНЯЕМЫЕ АВТОРСКИМ ПРАВОМ

№ 25762 от «4» мая 2022 года

Фамилия, имя, отчество, (если оно указано в документе, удостоверяющем личность) автора (ов):
САКЫПБЕКОВА МЕРУЕРТ ЖУМАБЕКОВНА, Исахов Алибек Абилашимович

Вид объекта авторского права: программа для ЭВМ

Название объекта: Разработка эффективного высокопроизводительного вычисления для задач несжимаемого вязкого течения за обратным уступом

Дата создания объекта: 03.05.2022

Подписано ЭЦП

А. Жумабекова

Рисунок А.1 – Авторское свидетельство

ПРИЛОЖЕНИЕ Б – Программное обеспечение задачи

```
__global__ void uCalculation(double * input_un, double * output_u, double * input_vn, double * input_Tn, unsigned int N, unsigned int M, double  hx, double hy, double dt) {

	float Re = 75.2;
	float Gr = 15.7;
	int i = blockDim.y * blockIdx.y + threadIdx.y;
	int j = blockDim.x * blockIdx.x + threadIdx.x;

	int ti = threadIdx.y;
	int tj = threadIdx.x;

	int d_ti= ti + RADIUS;
	int d_tj = tj + RADIUS;

	int d_ti_prev = d_ti- 1;
	int d_ti_next = d_ti+ 1;

	int d_tj_prev = d_tj - 1;
	int d_tj_next = d_tj + 1;

	__shared__ double sData_T[BLOCK_SIZE_Y + 2 * RADIUS][BLOCK_SIZE_X + 2 * RADIUS];
	__shared__ double sData_u[BLOCK_SIZE_Y + 2 * RADIUS][BLOCK_SIZE_X + 2 * RADIUS];
	__shared__ double sData_v[BLOCK_SIZE_Y + 2 * RADIUS][BLOCK_SIZE_X + 2 * RADIUS];
	unsigned int index = (i)* N + (j);


	if (ti<RADIUS)
	{
		if (blockIdx.y > 0) {
			sData_u[ti][d_tj] = input_un[index - RADIUS * N];
			sData_v[ti][d_tj] = input_vn[index - RADIUS * N];
			sData_T[ti][d_tj] = input_Tn[index - RADIUS * N];
		}

		if (blockIdx.y < (gridDim.y - 1)) {
			sData_u[d_ti+BLOCK_SIZE_Y][d_tj] = input_un[index + BLOCK_SIZE_Y * N];
			sData_v[d_ti+BLOCK_SIZE_Y][d_tj] = input_vn[index + BLOCK_SIZE_Y * N];
```

```
                sData_T[d_ti+ BLOCK_SIZE_Y][d_tj] = input_Tn[index + BLOCK_SIZE_Y * N];
            }
        }

        if (tj<RADIUS)
        {
            if (blockIdx.x > 0) {
                sData_u[d_ti][tj] = input_un[index - RADIUS];
                sData_v[d_ti][tj] = input_vn[index - RADIUS];
                sData_T[d_ti][tj] = input_Tn[index - RADIUS];
            }

            if (blockIdx.x < (gridDim.x - 1)) {
                sData_u[d_ti][d_tj + BLOCK_SIZE_X] = input_un[index + BLOCK_SIZE_X];
                sData_v[d_ti][d_tj + BLOCK_SIZE_X] = input_vn[index + BLOCK_SIZE_X];
                sData_T[d_ti][d_tj + BLOCK_SIZE_X] = input_Tn[index + BLOCK_SIZE_X];
            }
        }

        sData_u[d_ti][d_tj] = input_un[index];
        sData_v[d_ti][d_tj] = input_vn[index];
        sData_T[d_ti][d_tj] = input_Tn[index];
        __syncthreads();

        if (i >= 0 && i < 95 && j >= 21 && j < N) {
            output_u[i * N + j] = 0.0;
        }
        if (i >= 0 && i<M && j >= 0 && j<N) {
            if (i >= 0 && i < 95) {
                output_u[i*N + 20] = 0.0;
            }
            if (i >= 95 && i < M) {
                output_u[i*N + N - 1] = 0.0;
            }
            output_u[i*N + 0] = 0.0;// output_u[i*N + (N - 2)];
            if (j >= 0 && j < 21) {
                output_u[0 * N + j] = 0.0;// 6.0*(1.0 - j*hy)*(j*hy - 0.0);
            }
            if (j >= 21 && j < N) {
                output_u[95 * N + j] = 0.0;
```

```
		}
		output_u[((M - 1) * N) + j] = output_u[((M - 2) * N) + j];
	}

	if (i > 0 && j > 0 && i < (M - 1) && j < (N - 1)) {
		output_u[index] = sData_u[d_ti][d_tj] + dt * (-(sData_u[d_ti][d_tj] *
sData_u[d_ti][d_tj] - sData_u[d_ti][d_tj_prev] * sData_u[d_ti][d_tj_prev]) / hx
			- (sData_v[d_ti][d_tj] * sData_u[d_ti][d_tj] -
sData_v[d_ti_prev][d_tj] * sData_u[d_ti_prev][d_tj]) / hy
			+ 1.0f / Re * ((sData_u[d_ti][d_tj_next] - 2.0f *
sData_u[d_ti][d_tj] + sData_u[d_ti][d_tj_prev]) / pow(hx, 2)
				+ (sData_u[d_ti_next][d_tj] - 2.0f * sData_u[d_ti][d_tj] +
sData_u[d_ti_prev][d_tj]) / pow(hy, 2)));// -(Gr / (Re*Re)) *sData_T[d_ti][d_tj]);

	}

	if (i >= 0 && i<M && j >= 0 && j<N) {
		if (i >= 0 && i < 95) {
			output_u[i*N + 20] = 0.0;
		}
		if (i >= 95 && i < M) {
			output_u[i*N + N - 1] = 0.0;
		}
		output_u[i*N + 0] = 0.0;// output_u[i*N + (N - 2)];
		if (j >= 0 && j < 21) {
			output_u[0 * N + j] = 0.0;// 6.0*(1.0 - j*hy)*(j*hy - 0.0);
		}
		if (j >= 21 && j < N) {
			output_u[95 * N + j] = 0.0;
		}
		output_u[((M - 1) * N) + j] = output_u[((M - 2) * N) + j];
	}
}
```