ГОУ ВПО Российско-Армянский (Славянский) университет

Утверждено
Директор Института математики и информатики
Арамян Р.Г.
«21» марта 2025, протокол №9.1

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ

Наименование дисциплины: Основы ускоренных вычислений с использованием CUDA C/C++

Автор: **Акопян Гагик Дживанович**

Направление подготовки: 01.04.02 Прикладная математика и информатика
Наименование образовательной программы: «Системное программирование»

1. АННОТАЦИЯ

1.1. Краткое описание содержания данной дисциплины;

Этот курс предоставляет студентам понимание основ и продвинутых концепций параллельного программирования. Курс охватывает теоретические аспекты, практические техники и инструменты для разработки параллельных приложений. Студенты научатся эффективно использовать многопоточность, распределенные системы и различные параллельные модели для решения вычислительных задач.

1.2. Трудоемкость в академических кредитах и часах, формы итогового контроля (экзамен/зачет);

2-ой семестр - 6 ЗЕТ- экзамен,

1.3. Взаимосвязь дисциплины с другими дисциплинами учебного плана специальности (направления)

Для прохождения этого курса изучение других дисциплин не требуются.

1.4. Результаты освоения программы дисциплины:

Код компетенции	**Наименование компетенции**	**Код индикатора достижения компетенций**	**Наименование индикатора достижений компетенций**
УК-3	Способен организовывать и руководить работой команды, вырабатывая командную стратегию для достижения поставленной цели	**УК-3.1**	Разрабатывает командную стратегию для достижения поставленной цели
		УК-3.2	Умеет организовывать и руководить работой команды
		УК-3.3	Демонстрирует понимание результатов работы команды и личных действий в ней
ПК-7	способностью разрабатывать и оптимизировать бизнес-планы научно-прикладных проектов	**ПК-7.1**	Знает основные критерии эффективности и качества функционирования системы
		ПК-7.2	Умеет руководить выполнением коллективной деятельностью
		ПК-7.3	Владеет методами постановки задачи,

			проведением эксперимента работоспособности системы

2. УЧЕБНАЯ ПРОГРАММА

2.1. Цели и задачи дисциплины

- Изучить основы параллельного программирования и архитектуры.
- Понять принципы и модели параллельного выполнения.
- Овладеть методами разработки параллельных алгоритмов и программ.
- Научиться использовать современные инструменты и библиотеки для параллельного программирования.
- Развить навыки анализа производительности и оптимизации параллельных приложений.

2.2. Трудоемкость дисциплины и виды учебной работы (в академических часах и зачетных единицах)

Виды учебной работы	**Всего, в акад. часах**	**Распределение по семестрам**					
		I сем	**II сем**	**___ сем**	**___ сем.**	**___ сем**	**____ сем.**
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1. Общая трудоемкость изучения дисциплины по семестрам, в т. ч.:	**144**						
1.1. Аудиторные занятия, в т. ч.:	**64**	**64**					
1.1.1. Лекции	**32**	**32**					
1.1.2. Практические занятия, в т. ч.	**32**	**32**					
1.2. Самостоятельная работа, в т. ч.:	**44**	**44**					
1.3. Другие методы и формы занятий	**46**	**46**					
Итоговый контроль (Экзамен, Зачет, диф. зачет - указать)		**Диф.зачет**					

2.3. Содержание дисциплины

2.3.1. Тематический план и трудоемкость аудиторных занятий (модули, разделы дисциплины и виды занятий) по рабочему учебному плану

Разделы и темы дисциплины	**Всего (ак. часов)**	**Лекции (ак. часов)**	**Практ. занятия (ак. часов)**	**Семина-ры (ак. часов)**	**Лабор. (ак. часов)**	**Другие виды занятий (ак. часов)**
1	**2=3+4+5+6**	**3**	**4**	**5**	**6**	**7**

	+7					
Модуль 1.						
Тема 1. Основы CUDA и модель программирования	**4**	**2**	**2**			
Тема 2. Иерархия потоков и модель исполнения	**4**	**2**	**2**			
Тема 3. Выполнение на уровне warp и управление потоком (control flow)	**4**	**2**	**2**			
Тема 4. Типы памяти CUDA и управление ими	**4**	**2**	**2**			
Тема 5. Конфликты памяти и разделяемая память	**4**	**2**	**2**			
Тема 6. Потоки (streams) и события (events)	**4**	**2**	**2**			
Тема 7. Приёмы асинхронного исполнения	**4**	**2**	**2**			
Тема 8. Интринсики уровня warp — редукция	**4**	**2**	**2**			
Тема 9. Обмен данными на уровне warp	**4**	**2**	**2**			
Тема 10. Практические алгоритмы	**4**	**2**	**2**			
Тема 11. Текстуры и поверхности	**6**	**3**	**3**			
Тема 12. API графов CUDA	**6**	**3**	**3**			
Тема 13. Поведение кэша и оптимизация	**6**	**3**	**3**			
Тема 14. Библиотеки CUDA	**6**	**3**	**3**			
ИТОГО	**64**	**28**	**28**			

2.3.2. Краткое содержание разделов дисциплины в виде тематического плана

Тема 1. Основы CUDA и модель программирования

- Введение в CUDA
- Архитектура CUDA
- Программа на C/C++ с использованием CUDA

Тема 2. Архитектура SIMD; потоки, блоки, сетки

- Понятия `thread`, `block`, `grid`
- Индексация и размерности
- Согласование числа потоков с задачей

Тема 3. Warp и управление потоком

- Что такое warp
- Ветвление (`if`, `else`) и его влияние
- Циклы `for`, `while`; развёртка циклов (loop unrolling)

Тема 4. Управление памятью в CUDA

- Типы памяти: paged, pinned, mapped, unified
- Сравнение производительности
- Синтаксис `cudaMalloc`, `cudaHostAlloc`, `cudaMallocManaged`

Тема 5. Конфликты памяти и разделяемая память

- Банки памяти (memory banks) и конфликты
- Использование `__syncthreads()`
- Постоянная (constant) и шаговая (pitched) память

Тема 6. Потоки и события

- Потоки (streams) и параллелизм
- События CUDA (`cudaEvent_t`)
- Перекрытие операций копирования и вычислений

Тема 7. Асинхронное выполнение

- Синхронный и асинхронный запуск

- `cudaMemcpyAsync` и `cudaStreamSynchronize`
- Сравнение производительности: `sync` vs `async`

Тема 8. Программирование на уровне warp: редукция

- Использование `__shfl_down_sync`, `__syncwarp`
- Внутри-warp редукция (in-warp reduction)
- Минимизация глобальной синхронизации

Тема 9. Программирование на уровне warp: обмен данными

- Интринсики `__shfl_sync`, `__ballot_sync`
- Обмен информацией между потоками
- Оптимизация через warp shuffle

Тема 10. Сопоставление дескрипторов и матричное умножение

- Расстояние Хэмминга: побитовые операции
- Сравнение бинарных дескрипторов
- Оптимизированное умножение матриц с использованием тайлинга

Тема 11. Текстуры и поверхности

- Текстурная память (`cudaTextureObject_t`)
- Примеры масштабирования изображения (zoom)
- Работа с поверхностями (surfaces)

Тема 12. Программирование с CUDA Graph

- Запись и воспроизведение CUDA Graph
- `cudaGraphCreate`, `cudaGraphExec`
- Повторное использование графа

Тема 13. Кэш и его поведение

- Кэш L1 и L2

- Указание политики кэширования (`cudaFuncCache`)
- Постоянный кэш (persistent cache)

Тема 14. CUDA библиотеки

- cuRAND — генерация случайных чисел
- cuBLAS — линейная алгебра
- cuFFT — быстрое преобразование Фурье

2.3.3. Краткое содержание семинарских/практических занятий/лабораторного практикума

Решение задач согласно пройденной на лекционном занятии темы.

2.3.4. Материально-техническое обеспечение дисциплины

Компьютеры с интернет-браузером.

2.4. Модульная структура дисциплины с распределением весов по формам контролей

Формы контролей	Вес формы (форм) текущего контроля в результирующей оценке текущего контроля (по модулям)		Вес формы промежуточного контроля в итоговой оценке промежуточного контроля		Вес итоговой оценки промежуточного контроля в результирующей оценке промежуточных контролей		Вес итоговой оценки промежуточного контроля в результирующей оценке промежуточных контролей (семестровой оценке)	Веса результирующей оценки промежуточных контролей и оценки итогового контроля в результирующей оценке итогового контроля
Вид учебной работы/контроля	**M1**	**M2**	**M1**	**M2**	**M1**	**M2**		
Контрольная работа		1		1				

Письменные домашние задания *(при наличии)*								
Веса результирующих оценок текущих контролей в итоговых оценках промежуточных контролей								
Веса оценок промежуточных контролей в итоговых оценках промежуточных контролей					0.5	0.5		
Вес итоговой оценки 1-го промежуточного контроля в результирующей оценке промежуточных контролей							0	
Вес итоговой оценки 2-го промежуточного контроля в результирующей оценке промежуточных контролей							1	
Вес результирующей оценки промежуточных контролей в результирующей оценке итогового контроля								0.5
Вес итогового контроля (Экзамен/зачет) в результирующей оценке итогового контроля								0.5
	∑ = 1	**∑ = 1**	**∑ = 1**	**∑ = 1**	**∑ = 1**	**∑ = 1**	**∑ = 1**	**∑ = 1**

3. Теоретический блок

3.1. Материалы по теоретической части курса

3.1.1. Сандерс Дж., Кэндрот Э. Технология CUDA в примерах: Введение в программирование графических процессоров. – М.: ДМК Пресс, 2013. – 232 с. 3. Cheng J., Grossman M., McKercher T. Professional CUDA C Programming. – Indianapolis: John Wiley & Sons, Inc., 2014. – 528 p

3.1.2. CUDA C++ Programming Guide

3.1.3. Prof. Dr. W. Tichy. Parallel programming with CUDA Architecture, Analysis, Application

4. Фонды оценочных средств.

4.1. Планы практических и семинарских занятий
Основы CUDA: построение и запуск простейшего ядра.
Иерархия потоков и блоков: threadIdx, blockIdx, gridDim.
Работа с warp: расхождение потоков, управление ветвлением.
Глобальная, локальная, разделяемая память: замеры производительности.
Анализ конфликтов в разделяемой памяти и способы их устранения.
Использование cudaStream и cudaEvent для организации асинхронности.
Приёмы перекрытия передачи данных и вычислений.
Реализация редукции с использованием warp intrinsics (__shfl_*).
Эффективный обмен данными между потоками внутри warp.
Реализация параллельных алгоритмов (скан, сортировка, матричное умножение).
Использование текстур и поверхностей: работа с cudaTextureObject_t.
Пример использования CUDA Graph API.
Исследование кэш-поведения, профилирование с nsight.
Использование cuBLAS, cuDNN и других библиотек.

4.2. Планы лабораторных работ и практикумов
Написание ядра CUDA и сопоставление модели исполнения.
Исследование конфигураций block / grid и их влияние на производительность.
Анализ warp-divergence и оптимизация ветвления.
Оптимизация доступа к глобальной и разделяемой памяти.
Разработка тестов с конфликтами доступа в shared memory.
Работа с несколькими потоками выполнения (cudaStream), измерение задержек.
Демонстрация асинхронной передачи данных и вычислений.
Реализация редукции с использованием __shfl_down_sync.
Обмен данными между потоками через __shfl_sync.
Матричное умножение: последовательная и параллельная реализация.
Использование текстурной памяти для фильтрации изображения.
Использование CUDA Graph API для запуска нескольких операций.
Настройка кэша L1/L2, профилирование и замеры.
Применение cuBLAS для умножения матриц и cuDNN для нейросетей.

4.3. Материалы по практической части курса
4.3.1. Учебно-методические пособия
Sanders J., Kandrot E. CUDA by Example
Kirk D., Hwu W. Programming Massively Parallel Processors
NVIDIA CUDA Toolkit Documentation
4.3.2. Учебные справочники
https://docs.nvidia.com/cuda/
NVIDIA Developer Blog
Nsight Compute и Nsight Systems документация
4.3.3. Задачники (практикумы)
CUDA Hands-On Labs
NVIDIA CUDA Samples
A100 CUDA Optimization Examples
4.3.4. Наглядно-иллюстративные материалы
Слайды о модели CUDA

Схемы warp и блока, карты памяти
Демонстрации через Nsight Visual Profiler

4.3.5. Другие виды материалов

GitHub-проекты по оптимизации и API Graph
Видео-лекции от CUDA University
Интерактивные демо на платформе NGC (NVIDIA GPU Cloud)

4.4. Вопросы и задания для самостоятельной работы студентов

Реализовать параллельную редукцию с использованием warp-интринсиков.
Исследовать поведение памяти при работе с shared и global access.
Провести эксперименты по cudaMemcpyAsync.
Написать собственную абстракцию над cudaGraph_t.
Использовать cudaStream для одновременного выполнения нескольких ядер.

4.5. Тематика рефератов, эссе и других форм самостоятельных работ

Сравнение CUDA и OpenCL: архитектурные отличия
Роль потоков warp в архитектуре NVIDIA
История эволюции моделей памяти CUDA
CUDA Graphs как средство динамического управления графом задач
Поведение L1-кэша в различных вычислительных задачах

4.6. Образцы контрольных и промежуточных заданий

Фрагмент контрольной:
Назовите основные типы памяти CUDA.
Объясните понятие warp divergence и как его избежать.
Реализуйте простую операцию редукции с использованием __shfl_down_sync.

4.10. Банк тестов для самоконтроля

Примеры:
Какое максимальное количество потоков может быть в одном блоке?
☐ 512 ☐ 1024 ☐ 2048
Ответ: 1024
Что делает __shfl_down_sync(mask, val, delta)?
Ответ: Перемещает значение val вниз по warp на delta позиций.

4.11. Методики решения и ответы к тестам

Вопрос: Как избежать конфликта доступа в разделяемую память?
Ответ: Выравнивание данных, использование различных банков памяти.
Вопрос: Как измерить задержку между двумя асинхронными событиями?
Ответ: Использовать cudaEventRecord и cudaEventElapsedTime.