Содержание

Модуль 1. Введение. Аппаратно-программная архитектура CUDA

Презентации лекций: cuda_course_part1.pdf - cuda_course_part2.pdf

1. Преимущества CUDA, место CUDA в современном HPC
2. Архитектура GPU в сравнении с архитектурой CPU, понятие Compute Capability, Fermi и Kepler
3. Гибридное программирование CPU+GPU: сетки, ядра, запуск ядер, управление памятью GPU
4. Модель исполнения Nvidia SIMT: очередь блоков, выполнение блоков варпами, сравнение с SIMD и SMP, масштабируемость
5. Гибридное программирование CPU+GPU: выбор устройства, обработка ошибок, асинхронность в CUDA, потоки и события, вычисление времени выполнения
6. Компиляция и запуск программ с использованием CUDA

Модуль 2. Иерархия памяти

Презентации лекций: cuda_course_part2.pdf - cuda_course_part4.pdf

1. Глобальная память: выделение, кеши L1 и L2, транзакции, шаблоны доступа, распространеные приемы: набивка матриц, раздельное хранение полей структур
2. Разделяемая память: выделение и особенности использования; синхронизация; банки общей памяти, различия между Fermi и Kepler, банк-конфликты
3. Константная память: выделение и особенности использования; однородные обращения, причины неявного использования константной памяти.
4. Регистры и локальная память: особенности выделения регистров, различия между Fermi и Kepler; причины неявного использования локальной памяти
5. Расчет occupancy: -Xptxas –v и CUDA occupancy calculator

Модуль 3. Профилирование и отладка

Презентации лекций: cuda_course_part5.pdf

1. Профилировщик с графическим интерфейсом nvvp; события и метрики
2. Профилирование из консоли: command line profiler, nvprof
3. Удаленное профилирование
4. Отладка при помощи cuda-gdb: основные расширения, распространенные сценарии использования

Модуль 4. Продвинутое использование CUDA

Презентации лекций: cuda_course_part6.pdf - cuda_course_part8.pdf

1. Pinned-память: выделение и особенности, доступ к памяти CPU напрямую из ядер, неблокирующие копирования, ускорение блокирующих копирований
2. UVA – единое виртуальное адресное пространство всех GPU и CPU
3. CUDA-потоки: аппаратные возможности, разбор случаев применения, различия между Fermi и Kepler, синхронизация потоков
4. Multi-GPU: CUDA-контекст; multi-GPU с одним CPU-потоком, с использованием OpenMP, с использованием MPI
5. Peer-to-peer копирования между GPU, прямой доступ к памяти другого GPU

Модуль 5. Компиляция CUDA-кода

Презентации лекций: cuda_course_part9.pdf

1. Устройство драйвера компиляции NVCC: цепочка компиляции, вспомогательные программы, полезные флаги
2. Виртуальные и физические архитектуры: проблема совместимости, промежуточный код PTX, компиляция под различные архитектуры, неявная JIT-компиляция, Compiler Cache
3. Объектные модули: CUBIN, FATBIN, встраивание FATBIN в объектные файлы с CPU-кодом, cuobjdump
4. Линковка CUDA-модулей: relocalable код, схема линковки, флаги
5. Низкоуровневый анализ программ: PTX и ассемблер
6. Явная JIT-компиляция при помощи cudaDriverAPI

Задания

Задание 1. Перемножение матриц на GPU

Описание задания: cuda_course_task_matmul.pdf

Этап 1. Простейшее перемножение

Реализовать программу, выполняющую перемножение прямоугольных матриц с использованием CUDA и проверку результата. Программа принимает через командную строку размеры матриц и необязательный флаг проверки результата. Генерирует случайным образом матрицы, выполняет перемножение на GPU и CPU (если требуется проверить результат), выводит в командную строку время работы CUDA-ядра и результат проверки, если был задан флаг.

Этап 2. Оптимизация загруженности GPU и обращений в глобальную память

В программу из предыдущего этапа добавить набивка (padding) в матрицы для выравнивания начал строк.

Этап 3. Использование разделяемой памяти

В программу из предыдущего этапа добавить ручное дополнение матриц нулями до размеров, кратных 32. Реализовать ядро перемножения матриц с использованием общей памяти, при выполнении которого каждая нить рассчитывает два элемента матрицы-результата.

Отчет.

Таблица результатов тестов времени работы ядра

Размеры матриц	Простейшее перемножение	Перемножение матриц с набивкой		Перемножение матриц с общей памятью	
	Сетка из блоков 32x32	Сетка из блоков 32x32	Сетка из блоков 32x16	Нить считает один элемент, Сетка из блоков 32x32	Нить считает два элемента, Сетка из блоков 32x16
1792x1792					
1793x1793					
…					

2048x2048					

Представить графическое представление таблицы результатов тестов в виде графиков, объяснение полученных результатов, исходные тексты программ.

Задание 2. Однокубитовая квантовая операция на GPU

Описание задания: cuda_course_task_quantum.pdf

Этап 1. Однокубитовая операция

Реализовать программу, выполняющую однокубитовую квантовую операцию с использованием CUDA. Программа принимает через командную строку число кубитов, индекс кубита, над которым нужно провести операцию, и необязательный флаг проверки результата. Программа генерирует суперпозицию состояний квантового регистра и квантовый вентиль, применяет квантовый вентиль к указанному кубиту регистра на GPU и на CPU (если треубется проверить результат), выводит в командную строку время работы CUDA-ядра и копирований памяти между CPU и GPU, выводит результат проверки, если был задан флаг

Этап 2. Последовательность однокубитовых операций с общей памятью

В программе из предыдущего этапа одиночную однокубитовую операцию обобщить до последовательности однокубитовых, применяемых к последним кубитам регистра. Программа принимает через командную строку размер региста, число кубитов b и необязательный флаг проверки результата. Программа выполняет на GPU два CUDA-ядра, поочередно применяющие квантовый вентиль к b последним кубитам регистра: с хранением промежуточных результатов в разделяемой памяти и без, выводит в командную строку время работы ядра с разделяемой памятью, время работы ядра без разделяемой памяти и результаты проверки, если был задан флаг.

Отчет.

Таблица результатов тестов времени работы ядра при максимальном числе кубитов

Индекс кубита	Однокубитовая квантовая операция	
	Ядро	Копирования между CPU и GPU
0		

1		
..		
N-1		

b	Применение вентиля к b последним кубитам		
	Ядро без общей памяти	Ядро с общей памятью	
1			
…			
10			

Представить графическое представление таблицы результатов тестов в виде графиков, объяснение полученных результатов, исходные тексты программ.

Задание 3. Фильтрация изображений с матричным фильтром

Описание задания: cuda_course_task_filtering.pdf

Этап 1. Базовый вариант

Реализовать программу, выполняющую на GPU фильтрацию (корреляцию) изображения с квадратным матричным фильтром. Программа работает как с реальными данными (изображениями), так и с синтетическими – для тестов GPU-реализации. Для проверки результатов работы программа сохраняет на диск фильтрованные изображения. В качестве входных данных программа принимает из командной строки размер фильтра и путь к графическому файлу, либо размеры синтетической матрицы. Программа загружает входное изображение или генерирует синтетические данные, последовательно применяет на GPU два фильтра, выводит в консоль время работы ядер и копирований, сохраняет результующее изображение.

Этап 2. Использование потоков

В программу из предыдущего этапа добавить использование нескольких потоков для одновременного копирования данных и счета ядра, число потоков передается через параметр командной строки.

Этап 3. Использование multi-GPU

В программу из предыдущего этапа добавить использование нескольких GPU, число GPU передается через параметр командной строки.

Отчет.

Таблица результатов тестов на синтетических данных

Размер матрицы	Размер ядра фильтра	Суммарное время копирований + время счета для ядра		
		Базовый вариант	С четырьмя потоками	С четырьмя multi-GPU, на каждом 4 потока
1024x1024	N1			
1024x1024	N2			
2048x1024	N3			
…	….			

Представить графическое представление таблицы результатов тестов на синтетических данных в виде столбчатых диаграмм, объяснение полученных результатов, исходные тексты программ, скриншоты профилировщика nvvp для каждого этапа, демонстрирующие эффект от использования потоков и multi-GPU. Корректность работы на каждом этапе подтвердить примерами генерируемых изображений на реальных данных.