Теория параллелизма

**Отчет**

Лабораторная работа 8

Выполнил Кошелев Никита, 22931  
28.05.24

**Цель работы**: Реализовать и оптимизировать решение уравнение теплопроводности (разностная схема –пятиточечный шаблон) в двумерной области на равномерных сетках на GPU используя CUDA.

**Используемый компилятор:** **nvc++**  
**Используемый** профилировщик: “Nsight Systems”.  
Замер времени работы: Библиотека “chrono”

Реализация кода

* Выделение памяти на CUDA для массивов (а также stream and graph) и ее автоматическое удаление происходит с помощью   
    
  умных указателей: выделения памяти, если память выделяется с ошибкой, происходит завершение программы, а также удаление при неиспользовании
*   
  Задание размера и блоки для сетки (32\*32 максимальный размер для группы потоков которые могут использовать локальную память(блок), (n + block.x - 1)/ block.x для того, чтобы покрыть всю матрицу и правильного округления)
* Создание графа



   
начало создания графа в потоке stream

  
Код для вычисления 100 итераций перед вычислением ошибки

  
Эта функция для рассчета матрицы

С помощью blockIdx.x \* blockDim.x + threadIdx.x; вычисляется глобальные индексы элементов матрицы( blockIdx - индексы блоков, threadIdx - индексы потоков, blockDim — размер блока потоков.  
с помощью   
проверка, что не будут затрагиваться границы матрицы  
  
  
функция для проверки ошибки  
  
  
  
завершение записи графа и сохрения для дальнейшего использования

4.  В этой части кода выполняется сохраненный граф, ошибка копируется в хост для проверки  
  
  
Вычисление ошибки - находит максимум в массиве

Сравнение работы на вычислительных устройствах

**CPU**-onecore

|  |  |  |  |
| --- | --- | --- | --- |
| Размер сетки | Время выполнения | Точность | Количество операций |
| 128\*128 | 0.672911 | 0.000001 | 30101(30074) |
| 256\*256 | 9.620933 | 0.000001 | 102901(102885) |
| 512\*512 | 134.230674 | 0.000001 | 339601(339599) |

**CPU**-multicore

|  |  |  |  |
| --- | --- | --- | --- |
| Размер сетки | Время выполнения | Точность | Количество операций |
| 128\*128 | 2.329963 | 0.000001 | 30101(30074) |
| 256\*256 | 6.132657 | 0.000001 | 102901(102885) |
| 512\*512 | 37.022035 | 0.000001 | 339601(339599) |
| 1024\*1024 | 329.625234 | >0.000001 | 1000000 |

GPU - **оптимизированный** вариант

|  |  |  |  |
| --- | --- | --- | --- |
| Размер сетки | Время выполнения(c) | Точность | Количество операций |
| 128\*128 | 0.434266 | 0.000001 | 30074(до возвращения ошибки раз в 100 раз)  300101 |
| 256\*256 | 1.364479 | 0.000001 | 102885\102901 |
| 512\*512 | 4.240394 | 0.000001 | 339599\339601 |
| 1024\*1024 | 35.587202 | >0.000001 | 1000000 |

GPU - **оптимизированный** вариант + cuBLAS

|  |  |  |  |
| --- | --- | --- | --- |
| Размер сетки | Время выполнения(c) | Точность | Количество операций |
| 128\*128 | 0.305420 | 0.000001 | 300101 |
| 256\*256 | 1.173664 | 0.000001 | 102901 |
| 512\*512 | 4.485946 | 0.000001 | 339601 |
| 1024\*1024 | 35.504507 | >0.000001 | 1000000 |

GPU + CUDA

|  |  |  |  |
| --- | --- | --- | --- |
| Размер сетки | Время выполнения(c) | Точность | Количество операций |
| 128\*128 | 0.060651 | 0.000001 | 300100 |
| 256\*256 | 0.218702 | 0.000001 | 102900 |
| 512\*512 | 1.249609 | 0.000001 | 339600 |
| 1024\*1024 | 24.366812 | >0.000001 | 1000000 |

Диаграммы сравнения времени работы CPU-one, CPU-multi, GPU(**оптимизированный** вариант**),**   
GPU+ cuBLAS, GPU + CUDA для **разных** размеров **сеток**

  
Вычисление графа на 100 итерациях занимает 3мс  
  
  
На других программах в самом оптимизированном варианте 20мс  
  
  
**Вывод:**1. С помощью применения графов увеличивается производительность вычислений, так как нет необходимости постоянного вызова каждой функции (они группируются), внутри графа происходит улучшение распределения ресурсов, уменьшение задержек за счет параллельного выполнения узлов графа  
2. Использование «умных указателей» позволяет автоматически выделять память, а также освобождать ресурсы(RALL)  
3. Использование функции cub::DeviceReduce::Max позволяет быстро найти максимальное значение (ошибку) в матрице, так как функции из этой библиотеки высокоэффективно реализованы  
4. Разделение вычислений на блоки и сетки позволяет эффективно использовать архитектуру GPU   
5. По графикам можно увидеть что одно и тоже решение реализованное на разных вычислительных устройствах имеет самое большое ускорение при использовании CUDA