Кафедра прикладной математики и кибернетики

Выполнили:

Студенты 3 курса группы ИП-111  
Корнилов А.А.,  
Попов М.И.,

Толкач А.А.

Проверил:

Профессор кафедры ПМиК  
Малков Е.А.

ЛАБОРАТОРНАЯ РАБОТА №10

По дисциплине: «Программирование графических процессоров»

Сибирский государственный университет  
телекоммуникаций и информатики

Новосибирск, 2024

Министерство цифрового развития, связи  
и массовых коммуникаций Российской Федерации

**Задание:**

* сравнить время выполнения умножения матриц с использованием numpy.matmul и cuBLAS.

**Цель:** определить целесообразность использования python для перемножения матриц

**Оборудование**: Видеокарта GTX 1050TI (Pascal)

**Выполнение работы:**

Для выполнения работы была написана программа на языке python для перемножения матриц используя библиотеку numpy и её функцию matmul, для замера времени используется time.

|  |
| --- |
| import numpy as np from time import time  start = time() num = 1 << 12 rows = 2 \* num  A = np.empty((rows, rows)) B = np.empty((rows, rows))  for i in range(rows):  for j in range(rows):  A[i][j] = i+j  B[i][j] = i+j+1  #print(A) #print(B) С = np.matmul(A, B)  end = time() print('Функция выполнилась за ', end - start) |

Листинг 1 – программа LR10\_G1.py

Команда компиляции и результат работы программы:

|  |
| --- |
| miron@DESKTOP-UMC1Q46:/mnt/d/Projects/CUDA\_CMake/LR10/src$ time(python3 LR10\_1G.py)  Функция выполнилась за 54.802191972732544  ZZZZZZZZ  real 0m55,173s  user 2m15,848s  sys 0m5,120s |

Для второго задания взята программа для 8 лабораторной работы для перемножения матриц, замер времени проводился через cudaEvent и запуск через команду Measure-Command в PowerShell.

|  |
| --- |
| #include <iostream> #include <iomanip> #include <cuda\_runtime.h> #include <cublas\_v2.h>  void initMatrix(float \*matrix, int rows, int cols) {  for (int i = 0; i < rows; ++i) {  for (int j = 0; j < cols; ++j) {  matrix[i \* cols + j] = i + j;  }  } } void printMatrix(float \*matrix, int rows, int cols) {  for (int i = 0; i < rows; ++i) {  for (int j = 0; j < cols; ++j) {  std::cout << matrix[i \* cols + j] << "\t";  }  std::cout << std::endl;  } }  int main() {  const int num = 1 << 12;  int N = 2 \* num;  float elapsedTime = 0;  cudaEvent\_t start, stop;  cudaEventCreate(&start);  cudaEventCreate(&stop);   float \*h\_A = new float[N \* N];  float \*h\_B = new float[N \* N];  float \*h\_C = new float[N \* N];   initMatrix(h\_A, N, N);  initMatrix(h\_B, N, N);   cublasHandle\_t handle;  **cublasCreate**(&handle);   float \*d\_A, \*d\_B, \*d\_C;  cudaMalloc(&d\_A, N \* N \* sizeof(float));  cudaMalloc(&d\_B, N \* N \* sizeof(float));  cudaMalloc(&d\_C, N \* N \* sizeof(float));   cudaMemcpy(d\_A, h\_A, N \* N \* sizeof(float), cudaMemcpyHostToDevice);  cudaMemcpy(d\_B, h\_B, N \* N \* sizeof(float), cudaMemcpyHostToDevice);   float alpha = 1.0f, beta = 0.0f;   cudaEventRecord(start, 0);  **cublasSgemm**(handle, CUBLAS\_OP\_N, CUBLAS\_OP\_N, N, N, N, &alpha, d\_A, N, d\_B, N, &beta, d\_C, N);  cudaDeviceSynchronize();  cudaEventRecord(stop, 0);   cudaEventSynchronize(stop);  cudaEventElapsedTime(&elapsedTime, start, stop);  std::cout << "Time using cuBLAS code: " /\*<< std::setprecision(15)\*/ << elapsedTime << std::endl;   cudaMemcpy(h\_C, d\_C, N \* N \* sizeof(float), cudaMemcpyDeviceToHost);   cudaFree(d\_A);  cudaFree(d\_B);  cudaFree(d\_C);  **cublasDestroy**(handle);  delete[] h\_A;  delete[] h\_B;  delete[] h\_C;  return 0; } |

Листинг 1 – программа LR08\_1G.cu

Результат работы программы:

|  |
| --- |
| (venv) PS D:\Projects\CUDA\_CMake\LR10\src> Measure-Command {.\LR10\_1G.exe}  Days : 0  Hours : 0  Minutes : 0  Seconds : 0  Milliseconds : 296  Ticks : 2963658  TotalDays : 3,43015972222222E-06  TotalHours : 8,23238333333333E-05  TotalMinutes : 0,00493943  TotalSeconds : 0,2963658  TotalMilliseconds : 296,3658 |

|  |  |  |
| --- | --- | --- |
| секунды | numpy | cuBLAS |
| 1 << 2 | 0,30 | 0,30 |
| 1 << 4 | 0,16 | 0,3073171 |
| 1 << 6 | 0,163 | 0,3014205 |
| 1 << 8 | 0,363 | 0,3576197 |
| 1 << 9 | 0,919 | 0,451934 |
| 1 << 10 | 3,398 | 0,5080446 |
| 1 << 11 | 13,538 | 0,7352486 |
| 1 << 12 | 57,417 | 1,682821 |

Вывод: в ходе выполнения лабораторной работы, была исследована и применена работа с языком python и его библиотекой numpy, в ходе работы было выяснено что время выполнения программы для перемножения матриц сильно возрастает если использовать python3 и размер матриц более чем 2^9, если брать размер меньше то время будет не значительно различаться в отличии от программы написанной используя библиотеку cuBLAS.