Задание 1
Отчёт
по CUDA

Ши Хуэй shihuicollapsor@gmail.com

1. Постановка задачи

Первое задание состоит в оптимальном распараллеливании программы 3х мерного Якоби. Менять программу можно, главное, чтобы значения выходных массивов совпадали. Единственная трудность пока на данный момент – это редукция. Как ее распараллеливать будет рассказано на следующих презентациях.

Запускать программу нужно на максимальном количестве памяти, которое доступно на запускаемом ГПУ (соответственно, корректировать значение L).

Для сдачи нужно завести git репозиторий и туда закоммитить свой код. Там должен быть какой-либо простой makefile, с помощью которого я могу собрать эту программу на любом сервере с ГПУ и запустить. Должна быть возможность запуска на ЦПУ и ГПУ, а также должна быть возможность запуска в режиме сравнения выходных массивов, чтобы понять, что алгоритм считает верно. В качестве результата нужно оценить ускорение программы по отношению к последовательной версии (скомпилированной с максимальными опциями оптимизации).

2. Формат командной строки

nvcc jac3d_cuda2.cu -o cuda2

3. Спецификация системы

- Operating system : Linux 6.8.0-45-generic
- Vendor string and code : GenuineIntel (1, 0x1)
- Model string and code : Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz (165, 0xa5)
- CPU revision : 2.000000
- CPUID : Family/Model/Stepping 6/165/2, 0x06/0xa5/0x02
- CPU Max MHz : 5000
- CPU Min MHz : 800
- Total cores : 12
- SMT threads per core : 2
- Cores per socket : 6
- Sockets : 1
- Cores per NUMA region : 12

- NUMA regions : 1
- Running in a VM : no
- Number Hardware Counters : 10
- Max Multiplex Counters : 384

4. Описание алгоритмов

Код реализует 3D-итерационный алгоритм Якоби и в то же время реализует сравнение производительности алгоритма, запущенного на CPU и GPU.

```
#include <stdio.h>
#include <cuda_runtime.h>
#include <math.h>
#include <chrono>

#define MAXEPS 0.5f
#define ITMAX 100
#define BLOCK_SIZE 8
```

- MAXEPS: условие завершения итерации, остановка при ошибке (EPS) менее 0,5.
- ITMAX: максимальное количество итераций для предотвращения бесконечных циклов.
- BLOCK_SIZE: размер каждого блока CUDA графического процессора для оптимизации параллельной работы графического процессора.

```
#define cudaCheckError() {                                  \
    cudaError_t e = cudaGetLastError();                     \
    if (e != cudaSuccess) {                                 \
        printf("CUDA error %s:%d: %s\n", __FILE__, __LINE__,    \
            cudaGetErrorString(e));                         \
        exit(EXIT_FAILURE);                                 \
    }                                                       \
}

__device__ __host__ inline double Max(double a, double b) {
    return a > b ? a : b;
}

// GPU Kernel: Update A matrix
__global__ void jacobi_update_A(double *A, double *B, double *eps, int L) {
    int i = blockIdx.x * blockDim.x + threadIdx.x + 1;
    int j = blockIdx.y * blockDim.y + threadIdx.y + 1;
```

```
    int k = blockIdx.z * blockDim.z + threadIdx.z + 1;

    if (i < L - 1 && j < L - 1 && k < L - 1) {
      int idx = i * L * L + j * L + k;
      if (idx < L * L * L) { // Ensure index is within bounds
        double diff = fabs(B[idx] - A[idx]);
        atomicMax((unsigned long long int *)eps, __double_as_longlong(diff)); // Use atomic for reduction
        A[idx] = B[idx];
      }
    }
  }
```

- blockIdx.x/y/z: индекс блока потоков.
- blockDim.x/y/z: индекс потока в блоке.
- threadIdx.x/y/z: количество потоков в блоке потоков (то есть размер блока).

Глобальный индекс потока = индекс блока потоков × количество потоков в блоке + индекс потока внутри блока + 1.

```
// GPU Kernel: Update B matrix
__global__ void jacobi_update_B(double *A, double *B, int L) {
  int i = blockIdx.x * blockDim.x + threadIdx.x + 1;
  int j = blockIdx.y * blockDim.y + threadIdx.y + 1;
  int k = blockIdx.z * blockDim.z + threadIdx.z + 1;

  if (i < L - 1 && j < L - 1 && k < L - 1) {
    int idx = i * L * L + j * L + k;
    if (idx < L * L * L) { // Ensure index is within bounds
      B[idx] = (A[(i - 1) * L * L + j * L + k] +
            A[i * L * L + (j - 1) * L + k] +
            A[i * L * L + j * L + (k - 1)] +
            A[i * L * L + j * L + (k + 1)] +
            A[i * L * L + (j + 1) * L + k] +
            A[(i + 1) * L * L + j * L + k]) / 6.0;
    }
  }
}

// CPU version of Jacobi update
```

```
void jacobi_cpu(double *A, double *B, double &eps, int L) {
  eps = 0.0;
  for (int i = 1; i < L - 1; ++i) {
    for (int j = 1; j < L - 1; ++j) {
      for (int k = 1; k < L - 1; ++k) {
        int idx = i * L * L + j * L + k;
        double diff = fabs(B[idx] - A[idx]);
        eps = Max(eps, diff);
        A[idx] = B[idx];
      }
    }
  }

  for (int i = 1; i < L - 1; ++i) {
    for (int j = 1; j < L - 1; ++j) {
      for (int k = 1; k < L - 1; ++k) {
        int idx = i * L * L + j * L + k;
        B[idx] = (A[(i - 1) * L * L + j * L + k] +
                  A[i * L * L + (j - 1) * L + k] +
                  A[i * L * L + j * L + (k - 1)] +
                  A[i * L * L + j * L + (k + 1)] +
                  A[i * L * L + (j + 1) * L + k] +
                  A[(i + 1) * L * L + j * L + k]) / 6.0;
      }
    }
  }
}

// Initialize the matrices
void initialize(double *A, double *B, int L) {
  for (int i = 0; i < L; ++i) {
    for (int j = 0; j < L; ++j) {
      for (int k = 0; k < L; ++k) {
        int idx = i * L * L + j * L + k;
        A[idx] = 0;
        if (i == 0 || j == 0 || k == 0 || i == L - 1 || j == L - 1 || k == L - 1)
```

```
                B[idx] = 0;
            else
                B[idx] = 4 + i + j + k;
        }
    }
}

// Main program
int main(int argc, char **argv) {
    bool use_gpu = (argc > 1 && strcmp(argv[1], "gpu") == 0);
    int L = 384;

    size_t free_mem, total_mem;
    cudaMemGetInfo(&free_mem, &total_mem);
    L = pow((free_mem * 0.9 / (2.0 * sizeof(double))), 1.0 / 3.0);
```

Динамически настраивайте размер матрицы и динамически вычисляйте максимальный размер L на основе доступной памяти графического процессора

```
    if (use_gpu) {
        // GPU Execution
        printf("Running on GPU with L = %d\n", L);

        double *A, *B, *eps;

        cudaMallocManaged(&A, L * L * L * sizeof(double));
        cudaMallocManaged(&B, L * L * L * sizeof(double));
        cudaMallocManaged(&eps, sizeof(double));

        initialize(A, B, L);

        dim3 block(BLOCK_SIZE, BLOCK_SIZE, BLOCK_SIZE);
        dim3 grid((L - 2 + block.x - 1) / block.x, (L - 2 + block.y - 1) / block.y, (L - 2 +
block.z - 1) / block.z);
        cudaEvent_t start, stop;
        cudaEventCreate(&start);
        cudaEventCreate(&stop);
        cudaEventRecord(start);
```

```
    for (int it = 1; it <= ITMAX; it++) {
        *eps = 0.0;

        jacobi_update_A<<<grid, block>>>(A, B, eps, L);
        cudaDeviceSynchronize();
        cudaCheckError();

        jacobi_update_B<<<grid, block>>>(A, B, L);
        cudaDeviceSynchronize();
        cudaCheckError();

        printf(" IT = %4d   EPS = %14.7E\n", it, *eps);

        if (*eps < MAXEPS)
            break;
    }

    cudaEventRecord(stop);
    cudaEventSynchronize(stop);
    float milliseconds = 0;
    cudaEventElapsedTime(&milliseconds, start, stop);

    printf(" GPU Time (ms): %f\n", milliseconds);

    cudaFree(A);
    cudaFree(B);
    cudaFree(eps);
} else {
    // CPU Execution
    printf("Running on CPU with L = %d\n", L);

    double *A = (double *)malloc(L * L * L * sizeof(double));
    double *B = (double *)malloc(L * L * L * sizeof(double));
    double eps = 0.0;
```

```
        initialize(A, B, L);

        auto cpu_start = std::chrono::high_resolution_clock::now();

        for (int it = 1; it <= ITMAX; it++) {
            eps = 0.0;
            jacobi_cpu(A, B, eps, L);
            printf(" IT = %4d   EPS = %14.7E\n", it, eps);

            if (eps < MAXEPS)
                break;
        }

        auto cpu_stop = std::chrono::high_resolution_clock::now();
        std::chrono::duration<double> elapsed = cpu_stop - cpu_start;

        printf(" CPU Time (s): %f\n", elapsed.count());

        free(A);
        free(B);
    }

    printf(" Jacobi3D Benchmark Completed.\n");
    return 0;
}
```

5.Заключение

Для этой программы был написан файл Makefile.

Введите следующую команду в терминале:

nvcc jac3d_cuda.cu -o cuda2

После этого мы войдем:

make run_cpu

Программа будет запущена на CPU, и результаты показаны слева в таблице ниже.

make run_gpu

Программа будет запущена на GPU, и результаты показаны справа в таблице ниже.

collapsor@collapsor-G5-5500:~/Desktop/CUDA$ nvcc jac3d_cuda2.cu -o cuda2

collapsor@collapsor-G5-5500:~/Desktop/CUDA$ ./cuda2 cpu	collapsor@collapsor-G5-5500:~/Desktop/CUDA$ ./cuda2 gpu
Running on CPU with L = 693	Running on GPU with L = 693
IT = 1 EPS = 2.0770000E+03	IT = 1 EPS = 2.0770000E+03
IT = 2 EPS = 1.0390000E+03	IT = 2 EPS = 1.0390000E+03
IT = 3 EPS = 4.0383333E+02	IT = 3 EPS = 4.0383333E+02
IT = 4 EPS = 2.5951852E+02	IT = 4 EPS = 2.5951852E+02
IT = 5 EPS = 2.1141667E+02	IT = 5 EPS = 2.1141667E+02
IT = 6 EPS = 1.7935802E+02	IT = 6 EPS = 1.7935802E+02
IT = 7 EPS = 1.5145190E+02	IT = 7 EPS = 1.5145190E+02
IT = 8 EPS = 1.2541529E+02	IT = 8 EPS = 1.2541529E+02
IT = 9 EPS = 1.0719672E+02	IT = 9 EPS = 1.0719672E+02
IT = 10 EPS = 9.4896359E+01	IT = 10 EPS = 9.4896359E+01
IT = 11 EPS = 8.6348853E+01	IT = 11 EPS = 8.6348853E+01
IT = 12 EPS = 8.1156211E+01	IT = 12 EPS = 8.1156211E+01
IT = 13 EPS = 7.5512819E+01	IT = 13 EPS = 7.5512819E+01
IT = 14 EPS = 7.0219036E+01	IT = 14 EPS = 7.0219036E+01
IT = 15 EPS = 6.4945633E+01	IT = 15 EPS = 6.4945633E+01
IT = 16 EPS = 6.0200110E+01	IT = 16 EPS = 6.0200110E+01
IT = 17 EPS = 5.5628457E+01	IT = 17 EPS = 5.5628457E+01
IT = 18 EPS = 5.2037555E+01	IT = 18 EPS = 5.2037555E+01
IT = 19 EPS = 4.9018618E+01	IT = 19 EPS = 4.9018618E+01
IT = 20 EPS = 4.6646850E+01	IT = 20 EPS = 4.6646850E+01
IT = 21 EPS = 4.4740502E+01	IT = 21 EPS = 4.4740502E+01
IT = 22 EPS = 4.3024448E+01	IT = 22 EPS = 4.3024448E+01
IT = 23 EPS = 4.1354680E+01	IT = 23 EPS = 4.1354680E+01
IT = 24 EPS = 3.9664887E+01	IT = 24 EPS = 3.9664887E+01
IT = 25 EPS = 3.8051892E+01	IT = 25 EPS = 3.8051892E+01
IT = 26 EPS = 3.6448486E+01	IT = 26 EPS = 3.6448486E+01
IT = 27 EPS = 3.4934067E+01	IT = 27 EPS = 3.4934067E+01

collapsor@collapsor-G5-5500:~/Desktop/CUDA$ ./cuda2 cpu	collapsor@collapsor-G5-5500:~/Desktop/CUDA$ ./cuda2 gpu
IT = 28 EPS = 3.3444865E+01	IT = 28 EPS = 3.3444865E+01
IT = 29 EPS = 3.2046667E+01	IT = 29 EPS = 3.2046667E+01
IT = 30 EPS = 3.0896418E+01	IT = 30 EPS = 3.0896418E+01
IT = 31 EPS = 2.9810342E+01	IT = 31 EPS = 2.9810342E+01
IT = 32 EPS = 2.8904726E+01	IT = 32 EPS = 2.8904726E+01
IT = 33 EPS = 2.8061452E+01	IT = 33 EPS = 2.8061452E+01
IT = 34 EPS = 2.7382036E+01	IT = 34 EPS = 2.7382036E+01
IT = 35 EPS = 2.6691393E+01	IT = 35 EPS = 2.6691393E+01
IT = 36 EPS = 2.6015685E+01	IT = 36 EPS = 2.6015685E+01
IT = 37 EPS = 2.5336015E+01	IT = 37 EPS = 2.5336015E+01
IT = 38 EPS = 2.4675740E+01	IT = 38 EPS = 2.4675740E+01
IT = 39 EPS = 2.4016497E+01	IT = 39 EPS = 2.4016497E+01
IT = 40 EPS = 2.3379181E+01	IT = 40 EPS = 2.3379181E+01
IT = 41 EPS = 2.2746260E+01	IT = 41 EPS = 2.2746260E+01
IT = 42 EPS = 2.2136460E+01	IT = 42 EPS = 2.2136460E+01
IT = 43 EPS = 2.1533251E+01	IT = 43 EPS = 2.1533251E+01
IT = 44 EPS = 2.0973148E+01	IT = 44 EPS = 2.0973148E+01
IT = 45 EPS = 2.0478860E+01	IT = 45 EPS = 2.0478860E+01
IT = 46 EPS = 1.9995550E+01	IT = 46 EPS = 1.9995550E+01
IT = 47 EPS = 1.9585132E+01	IT = 47 EPS = 1.9585132E+01
IT = 48 EPS = 1.9184011E+01	IT = 48 EPS = 1.9184011E+01
IT = 49 EPS = 1.8834097E+01	IT = 49 EPS = 1.8834097E+01
IT = 50 EPS = 1.8504304E+01	IT = 50 EPS = 1.8504304E+01
IT = 51 EPS = 1.8179363E+01	IT = 51 EPS = 1.8179363E+01
IT = 52 EPS = 1.7852687E+01	IT = 52 EPS = 1.7852687E+01
IT = 53 EPS = 1.7531990E+01	IT = 53 EPS = 1.7531990E+01
IT = 54 EPS = 1.7210867E+01	IT = 54 EPS = 1.7210867E+01
IT = 55 EPS = 1.6896505E+01	IT = 55 EPS = 1.6896505E+01
IT = 56 EPS = 1.6582722E+01	IT = 56 EPS = 1.6582722E+01

collapsor@collapsor-G5-5500:~/Desktop/CUDA$ ./cuda2 cpu	collapsor@collapsor-G5-5500:~/Desktop/CUDA$ ./cuda2 gpu
IT = 57 EPS = 1.6276213E+01	IT = 57 EPS = 1.6276213E+01
IT = 58 EPS = 1.5971044E+01	IT = 58 EPS = 1.5971044E+01
IT = 59 EPS = 1.5673455E+01	IT = 59 EPS = 1.5673455E+01
IT = 60 EPS = 1.5377776E+01	IT = 60 EPS = 1.5377776E+01
IT = 61 EPS = 1.5089821E+01	IT = 61 EPS = 1.5089821E+01
IT = 62 EPS = 1.4823532E+01	IT = 62 EPS = 1.4823532E+01
IT = 63 EPS = 1.4573695E+01	IT = 63 EPS = 1.4573695E+01
IT = 64 EPS = 1.4329295E+01	IT = 64 EPS = 1.4329295E+01
IT = 65 EPS = 1.4115757E+01	IT = 65 EPS = 1.4115757E+01
IT = 66 EPS = 1.3905279E+01	IT = 66 EPS = 1.3905279E+01
IT = 67 EPS = 1.3702972E+01	IT = 67 EPS = 1.3702972E+01
IT = 68 EPS = 1.3527626E+01	IT = 68 EPS = 1.3527626E+01
IT = 69 EPS = 1.3351323E+01	IT = 69 EPS = 1.3351323E+01
IT = 70 EPS = 1.3177138E+01	IT = 70 EPS = 1.3177138E+01
IT = 71 EPS = 1.3002433E+01	IT = 71 EPS = 1.3002433E+01
IT = 72 EPS = 1.2830139E+01	IT = 72 EPS = 1.2830139E+01
IT = 73 EPS = 1.2657688E+01	IT = 73 EPS = 1.2657688E+01
IT = 74 EPS = 1.2487871E+01	IT = 74 EPS = 1.2487871E+01
IT = 75 EPS = 1.2318196E+01	IT = 75 EPS = 1.2318196E+01
IT = 76 EPS = 1.2151319E+01	IT = 76 EPS = 1.2151319E+01
IT = 77 EPS = 1.1984829E+01	IT = 77 EPS = 1.1984829E+01
IT = 78 EPS = 1.1821254E+01	IT = 78 EPS = 1.1821254E+01
IT = 79 EPS = 1.1658264E+01	IT = 79 EPS = 1.1658264E+01
IT = 80 EPS = 1.1498266E+01	IT = 80 EPS = 1.1498266E+01
IT = 81 EPS = 1.1339012E+01	IT = 81 EPS = 1.1339012E+01
IT = 82 EPS = 1.1182794E+01	IT = 82 EPS = 1.1182794E+01
IT = 83 EPS = 1.1040310E+01	IT = 83 EPS = 1.1040310E+01
IT = 84 EPS = 1.0901879E+01	IT = 84 EPS = 1.0901879E+01
IT = 85 EPS = 1.0765669E+01	IT = 85 EPS = 1.0765669E+01

collapsor@collapsor-G5-5500:~/Desktop/CUDA$ ./cuda2 cpu	collapsor@collapsor-G5-5500:~/Desktop/CUDA$ ./cuda2 gpu
IT = 86 EPS = 1.0641961E+01	IT = 86 EPS = 1.0641961E+01
IT = 87 EPS = 1.0522208E+01	IT = 87 EPS = 1.0522208E+01
IT = 88 EPS = 1.0403990E+01	IT = 88 EPS = 1.0403990E+01
IT = 89 EPS = 1.0295375E+01	IT = 89 EPS = 1.0295375E+01
IT = 90 EPS = 1.0192523E+01	IT = 90 EPS = 1.0192523E+01
IT = 91 EPS = 1.0090620E+01	IT = 91 EPS = 1.0090620E+01
IT = 92 EPS = 9.9884684E+00	IT = 92 EPS = 9.9884684E+00
IT = 93 EPS = 9.8873683E+00	IT = 93 EPS = 9.8873683E+00
IT = 94 EPS = 9.7861476E+00	IT = 94 EPS = 9.7861476E+00
IT = 95 EPS = 9.6860624E+00	IT = 95 EPS = 9.6860624E+00
IT = 96 EPS = 9.5859680E+00	IT = 96 EPS = 9.5859680E+00
IT = 97 EPS = 9.4870762E+00	IT = 97 EPS = 9.4870762E+00
IT = 98 EPS = 9.3882709E+00	IT = 98 EPS = 9.3882709E+00
IT = 99 EPS = 9.2907209E+00	IT = 99 EPS = 9.2907209E+00
IT = 100 EPS = 9.1933395E+00	IT = 100 EPS = 9.1933395E+00
CPU Time (s): 573.603171	GPU Time (ms): 33936.792969
Jacobi3D Benchmark Completed.	Jacobi3D Benchmark Completed.