并行与分布式作业

第六次作业

姓名：陈琮昊
班级：人工智能与大数据
学号：18340013

一、问题描述：

(1)Start from the provided skeleton code error-test.cu that provides some convenience macros for error checking. The macros are defined in the header file error_checks_1.h. Add the missing memory allocations and copies and the kernel launch and check that your code works.

1. What happens if you try to launch kernel with too large block size? When do you catch the error if you remove the cudaDeviceSynchronize() call?

2. What happens if you try to dereference a pointer to device memory in host code?

3. What if you try to access host memory from the kernel? Remember that you can use also cuda-memcheck! If you have time, you can also check what happens if you remove all error checks and do the same tests again.

(2)In this exercise we will implement a Jacobi iteration which is a very simple finite-difference scheme. Familiarize yourself with the provided skeleton. Then implement following things:

1.Write the missing CUDA kernel sweepGPU that implements the same algorithm as the sweepCPU function. Check that

the reported averate difference is in the order of the numerical accuracy.

2.Experiment with different grid and block sizes and compare the execution times.

二、解决方案：

(1)首先对 error-test.cu 进行修改：

①核函数：

```
__global__ void vector_add(double *C, const double *A, const double *B, int N)
{
    // Add the kernel code
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    // Do not try to access past the allocated memory
    if (idx < N) {
        C[idx] = A[idx] + B[idx];
    }
}
```

②在显卡上申请内存并从主机上拷贝：

```
CUDA_CHECK( cudaMalloc((void**)&dA, sizeof(double)*N) );
CUDA_CHECK( cudaMalloc((void**)&dB, sizeof(double)*N) );
CUDA_CHECK( cudaMalloc((void**)&dC, sizeof(double)*N) );
CUDA_CHECK( cudaMemcpy(dA, hA, sizeof(double) * N, cudaMemcpyHostToDevice));
CUDA_CHECK( cudaMemcpy(dB, hB, sizeof(double) * N, cudaMemcpyHostToDevice));
```

③将运算结果从设备拷贝回主机，并释放之前在显卡上分配的内存：

```
CUDA_CHECK( cudaMemcpy(hC, dC, sizeof(double) * N, cudaMemcpyDeviceToHost));
CUDA_CHECK( cudaFree(dA));
CUDA_CHECK( cudaFree(dB));
CUDA_CHECK( cudaFree(dC));
```

对于 3 个问题在实验结果里给出答案。

(2)编写 GPU 的 sweepGPU 函数如下：

```
__global__
void sweepGPU(double *phi, const double *phiPrev, const double *source,
              double h2, int N)
{
    // #error Add here the GPU version of the update routine (see sweepCPU above)
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    int j = blockIdx.y * blockDim.y + threadIdx.y;
```

```
8.        if (i > 0 && j > 0 && i < N - 1 && j < N - 1) {
9.            int index = i + j * N;
10.           int i1 = (i - 1) + j * N;
11.           int i2 = (i + 1) + j * N;
12.           int i3 = i + (j - 1) * N;
13.           int i4 = i + (j + 1) * N;
14.           phi[index] = 0.25 * (phiPrev[i1] + phiPrev[i2] + phiPrev[i3] + phiPr
    ev[i4] - h2 * source[index]);
15.       }
16.   }
```

运行结果见实验结果一栏。

三、实验结果：

(1)在对 error-test.cu 进行修改以后，进行编译得到的结果如下：

```
0.0
2.0
6.0
12.0
20.0
30.0
42.0
56.0
72.0
90.0
110.0
132.0
156.0
182.0
210.0
240.0
272.0
306.0
342.0
380.0
```

现回答对于(1)中的 3 个问题：

1. What happens if you try to launch kernel with too large block size? When do you catch the error if you remove the cudaDeviceSynchronize() call?

在调用核函数时修改 block size 大小为 1025，编译时不会报错，但运行时会出现如下错误信息：

```
Error: vector_add kernel at error-test.cu(52): invalid configuration argument
```

将函数 cudaDeviceSynchronize()移除后，发现并没有出现报错，且对结果也没有产生影响。

2. What happens if you try to dereference a pointer to device memory in host code?

编译时没有报错，但在运行时出现段错误：

```
Segmentation fault (core dumped)
```

3. What if you try to access host memory from the kernel?

定义一个全局数组，当在核函数中访问这个全局变量时会在编译阶段报错：

```
error-test.cu(13): warning: a host variable "hb" cannot be directly written in a device function
1 error detected in the compilation of "/tmp/tmpxft_00000229_00000000-8_error-test.cpp1.ii".
```

(2)运行结果如下：

```
100 0.00972858
200 0.00479832
300 0.00316256
400 0.00234765
500 0.00186023
600 0.00153621
700 0.0013054
800 0.00113277
900 0.000998881
1000 0.000892078
1100 0.00080496
1200 0.000732594
1300 0.000671564
1400 0.000619434
1500 0.000574415
1600 0.000535167
1700 0.000500665
1800 0.000470113
CPU Jacobi: 3.41514 seconds, 1800 iterations
100 0.00972858
200 0.00479832
300 0.00316256
400 0.00234765
500 0.00186023
600 0.00153621
700 0.0013054
800 0.00113277
900 0.000998881
1000 0.000892078
1100 0.00080496
1200 0.000732594
1300 0.000671564
1400 0.000619434
```

```
1500 0.000574415
1600 0.000535167
1700 0.000500665
1800 0.000470113
GPU Jacobi: 0.018596 seconds, 1800 iterations
Average difference is 4.12709
```

修改 blocksize，然后看运行时间，结果如下：

blocksize	2	4	8	16	32
time	0.199464	0.061741	0.025676	0.022241	0.018811

随着 blocksize 的增大，运行时间不断缩短，但缩短的量越来越小。

四、遇到的问题及解决方法：

本次实验是 CUDA 编程的第一个作业，在老师给的文件夹里已经给出了代码，其实主要做的工作就是理解并读懂代码、然后进行增删与修改来实现要求的内容。通过这次实验我还了解到了 CUDA 的内存分布，从实践上体会到了 CUDA 的力量。