CSED490C Heterogeneous Parallel Computing 2023 Fall

Lab Assignment 1 Report

20232241 Dookyung Kang
2023. 10. 02

The written code is shown below.

```
#include <gputk.h>

__global__ void vecAdd(float *in1, float *in2, float *out, int len) {
  //@@ Insert code to implement vector addition here
  int idx = threadIdx.x + blockIdx.x * blockDim.x;
  if (idx < len) {
    out[idx] = in1[idx] + in2[idx];
  }
}
...
  //@@ Allocate GPU memory here
  cudaMalloc((void**)&deviceInput1, sizeof(float) * inputLength);
  cudaMalloc((void**)&deviceInput2, sizeof(float) * inputLength);
  cudaMalloc((void**)&deviceOutput, sizeof(float) * inputLength);

  gpuTKTime_stop(GPU, "Allocating GPU memory.");

  gpuTKTime_start(GPU, "Copying input memory to the GPU.");
  //@@ Copy memory to the GPU here
  cudaMemcpy(deviceInput1, hostInput1, sizeof(float) * inputLength,
     cudaMemcpyHostToDevice);
  cudaMemcpy(deviceInput2, hostInput2, sizeof(float) * inputLength,
     cudaMemcpyHostToDevice);

  gpuTKTime_stop(GPU, "Copying input memory to the GPU.");

  //@@ Initialize the grid and block dimensions here
  int blockSize = 256;
  int gridSize = (inputLength + 255) / 256;
  gpuTKTime_start(Compute, "Performing CUDA computation");
  //@@ Launch the GPU Kernel here
  vecAdd<<<gridSize, blockSize>>>(deviceInput1, deviceInput2, deviceOutput,
     inputLength);

  cudaDeviceSynchronize();
  gpuTKTime_stop(Compute, "Performing CUDA computation");

  gpuTKTime_start(Copy, "Copying output memory to the CPU");
  //@@ Copy the GPU memory back to the CPU here
  cudaMemcpy(hostOutput, deviceOutput, sizeof(float) * inputLength,
     cudaMemcpyDeviceToHost);

  gpuTKTime_stop(Copy, "Copying output memory to the CPU");
```

```
  gpuTKTime_start(GPU, "Freeing GPU Memory");
  //@@ Free the GPU memory here
  cudaFree(deviceInput1);
  cudaFree(deviceInput2);
  cudaFree(deviceOutput);
...
```

1. When the GPU kernel runs, `blockSize * gridSize` threads are executed according to the code. Since grid size per block is 25 for this assignment, entire thread size is equal to $256 \times \left\lceil \frac{\texttt{inputLength}}{256} \right\rceil$.

   (a) As the kernel code contains single FP operations per thread, the answer is equal to the number of threads.

Length	16	64	93	112	1120	9921	14000	25365	48000	96000
#FP operations	256	256	256	256	1280	9984	14080	25600	48128	96000

   (b) As the kernel code contains 2 global memory read for two operands from each vectors, the answer is twice the number of threads.

Length	16	64	93	112	1120	9921	14000	25365	48000	96000
#Global reads	512	512	512	512	2560	19968	28160	51200	96256	192000

   (c) As the kernel code contains single global memory write for result vector, the answer is equal to the number of threads.

Length	16	64	93	112	1120	9921	14000	25365	48000	96000
#Global writes	256	256	256	256	1280	9984	14080	25600	48128	96000

2. Evaluation is performed in NVIDIA RTX A5000, with CUDA 12.0 environment.

3. Elapsed times are indicated on the Table 1.

Data Length	16	64	93	112	1120
Importing data and creating memory on host (ms)	0.201849	0.438982	0.585114	0.672965	5.68956
Allocating GPU memory (ms)	0.15499	0.292911	0.14632	0.148487	0.148215
Copying input memory to the GPU (ms)	0.03406	0.046694	0.03347	0.033901	0.03294
Performing CUDA computation (ms)	0.022348	0.02477	0.0225	0.021931	0.020554
Copying output memory to the GPU (ms)	0.015048	0.016942	0.014507	0.013754	0.014905
Freeing GPU Memory (ms)	0.124746	0.159609	0.121834	0.121961	0.123701
Data Length	9921	14000	25365	48000	96000
Importing data and creating memory on host (ms)	49.4532	69.8133	126.141	239.206	478.397
Allocating GPU memory (ms)	0.156642	0.161043	0.163065	0.151954	0.16458
Copying input memory to the GPU (ms)	0.053163	0.063602	0.062271	0.081346	0.141132
Performing CUDA computation (ms)	0.021174	0.022703	0.02264	0.022969	0.023696
Copying output memory to the GPU (ms)	0.022384	0.029302	0.028408	0.039425	0.064933
Freeing GPU Memory (ms)	0.124083	0.125385	0.128492	0.126341	0.135912

Table 1: Execution times of the kernel with the input data generated by the dataset generator