1. Problem1
    a. 2 times synchronize
    b. Maximum number of “real” operations: 8; Minimum number of “real” operations: 0; Average number of “real” operations: 4.

Help vector_reduction.cu - hw2 - Visual Studio

vector_reduction.cu × vector_reduction_adv.cu

Reduction > vector_reduction.cu > NUM_ELEMENTS

```
    const unsigned int array_mem_size = sizeof( float) * num_elements;

    // allocate host memory to store the input data
    float* h_data = (float*) malloc( array_mem_size);

    // * No arguments: Randomly generate input data and compare against the
    //   host's result.
    // * One argument: Read the input data array from the given file.
    switch(argc-1)
    {
        // One Argument
        case 1:
            if(readFile(argv[1], h_data) != 1)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS E:\cs677\hw2\Reduction> nvcc -o vec vector_reduction.cu
vector_reduction.cu
   Creating library vec.lib and object vec.exp
PS E:\cs677\hw2\Reduction> ./vec
Test PASSED
device: 254095.000000  host: 254095.000000
PS E:\cs677\hw2\Reduction> ./vec data.txt
Array read successfully
Test PASSED
device: 130816.000000  host: 130816.000000
PS E:\cs677\hw2\Reduction>
```

2. Problem2
    a. I add another kernel in ‘vector_reduction_kernel.cu’ named reductionLarge() which take case of vector size greater than 512.
    b. In reductionLarge() function,

```
int temp = arr_size / 512;
int remainder = arr_size % 512;
int division = remainder == 0 ? temp : temp+1;
```

it can find out how many times this vector’s size is 512. Then, I use a for-loop to load data to threads and calculate the sub-value. After the calculation, storing them into a shared memory.

Help vector_reduction_kernel.cu - hw2 - V

vector_reduction.cu | vector_reduction_adv.cu | vector_reduction_kernel.cu ×

Reduction > vector_reduction_kernel.cu > reductionLarge(float *, int)

```
    __shared__ float partialSum[1024];
    unsigned int t = threadIdx.x;
    unsigned int start = 2*blockIdx.x*blockDim.x;
    partialSum[t] =g_data[t+start];
    partialSum[blockDim.x+t] = g_data[start+blockDim.x+t];

    for(unsigned int stride = blockDim.x/2; stride >= 1; stride >>= 1)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS E:\cs677\hw2\Reduction> nvcc -o vec2 vector_reduction_adv.cu
   Creating library vec2.lib and object vec2.exp
PS E:\cs677\hw2\Reduction> ./vec2
length of vector: 512
Test PASSED
PS E:\cs677\hw2\Reduction> ./vec2 0
length of vector: 512
Test PASSED
device: 254095.000000  host: 254095.000000
PS E:\cs677\hw2\Reduction> ./vec2 1
length of vector: 512
Test PASSED
device: 254095.000000  host: 254095.000000
PS E:\cs677\hw2\Reduction> ./vec2 0 data.txt
Array read successfully
length of vector: 512
Test PASSED
device: 130816.000000  host: 130816.000000
PS E:\cs677\hw2\Reduction> ./vec2 1 data.txt
Array read successfully
length of vector: 512
Test PASSED
device: 130816.000000  host: 130816.000000
PS E:\cs677\hw2\Reduction>
```

c.

d.

```cuda
int temp = arr_size / 512;
int remainder = arr_size % 512;
int division = remainder == 0 ? temp : temp+1;

__shared__ float partialSum[1024];
//__shared__ float* temp;
unsigned int t = threadIdx.x;
unsigned int start = 2*blockIdx.x*blockDim.x;

for(int i = 0; i < division; ++i)
{
    if(start + t < 512){
        partialSum[t] += g_data[t+start + i * 512];
    }
    if(start+t+blockDim.x < 512){
        partialSum[t+blockDim.x] += g_data[start+t+blockDim.x + i * 512];
    }
}

for(unsigned int stride = blockDim.x/2; stride >= 1; stride >>= 1)
{
    __syncthreads();
    if(t < stride )
    {
        partialSum[t] += partialSum[t+stride];
    }
}
  syncthreads();
```

```
PS E:\cs677\hw2\Reduction> nvcc -o vec2 vector_reduction_adv.cu
vector_reduction_adv.cu
   Creating library vec2.lib and object vec2.exp
PS E:\cs677\hw2\Reduction> ./vec2 1
(length of vector, kernel) --> (6000, 1)
Test PASSED
device: 3018652.000000  host: 3018652.000000
PS E:\cs677\hw2\Reduction> nvcc -o vec2 vector_reduction_adv.cu
vector_reduction_adv.cu
   Creating library vec2.lib and object vec2.exp
PS E:\cs677\hw2\Reduction> ./vec2 1
(length of vector, kernel) --> (10000, 1)
Test PASSED
device: 5032845.000000  host: 5032845.000000
PS E:\cs677\hw2\Reduction>
```

3. Problem3

a.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

GPU computation complete
CPU computation complete
Test PASSED
PS E:\cs677\hw2\MatMul> nvcc -o mat matrixmul.cu
matrixmul.cu
   Creating library mat.lib and object mat.exp
PS E:\cs677\hw2\MatMul> ./mat
GPU computation complete
CPU computation complete
Test PASSED
PS E:\cs677\hw2\MatMul>
```

4. Problem 4
    a. 1024 * 256 = 262,144
    b. Two global memory loads and one global memory store for each thread.
    c. 256 * 2 + (128 + 64 + 32 + 16 + 8 + 4 +2+1) * 3 = 1277
    d. None
    e. 128 / 2 / 2 / 2 / 2 / 2 / 2 / 2 / 2 = 0; So, there is 8 iterations of the for loop in total and 5 iterations will have branch divergence.
    f. There are 256 times to write back to the global memory. By reduction, we can store all value to thread 0 or the last thread then let that thread access the global memory, which can eliminate 1024 * 255 accesses.