Jose F. Martinez Rivera

Course: ICOM4036 – 040

Prof. Wilson Rivera

April 29, 2013

**Problem – 1:** Discussion of Implementation and Results

**Implementation:**

The program ‘Problem-1.cu’ was implemented in CUDA from NVIDIA. The most notable thing is the nested for loop in the beginning of the program. It’s purpose is to loop through the whole main function and increase the size of the array (N) and the threads (32, 1024). Every time it loops it reallocates the memory needed for the program as well as cleaning it up afterwards. To dynamically change the number of blocks needed per thread we knew we had a # of threads per block (32, 1024) and a number of N. So to get a number of blocks we could just divide N by the number of threads per block to get the number of blocks we need:

Line 178 add\_Vector<<<N/threads\_per\_block, threads\_per\_block>>>(dA,dB, dC);

The kernel parameters change the block size.

One of the issues that occurred was the conversion from double to float data type in the code because of the architecture of the GPUs. Unfortunately since float types only cover a shorter range than double the check for precision seems like a waste. Perhaps if we had GPUs that support double the problem would be resolved.

**Results:**

From the timing results (Timing-Results-Problem-1.txt) we can graph the different times vs. the size of the array (N):

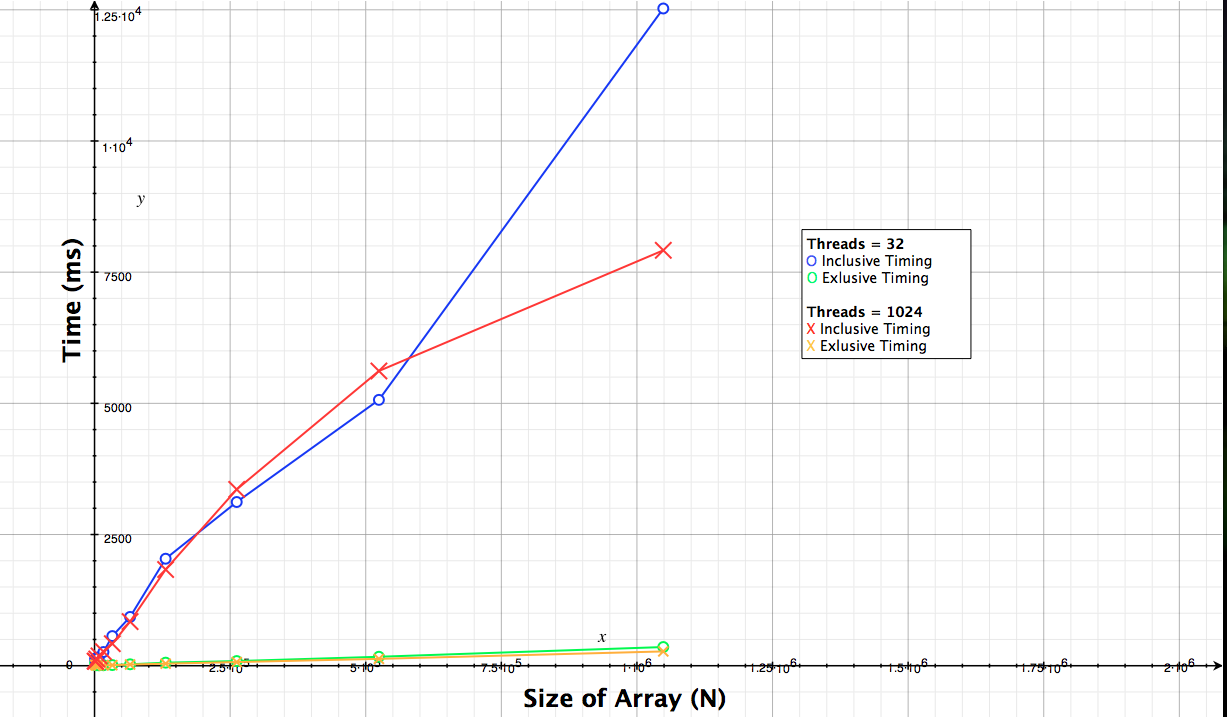


Figure 1 – Time (ms) vs. Size of Array(N)

We can clearly see that for both of the threads per block = 32 and 1024 the exclusive time is almost the same meaning the time to execute the kernel was close to equal. When we see the inclusive timing for both tests we can see that as the size of array increases they both increase linearly but when we have fewer threads (32) the time it takes to complete the task increases faster than the time it takes to complete the task with 1024 threads.

It could be a clear conclusion that executing the kernel for any number of threads is actually a pretty constant value for this test. But the time needed to complete the task of allocating the memory needed and running the kernel decreases with the number of threads we use. The results clearly show that adding more threads will help decrease the time to execute a task and proving that parallel programming is an efficient way of completing tasks.

**Data Results:**

*Threads: 32*

Value of N: 1024

Exclusive Time was: 0.052832

Inclusive Time was: 0.133600

Total Time: 0.186432

Value of N: 2048

Exclusive Time was: 0.013472

Inclusive Time was: 0.093952

Total Time: 0.107424

Value of N: 4096

Exclusive Time was: 0.009632

Inclusive Time was: 0.123712

Total Time: 0.133344

Value of N: 8192

Exclusive Time was: 0.007072

Inclusive Time was: 0.163232

Total Time: 0.170304

Value of N: 16384

Exclusive Time was: 0.008512

Inclusive Time was: 0.260480

Total Time: 0.268992

Value of N: 32768

Exclusive Time was: 0.012256

Inclusive Time was: 0.564608

Total Time: 0.576864

Value of N: 65536

Exclusive Time was: 0.026816

Inclusive Time was: 0.929088

Total Time: 0.955904

Value of N: 131072

Exclusive Time was: 0.057984

Inclusive Time was: 2.038688

Total Time: 2.096672

Value of N: 262144

Exclusive Time was: 0.088608

Inclusive Time was: 3.119520

Total Time: 3.208128

Value of N: 524288

Exclusive Time was: 0.170304

Inclusive Time was: 5.066272

Total Time: 5.236576

Value of N: 1048576

Exclusive Time was: 0.355712

Inclusive Time was: 12.524928

Total Time: 12.880640

*Threads: 1024*

Value of N: 1024

Exclusive Time was: 0.021376

Inclusive Time was: 0.087552

Total Time: 0.108928

Value of N: 2048

Exclusive Time was: 0.008032

Inclusive Time was: 0.082016

Total Time: 0.090048

Value of N: 4096

Exclusive Time was: 0.006464

Inclusive Time was: 0.127648

Total Time: 0.134112

Value of N: 8192

Exclusive Time was: 0.006464

Inclusive Time was: 0.186912

Total Time: 0.193376

Value of N: 16384

Exclusive Time was: 0.008512

Inclusive Time was: 0.267488

Total Time: 0.276000

Value of N: 32768

Exclusive Time was: 0.010688

Inclusive Time was: 0.419680

Total Time: 0.430368

Value of N: 65536

Exclusive Time was: 0.022240

Inclusive Time was: 0.817984

Total Time: 0.840224

Value of N: 131072

Exclusive Time was: 0.038816

Inclusive Time was: 1.832992

Total Time: 1.871808

Value of N: 262144

Exclusive Time was: 0.069312

Inclusive Time was: 3.359680

Total Time: 3.428992

Value of N: 524288

Exclusive Time was: 0.130464

Inclusive Time was: 5.616576

Total Time: 5.747040

Value of N: 1048576

Exclusive Time was: 0.272640

Inclusive Time was: 7.916512

Total Time: 8.189152