Cuda Exercises

Vicent Santamarta Martinez

1. hello_world.cu

In the code for this exercise we have a program that prints once a “hello world” message once from the host code.

At the beginning we have the function `__global__ void mykernel (void)`, which is the kernel function that will be executed in the GPU and is called by the host. In the main function we have the call `mykernel <<<1,1>>>()` which is the call to the cuda function. The (1,1) parameters indicate the number of blocks and threads that we are going to launch.

2. pi_par_loop.cu

Blocks	Threads	Steps	Speedup
1	256	1e7	1.755684376
1	256	5e7	1.805247426
1	256	1e8	1.778832436
1	256	5e8	2.217321396
1	512	1e7	1.806041241
1	512	5e7	1.851584196
1	512	1e8	1.853107691
1	512	5e8	2.274461746
1	1024	1e7	1.78277123
1	1024	5e7	1.84284997
1	1024	1e8	1.837530971
1	1024	5e8	2.281101704
2	256	1e7	3.433465958
2	256	5e7	3.53213048
2	256	1e8	3.547942638
2	256	5e8	4.006081104
2	512	1e7	3.407314539
2	512	5e7	3.623372078
2	512	1e8	3.652956724

2	512	5e8	4.166157246
2	1024	1e7	2.049420834
2	1024	5e7	3.66221714
2	1024	1e8	3.688157558
2	1024	5e8	4.169987202
3	256	1e7	5.171441078
3	256	5e7	5.284718513
3	256	1e8	5.374106884
3	256	5e8	5.800917149
3	512	1e7	5.294831276
3	512	5e7	5.43606472
3	512	1e8	5.440898895
3	512	5e8	5.988674641
3	1024	1e7	4.691909313
3	1024	5e7	5.152007103
3	1024	1e8	5.336522579
3	1024	5e8	5.9295187

For this exercise we have implemented a loop within each warp and then added all together with an atomicAdd function. We had to implement a supplementary device kernel to be able to handle doubles instead of floats in patan.

From the results we can see how the speed up increases as we increase the number of blocks and threads. Through exploration we’ve seen that running the code with more than 3 blocks did not resume an increasing speedup or efficiency values.