CUDA Parallel Programming Problem Set 5

R08244002 Shin-Rong Tsai

- ## Heat Diffusion with $\omega = 1$

I. Result

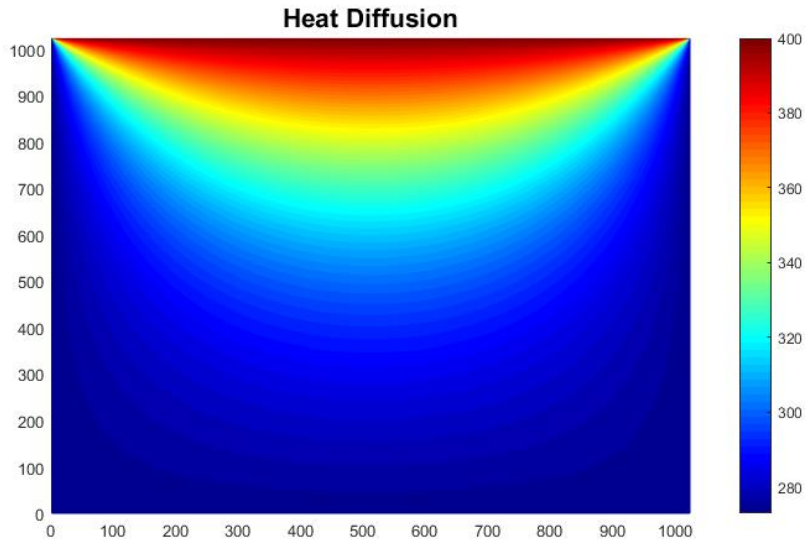
"Slightly" change the code *laplace2D_NGPU.cu* given in the course and saved it as *heat_diffusion.cu*, then compiled and run. To find the optimal block size, add a loop to loop through all the block size, saved in file *heat_diffusion_Optimize.cu*.

```
r08244002@twqcd58:~/PS5$ make
nvcc -arch=compute_52 -code=sm_52,sm_52 -O3 --compiler-options -fopenmp -c heat_diffusion_Optimize.cu -o heat_diffusion_Optimize.o
nvcc -o heat_diffusion_Optimize -arch=compute_52 -code=sm_52,sm_52 -O3 --compiler-options -fopenmp heat_diffusion_Optimize.o
r08244002@twqcd58:~/PS5$ ./heat_diffusion_Optimize
* Initial parameters:
  Enter the number of GPUs (NGx, NGy): 1 2
1 2
  * Enter the GPU ID (0/1/...): 0
0
  * Enter the GPU ID (0/1/...): 1
1
  Solve Heat Diffusion on 2D lattice with boundary conditions
  Enter the size (Nx, Ny) of the 2D lattice: 1024 1024
1024 1024
  Enter omega of the alogrithm: 1
1.000

* Allocate working space for GPUs ....
  Data input time for GPU: 9.787488 (ms)

* Compute GPU solution ....
```

Which gives the final result:



Compare total time used of 2 GPU with 1 GPU, unit in second:

Block Size / Time Used (s)	(4, 4)	(8, 8)	(16, 16)	(32, 32)
2 GPU	485.80	222.54	233.91	242.52
1 GPU	791.82	355.68	338.89	405.44
Speed Up	1.63	1.60	1.45	1.67

II. Discussion

1. The optimal block size for 2 GPU is (8,8), 1 GPU is (16,16). It is quite interesting that all the algorithm about lattice has optimal block size either be 16 or near 16.
2. The speed up rate between 2 GPU and 1 GPU is far from 2, which means effects like direct access to other GPU's memory may take up some time.
3. Compare different ω with block size (16, 16) and 2 GPU:

ω	Iteration	Total Time Used (sec)
1.5	×	×
1	1214037	233.91
0.9	1274320	178.92

Though $\omega = 0.9$ has more iterations, it uses less time. I think it is just because of how the hardware performed then. And since

$$\omega = \frac{4c^2\delta t}{\Delta^2}$$

if we fixed δt and Δ, bigger ω means bigger velocity, and faster diffusion rate. So it needs less iteration to reach equilibrium. $\omega = 1$ and $\omega = 0.9$ give the same lattice result, even though they used different iterations.

But if the diffusion rate is too big that the neighbor coordinate cannot cover, then the whole algorithm breaks down. Just like $\omega = 1.5$ case, which gives a wrong simulation.