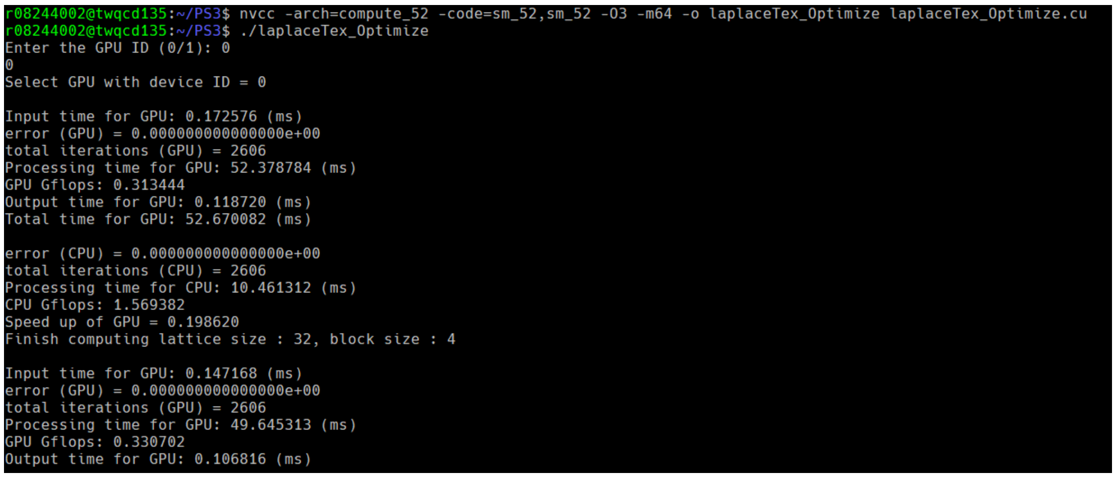
CUDA Parallel Programming Problem Set 3

R08244002 Shin-Rong Tsai

- **Compare the performances of CPU, GPU using device memory, and GPU using texture memory.**

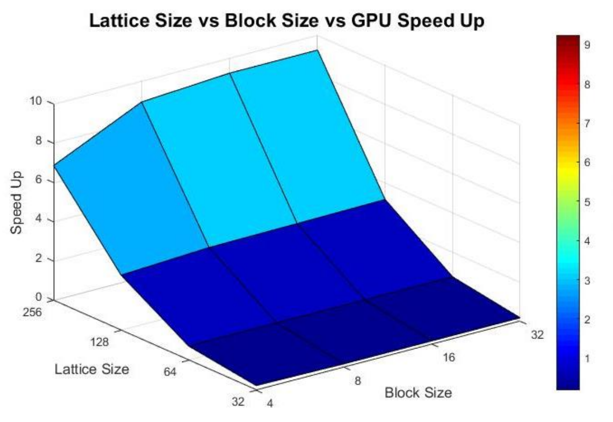
I. Result

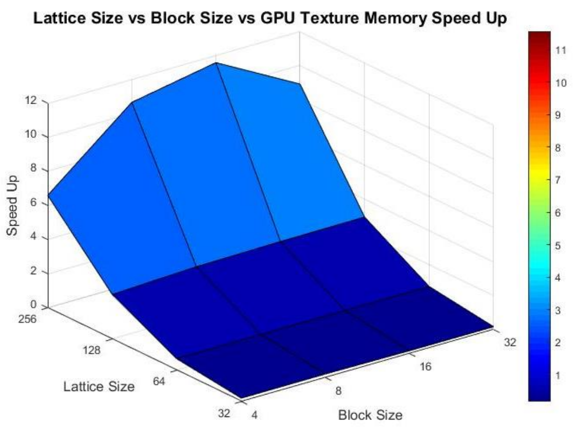
Change the code introduce in course *laplace.cu* and *laplaceTex.cu*, add a for loop to loop through all the given lattice size and block size and output each of their performances to files, then saved the code in *laplace_Optimize.cu* and *laplaceTex_Optimize.cu* respectively.



```
r08244002@twqcd135:~/PS3$ nvcc -arch=compute_52 -code=sm_52,sm_52 -O3 -m64 -o laplaceTex_Optimize laplaceTex_Optimize.cu
r08244002@twqcd135:~/PS3$ ./laplaceTex_Optimize
Enter the GPU ID (0/1): 0
0
Select GPU with device ID = 0

Input time for GPU: 0.172576 (ms)
error (GPU) = 0.000000000000000e+00
total iterations (GPU) = 2606
Processing time for GPU: 52.378784 (ms)
GPU Gflops: 0.313444
Output time for GPU: 0.118720 (ms)
Total time for GPU: 52.670082 (ms)

error (CPU) = 0.000000000000000e+00
total iterations (CPU) = 2606
Processing time for CPU: 10.461312 (ms)
CPU Gflops: 1.569382
Speed up of GPU = 0.198620
Finish computing lattice size : 32, block size : 4

Input time for GPU: 0.147168 (ms)
error (GPU) = 0.000000000000000e+00
total iterations (GPU) = 2606
Processing time for GPU: 49.645313 (ms)
GPU Gflops: 0.330702
Output time for GPU: 0.106816 (ms)
```

Since in the separate run of GPU and GPU with texture memory, the performances of CPU are the same, we can directly compare the performances measured by time in these two different run.

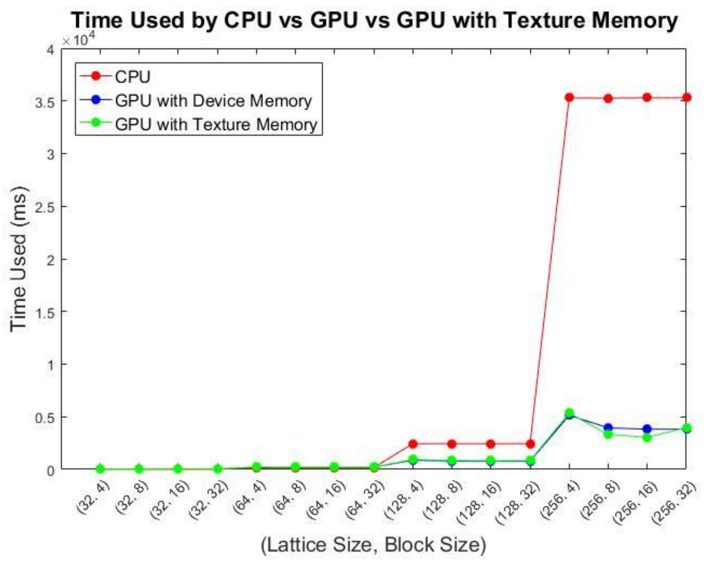




II. Discussion

1. We can see that only if the computational process is really intense, we will need GPU to speed up, no matter if we use texture memory or not.
2. In the below figure, the lattice size 128 and 256 separate each of their performances. In lattice size 128, line- (CPU Time Used) separates from GPU, since the computation is intensive. In lattice size 256, two

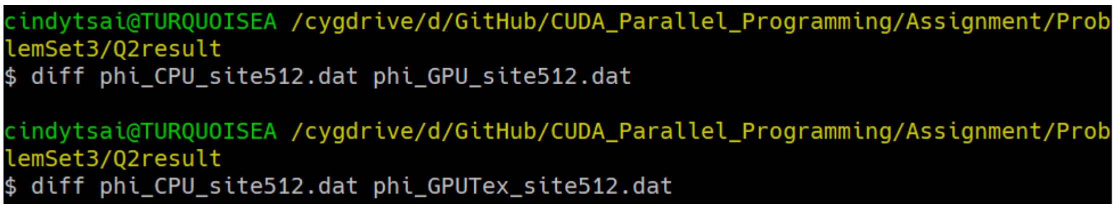
GPU lines separate, probably because the time difference of reading from device memory and reading from texture memory are amplified, due to the big amount of site should be compute. And we can see that there also exists an optimize block size when using GPU with texture memory. The optimal block size is near (16, 16).



- **Solve the Laplace Equation on the 512x512 square lattice.**

I. Result

Change the code from the previous question, and saved them in file *lattice_site_CPU.cu*, *lattice_site512_GPU.cu*, and *lattice_site_GPUTex.cu* respectively. Then run *diff* command to compare their final result:

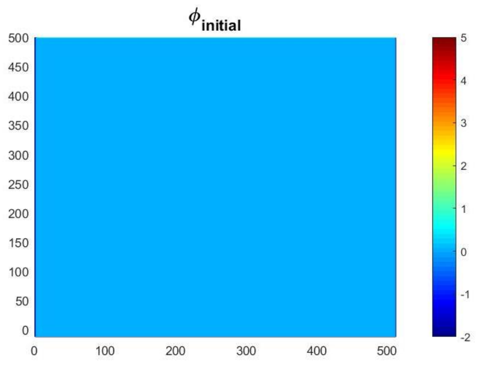


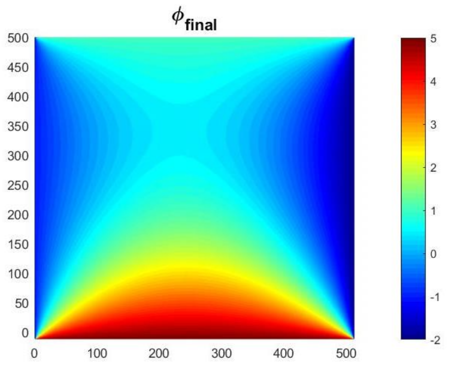
```
cindytsai@TURQUOISEA /cygdrive/d/GitHub/CUDA_Parallel_Programming/Assignment/Prob
lemSet3/Q2result
$ diff phi_CPU_site512.dat phi_GPU_site512.dat

cindytsai@TURQUOISEA /cygdrive/d/GitHub/CUDA_Parallel_Programming/Assignment/Prob
lemSet3/Q2result
$ diff phi_CPU_site512.dat phi_GPUTex_site512.dat
```

It has no output, so three of their results are the same.

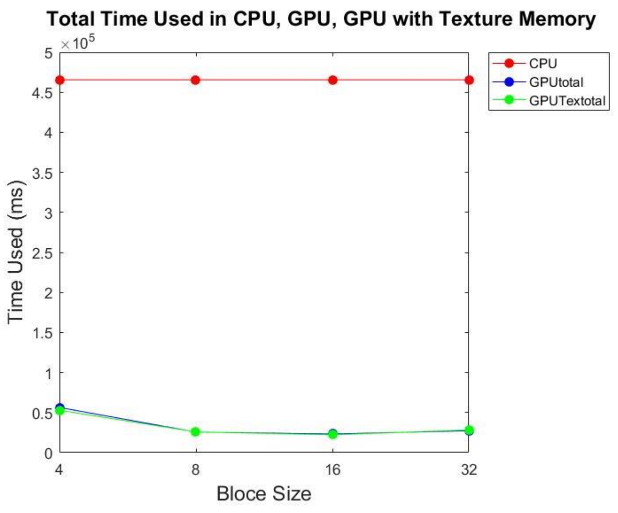
II. Discussion

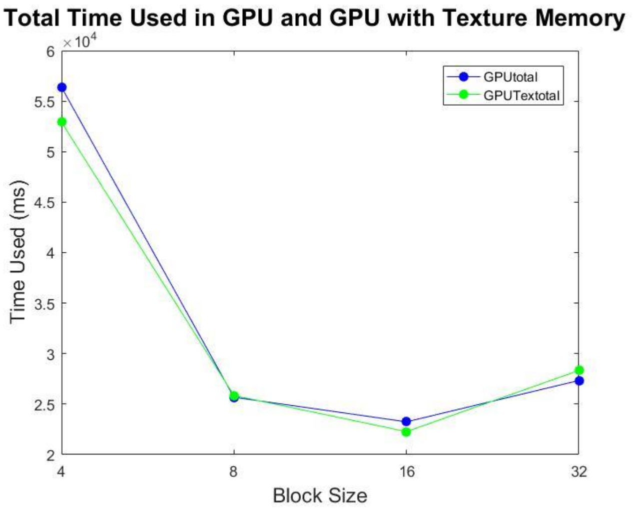
1. $\phi_{initial}$ and ϕ_{final} :





2. Time Used in CPU, GPU, and GPU with texture memory respectively are:



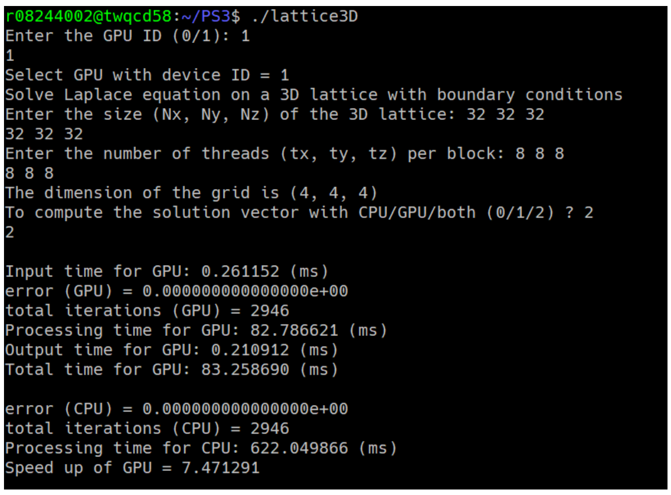


The optimal block size in both GPU and GPU with texture memory are 16, which is the same. Since whether using texture memory or not only makes getting data faster, it has nothing to do with distributing the GPU resource like streaming multiprocessors, memory space, etc…

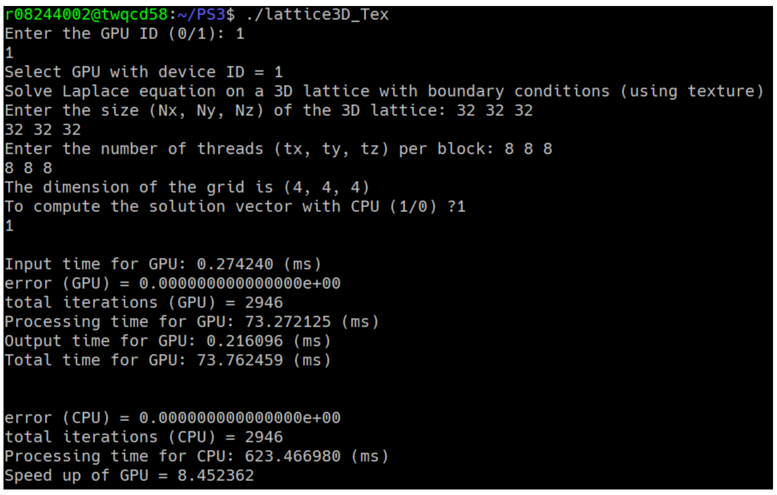
- **Develop a CUDA code for solving the Laplace Equation on 3D lattice.**

I. Result

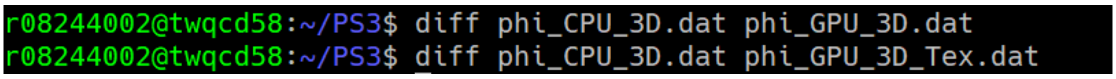
Save the code in *lattice3D.cu* and *lattice3D_Tex.cu*. Compare the result using *diff*, it gives no output, which means all of their results are the same.



```
r08244002@twqcd58:~/PS3$ ./lattice3D
Enter the GPU ID (0/1): 1
1
Select GPU with device ID = 1
Solve Laplace equation on a 3D lattice with boundary conditions
Enter the size (Nx, Ny, Nz) of the 3D lattice: 32 32 32
32 32 32
Enter the number of threads (tx, ty, tz) per block: 8 8 8
8 8 8
The dimension of the grid is (4, 4, 4)
To compute the solution vector with CPU/GPU/both (0/1/2) ? 2
2

Input time for GPU: 0.261152 (ms)
error (GPU) = 0.000000000000000e+00
total iterations (GPU) = 2946
Processing time for GPU: 82.786621 (ms)
Output time for GPU: 0.210912 (ms)
Total time for GPU: 83.258690 (ms)

error (CPU) = 0.000000000000000e+00
total iterations (CPU) = 2946
Processing time for CPU: 622.049866 (ms)
Speed up of GPU = 7.471291
```



```
r08244002@twqcd58:~/PS3$ ./lattice3D_Tex
Enter the GPU ID (0/1): 1
1
Select GPU with device ID = 1
Solve Laplace equation on a 3D lattice with boundary conditions (using texture)
Enter the size (Nx, Ny, Nz) of the 3D lattice: 32 32 32
32 32 32
Enter the number of threads (tx, ty, tz) per block: 8 8 8
8 8 8
The dimension of the grid is (4, 4, 4)
To compute the solution vector with CPU (1/0) ?1
1

Input time for GPU: 0.274240 (ms)
error (GPU) = 0.000000000000000e+00
total iterations (GPU) = 2946
Processing time for GPU: 73.272125 (ms)
Output time for GPU: 0.216096 (ms)
Total time for GPU: 73.762459 (ms)


error (CPU) = 0.000000000000000e+00
total iterations (CPU) = 2946
Processing time for CPU: 623.466980 (ms)
Speed up of GPU = 8.452362
```



```
r08244002@twqcd58:~/PS3$ diff phi_CPU_3D.dat phi_GPU_3D.dat
r08244002@twqcd58:~/PS3$ diff phi_CPU_3D.dat phi_GPU_3D_Tex.dat
```

II. Discussion

1. Implementation of the code:

   Since we are now in 3D,

$$\nabla^2\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2} = 0$$

$$\rightarrow \frac{\phi(x+a,y,z)+\phi(x-a,y,z)-2\phi(x,y,z)}{a^2} + \frac{\phi(x,y+a,z)+\phi(x,y-a,z)-2\phi(x,y,z)}{a^2} + \frac{\phi(x,y,z+a)+\phi(x,y,z-a)-2\phi(x,y,z)}{a^2} = 0$$

$$\Rightarrow \phi(x,y,z) = \frac{1}{6}\cdot[\phi(x+a,y,z)+\phi(x-a,y,z)+\phi(x,y+a,z)+\phi(x,y-a,z) + \phi(x,y,z+a)+\phi(x,y,z-a)]$$

2. From the first problem, we know that more lattice means more reading from the data, which amplifies the time difference between reading from device memory and from texture memory.

   We can see that the time spent by GPU with texture memory are generally less than GPU with device memory only, but the speed up rate between these two does not grow with the lattice size. The problem is that we can't guarantee the resource of the node are the same in all runs.

Lattice Size (x direction)	Block Size (x direction)	GPU Total (ms)	GPU Texture Total (ms)	$\frac{\text{GPU}}{\text{GPU Texture}}$
32	8	83.25869	73.762459	1.128740705
64	8	760.494873	759.118286	1.001813403
128	8	17022.71289	16368.66016	1.039957622
256	8	503148.0313	455896.1453	1.103646162