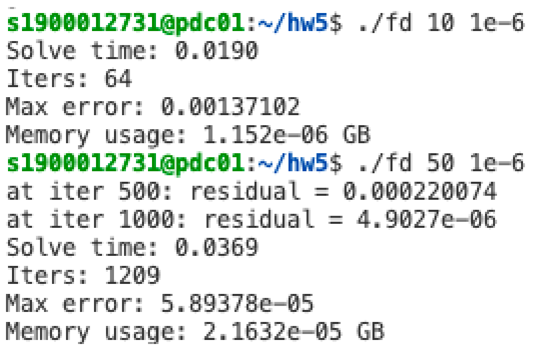
使用CUDA并行加速有限差分法求解二维拉普拉斯方程

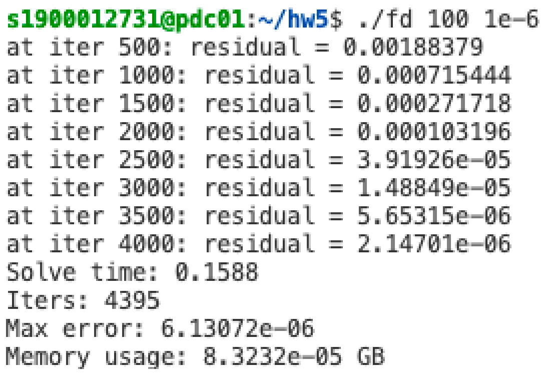
1900012731 王一鸣

测试并行版本的可扩展性

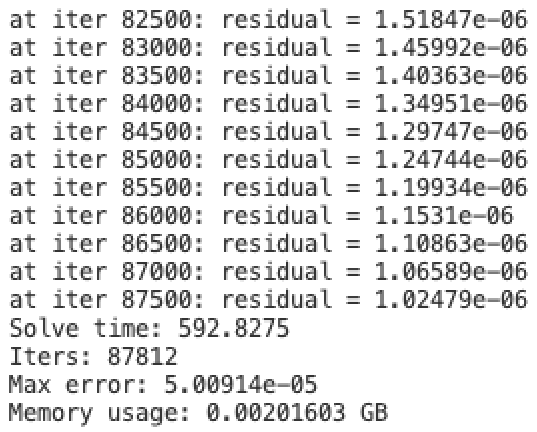
使用OpenMP作业中提供的CycleTimer代码计算运行时间。



```
s1900012731@pdc01:~/hw5$ ./fd 10 1e-6
Solve time: 0.0190
Iters: 64
Max error: 0.00137102
Memory usage: 1.152e-06 GB
s1900012731@pdc01:~/hw5$ ./fd 50 1e-6
at iter 500: residual = 0.000220074
at iter 1000: residual = 4.9027e-06
Solve time: 0.0369
Iters: 1209
Max error: 5.89378e-05
Memory usage: 2.1632e-05 GB
```



```
s1900012731@pdc01:~/hw5$ ./fd 100 1e-6
at iter 500: residual = 0.00188379
at iter 1000: residual = 0.000715444
at iter 1500: residual = 0.000271718
at iter 2000: residual = 0.000103196
at iter 2500: residual = 3.91926e-05
at iter 3000: residual = 1.48849e-05
at iter 3500: residual = 5.65315e-06
at iter 4000: residual = 2.14701e-06
Solve time: 0.1588
Iters: 4395
Max error: 6.13072e-06
Memory usage: 8.3232e-05 GB
```



```
at iter 82500: residual = 1.51847e-06
at iter 83000: residual = 1.45992e-06
at iter 83500: residual = 1.40363e-06
at iter 84000: residual = 1.34951e-06
at iter 84500: residual = 1.29747e-06
at iter 85000: residual = 1.24744e-06
at iter 85500: residual = 1.19934e-06
at iter 86000: residual = 1.1531e-06
at iter 86500: residual = 1.10863e-06
at iter 87000: residual = 1.06589e-06
at iter 87500: residual = 1.02479e-06
Solve time: 592.8275
Iters: 87812
Max error: 5.00914e-05
Memory usage: 0.00201603 GB
```

通过运行串行版本的代码，发现N=500时运行时间便已经难以接受了(590+ 秒)。

观察Memory Usage发现，即使N=500的情况下也只使用了2MB左右的内存，远小于服务器的运行内存，说明是运行时间限制了程序进行更大规模的运算。

利用CUDA在GPU上加速

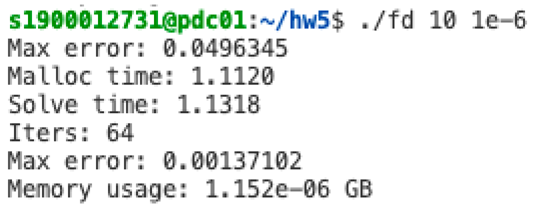
代码思路：

核函数：进行有限差分法的核心逻辑，不在循环中处理，而是每个thread处理自己的一部分。

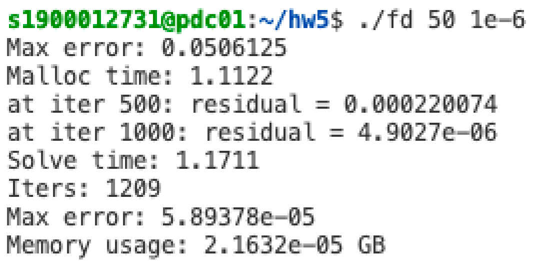
thread (i, j)需要访问的数据为u(i, j),u(i-1, j), u(i+1, j), u(i, j-1), u(i, j+1).

定义二维的block，每个block里装有16*16个thread 。注意边界thread的处理，坐标不在[1, N] $\times$ [1, N]的thread显式停止执行任务，否则会影响正确性。

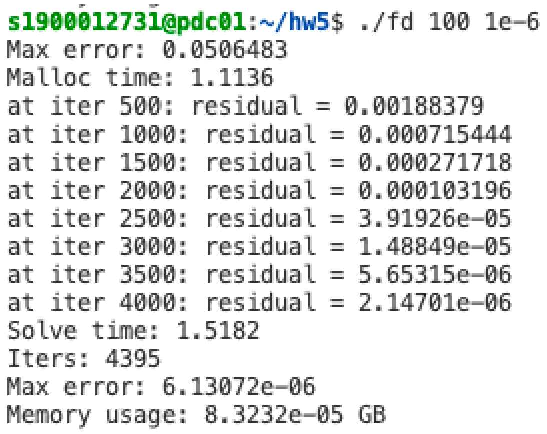
加速后测试结果如下：



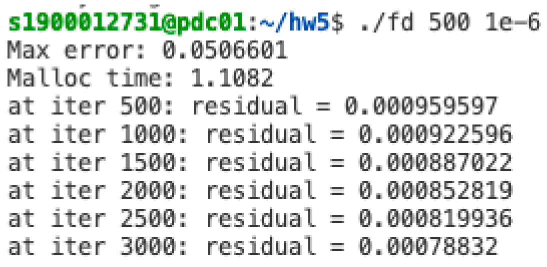
```
s1900012731@pdc01:~/hw5$ ./fd 10 1e-6
Max error: 0.0496345
Malloc time: 1.1120
Solve time: 1.1318
Iters: 64
Max error: 0.00137102
Memory usage: 1.152e-06 GB
```



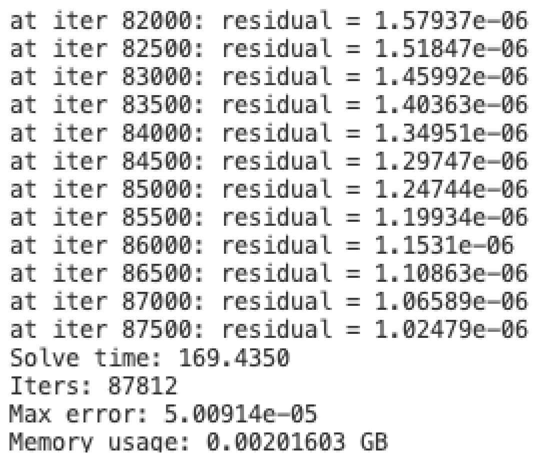
```
s1900012731@pdc01:~/hw5$ ./fd 50 1e-6
Max error: 0.0506125
Malloc time: 1.1122
at iter 500: residual = 0.000220074
at iter 1000: residual = 4.9027e-06
Solve time: 1.1711
Iters: 1209
Max error: 5.89378e-05
Memory usage: 2.1632e-05 GB
```



```
s1900012731@pdc01:~/hw5$ ./fd 100 1e-6
Max error: 0.0506483
Malloc time: 1.1136
at iter 500: residual = 0.00188379
at iter 1000: residual = 0.000715444
at iter 1500: residual = 0.000271718
at iter 2000: residual = 0.000103196
at iter 2500: residual = 3.91926e-05
at iter 3000: residual = 1.48849e-05
at iter 3500: residual = 5.65315e-06
at iter 4000: residual = 2.14701e-06
Solve time: 1.5182
Iters: 4395
Max error: 6.13072e-06
Memory usage: 8.3232e-05 GB
```



```
s1900012731@pdc01:~/hw5$ ./fd 500 1e-6
Max error: 0.0506601
Malloc time: 1.1082
at iter 500: residual = 0.000959597
at iter 1000: residual = 0.000922596
at iter 1500: residual = 0.000887022
at iter 2000: residual = 0.000852819
at iter 2500: residual = 0.000819936
at iter 3000: residual = 0.00078832
```



```
at iter 82000: residual = 1.57937e-06
at iter 82500: residual = 1.51847e-06
at iter 83000: residual = 1.45992e-06
at iter 83500: residual = 1.40363e-06
at iter 84000: residual = 1.34951e-06
at iter 84500: residual = 1.29747e-06
at iter 85000: residual = 1.24744e-06
at iter 85500: residual = 1.19934e-06
at iter 86000: residual = 1.1531e-06
at iter 86500: residual = 1.10863e-06
at iter 87000: residual = 1.06589e-06
at iter 87500: residual = 1.02479e-06
Solve time: 169.4350
Iters: 87812
Max error: 5.00914e-05
Memory usage: 0.00201603 GB
```

可以看到，在规模较小时优化效果反而不好，这是因为进行 `cudaMalloc()` 的时间占了大部分的处理时间。在问题规模逐渐变大后，并行加速的效果开始显现，在N=500时总运行时间缩短了大约75%。