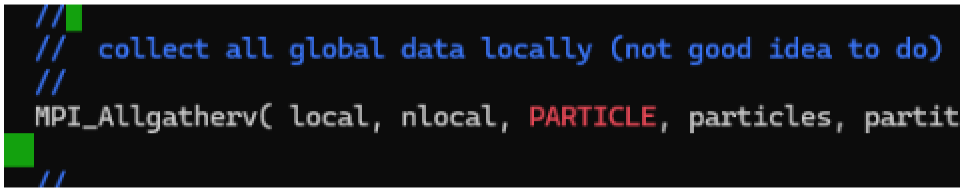
hw2 HPC - Cameron Stiver

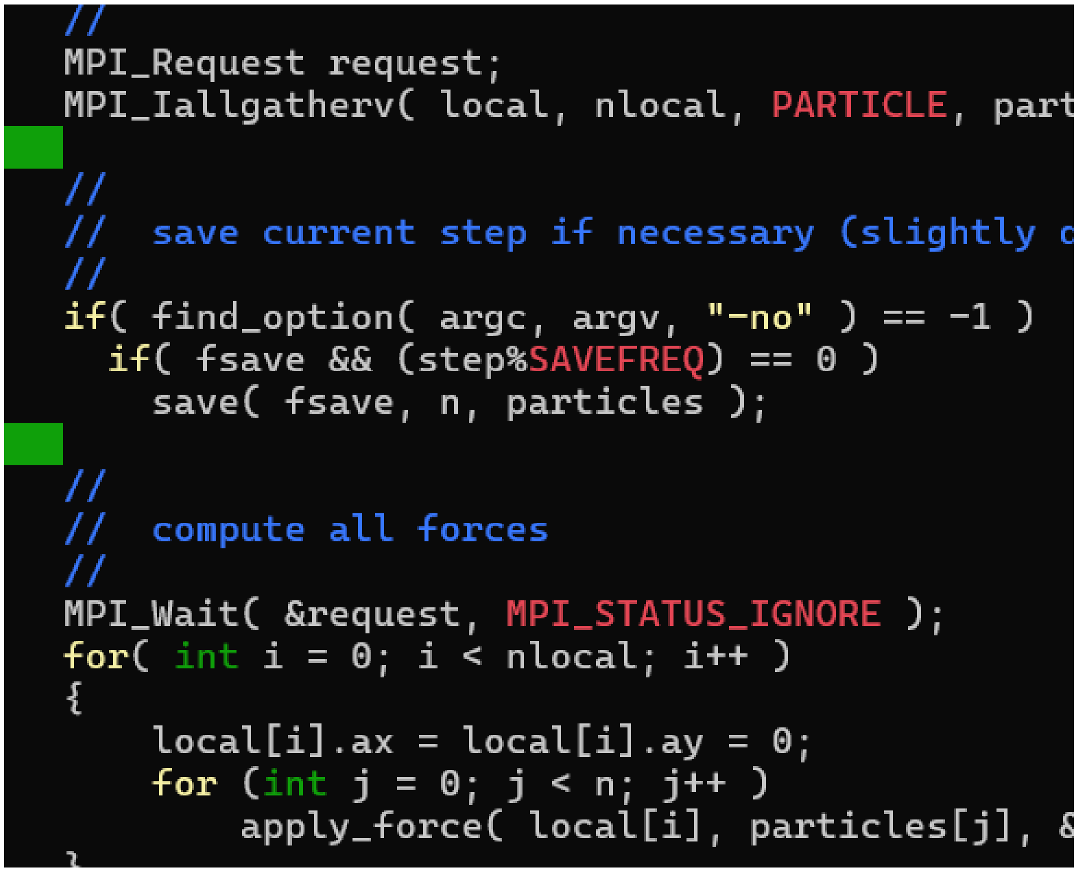
I tried a few things with openMP and actually somehow got it to run slower, so I mostly left openMP alone. It was already one of the best, I actually found the best performance increase in changing things around it like the apply force method called by everyone in the common.cpp. This isn't necessarily specific to one of the four types, so it might not be what this assignment was about, but I figured it would be a good thing to maximize efficiency to make small improvements to the methods that are called a lot like applyforce.

In MPI I moved the order of a couple things and tried to eliminate some of the calls to gather all, which can slow things down, but I learned it was not easy for me to get it working with alternatives.



```
//
//  collect all global data locally (not good idea to do)
//
MPI_Allgatherv( local, nlocal, PARTICLE, particles, partit
```

Like this comment mentions, I think alternatives for this allgatherv would greatly help. I tried switching out this call for a gather neighbor version that seems ideal but I couldn't get it to work. I then found a non-blocking version where I could call it then await it later on like how most async focused programming works, but I didn't know the program well enough to fully utilize this.

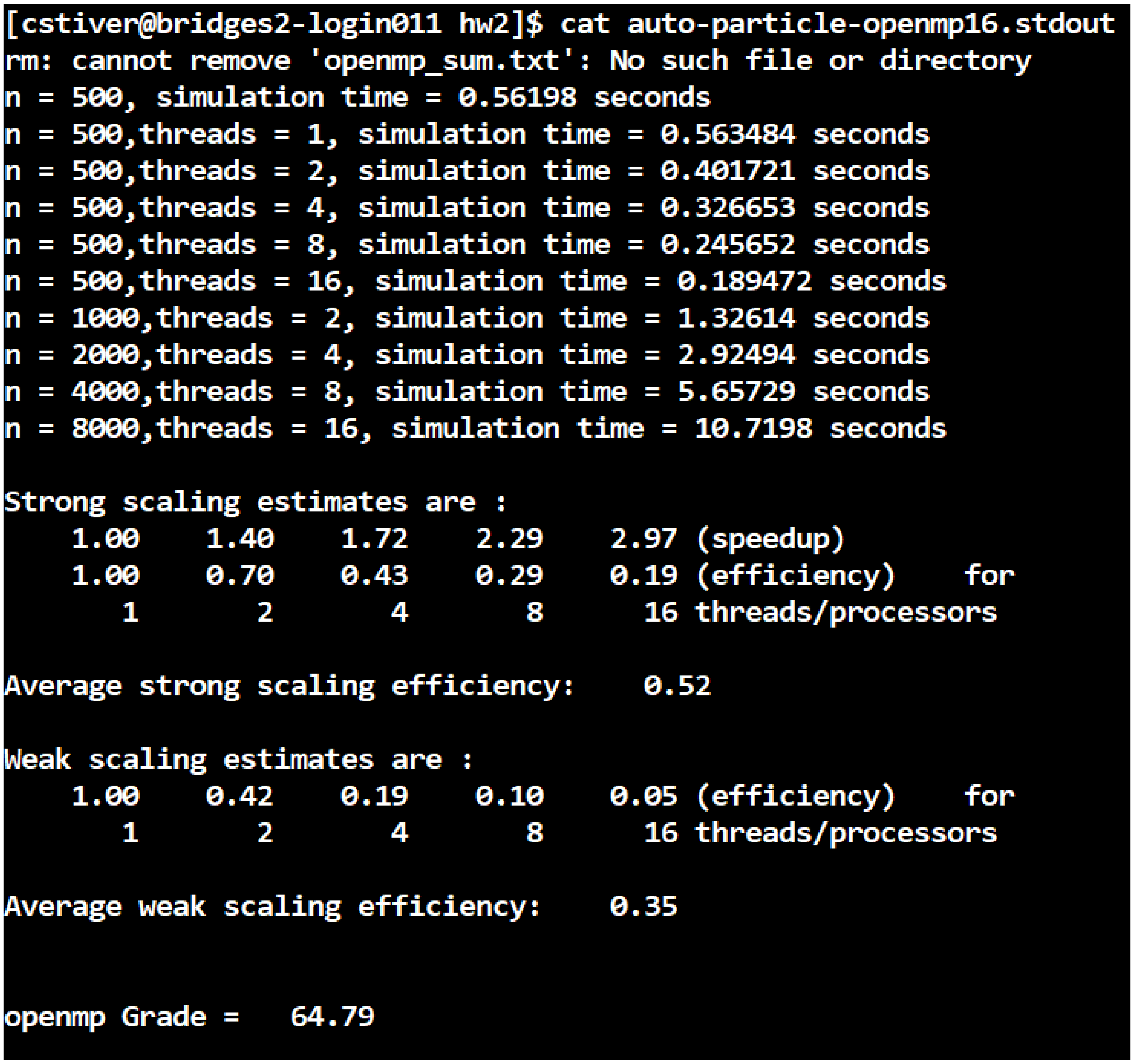


```
//
MPI_Request request;
MPI_Iallgatherv( local, nlocal, PARTICLE, part

//
//  save current step if necessary (slightly o
//
if( find_option( argc, argv, "-no" ) == -1 )
  if( fsave && (step%SAVEFREQ) == 0 )
    save( fsave, n, particles );

//
//  compute all forces
//
MPI_Wait( &request, MPI_STATUS_IGNORE );
for( int i = 0; i < nlocal; i++ )
{
    local[i].ax = local[i].ay = 0;
    for (int j = 0; j < n; j++ )
        apply_force( local[i], particles[j], &
```

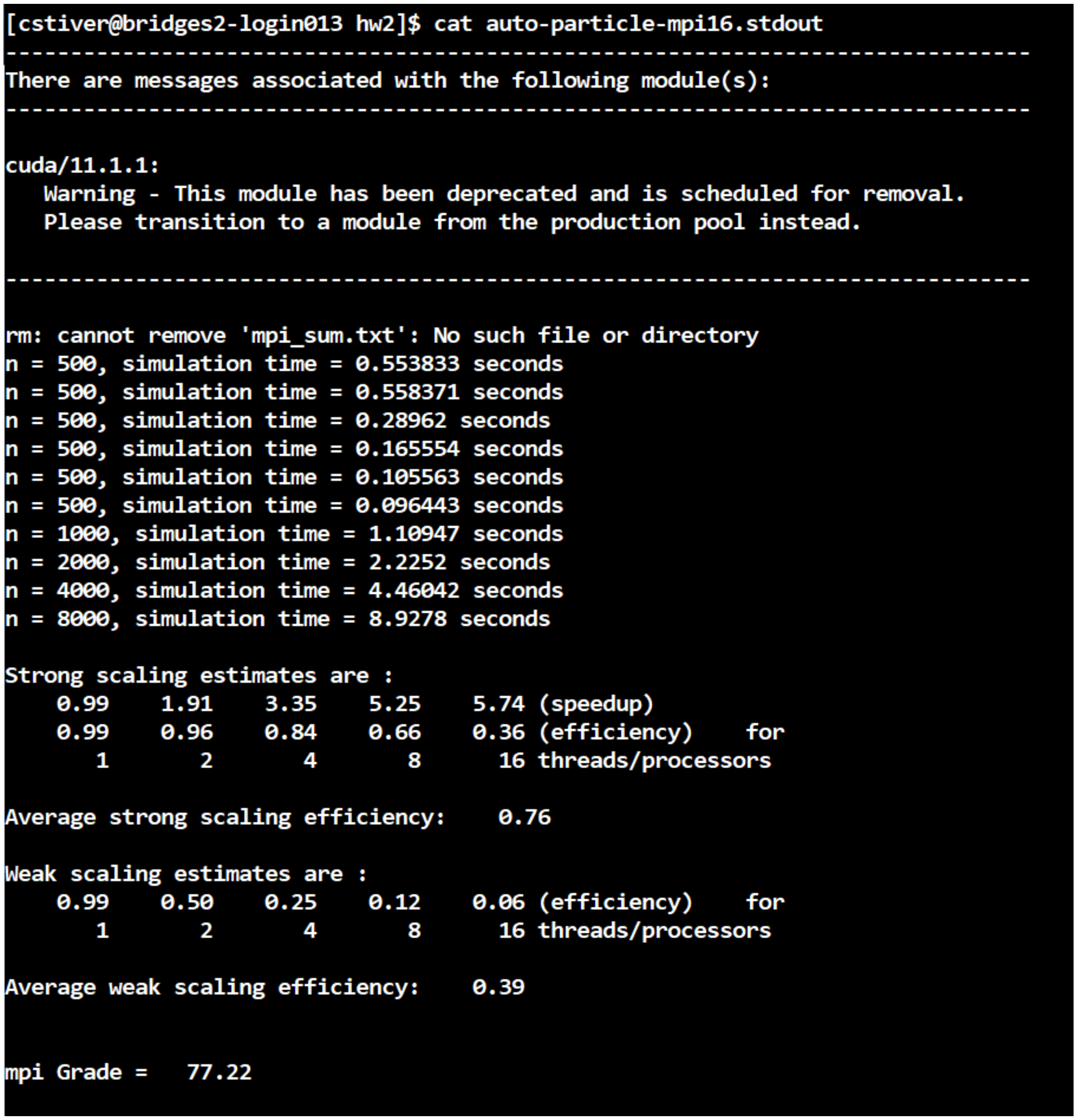
This might not actually even do anything beneficial. Ideally you would spread things out enough to where you can make async calls, do some other things that don't rely on those results, then make a Wait call when you actually need the results, which would be done in the background while you perform something else on the main thread. I know what I WANT to happen but learning all the individual interfaces/strategies has proven difficult.

The pthreads one I figured changing the number of threads and particles per thread would have an impact but I didn't really see as much of a difference as I expected. This was mostly from the launch arguments which are in how you call the procedure when batching it. The pthreads thread routine is probably something that could be re-written but I wasted too much time messing with it. The threads are called to wait multiple times. The thread routine could pretty much just run a cleaned up version of the serial methods where you do things like loop unrolling. Saving time in the routine would save each thread quite some time, but loop unrolling is something I already did elsewhere so I tried different approaches. There's also some give-and-take with the particles per threads kind of like with the block size from the previous homework. Spinning up new threads is a semi expensive operation, so there's some sweet spot where you don't spin up threads for everything but also don't run everything on one thread. You can also spin up a couple threads and use them multiple separate times rather than spinning up many threads just to run one specific operation like how the pthreads currently run a main on one thread that does many different things, then run a subroutine on many separated threads which end up having to wait on each other. I guess what I'm suggesting is actually more like how openMP does it, but I don't want to just force the other files to just run openMP. I tried to stay true to the focus of the assignment and just make each file better at what it was designed for, but I think I wasted a ton of time going down rabbit holes and ending up in dead ends.

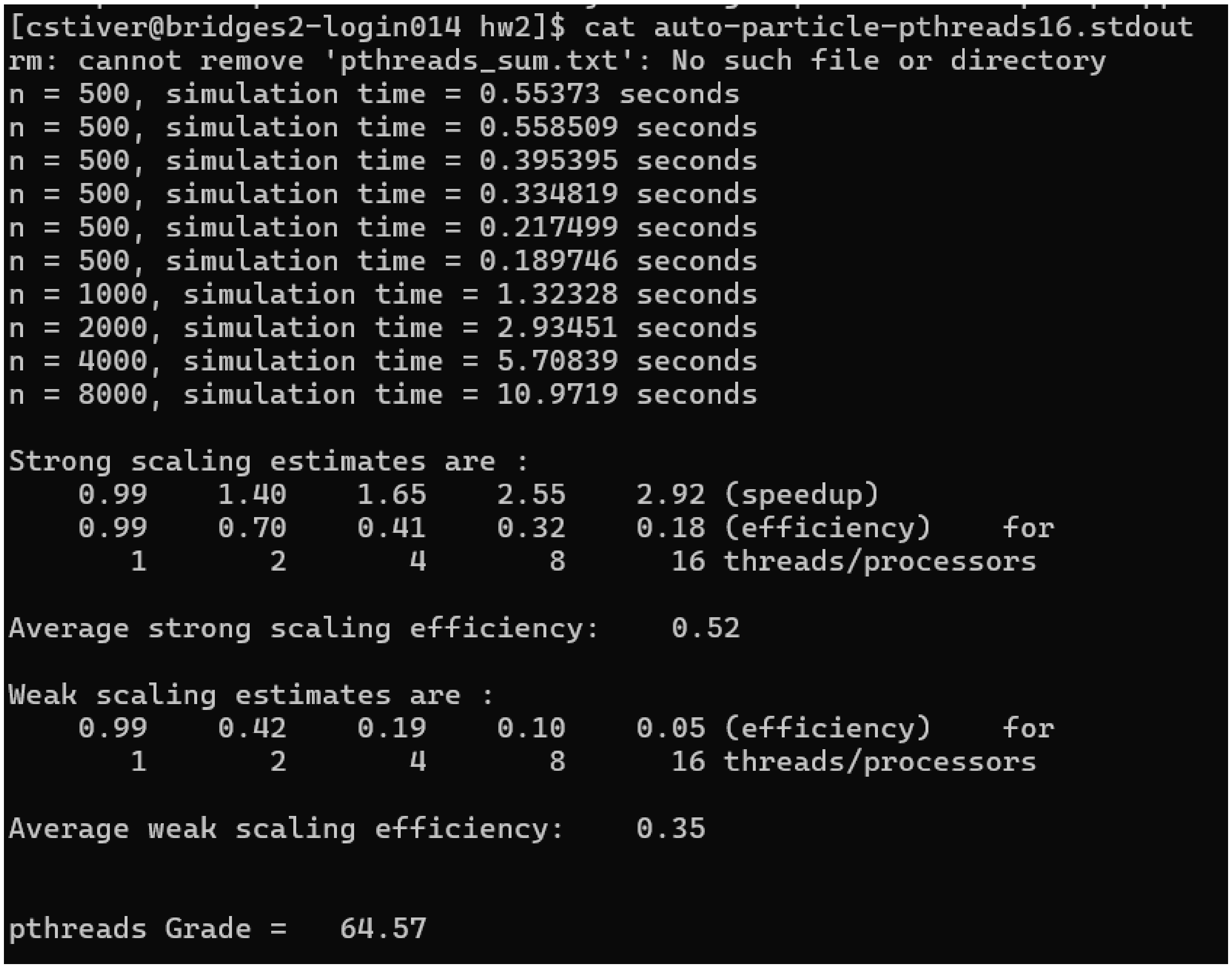
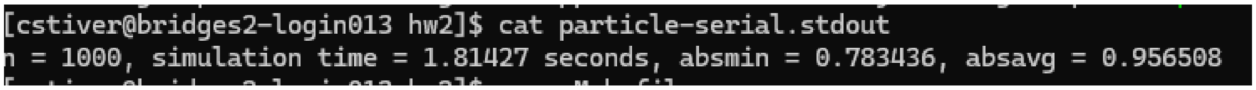
before changes:

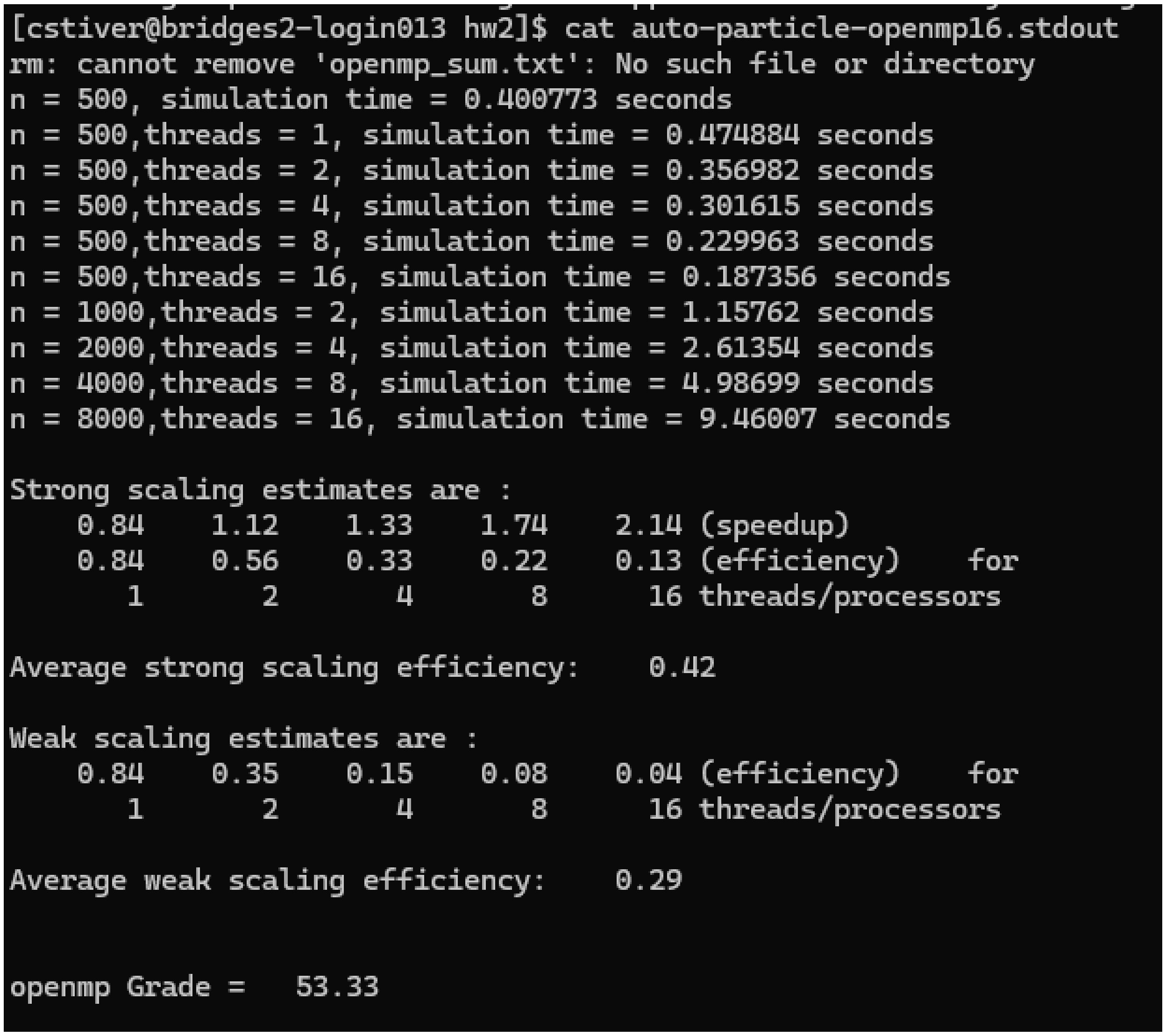


```
[cstiver@bridges2-login011 hw2]$ cat auto-particle-openmp16.stdout
rm: cannot remove 'openmp_sum.txt': No such file or directory
n = 500, simulation time = 0.56198 seconds
n = 500,threads = 1, simulation time = 0.563484 seconds
n = 500,threads = 2, simulation time = 0.401721 seconds
n = 500,threads = 4, simulation time = 0.326653 seconds
n = 500,threads = 8, simulation time = 0.245652 seconds
n = 500,threads = 16, simulation time = 0.189472 seconds
n = 1000,threads = 2, simulation time = 1.32614 seconds
n = 2000,threads = 4, simulation time = 2.92494 seconds
n = 4000,threads = 8, simulation time = 5.65729 seconds
n = 8000,threads = 16, simulation time = 10.7198 seconds

Strong scaling estimates are :
    1.00    1.40    1.72    2.29    2.97 (speedup)
    1.00    0.70    0.43    0.29    0.19 (efficiency)    for
       1       2       4       8      16 threads/processors

Average strong scaling efficiency:    0.52

Weak scaling estimates are :
    1.00    0.42    0.19    0.10    0.05 (efficiency)    for
       1       2       4       8      16 threads/processors

Average weak scaling efficiency:    0.35


openmp Grade =   64.79
```

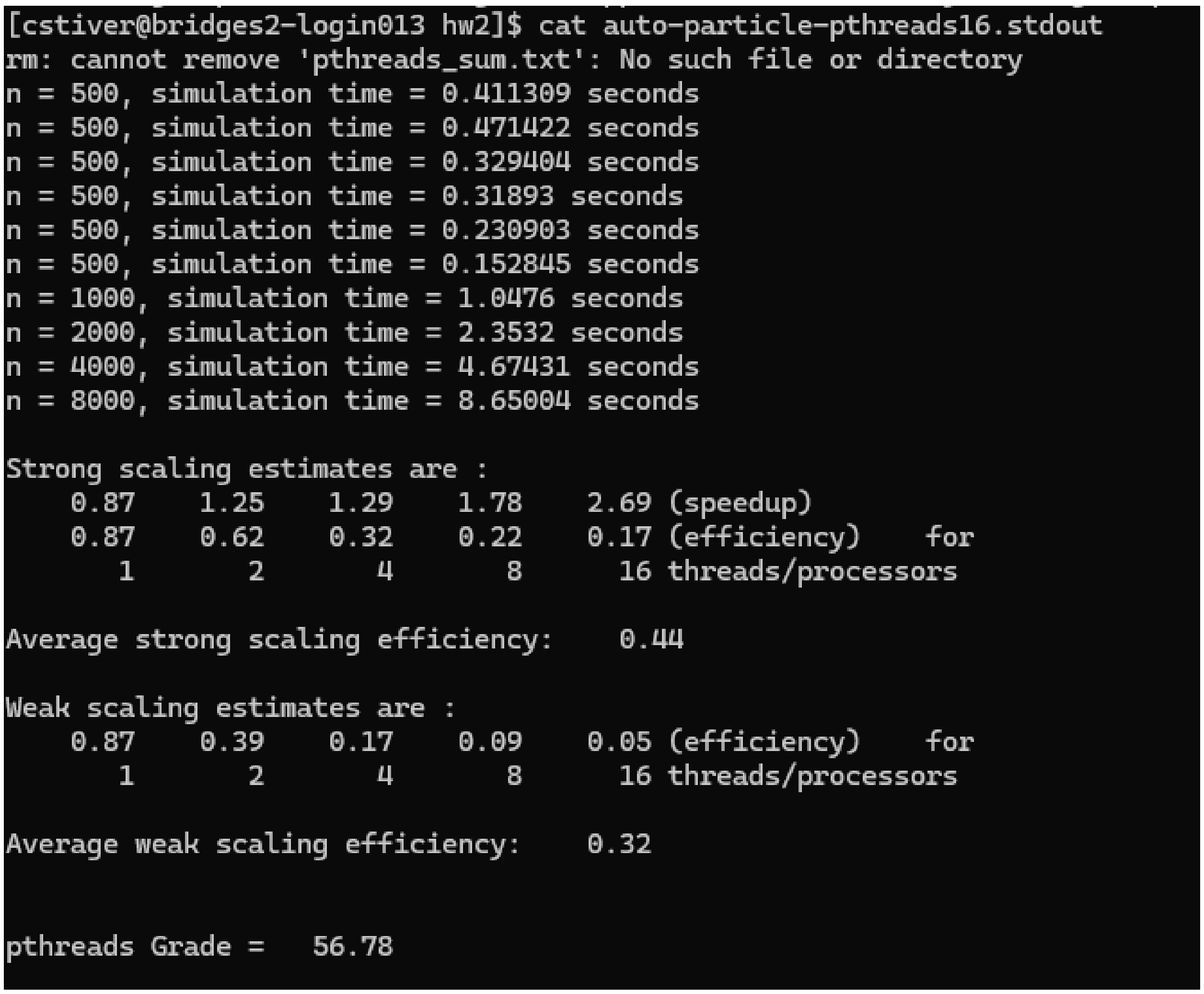


```
[cstiver@bridges2-login013 hw2]$ cat auto-particle-mpi16.stdout
-------------------------------------------------------------------------------
There are messages associated with the following module(s):
-------------------------------------------------------------------------------

cuda/11.1.1:
   Warning - This module has been deprecated and is scheduled for removal.
   Please transition to a module from the production pool instead.

-------------------------------------------------------------------------------

rm: cannot remove 'mpi_sum.txt': No such file or directory
n = 500, simulation time = 0.553833 seconds
n = 500, simulation time = 0.558371 seconds
n = 500, simulation time = 0.28962 seconds
n = 500, simulation time = 0.165554 seconds
n = 500, simulation time = 0.105563 seconds
n = 500, simulation time = 0.096443 seconds
n = 1000, simulation time = 1.10947 seconds
n = 2000, simulation time = 2.2252 seconds
n = 4000, simulation time = 4.46042 seconds
n = 8000, simulation time = 8.9278 seconds

Strong scaling estimates are :
    0.99    1.91    3.35    5.25    5.74 (speedup)
    0.99    0.96    0.84    0.66    0.36 (efficiency)    for
       1       2       4       8      16 threads/processors

Average strong scaling efficiency:    0.76

Weak scaling estimates are :
    0.99    0.50    0.25    0.12    0.06 (efficiency)    for
       1       2       4       8      16 threads/processors

Average weak scaling efficiency:    0.39


mpi Grade =   77.22
```

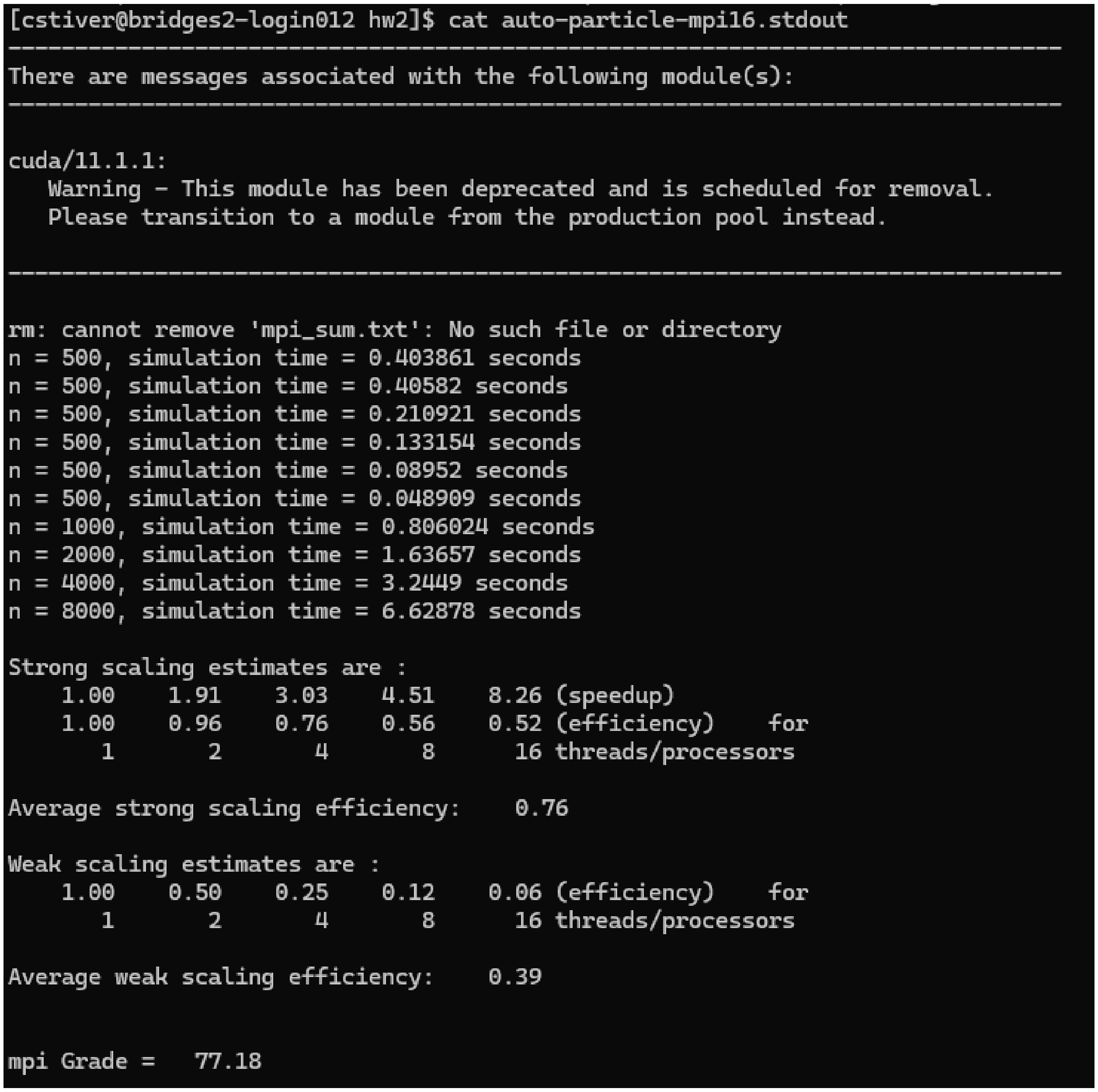

```
[cstiver@bridges2-login014 hw2]$ cat auto-particle-pthreads16.stdout
rm: cannot remove 'pthreads_sum.txt': No such file or directory
n = 500, simulation time = 0.55373 seconds
n = 500, simulation time = 0.558509 seconds
n = 500, simulation time = 0.395395 seconds
n = 500, simulation time = 0.334819 seconds
n = 500, simulation time = 0.217499 seconds
n = 500, simulation time = 0.189746 seconds
n = 1000, simulation time = 1.32328 seconds
n = 2000, simulation time = 2.93451 seconds
n = 4000, simulation time = 5.70839 seconds
n = 8000, simulation time = 10.9719 seconds

Strong scaling estimates are :
    0.99    1.40    1.65    2.55    2.92 (speedup)
    0.99    0.70    0.41    0.32    0.18 (efficiency)    for
       1       2       4       8      16 threads/processors

Average strong scaling efficiency:    0.52

Weak scaling estimates are :
    0.99    0.42    0.19    0.10    0.05 (efficiency)    for
       1       2       4       8      16 threads/processors

Average weak scaling efficiency:    0.35


pthreads Grade =   64.57
```

```
[cstiver@bridges2-login013 hw2]$ cat particle-serial.stdout
n = 1000, simulation time = 1.81427 seconds, absmin = 0.783436, absavg = 0.956508
```
after changes below, either grade went down or sim times went down (but not necessarily grade??)



```
[cstiver@bridges2-login013 hw2]$ cat auto-particle-openmp16.stdout
rm: cannot remove 'openmp_sum.txt': No such file or directory
n = 500, simulation time = 0.400773 seconds
n = 500,threads = 1, simulation time = 0.474884 seconds
n = 500,threads = 2, simulation time = 0.356982 seconds
n = 500,threads = 4, simulation time = 0.301615 seconds
n = 500,threads = 8, simulation time = 0.229963 seconds
n = 500,threads = 16, simulation time = 0.187356 seconds
n = 1000,threads = 2, simulation time = 1.15762 seconds
n = 2000,threads = 4, simulation time = 2.61354 seconds
n = 4000,threads = 8, simulation time = 4.98699 seconds
n = 8000,threads = 16, simulation time = 9.46007 seconds

Strong scaling estimates are :
    0.84    1.12    1.33    1.74    2.14 (speedup)
    0.84    0.56    0.33    0.22    0.13 (efficiency)    for
       1       2       4       8      16 threads/processors

Average strong scaling efficiency:    0.42

Weak scaling estimates are :
    0.84    0.35    0.15    0.08    0.04 (efficiency)    for
       1       2       4       8      16 threads/processors

Average weak scaling efficiency:    0.29


openmp Grade =   53.33
```

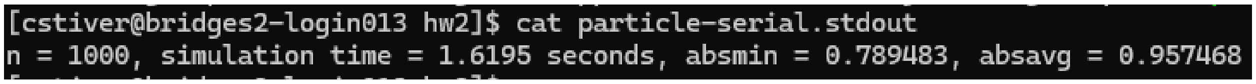


```
[cstiver@bridges2-login013 hw2]$ cat auto-particle-pthreads16.stdout
rm: cannot remove 'pthreads_sum.txt': No such file or directory
n = 500, simulation time = 0.411309 seconds
n = 500, simulation time = 0.471422 seconds
n = 500, simulation time = 0.329404 seconds
n = 500, simulation time = 0.31893 seconds
n = 500, simulation time = 0.230903 seconds
n = 500, simulation time = 0.152845 seconds
n = 1000, simulation time = 1.0476 seconds
n = 2000, simulation time = 2.3532 seconds
n = 4000, simulation time = 4.67431 seconds
n = 8000, simulation time = 8.65004 seconds

Strong scaling estimates are :
    0.87    1.25    1.29    1.78    2.69 (speedup)
    0.87    0.62    0.32    0.22    0.17 (efficiency)    for
       1       2       4       8      16 threads/processors

Average strong scaling efficiency:    0.44

Weak scaling estimates are :
    0.87    0.39    0.17    0.09    0.05 (efficiency)    for
       1       2       4       8      16 threads/processors

Average weak scaling efficiency:    0.32


pthreads Grade =   56.78
```



```
[cstiver@bridges2-login012 hw2]$ cat auto-particle-mpi16.stdout
-------------------------------------------------------------------------------
There are messages associated with the following module(s):
-------------------------------------------------------------------------------


cuda/11.1.1:
   Warning - This module has been deprecated and is scheduled for removal.
   Please transition to a module from the production pool instead.

-------------------------------------------------------------------------------


rm: cannot remove 'mpi_sum.txt': No such file or directory
n = 500, simulation time = 0.403861 seconds
n = 500, simulation time = 0.40582 seconds
n = 500, simulation time = 0.210921 seconds
n = 500, simulation time = 0.133154 seconds
n = 500, simulation time = 0.08952 seconds
n = 500, simulation time = 0.048909 seconds
n = 1000, simulation time = 0.806024 seconds
n = 2000, simulation time = 1.63657 seconds
n = 4000, simulation time = 3.2449 seconds
n = 8000, simulation time = 6.62878 seconds

Strong scaling estimates are :
    1.00    1.91    3.03    4.51    8.26 (speedup)
    1.00    0.96    0.76    0.56    0.52 (efficiency)    for
       1       2       4       8      16 threads/processors

Average strong scaling efficiency:    0.76

Weak scaling estimates are :
    1.00    0.50    0.25    0.12    0.06 (efficiency)    for
       1       2       4       8      16 threads/processors

Average weak scaling efficiency:    0.39


mpi Grade =   77.18
```

grade didnt change but final simulation time was 2.3 seconds faster from a 8.9 to 6.6 seconds?



```
[cstiver@bridges2-login013 hw2]$ cat particle-serial.stdout
n = 1000, simulation time = 1.6195 seconds, absmin = 0.789483, absavg = 0.957468
```