

hpc hw 3 cameron stiver

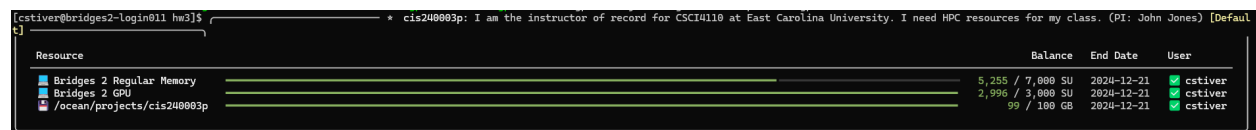
Getting the gpu to work at all was a headache, it wouldn't compile with gpu-shared and complained about the QoS.

```
[cstiver@bridges2-login014 hw3]$ cat job-bridges-gpu
#!/bin/bash
##SBATCH -A see180004p # 2021 Allocation -- This might change in the following years.
#SBATCH -J particle-gpu
#SBATCH -o particle-gpu.stdout
#SBATCH -n 1
#SBATCH -p GPU
#SBATCH --gres=gpu:v100-32:8
#SBATCH -t 00:10:00
source modules.sh
./gpu -n 2000 -o gpu.txt
./autocorrect -s gpu.txt
```

I eventually got to a point where I could compile but couldn't run the gpu code, but I could run serial. I wasn't sure if we were supposed to edit serial, so i left it there.

```
[cstiver@bridges2-login014 hw3]$ cat particle-serial.stdout
n = 2000, simulation time = 8.02069 seconds
n = 2000, absmin = 0.812940, absavg = 0.961260
```

I eventually saw this, below, and got an email from PSC at 3am on 4/3 saying I could use the GPUs now.



Resource	Balance	End Date	User
Bridges 2 Regular Memory	5,285 / 7,000 SU	2024-12-21	cstiver
Bridges 2 GPU	2,996 / 3,000 SU	2024-12-21	cstiver
/ocean/projects/cis240003p	99 / 100 GB	2024-12-21	cstiver

I first thought I'd adjust the number of threads.

num threads

128

```
[cstiver@bridges2-login013 hw3]$ cat particle-gpu.stdout
CPU-GPU copy time = 9.8e-05 seconds
n = 2000, simulation time = 0.485296 seconds
n = 2000, absmin = 0.813039, absavg = 0.958637
```

256

```
[cstiver@bridges2-login014 hw3]$ cat particle-gpu.stdout
/var/spool/slurm/d/job23233346/slurm_script: line 9: source: command not found
CPU-GPU copy time = 9.8e-05 seconds
n = 2000, simulation time = 0.466635 seconds
n = 2000, absmin = 0.786817, absavg = 0.961142
```

512 seems good

```
[cstiver@bridges2-login013 hw3]$ cat particle-gpu.stdout
CPU-GPU copy time = 9.1e-05 seconds
n = 2000, simulation time = 0.454268 seconds
n = 2000, absmin = 0.805867, absavg = 0.961590
```

1024

```
[cstiver@bridges2-login013 hw3]$ cat particle-gpu.stdout
CPU-GPU copy time = 8.2e-05 seconds
n = 2000, simulation time = 0.472171 seconds
n = 2000, absmin = 0.798598, absavg = 0.958101
[cstiver@bridges2-login013 hw3]$
```

I changed a method call in `apply_force_gpu` to assign `r2` with a new value from calling the `fmax` function, i think it's actually the same thing as what's commented out.

I noticed sometimes my simulation time was lower but my cpu-gpu copy time was higher? I couldn't really figure the relationship between those. I thought my speed was going down for simulation time, but my copy time went up randomly.

I saw the number of blocks was being computed by the `NUM_THREADS`. I thought, like the first assignment with the blocked matrix multiplication, maybe adjusting the block number could change some things.

I tried lowering it, like:

set `blks = n / NUM_THREADS`

with

256 threads

```
[cstiver@bridges2-login014 hw3]$ cat particle-gpu.stdout
CPU-GPU copy time = 8.8e-05 seconds
n = 2000, simulation time = 0.482356 seconds
n = 2000, absmin = 0.812098, absavg = 0.954275
```

512 threads

```
[cstiver@bridges2-login014 hw3]$ cat particle-gpu.stdout
CPU-GPU copy time = 8.6e-05 seconds
n = 2000, simulation time = 0.469803 seconds
n = 2000, absmin = 0.812422, absavg = 0.956627
```

256 threads, `blks = n / 2 / num_threads`

```
[cstiver@bridges2-login012 hw3]$ cat particle-gpu.stdout
CPU-GPU copy time = 8.2e-05 seconds
n = 2000, simulation time = 0.470834 seconds
n = 2000, absmin = 0.806684, absavg = 0.954784
```

256 threads, $\text{blks} = n / 4 / \text{num_threads}$

```
[cstiver@bridges2-login012 hw3]$ cat particle-gpu.stdout
CPU-GPU copy time = 8.3e-05 seconds
n = 2000, simulation time = 0.476538 seconds
n = 2000, absmin = 0.838227, absavg = 0.964983
```

1024 threads, $\text{blks} = n / 4 / \text{num_threads}$

It was about this time I learned I should probably stop messing with blocks. I thought it would be like the job-blocked from earlier in our semester where lowering the block size led to better performance, but I actually just broke things doing this.

```
[cstiver@bridges2-login012 hw3]$ cat particle-gpu.stdout
CPU-GPU copy time = 6.2e-05 seconds
n = 2000, simulation time = 0.083091 seconds
n = 2000, absmin = 1.000000, absavg = 1.000000
```

This time I broke the simulation entirely because the division on number of blocks lead to there being 0 blocks. It executes very quickly!!!! but it doesnt do anything as the absmin and avg are both 1.0 so i assume nothing is actually moving.

about this time i went back to the expected blocks count and 512 threads seemed like a good number. The copy time was much lower than most of the prior runs, but again im not sure if that makes a difference.

```
[cstiver@bridges2-login012 hw3]$ cat particle-gpu.stdout
CPU-GPU copy time = 6.6e-05 seconds
n = 2000, simulation time = 0.457862 seconds
n = 2000, absmin = 0.835940, absavg = 0.964211
```

Never seen this before, seems like an issue

```
[cstiver@bridges2-login012 hw3]$ squeue -u cstiver
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
23511066 GPU particle cstiver PD 0:00 1 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher priority partitions)
```

```

Please contact help@bridges21.org with any comments/concerns.
Last login: Thu Apr 18 17:54:35 2024 from 50.25.142.122

* cis240003p: I am the instructor of record for CSCI4110 at East Carolina University. I need HPC resources for my

Resource                                     Balance    End Date    User
-----
Bridges 2 Regular Memory  ██████████  5,247 / 7,000 SU  2024-12-21  ✓ cstiver
Bridges 2 GPU             ██████████  2,995 / 3,000 SU  2024-12-21  ✓ cstiver
Bridges 2 Ocean Storage   ██████████  100* / 100 GB    2024-12-21  ✓ cstiver

* Use the projects command for up-to-date storage usage.

[cstiver@bridges2-login011 ~]$ cd hw3
[cstiver@bridges2-login011 hw3]$ cd csci4110/
[cstiver@bridges2-login011 csci4110]$ cd hw3
[cstiver@bridges2-login011 hw3]$ ls
Makefile      autograder    common.h      gpu.cu        gpu.txt       particle-gpu.stdout
README        autograder.cu common.o      gpu.cu.save   job-bridges-gpu serial
auto-bridges-gpu autograder.o  gen_gpusum.py gpu.cu.save.1 job-bridges-serial serial.cu
autocorrect   common.cu     gpu          gpu.o         modules.sh    serial.o
[cstiver@bridges2-login011 hw3]$ squeue -u cstiver
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
23513855 GPU particle cstiver PD 0:00 1 (Nodes required for job are DOWN, DRAINED or reserved for jobs in higher priority partitions)

```

It's now midnight, this has been there for like 6 hours. I don't know if I could do more if I even wanted to.

I had about 3 bridges or setup issues this assignment, even more than the first ones where we kind of expected things like that. Not sure how i feel about CUDA coding but I didn't really get to actually restructure things much. I think if the particles were split up in such a way that they only had to pay attention to their direct neighbors then we could parallelize that super well. That being said, that would take a lot of changing. I've also read about Streams, which seem like a pretty good way to utilize some of this CUDA concurrent execution. It seems more efficient than sometimes needing to call `cudaDeviceSynchronize`, which seems to have to do more work, but I think again I would have to change much more to utilize them.