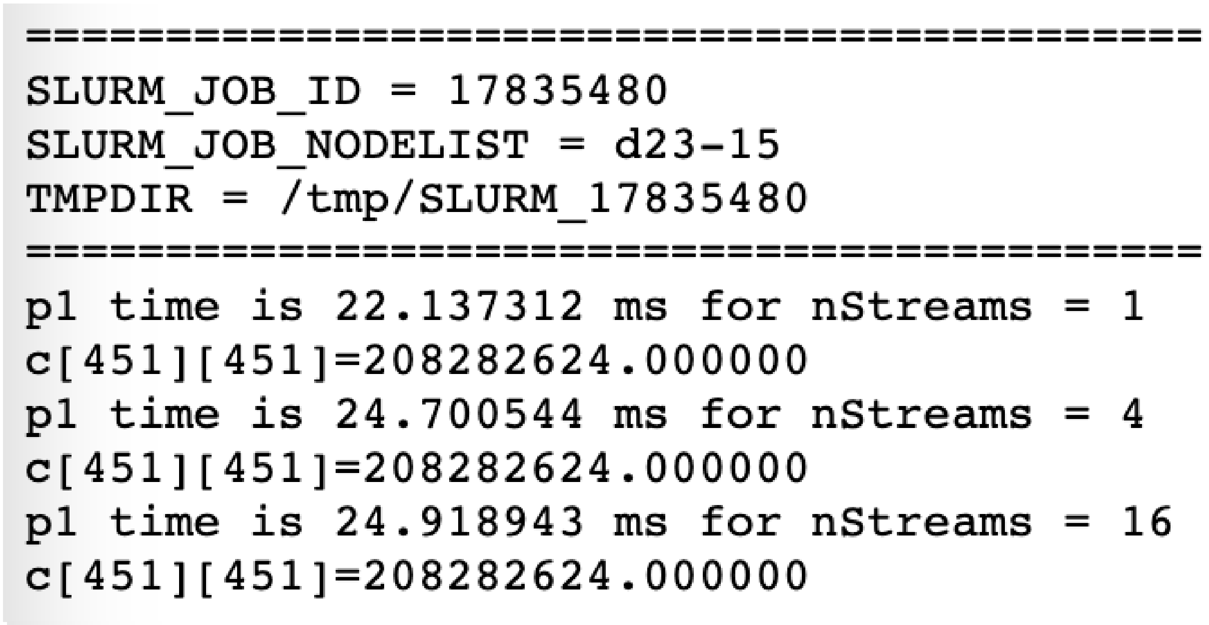
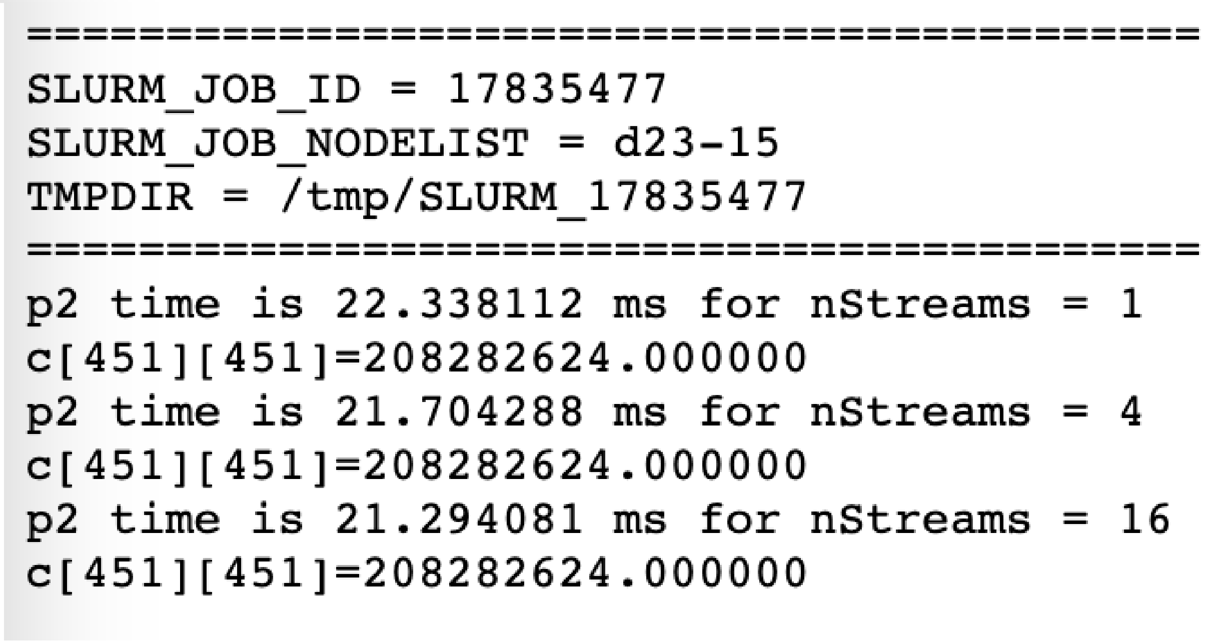
Tingqi (Ting) Wang

## Problem 1: Matrix Multiplication

The following is the output from running two versions of matrix multiplication implementation using CUDA to run on the GPU that overlaps computation and communication on the HPC for number of streams = {1, 4, 16}. p1 is the implementation that loops over {memory copy, kernel function, memory copy}and p2 is the implementation that separately loops for memory copy, kernel function, and memory copy.



```
========================================
SLURM_JOB_ID = 17835480
SLURM_JOB_NODELIST = d23-15
TMPDIR = /tmp/SLURM_17835480
========================================
p1 time is 22.137312 ms for nStreams = 1
c[451][451]=208282624.000000
p1 time is 24.700544 ms for nStreams = 4
c[451][451]=208282624.000000
p1 time is 24.918943 ms for nStreams = 16
c[451][451]=208282624.000000
```



```
========================================
SLURM_JOB_ID = 17835477
SLURM_JOB_NODELIST = d23-15
TMPDIR = /tmp/SLURM_17835477
========================================
p2 time is 22.338112 ms for nStreams = 1
c[451][451]=208282624.000000
p2 time is 21.704288 ms for nStreams = 4
c[451][451]=208282624.000000
p2 time is 21.294081 ms for nStreams = 16
c[451][451]=208282624.000000
```

Looking at the results, p1 had an overall higher runtime for all three nStreams value compared to p2. This is likely a result of the second approach being more effective in allowing for more overlap between memory transfer and kernel execution so as to better utilize the GPU's ability to handle multiple tasks concurrently.

In addition, the runtime of p1 increased with an increase in nStreams, whereas the runtime of p2 decreased with an increase in nStreams. The likely reason for the increase in runtime as nStreams increase for p1 is due to the synchronization overhead that becomes more problematic as the number of streams increases. This is because the GPU becomes idle when waiting for each stream to complete before moving to the next one.