

# Parallel Programming Homework 5 Report

The following codes have been run on HPC, on a single GPU

p1.cu - Vector\_addition.cu

p2\_1.cu - un-optimized matrix multiplication

p2\_2.cu - shared matrix multiplication

The executable names generated using the above codes are

1. p1.cu - p1
2. p2\_1.cu - p2\_1
3. p2\_2.cu - p2\_2

## 1. Vector-Addition

Screenshot:



```
csci545@ubuntu: ~/Desktop/Parallel_Computing/EE_451_F_2019_PHW_5
anirudpk@hpc3139:~$ nvcc vector_add.cu -o go
anirudpk@hpc3139:~$ srun -n1 ./go
time is 606756.000000 ns
c[0]=3 c[1]=3 c[2]=3 c[3]=3 c[4]=3 c[5]=3 c[6]=3 c[7]=3 c[8]=3 c[9]=3 c[10]=3 c[11]=3 c[12]=3 c[13]=3 c[14]=3 c[15]=3 c[16]=3 c[17]=3 c[18]=
3 c[19]=3 c[20]=3 c[21]=3 c[22]=3 c[23]=3 c[24]=3 c[25]=3 c[26]=3 c[27]=3 c[28]=3 c[29]=3 c[30]=3 c[31]=3 c[32]=3 c[33]=3 c[34]=3 c[35]=3
c[36]=3 c[37]=3 c[38]=3 c[39]=3 c[40]=3 c[41]=3 c[42]=3 c[43]=3 c[44]=3 c[45]=3 c[46]=3 c[47]=3 c[48]=3 c[49]=3 c[50]=3 c[51]=3 c[52]=3 c[5
3]=3 c[54]=3 c[55]=3 c[56]=3 c[57]=3 c[58]=3 c[59]=3 c[60]=3 c[61]=3 c[62]=3 c[63]=3 c[64]=3 c[65]=3 c[66]=3 c[67]=3 c[68]=3 c[69]=3 c[70]=
3 c[71]=3 c[72]=3 c[73]=3 c[74]=3 c[75]=3 c[76]=3 c[77]=3 c[78]=3 c[79]=3 c[80]=3 c[81]=3 c[82]=3 c[83]=3 c[84]=3 c[85]=3 c[86]=3 c[87]=3 c
[88]=3 c[89]=3 c[90]=3 c[91]=3 c[92]=3 c[93]=3 c[94]=3 c[95]=3 c[96]=3 c[97]=3 c[98]=3 c[99]=3 c[100]=3 c[101]=3 c[102]=3 c[103]=3 c[104]=3
c[105]=3 c[106]=3 c[107]=3 c[108]=3 c[109]=3 c[110]=3 c[111]=3 c[112]=3 c[113]=3 c[114]=3 c[115]=3 c[116]=3 c[117]=3 c[118]=3 c[119]=3 c[1
20]=3 c[121]=3 c[122]=3 c[123]=3 c[124]=3 c[125]=3 c[126]=3 c[127]=3 c[128]=3 c[129]=3 c[130]=3 c[131]=3 c[132]=3 c[133]=3 c[134]=3 c[135]=
3 c[136]=3 c[137]=3 c[138]=3 c[139]=3 c[140]=3 c[141]=3 c[142]=3 c[143]=3 c[144]=3 c[145]=3 c[146]=3 c[147]=3 c[148]=3 c[149]=3 c[150]=3 c[
151]=3 c[152]=3 c[153]=3 c[154]=3 c[155]=3 c[156]=3 c[157]=3 c[158]=3 c[159]=3 c[160]=3 c[161]=3 c[162]=3 c[163]=3 c[164]=3 c[165]=3 c[166]=
3 c[167]=3 c[168]=3 c[169]=3 c[170]=3 c[171]=3 c[172]=3 c[173]=3 c[174]=3 c[175]=3 c[176]=3 c[177]=3 c[178]=3 c[179]=3 c[180]=3 c[181]=3 c
[182]=3 c[183]=3 c[184]=3 c[185]=3 c[186]=3 c[187]=3 c[188]=3 c[189]=3 c[190]=3 c[191]=3 c[192]=3 c[193]=3 c[194]=3 c[195]=3 c[196]=3 c[197
]=3 c[198]=3 c[199]=3 c[200]=3 c[201]=3 c[202]=3 c[203]=3 c[204]=3 c[205]=3 c[206]=3 c[207]=3 c[208]=3 c[209]=3 c[210]=3 c[211]=3 c[212]=3
c[213]=3 c[214]=3 c[215]=3 c[216]=3 c[217]=3 c[218]=3 c[219]=3 c[220]=3 c[221]=3 c[222]=3 c[223]=3 c[224]=3 c[225]=3 c[226]=3 c[227]=3 c[22
8]=3 c[229]=3 c[230]=3 c[231]=3 c[232]=3 c[233]=3 c[234]=3 c[235]=3 c[236]=3 c[237]=3 c[238]=3 c[239]=3 c[240]=3 c[241]=3 c[242]=3 c[243]=3
c[244]=3 c[245]=3 c[246]=3 c[247]=3 c[248]=3 c[249]=3 c[250]=3 c[251]=3 c[252]=3 c[253]=3 c[254]=3 c[255]=3 anirudpk@hpc3139:~$
```

## 2. Unoptimized Matrix multiplication

```
csci545@ubuntu: ~/Desktop/Parallel_Computing /EE_451_F_2019_PHW_5
anirudpk@hpc-login3:~$ salloc --gres=gpu:1 --ntasks=2
salloc: Pending job allocation 4996150
salloc: job 4996150 queued and waiting for resources
salloc: job 4996150 has been allocated resources
salloc: Granted job allocation 4996150
salloc: Waiting for resource configuration
salloc: Nodes hpc3139 are ready for job
----- Begin SLURM Prolog -----
Job ID:          4996150
Username:        anirudpk
Accountname:     lc_vkp2
Name:            sh
Partition:       quick
Nodelist:        hpc3139
TasksPerNode:    2
CPUsPerTask:     Default[1]
TMPDIR:          /tmp/4996150.quick
SCRATCHDIR:      /staging/scratch/4996150
Cluster:         uschpc
HSDA Account:    false
----- 2019-11-09 10:58:48 -----
anirudpk@hpc3139:~$ nvcc matrix_mul.cu -o go
bash: nvcc: command not found
anirudpk@hpc3139:~$ source /usr/usc/cuda/default/setup.sh
anirudpk@hpc3139:~$ nvcc matrix_mul.cu -o go
anirudpk@hpc3139:~$ srun -n1 ./go
time is 136549536.000000 ns
C[451][451] = 2048 anirudpk@hpc3139:~$
```

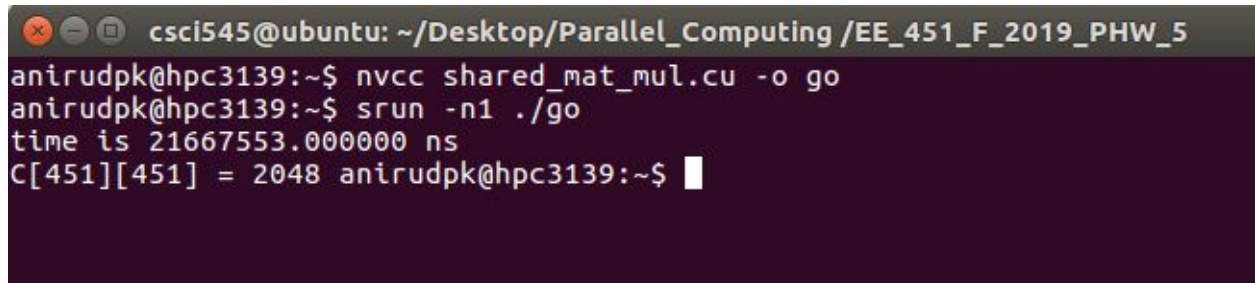
In this method, 1 thread is responsible for 1 output element.

As we can see from the above screenshot that the time taken for un-optimized matrix multiplication is 136549536 ns i.e 0.136 sec

This is a memory bound operation as for each element we have to fetch data from global memory.

This execution speed is very fast as compared to serial execution time, which will be order of  $n^3$  ( $1024*1024*1024$ )

### 3. Shared Matrix multiplication

A terminal window with a dark background and light-colored text. The window title is 'csci545@ubuntu: ~/Desktop/Parallel\_Computing /EE\_451\_F\_2019\_PHW\_5'. The terminal shows the following commands and output:

```
anirudpk@hpc3139:~$ nvcc shared_mat_mul.cu -o go
anirudpk@hpc3139:~$ srun -n1 ./go
time is 21667553.000000 ns
C[451][451] = 2048 anirudpk@hpc3139:~$
```

In this method, each thread is responsible for 1 element of the output. And we perform block matrix multiplication.

As we can see from the above screenshot, the time taken for shared matrix multiplication is 21667553 ns i.e 0.021 sec.

If we compare the time taken for un-optimized matrix multiplication and shared matrix multiplication, we can see that

1. Un-optimized time: 0.13 sec
2. Shared time: 0.021 sec

Ratio of 1 and 2 = 6.190 i.e shared matrix multiplication is approximately 6 times faster as compared to unoptimized technique.

Shared matrix multiplication is faster because for each element, we bring the block of matrix from shared memory, instead of fetching the data from global memory. Fetching from shared memory is faster as compared to fetching data from global memory.