High Performance Computing

Assignment 2

Theory Questions

1. **Total Number of Nodes = 260**
   Number of GPU Nodes = 64
   Number of large memory capacity nodes = 36
   Number of nodes required for high priority jobs = 128
   Number of regular nodes left = 32

   a. A normal partitioning scheme could have been diving all 64 GPU nodes into one partition, similarly all 36 large memory capacity nodes as a partition, all 128 nodes needed for high priority jobs as one and last partition of 32 nodes. But that would not be the best way, probably.

   Instead, one partition could be of 128 non-accelerated nodes that are exclusively for high priority jobs. This will ensure all high priority jobs can be executed as quickly as possible without interference. The one partition overlap could be if low priority jobs are running on this partition. But that can be resolved by pre-emption. That is, if a new high priority job comes, then the running low priority jobs can be pre-empted and continued when the high priority job is completed.

   Other partitions will include a few regular nodes, few GPU nodes and few large memory nodes. Making these partitions to be used for normal scheduling, effective resource utilization and also GRES (I know this is SLURM specific, but it could be useful for high computational power jobs without disrupting the scheduling of other jobs). So, maybe 2 partitions, one could be TST env other PROD env, could be of 16 regular nodes, 30 GPU Nodes, and 15 large memory capacity. Another one partition(DEV env) could be left over 4 GPU Nodes and 6 nodes of large memory capacity nodes.

   b. Some provisions that should be implemented to facilitate parallel job debugging would be:

   - Use DEV and TST for development and testing debugging is done in TST env
   - Using the partitions with regular nodes, GPU and high memory nodes to parallelize jobs effectively
   - Implement parallel programming models and libraries like OpenMP, MPI to be able to actually run parallelization code for all resource management suite (SLURM,

Hadoop YARN, OpenLava, PBS). Also as administrator, implementing any tools that help in optimizing the code and identifying performance issues would be beneficial

- Nodes with similar configurations and properties should be assigned for parallelization in each partition to ensure maximum and efficient parallelization
- When assigning nodes, as an administrator, providing all necessary details of configurations to ensure optimal resource utilization and effective parallelization

c. Some strategies and features of some schedulers that can minimize the impact of conflicts between jobs of different priorities are (Not SLURM specific):

- Event Triggered Scheduling: Whenever a new event is added or system configurations are changed (trigger), only a few jobs at the front of the queue are considered and scheduled. This is to ensure no starvation of low priority jobs
    - A feature that can be added to this is the considering of ALL the jobs in the queue to make scheduling decisions. But this requires a lot of overhead, therefore can only be done at infrequent intervals
- Gang Scheduling: This is an efficient scheduling to ensure no starvation and proper resource allocation based on priority and job requirements. The jobs that have similar properties and configurations are allocated similar resources and the resources are alternated between them. Time slicer is added to keep checking if any job has been suspended for too long and then the active jobs are suspended and added to back of queue. This is to ensure the low priority job that has been suspended for too long gets the chance to start and finish
    - Pre-emption: A variant of gang scheduling, This allows high priority jobs to interrupt any running low priority jobs if there is a resource allocation conflict. The low priority jobs are suspended and continue once the high priority jobs are completed

2. Required: peak 100Tflops machine

Unit1 = dual socket CPU with 12-Core processor operating at 3.4GHz
Unit2 = dual socket CPU with 20-Core processor operating at 2.5GHz
Number of Floating point operations per clock cycle = 4
Floor Space = 32U for nodes in each rack
Nodes per rack = 16
Floor space taken by 1 rack = 1U
Max Number of nodes possible = 32

a. For Unit 1,
Per clock cycle = 4 floating point operations
Clock Speed = 3.4 GHz
Number of cores = 12

Peak Performance per node = (3.4GHz * 4FLOPS/cycle * 12cores)GFLOPS = 0.1632TFLOPS
For 16 nodes per rack = 2.6112 TFLOPS

For Unit 2,
Per clock cycle = 4 floating point operations
Clock Speed = 2.5 GHz
Number of cores = 20
Peak performance per node = (2.5GHz * 4FLOPS/Cycle * 20cores)GFLOPS = 0.2000TFLOPS
For 16 nodes per rack = 3.2 TFLOPS

The 20-core processors use less floor space and provide higher TFLOPS. Therefore, recommended option would be **20-Core Processor 2.5 GHz Dual socket CPUs**

b. Required performance = 100 TFLOPS
1 rack = 3.2 TFLOPS
Number of racks = 100/3.2 = 31.25 Racks
Instead of getting 0.25 racks, we would need **32 Racks.** That means it will occupy exactly 32U floor space.

c. 32 Racks * 3.2 TFLOPS = 102.4 TFLOPS
We get computational performance of 102.4 TFLOPS.
102.4 tera floating point operations can be performed every second taking 32U floor space.

3. Execution of a 1 million -instruction program takes 2.5 ms on a 2.5GHz core. The Hardware monitor reports a cache miss ratio of 6% for the application. Main Memory access takes on average 80 ns, while cache access has a latency of 800ps. Given that all ALU instructions are executed effectively in a single clock cycle.

T = total execution time
T_{cycle} = time for a single processor cycle = 2.5ms
I_{count} = total number of instructions = 1000000
I_{ALU} = number of ALU instructions (e.g. register – register)
I_{MEM} = number of memory access instructions (e.g. load, store)
CPI = average cycles per instructions = 2.5GHz
CPI_{ALU} = average cycles per ALU instructions = 1
CPI_{MEM} *= average cycles per memory instruction = 5.6*
r_{miss} = cache miss rate = 6%
r_{hit} = cache hit rate = 94%

$CPI_{MEM\text{-}MISS}$ = cycles per cache miss = 80ns
$CPI_{MEM\text{-}HIT}$=cycles per cache hit = 800ps = 0.8ns
M_{ALU} = instruction mix for ALU instructions = 0.67
M_{MEM} = instruction mix for memory access instruction

$CPI_{MEM} = CPI_{Mem\text{-}HIT} + r_{miss} * CPI_{MEM\text{-}MISS}$
$= 0.8 + 0.06*80$
$= 0.8 + 4.8$
$= 5.6$

$M_{ALU} + M_{MEM} = 1$
$M_{MEM} = 1 – M_{ALU}$

$CPI = (M_{ALU} * CPI_{ALU}) + (M_{MEM} * CPI_{MEM})$
$2.5 = (M_{ALU} * 1) + ((1-M_{ALU})* 5.6)$
$2.5 = M_{ALU} + 5.6 – 5.6M_{ALU}$
$4.6M_{ALU} = 5.6-2.5$
$M_{ALU} = 3.1/4.6 = 0.67$

a. Fraction of ALU instructions
$I_{ALU} = M_{ALU} * I_{COUNT} = 0.67 * 1000000 = 670000$

b. Cache Size = 16KB
To decrease execution time by 50%, cache size should be increased and miss rate therefore decreased. We know, miss rate decreases by 1% upon doubling cache size => cache size increases to the power 2^1
Since they are directly proportional,
$T_1/ (0.5 * T_1) \sim= r_{miss}/r_{miss\text{-}new}$
$r_{miss\text{-}new} = \sim r_{miss} * 0.5$
$r_{miss\text{-}new} = \sim 6\% * 0.5 = 3\%$

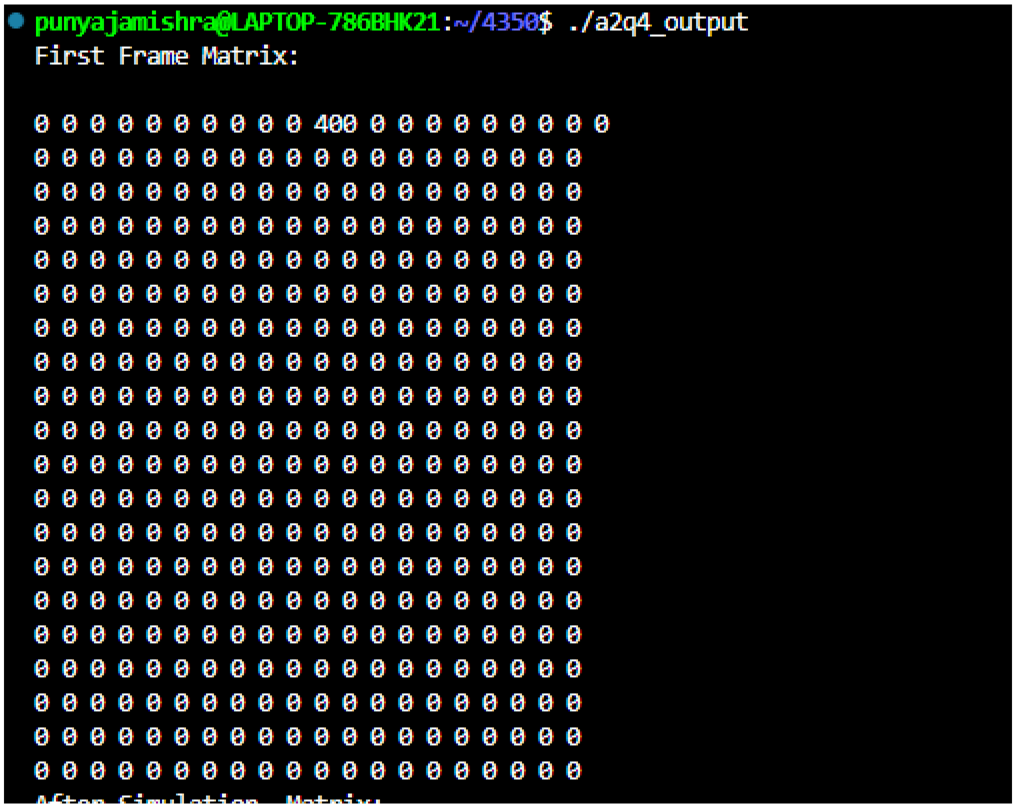
since doubling cache size decreases miss rate by 1%, to decrease miss rate TO 3%, doing some math, the size should double almost 3 times (6% to 3%) = 2^3

c. Applying the final formula to calculate the program runtime if all accessed data fits in cache, meaning run rate is 3%
$T = I_{count} * [(M_{ALU} * CPI_{ALU}) + M_{MEM} * (CPI_{MEM\text{-}HIT} + r_{miss} * CPI_{MEM\text{-}MISS})] * T_{cycle}$
$T = 1000000 * [(0.67 * 1) + 0.33*(3.2)] * 2.5$
$T = 1000000 * 1.726 * 2.5$
$T = 4315000 = 4315s$

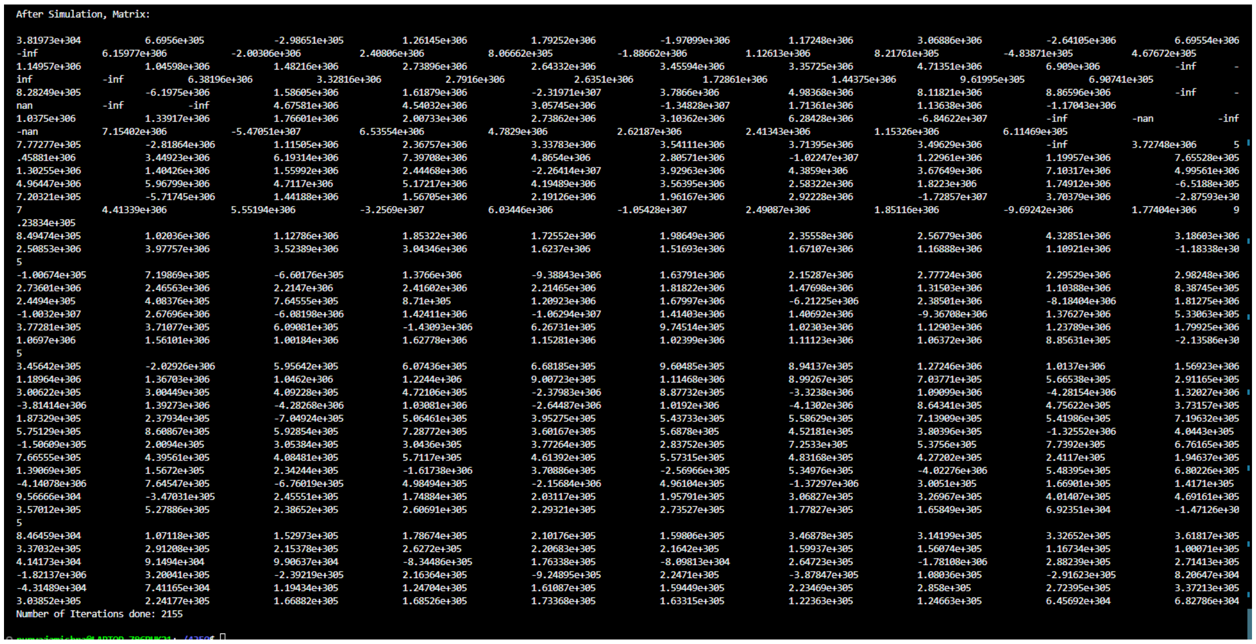
Programming Questions

4. I used the math directly from the question, the program runs but somehow my values are going VERY NAN. I tried debugging and the values are just very small and close to 0. The interesting thing is that I added a condition that simulation ends when all values are 12 or less, and still somehow simulation ends with a matrix full of "NAN" values.
   So instead of 100 by 100 matrix, I am doing for 20X20 matrix to avoid too small values. And make output seem like it is broken

   Screenshot :

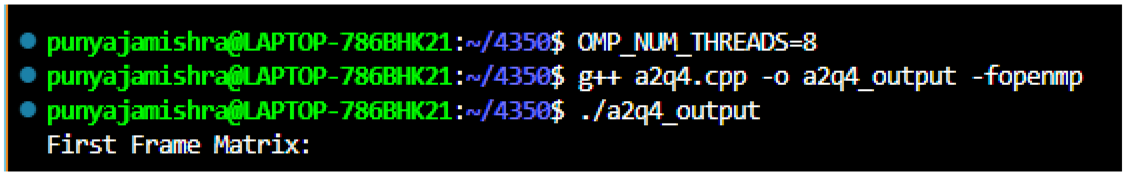


```
punyajamishra@LAPTOP-786BHK21:~/4350$ ./a2q4_output
First Frame Matrix:

0 0 0 0 0 0 0 0 0 0 400 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
After Simulation  Matrix:
```

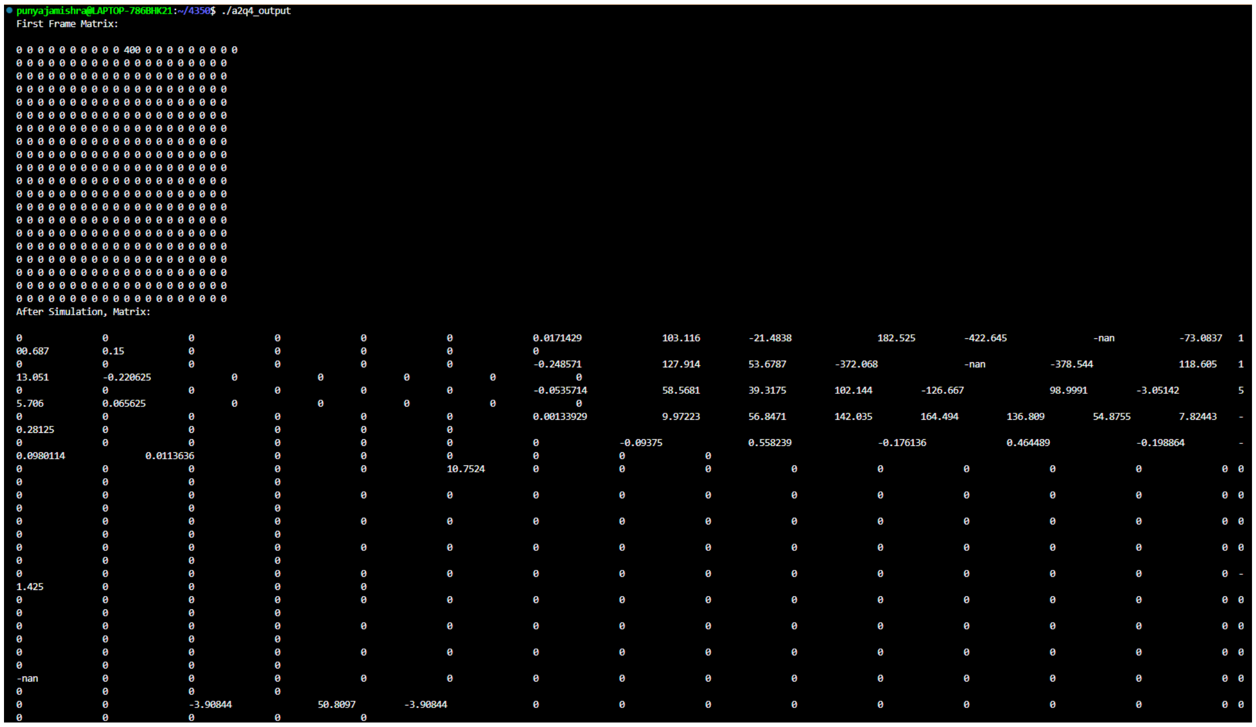


```
After Simulation, Matrix:

3.81973e+304    6.6956e+305     -2.98651e+305   1.26145e+306    1.79252e+306    -1.97099e+306   1.17248e+306    3.06886e+306    -2.64105e+306   6.69554e+306
-inf    6.15977e+306    -2.00306e+306   2.40806e+306    8.06662e+305    -1.88662e+306   1.12613e+306    8.21761e+305    -4.83871e+305   4.67672e+305
1.14957e+306    1.04598e+306    1.48216e+306    2.73896e+306    2.64332e+306    3.45594e+306    3.35725e+306    4.71351e+306    6.909e+306      -inf    -
inf     -inf    6.38196e+306    3.32816e+306    2.7916e+306     2.6351e+306     1.72861e+306    1.44375e+306    9.61995e+305    6.90741e+305
8.28249e+305    -6.1975e+306    1.58605e+306    1.61879e+306    -2.31971e+307   3.7866e+306     4.98368e+306    8.11821e+306    8.86596e+306    -inf    -
nan     -inf    -inf    4.67581e+306    4.54032e+306    3.05745e+306    -1.34828e+307   1.71361e+306    1.13638e+306    -1.17043e+306
1.0375e+306     1.33917e+306    1.76601e+306    2.00733e+306    2.73862e+306    3.10362e+306    6.28428e+306    -6.84622e+307   -inf    -nan    -inf
-nan    7.15402e+306    -5.47051e+307   6.53554e+306    4.7829e+306     2.62187e+306    2.41343e+306    1.15326e+306    6.11469e+305
7.77277e+305    -2.81864e+306   1.11505e+306    2.36757e+306    3.33783e+306    3.54111e+306    3.71395e+306    3.49629e+306    -inf    3.72748e+306    5
.45881e+306     3.44923e+306    6.19314e+306    7.39708e+306    4.8654e+306     2.80571e+306    -1.02247e+307   1.22961e+306    1.19957e+306    7.65528e+305
1.30255e+306    1.40426e+306    1.55992e+306    2.44468e+306    -2.26414e+307   3.92963e+306    4.3859e+306     3.67649e+306    7.10317e+306    4.99561e+306
4.96447e+306    5.96799e+306    4.7117e+306     5.17217e+306    4.19489e+306    3.56395e+306    2.58322e+306    1.8223e+306     1.74912e+306    -6.5188e+305
7.20321e+305    -5.71745e+306   1.44188e+306    1.56705e+306    2.19126e+306    1.96167e+306    2.92228e+306    -1.72857e+307   3.70379e+306    -2.87593e+30
7       4.41339e+306    5.55194e+306    -3.2569e+307    6.03446e+306    -1.05428e+307   2.49087e+306    1.85116e+306    -9.69242e+306   1.77404e+306    9
.23834e+305
8.49474e+305    1.02036e+306    1.12786e+306    1.85322e+306    1.72552e+306    1.98649e+306    2.35558e+306    2.56779e+306    4.32851e+306    3.18603e+306
2.50853e+306    3.97757e+306    3.52389e+306    3.04346e+306    1.6237e+306     1.51693e+306    1.67107e+306    1.16888e+306    1.10921e+306    -1.18338e+30
5
-1.00674e+305   7.19869e+305    -6.60176e+305   1.3766e+306     -9.38843e+306   1.63791e+306    2.15287e+306    2.77724e+306    2.29529e+306    2.98248e+306
2.73601e+306    2.46563e+306    2.2147e+306     2.41602e+306    2.21465e+306    1.81822e+306    1.47698e+306    1.31503e+306    1.10388e+306    8.38745e+305
2.4494e+305     4.08376e+305    7.64555e+305    8.71e+305       1.20923e+306    1.67997e+306    -6.21225e+306   2.38501e+306    -8.18404e+306   1.81275e+306
-1.0032e+307    2.67696e+306    -6.08198e+306   1.42411e+306    -1.06294e+307   1.41403e+306    1.40692e+306    -9.36708e+306   1.37627e+306    5.33063e+305
3.77281e+305    3.71077e+305    6.09081e+305    -1.43093e+306   6.26731e+305    9.74514e+305    1.02303e+306    1.12903e+306    1.23789e+306    1.79925e+306
1.0697e+306     1.56101e+306    1.00184e+306    1.62778e+306    1.15281e+306    1.02399e+306    1.11123e+306    1.06372e+306    8.85631e+305    -2.13586e+30
5
3.45642e+305    -2.02926e+306   5.95642e+305    6.07436e+305    6.68185e+305    9.60485e+305    8.94137e+305    1.27246e+306    1.0137e+306     1.56923e+306
1.18964e+306    1.36703e+306    1.0462e+306     1.2244e+306     9.00723e+305    1.11468e+306    8.99267e+305    7.03771e+305    5.66538e+305    2.91165e+305
3.00622e+305    3.00449e+305    4.09228e+305    4.72106e+305    -2.37983e+306   8.87732e+305    -3.3238e+306    1.09099e+306    -4.28154e+306   1.32027e+306
-3.81414e+306   1.39273e+306    -4.28268e+306   1.03081e+306    -2.64487e+306   1.0192e+306     -4.1302e+306    8.64341e+305    4.75622e+305    3.73157e+305
1.87329e+305    2.37934e+305    -7.04924e+305   5.06461e+305    3.95275e+305    5.43733e+305    5.58629e+305    7.13909e+305    5.41986e+305    7.19632e+305
5.75129e+305    8.60867e+305    5.92854e+305    7.28772e+305    3.60167e+305    5.6878e+305     4.52181e+305    3.80396e+305    -1.32552e+306   4.0443e+305
-1.50609e+305   2.0094e+305     3.05384e+305    3.0436e+305     3.77264e+305    2.83752e+305    7.2533e+305     5.3756e+305     7.7392e+305     6.76165e+305
7.66555e+305    4.39561e+305    4.08481e+305    5.7117e+305     4.61392e+305    5.57315e+305    4.83168e+305    4.27202e+305    2.4117e+305     1.94637e+305
1.39069e+305    1.5672e+305     2.34244e+305    -1.61738e+306   3.70886e+305    -2.56966e+305   5.34976e+305    -4.02276e+306   5.48395e+305    6.80226e+305
-4.14078e+306   7.64547e+305    -6.76019e+305   4.98494e+305    -2.15684e+306   4.96104e+305    -1.37297e+306   3.0051e+305     1.66901e+305    1.4171e+305
9.56666e+304    -3.47031e+305   2.45551e+305    1.74884e+305    2.03117e+305    1.95791e+305    3.06827e+305    3.26967e+305    4.01407e+305    4.69161e+305
3.57012e+305    5.27886e+305    2.38652e+305    2.60691e+305    2.29321e+305    2.73527e+305    1.77827e+305    1.65849e+305    6.92351e+304    -1.47126e+30
5
8.46459e+304    1.07118e+305    1.52973e+305    1.78674e+305    2.10176e+305    1.59806e+305    3.46878e+305    3.14199e+305    3.32652e+305    3.61817e+305
3.37032e+305    2.91208e+305    2.15378e+305    2.6272e+305     2.20683e+305    2.1642e+305     1.59937e+305    1.56074e+305    1.16734e+305    1.00071e+305
4.14173e+304    9.1494e+304     9.90637e+304    -8.34486e+305   1.76338e+305    -8.09813e+304   2.64723e+305    -1.78108e+306   2.88239e+305    2.71413e+305
-1.82137e+306   3.20041e+305    -2.39219e+305   2.16364e+305    -9.24895e+305   2.2471e+305     -3.87847e+305   1.08036e+305    -2.91623e+305   8.20647e+304
-4.31489e+304   7.41165e+304    1.19434e+305    1.24704e+305    1.61087e+305    1.59449e+305    2.23469e+305    2.858e+305      2.72395e+305    3.37213e+305
3.03852e+305    2.24177e+305    1.66882e+305    1.68526e+305    1.73368e+305    1.63315e+305    1.22363e+305    1.24663e+305    6.45692e+304    6.82786e+304
Number of Iterations done: 2155
```

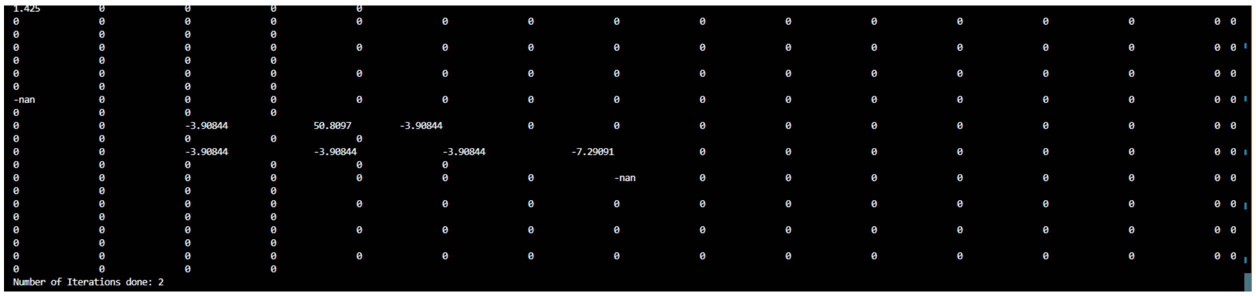
After Parallelizing with OpenMP :



```
● punyajamishra@LAPTOP-786BHK21:~/4350$ OMP_NUM_THREADS=8
● punyajamishra@LAPTOP-786BHK21:~/4350$ g++ a2q4.cpp -o a2q4_output -fopenmp
● punyajamishra@LAPTOP-786BHK21:~/4350$ ./a2q4_output
  First Frame Matrix:
```

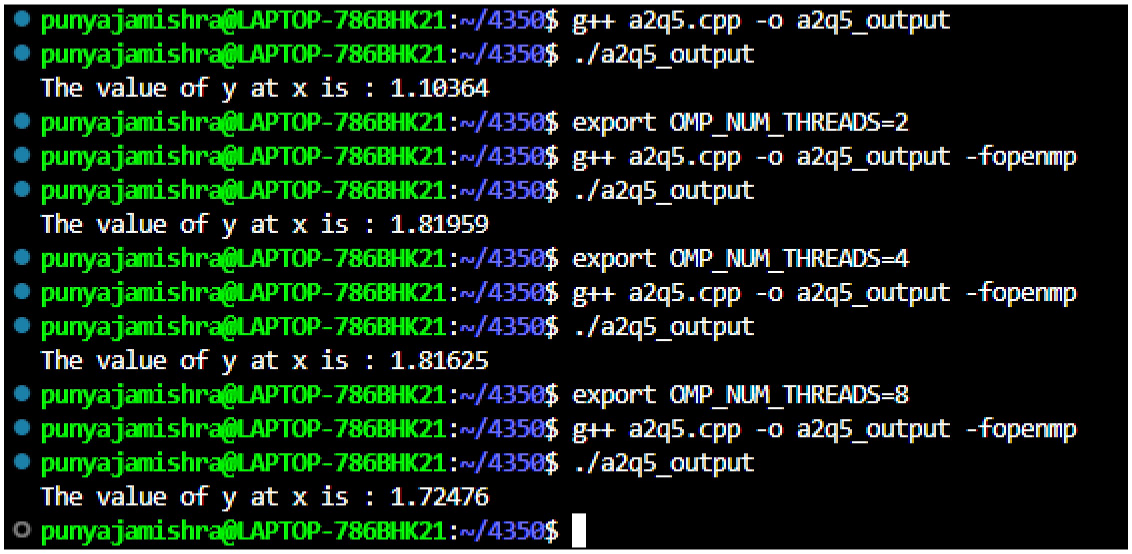


```
● punyajamishra@LAPTOP-786BHK21:~/4350$ ./a2q4_output
  First Frame Matrix:

  0 0 0 0 0 0 0 0 0 0 400 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  After Simulation, Matrix:

  0          0          0          0          0          0          0.0171429       103.116      -21.4838        182.525     -422.645          -nan         -73.0837   1
  00.687     0.15       0          0          0          0          0
  0          0          0          0          0          0          -0.248571       127.914      53.6787     -372.068          -nan        -378.544          118.605   1
  13.051     -0.220625       0          0          0          0          0
  0          0          0          0          0          0          -0.0535714      58.5681      39.3175     102.144     -126.667          98.9991     -3.05142          5
  5.706      0.065625        0          0          0          0          0
  0          0          0          0          0          0          0.00133929      9.97223      56.8471     142.035     164.494     136.809     54.8755     7.82443   -
  0.28125    0          0          0          0          0
  0          0          0          0          0          0          0          -0.09375          0.558239          -0.176136          0.464489          -0.198864          -
  0.0980114       0.0113636       0          0          0          0          0          0
  0          0          0          0          0          10.7524    0          0          0          0          0          0          0          0          0  0
  0          0          0          0
  0          0          0          0          0          0          0          0          0          0          0          0          0          0          0  0
  0          0          0          0
  0          0          0          0          0          0          0          0          0          0          0          0          0          0          0  0
  0          0          0          0
  0          0          0          0          0          0          0          0          0          0          0          0          0          0          0  0
  0          0          0          0
  0          0          0          0          0          0          0          0          0          0          0          0          0          0          0  -
  1.425      0          0          0          0
  0          0          0          0          0          0          0          0          0          0          0          0          0          0          0  0
  0          0          0          0
  0          0          0          0          0          0          0          0          0          0          0          0          0          0          0  0
  0          0          0          0
  0          0          0          0          0          0          0          0          0          0          0          0          0          0          0  0
  0          0          0          0
  -nan       0          0          0          0          0          0          0          0          0          0          0          0          0          0  0
  0          0          0          0
  0          0          -3.90844        50.8097         -3.90844        0          0          0          0          0          0          0          0          0  0
  0          0          0          0          0
```



```
1.425   0       0       0       0
0       0       0       0       0       0       0       0       0       0       0       0       0       0       0 0
0       0       0       0
0       0       0       0       0       0       0       0       0       0       0       0       0       0       0 0
0       0       0       0
0       0       0       0       0       0       0       0       0       0       0       0       0       0       0 0
0       0       0       0
-nan    0       0       0       0       0       0       0       0       0       0       0       0       0       0 0
0       0       0       0
0       0       -3.90844    50.8097 -3.90844    0       0       0       0       0       0       0       0       0 0
0       0       0       0       0
0       0       -3.90844    -3.90844    -3.90844    -7.29091    0       0       0       0       0       0       0 0
0       0       0       0       0       0
0       0       0       0       0       0       0       -nan    0       0       0       0       0       0       0 0
0       0       0       0
0       0       0       0       0       0       0       0       0       0       0       0       0       0       0 0
0       0       0       0
0       0       0       0       0       0       0       0       0       0       0       0       0       0       0 0
0       0       0       0
0       0       0       0       0       0       0       0       0       0       0       0       0       0       0 0
0       0       0       0
Number of Iterations done: 2
```

5. I used the pragma omp parallel shared default with reduction on the value of ‘y’
   The value changed clearly. Y value depends on the previous y value so yes, I don’t know how you parallelize without breaking the calculations.



```
● punyajamishra@LAPTOP-786BHK21:~/4350$ g++ a2q5.cpp -o a2q5_output
● punyajamishra@LAPTOP-786BHK21:~/4350$ ./a2q5_output
  The value of y at x is : 1.10364
● punyajamishra@LAPTOP-786BHK21:~/4350$ export OMP_NUM_THREADS=2
● punyajamishra@LAPTOP-786BHK21:~/4350$ g++ a2q5.cpp -o a2q5_output -fopenmp
● punyajamishra@LAPTOP-786BHK21:~/4350$ ./a2q5_output
  The value of y at x is : 1.81959
● punyajamishra@LAPTOP-786BHK21:~/4350$ export OMP_NUM_THREADS=4
● punyajamishra@LAPTOP-786BHK21:~/4350$ g++ a2q5.cpp -o a2q5_output -fopenmp
● punyajamishra@LAPTOP-786BHK21:~/4350$ ./a2q5_output
  The value of y at x is : 1.81625
● punyajamishra@LAPTOP-786BHK21:~/4350$ export OMP_NUM_THREADS=8
● punyajamishra@LAPTOP-786BHK21:~/4350$ g++ a2q5.cpp -o a2q5_output -fopenmp
● punyajamishra@LAPTOP-786BHK21:~/4350$ ./a2q5_output
  The value of y at x is : 1.72476
○ punyajamishra@LAPTOP-786BHK21:~/4350$
```