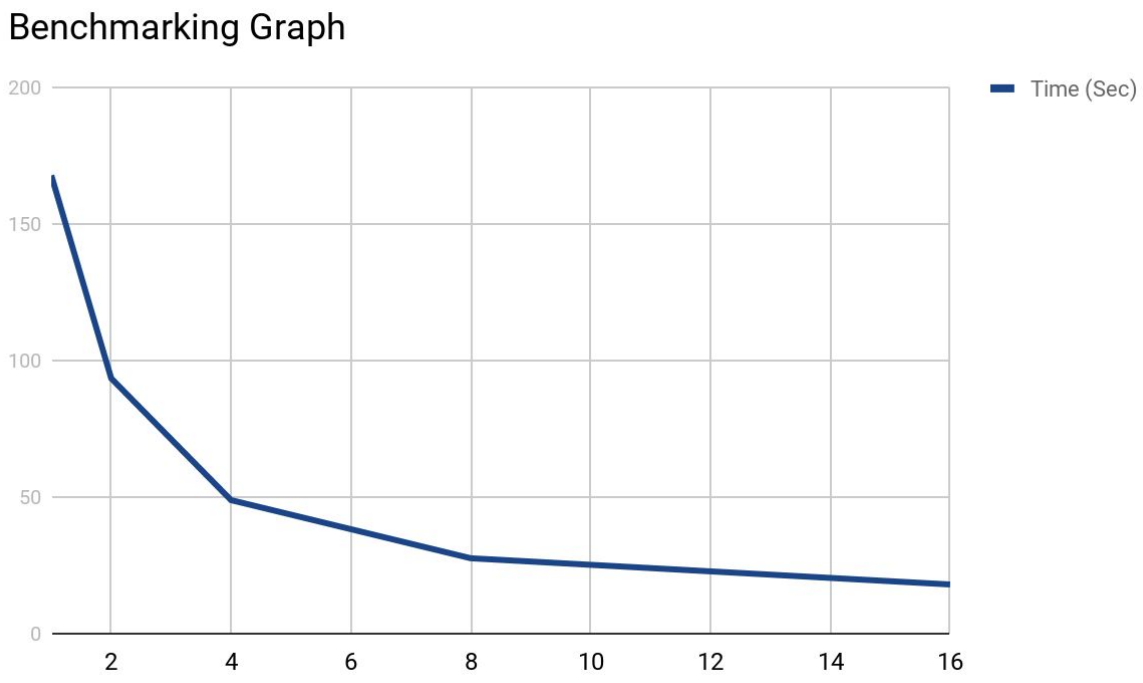
# Assignment 1: Heat Transfer


| | Member |
|---|---|
| Nathaphop Sundarabhogin | 60070503420 |
| Supakit Artsamart | 60070503461 |


## Normal Code

| Core | Time (Sec) |
|---|---|
| 1 | 167.8 |
| 2 | 93.47 |
| 4 | 48.88 |
| 8 | 27.6 |
| 16 | 18 |

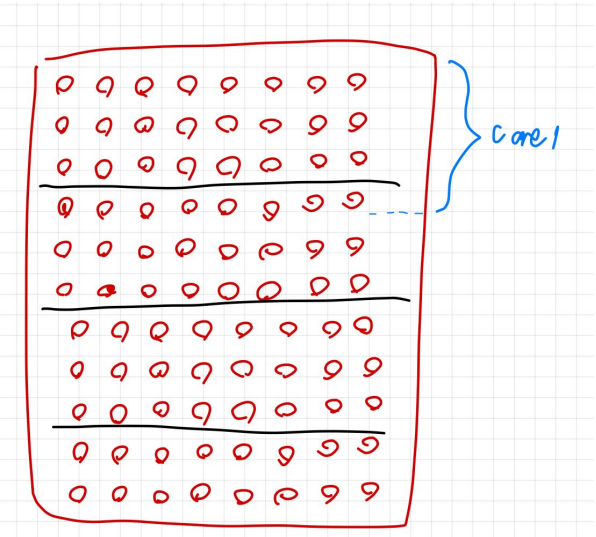


```
Compile Log:
cmd: timeout 50 mpirun -f mpi_host -n 16 ./60070503420 heatMatrixTest.txt 60070503420_out 2000
time: 18.001744173001498
stdout: Time calculation: 0.000000
Write time: 1.245908

Output is correct
---------------------------------

cmd: timeout 75 mpirun -f mpi_host -n 8 ./60070503420 heatMatrixTest.txt 60070503420_out 2000
time: 27.602286436012946
stdout: Time calculation: 128.000000
Write time: 1.244158

Output is correct
---------------------------------

cmd: timeout 100 mpirun -f mpi_host -n 4 ./60070503420 heatMatrixTest.txt 60070503420_out 2000
time: 48.8866252360167
stdout: Time calculation: 0.000000
Write time: 1.230531

Output is correct
---------------------------------

cmd: timeout 150 mpirun -f mpi_host -n 2 ./60070503420 heatMatrixTest.txt 60070503420_out 2000
time: 93.47432071890216
stdout: Time calculation: 128.000000
Write time: 1.258972

Output is correct
---------------------------------

cmd: timeout 200 mpirun -f mpi_host -n 1 ./60070503420 heatMatrixTest.txt 60070503420_out 2000
time: 167.80714818299748
stdout:
Output is correct
```

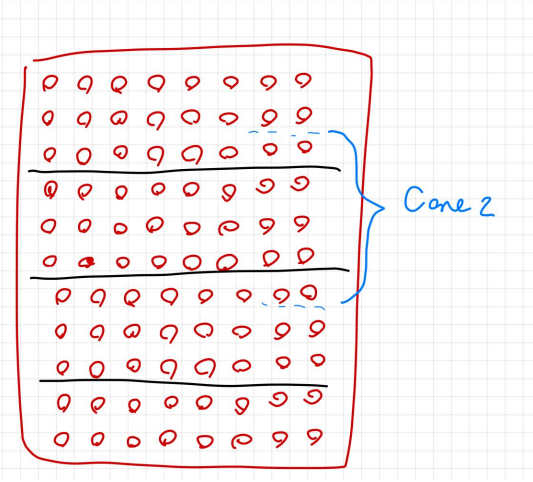
Explanation

From the execution 1 core, take 167.80 secs, 2 cores take 93.47 secs, 4 cores take 48.88 secs, 8 cores take 27.60 secs, and 16 cores take 18 secs.

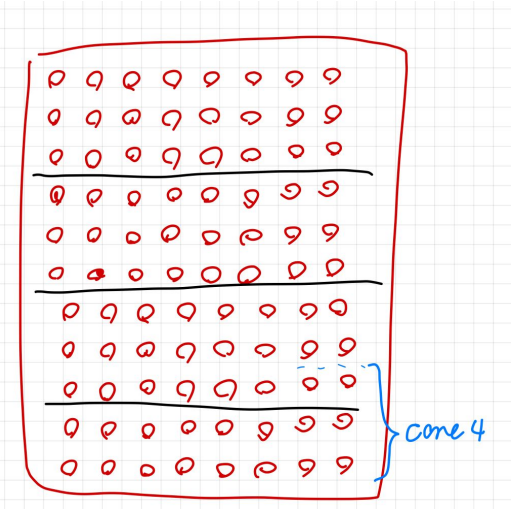
Our code calculate by sending the row plus row edge to each core. Assume we have 11 rows and 8 columns. We split data equally to calculate. But we will send the edge row to each core. See the exampl below.




For the first core, it calculates the first 3 rows. The first row is a constant value, we no need to worry about it. While the last row it needs another row to calculate. So, we will add one more row to it.




While the second core will calculate row forth until row sixth. Row forth and row sixth need one more row to calculate. So, we will add row second and row seventh to calculate. **And we will do this until the last core**



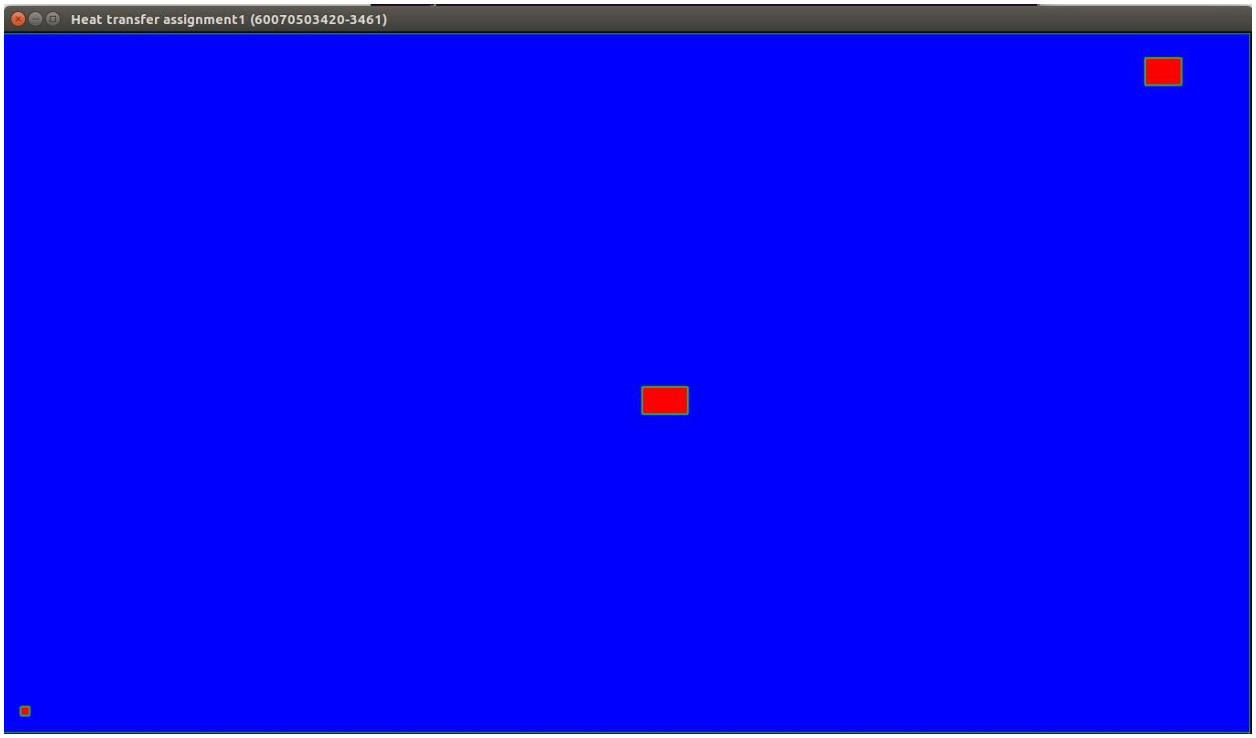

For the last core, it will do like the first core but in the opposite side.

We've tried to optimize the loop in computation by decreasing the amount of loop and decrease computation for the position when computation. From our guess, it should decrease the time usage, but it is increasing. We are not sure why it causes this.

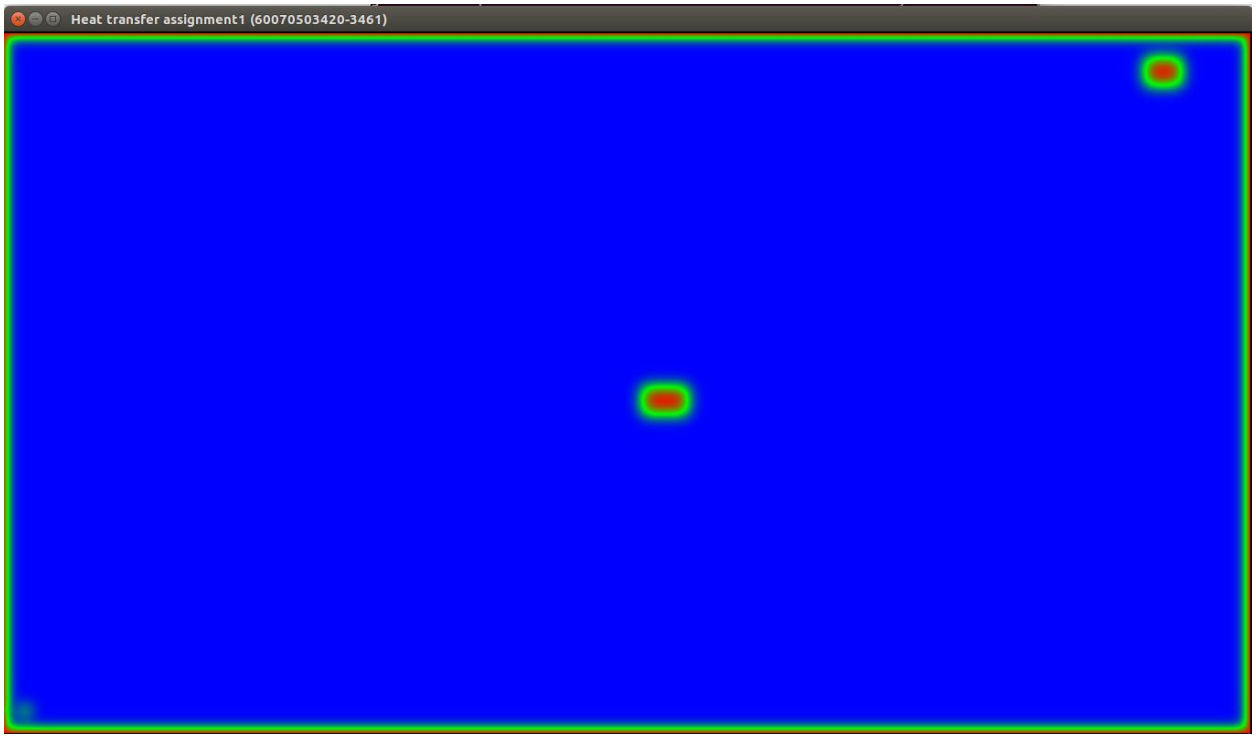
Also, we've tried to change from blocking to nonblocking and do the computation over communication. But the difference is not changed too much and used more time than blocking. So, we select to use the first file that sends to the system because it is the fastest code that we can do it.

## Open GL Simulation

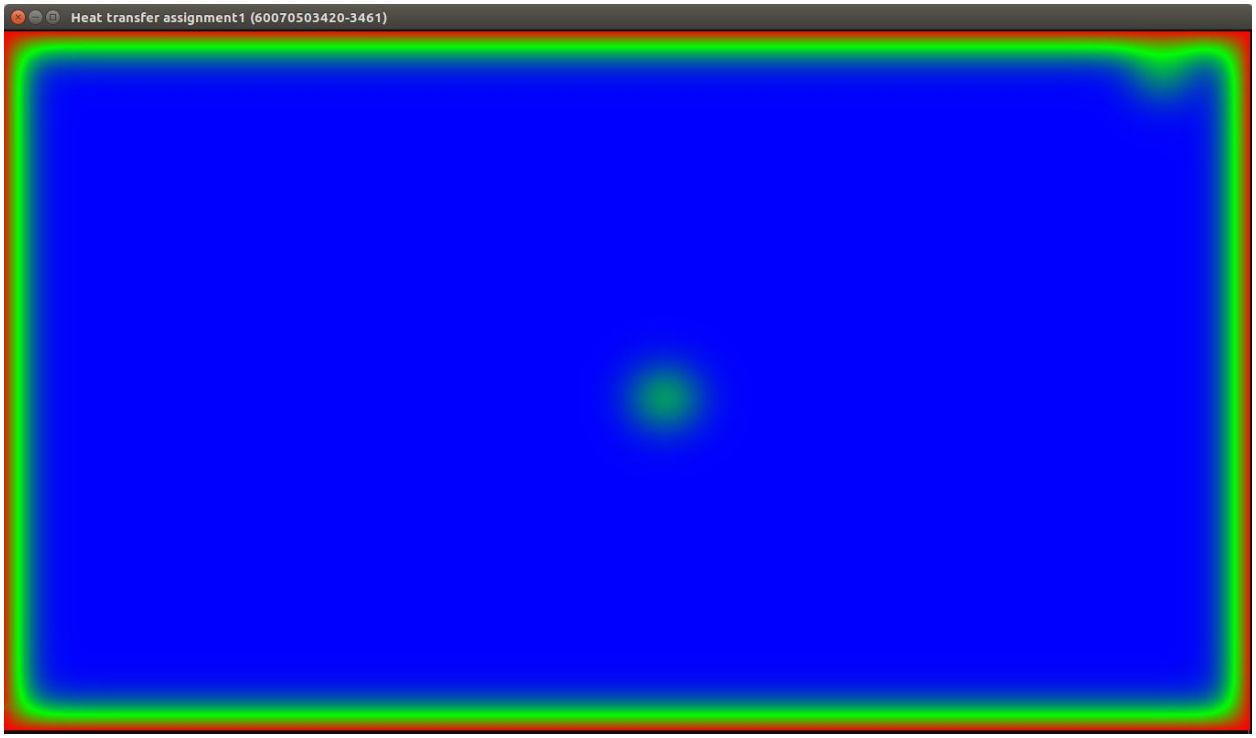
Iteration: 1



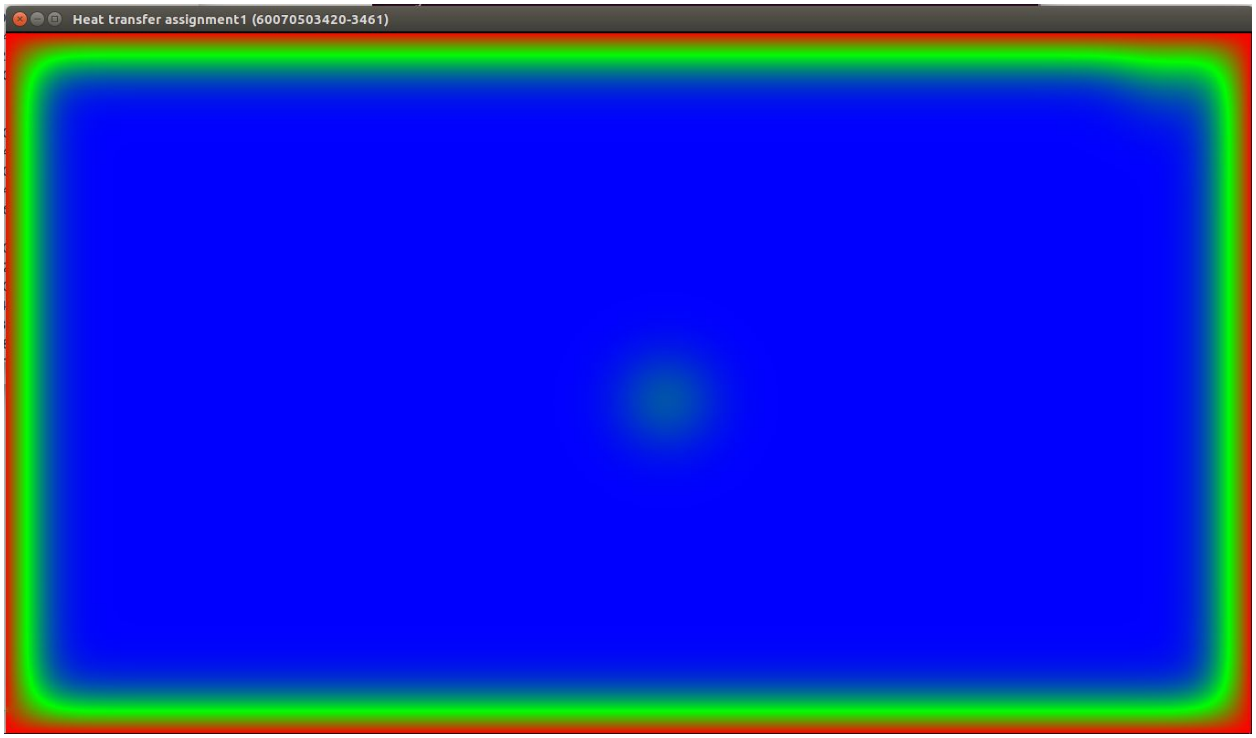

Iteration: 100



Iteration: 1000




Iteration: 2000



Iteration:10000