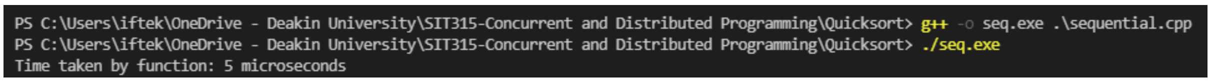
Complex Distributed Computing

Let’s first look at the Quicksort sequential runtimes that I calculated-

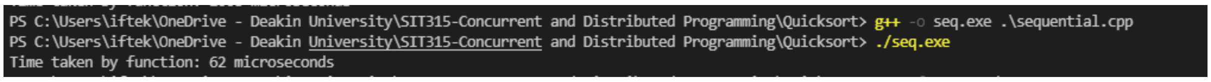
For sorting a size 100 array:



```
PS C:\Users\iftek\OneDrive - Deakin University\SIT315-Concurrent and Distributed Programming\Quicksort> g++ -o seq.exe .\sequential.cpp
PS C:\Users\iftek\OneDrive - Deakin University\SIT315-Concurrent and Distributed Programming\Quicksort> ./seq.exe
Time taken by function: 5 microseconds
```

Sequential Takes 5ms

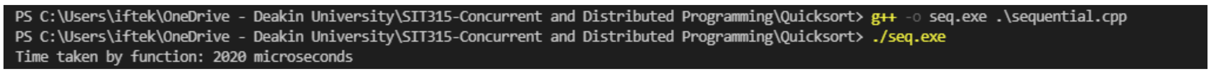
For sorting a size 1000 array:



```
PS C:\Users\iftek\OneDrive - Deakin University\SIT315-Concurrent and Distributed Programming\Quicksort> g++ -o seq.exe .\sequential.cpp
PS C:\Users\iftek\OneDrive - Deakin University\SIT315-Concurrent and Distributed Programming\Quicksort> ./seq.exe
Time taken by function: 62 microseconds
```

Sequential Takes 62ms

For sorting a size 10000 array:

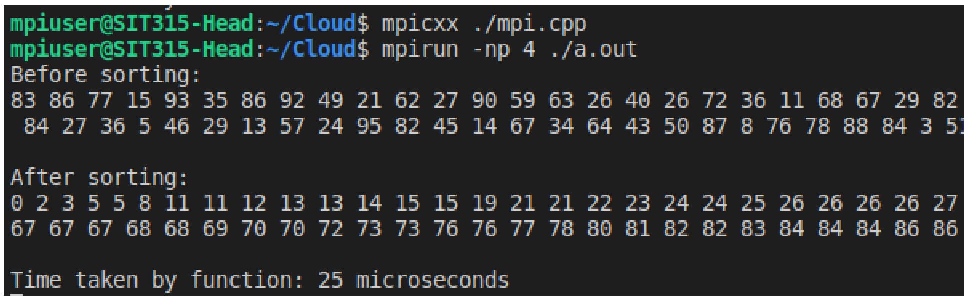


```
PS C:\Users\iftek\OneDrive - Deakin University\SIT315-Concurrent and Distributed Programming\Quicksort> g++ -o seq.exe .\sequential.cpp
PS C:\Users\iftek\OneDrive - Deakin University\SIT315-Concurrent and Distributed Programming\Quicksort> ./seq.exe
Time taken by function: 2020 microseconds
```

Sequential Takes 2020ms

Let’s look at the MPI Quicksort runtimes-

For sorting a size 100 array:



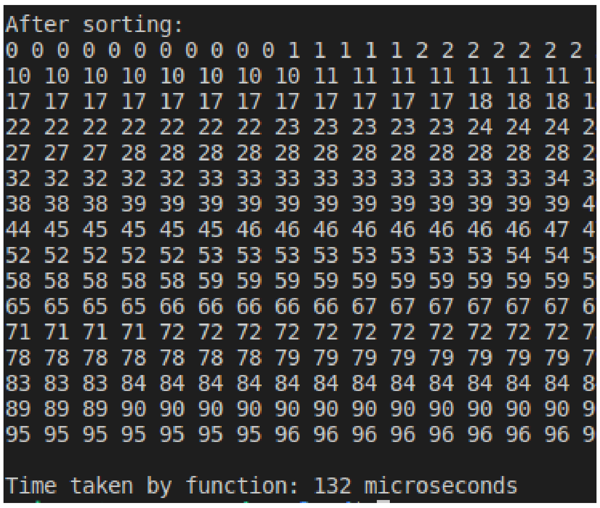
```
mpiuser@SIT315-Head:~/Cloud$ mpicxx ./mpi.cpp
mpiuser@SIT315-Head:~/Cloud$ mpirun -np 4 ./a.out
Before sorting:
83 86 77 15 93 35 86 92 49 21 62 27 90 59 63 26 40 26 72 36 11 68 67 29 82
 84 27 36 5 46 29 13 57 24 95 82 45 14 67 34 64 43 50 87 8 76 78 88 84 3 5

After sorting:
0 2 3 5 5 8 11 11 12 13 13 14 15 15 19 21 21 22 23 24 24 25 26 26 26 26 27
67 67 67 68 68 69 70 70 72 73 73 76 76 77 78 80 81 82 82 83 84 84 84 86 86

Time taken by function: 25 microseconds
```

MPI Takes 25ms

For sorting a size 1000 array:

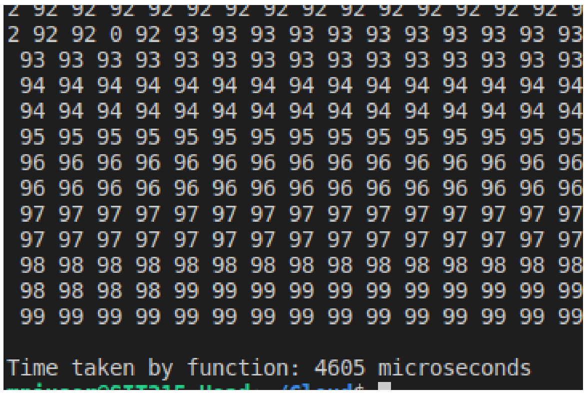


```
After sorting:
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 2 2
10 10 10 10 10 10 10 10 11 11 11 11 11 11 11 1
17 17 17 17 17 17 17 17 17 17 17 17 18 18 18 1
22 22 22 22 22 22 22 23 23 23 23 23 24 24 24 2
27 27 27 28 28 28 28 28 28 28 28 28 28 28 28 2
32 32 32 32 32 33 33 33 33 33 33 33 33 33 34 3
38 38 38 39 39 39 39 39 39 39 39 39 39 39 39 4
44 45 45 45 45 45 46 46 46 46 46 46 46 46 47 4
52 52 52 52 52 53 53 53 53 53 53 53 53 54 54 5
58 58 58 58 58 59 59 59 59 59 59 59 59 59 59 5
65 65 65 65 66 66 66 66 66 67 67 67 67 67 67 6
71 71 71 71 72 72 72 72 72 72 72 72 72 72 72 7
78 78 78 78 78 78 78 79 79 79 79 79 79 79 79 7
83 83 83 84 84 84 84 84 84 84 84 84 84 84 84 8
89 89 89 90 90 90 90 90 90 90 90 90 90 90 90 9
95 95 95 95 95 95 95 96 96 96 96 96 96 96 96 9

Time taken by function: 132 microseconds
```

MPI takes 132ms

For sorting a size 10000 array:



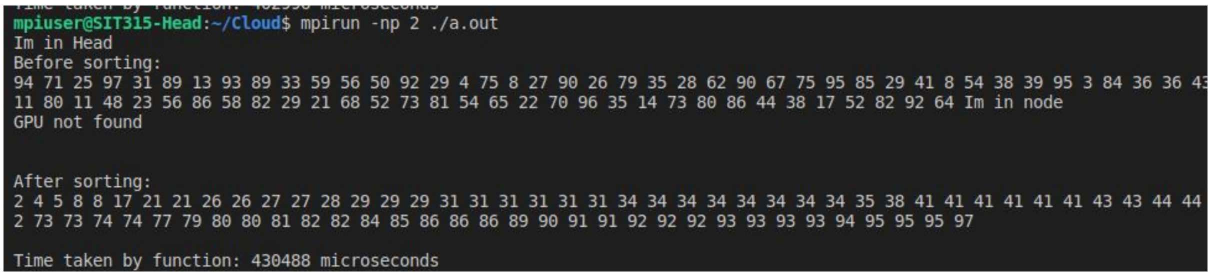
```
2 92 92 92 92 92 92 92 92 92 92 92 92 92 92 9
2 92 92 0 92 93 93 93 93 93 93 93 93 93 93 93
 93 93 93 93 93 93 93 93 93 93 93 93 93 93 93
 94 94 94 94 94 94 94 94 94 94 94 94 94 94 94
 94 94 94 94 94 94 94 94 94 94 94 94 94 94 94
 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95
 96 96 96 96 96 96 96 96 96 96 96 96 96 96 96
 96 96 96 96 96 96 96 96 96 96 96 96 96 96 96
 97 97 97 97 97 97 97 97 97 97 97 97 97 97 97
 97 97 97 97 97 97 97 97 97 97 97 97 97 97 97
 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98
 98 98 98 98 99 99 99 99 99 99 99 99 99 99 99
 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99

Time taken by function: 4605 microseconds
```

MPI takes 4605ms

Let’s look at the MPI+OpenCL Quicksort runtimes-

For sorting a size 100 array:



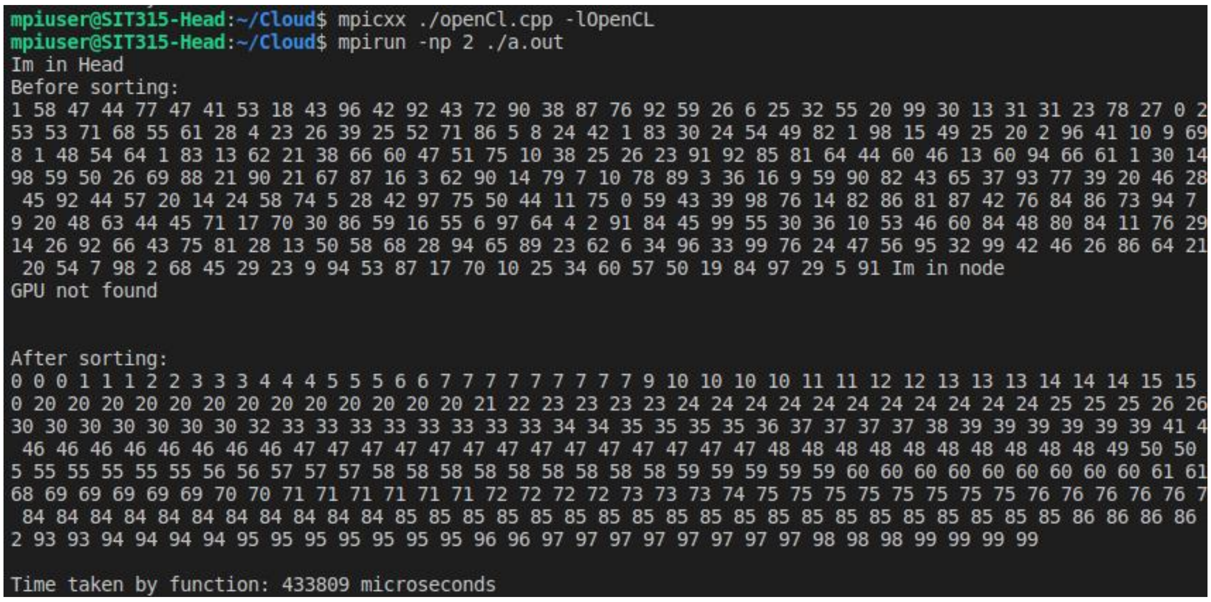
```
mpiuser@SIT315-Head:~/Cloud$ mpirun -np 2 ./a.out
Im in Head
Before sorting:
94 71 25 97 31 89 13 93 89 33 59 56 50 92 29 4 75 8 27 90 26 79 35 28 62 90 67 75 95 85 29 41 8 54 38 39 95 3 84 36 36 43
11 80 11 48 23 56 86 58 82 29 21 68 52 73 81 54 65 22 70 96 35 14 73 80 86 44 38 17 52 82 92 64 Im in node
GPU not found


After sorting:
2 4 5 8 8 17 21 21 26 26 27 27 28 29 29 29 31 31 31 31 31 31 34 34 34 34 34 34 34 34 35 38 41 41 41 41 41 41 43 43 44 44
2 73 73 74 74 77 79 80 80 81 82 82 84 85 86 86 86 89 90 91 91 92 92 92 93 93 93 93 94 95 95 95 97

Time taken by function: 430488 microseconds
```

OpenCL takes 430488ms

For sorting a size 500 array:



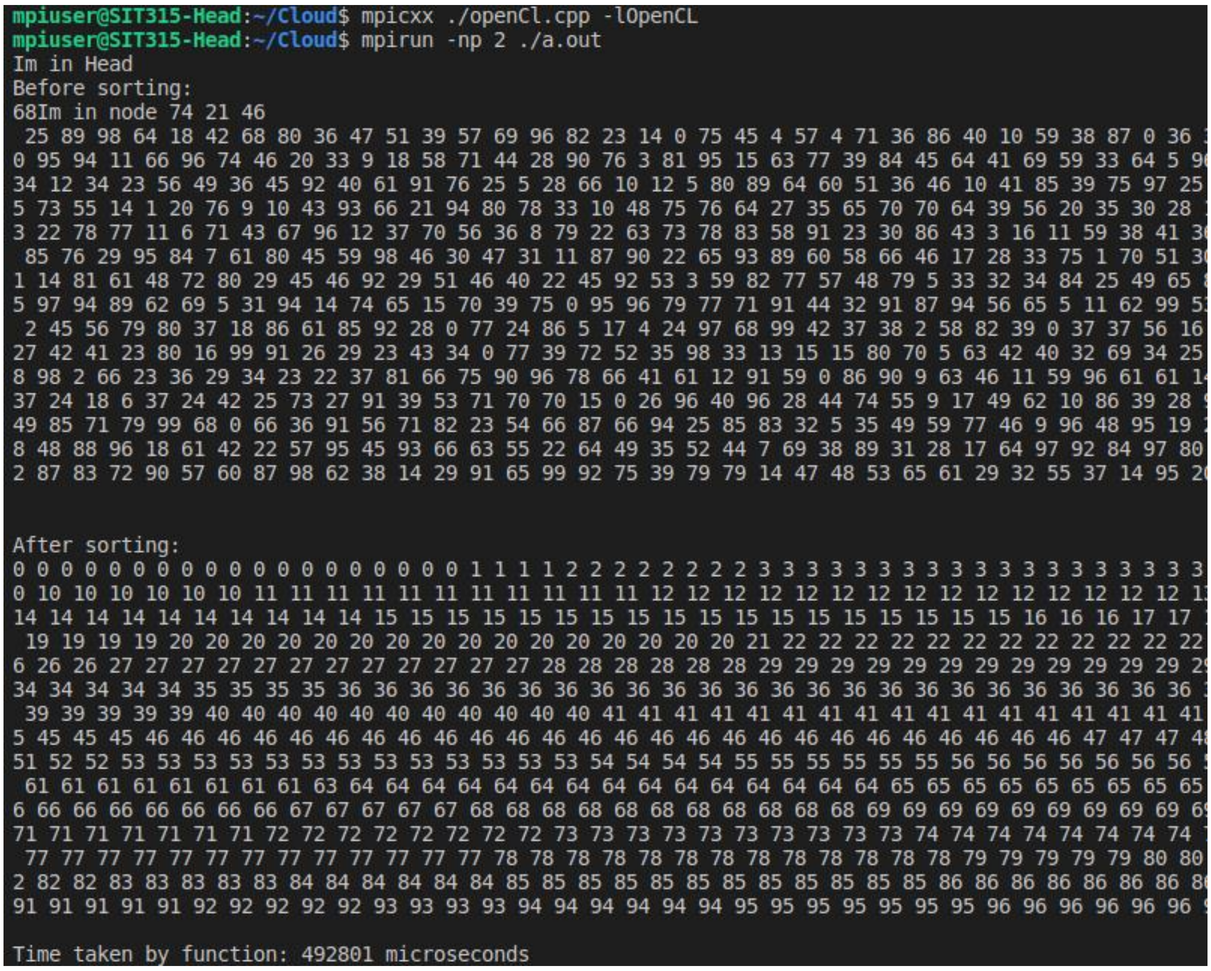
```
mpiuser@SIT315-Head:~/Cloud$ mpicxx ./openCl.cpp -lOpenCL
mpiuser@SIT315-Head:~/Cloud$ mpirun -np 2 ./a.out
Im in Head
Before sorting:
1 58 47 44 77 47 41 53 18 43 96 42 92 43 72 90 38 87 76 92 59 26 6 25 32 55 20 99 30 13 31 31 23 78 27 0 2
53 53 71 68 55 61 28 4 23 26 39 25 52 71 86 5 8 24 42 1 83 30 24 54 49 82 1 98 15 49 25 20 2 96 41 10 9 69
8 1 48 54 64 1 83 13 62 21 38 66 60 47 51 75 10 38 25 26 23 91 92 85 81 64 44 60 46 13 60 94 66 61 1 30 14
98 59 50 26 69 88 21 90 21 67 87 16 3 62 90 14 79 7 10 78 89 3 36 16 9 59 90 82 43 65 37 93 77 39 20 46 28
 45 92 44 57 20 14 24 58 74 5 28 42 97 75 50 44 11 75 0 59 43 39 98 76 14 82 86 81 87 42 76 84 86 73 94 7
9 20 48 63 44 45 71 17 70 30 86 59 16 55 6 97 64 4 2 91 84 45 99 55 30 36 10 53 46 60 84 48 80 84 11 76 29
14 26 92 66 43 75 81 28 13 50 58 68 28 94 65 89 23 62 6 34 96 33 99 76 24 47 56 95 32 99 42 46 26 86 64 21
 20 54 7 98 2 68 45 29 23 9 94 53 87 17 70 10 25 34 60 57 50 19 84 97 29 5 91 Im in node
GPU not found


After sorting:
0 0 0 1 1 1 2 2 3 3 3 4 4 4 5 5 5 6 6 7 7 7 7 7 7 7 7 7 9 10 10 10 10 11 11 12 12 13 13 13 14 14 14 15 15
0 20 20 20 20 20 20 20 20 20 20 20 20 20 21 22 23 23 23 23 24 24 24 24 24 24 24 24 24 24 24 25 25 25 26 26
30 30 30 30 30 30 30 32 33 33 33 33 33 33 33 33 34 34 35 35 35 35 36 37 37 37 37 38 39 39 39 39 39 39 41 4
 46 46 46 46 46 46 46 46 47 47 47 47 47 47 47 47 47 47 47 47 47 47 48 48 48 48 48 48 48 48 48 48 49 50 50
5 55 55 55 55 55 56 56 57 57 57 58 58 58 58 58 58 58 58 58 59 59 59 59 59 60 60 60 60 60 60 60 60 60 61 61
68 69 69 69 69 69 70 70 71 71 71 71 71 71 72 72 72 72 73 73 73 74 75 75 75 75 75 75 75 75 76 76 76 76 76 7
 84 84 84 84 84 84 84 84 84 84 84 85 85 85 85 85 85 85 85 85 85 85 85 85 85 85 85 85 85 85 85 86 86 86 86
2 93 93 94 94 94 94 95 95 95 95 95 95 95 96 96 97 97 97 97 97 97 97 97 98 98 98 99 99 99 99

Time taken by function: 433809 microseconds
```

OpenCL takes 433809ms

For sorting a size 1000 array:



```
mpiuser@SIT315-Head:~/Cloud$ mpicxx ./openCl.cpp -lOpenCL
mpiuser@SIT315-Head:~/Cloud$ mpirun -np 2 ./a.out
Im in Head
Before sorting:
68Im in node 74 21 46
 25 89 98 64 18 42 68 80 36 47 51 39 57 69 96 82 23 14 0 75 45 4 57 4 71 36 86 40 10 59 38 87 0 36
0 95 94 11 66 96 74 46 20 33 9 18 58 71 44 28 90 76 3 81 95 15 63 77 39 84 45 64 41 69 59 33 64 5 9
34 12 34 23 56 49 36 45 92 40 61 91 76 25 5 28 66 10 12 5 80 89 64 60 51 36 46 10 41 85 39 75 97 25
5 73 55 14 1 20 76 9 10 43 93 66 21 94 80 78 33 10 48 75 76 64 27 35 65 70 70 64 39 56 20 35 30 28
3 22 78 77 11 6 71 43 67 96 12 37 70 56 36 8 79 22 63 73 78 83 58 91 23 30 86 43 3 16 11 59 38 41 3
 85 76 29 95 84 7 61 80 45 59 98 46 30 47 31 11 87 90 22 65 93 89 60 58 66 46 17 28 33 75 1 70 51 3
1 14 81 61 48 72 80 29 45 46 92 29 51 46 40 22 45 92 53 3 59 82 77 57 48 79 5 33 32 34 84 25 49 65
5 97 94 89 62 69 5 31 94 14 74 65 15 70 39 75 0 95 96 79 77 71 91 44 32 91 87 94 56 65 5 11 62 99 5
 2 45 56 79 80 37 18 86 61 85 92 28 0 77 24 86 5 17 4 24 97 68 99 42 37 38 2 58 82 39 0 37 37 56 16
27 42 41 23 80 16 99 91 26 29 23 43 34 0 77 39 72 52 35 98 33 13 15 15 80 70 5 63 42 40 32 69 34 25
8 98 2 66 23 36 29 34 23 22 37 81 66 75 90 96 78 66 41 61 12 91 59 0 86 90 9 63 46 11 59 96 61 61 1
37 24 18 6 37 24 42 25 73 27 91 39 53 71 70 70 15 0 26 96 40 96 28 44 74 55 9 17 49 62 10 86 39 28
49 85 71 79 99 68 0 66 36 91 56 71 82 23 54 66 87 66 94 25 85 83 32 5 35 49 59 77 46 9 96 48 95 19
8 48 88 96 18 61 42 22 57 95 45 93 66 63 55 22 64 49 35 52 44 7 69 38 89 31 28 17 64 97 92 84 97 80
2 87 83 72 90 57 60 87 98 62 38 14 29 91 65 99 92 75 39 79 79 14 47 48 53 65 61 29 32 55 37 14 95 2


After sorting:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
0 10 10 10 10 10 10 11 11 11 11 11 11 11 11 11 11 11 12 12 12 12 12 12 12 12 12 12 12 12 12 12 1
14 14 14 14 14 14 14 14 14 14 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 16 16 16 17 17
 19 19 19 19 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 21 22 22 22 22 22 22 22 22 22 22 22 22
6 26 26 27 27 27 27 27 27 27 27 27 27 27 27 28 28 28 28 28 28 29 29 29 29 29 29 29 29 29 29 29 29 2
34 34 34 34 34 35 35 35 35 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
 39 39 39 39 39 40 40 40 40 40 40 40 40 40 40 40 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
5 45 45 45 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 47 47 47 4
51 52 52 53 53 53 53 53 53 53 53 53 53 53 53 53 54 54 54 54 55 55 55 55 55 55 56 56 56 56 56 56 56
 61 61 61 61 61 61 61 61 63 64 64 64 64 64 64 64 64 64 64 64 64 64 64 65 65 65 65 65 65 65 65 65
6 66 66 66 66 66 66 66 67 67 67 67 67 68 68 68 68 68 68 68 68 68 68 68 69 69 69 69 69 69 69 69 69 6
71 71 71 71 71 71 71 72 72 72 72 72 72 72 72 73 73 73 73 73 73 73 73 73 73 74 74 74 74 74 74 74 74
 77 77 77 77 77 77 77 77 77 77 77 77 77 78 78 78 78 78 78 78 78 78 78 78 78 78 79 79 79 79 79 80 80
2 82 82 83 83 83 83 83 84 84 84 84 84 84 85 85 85 85 85 85 85 85 85 85 85 85 86 86 86 86 86 86 86 8
91 91 91 91 91 92 92 92 92 92 93 93 93 93 94 94 94 94 94 94 95 95 95 95 95 95 95 96 96 96 96 96 96

Time taken by function: 492801 microseconds
```

OpenCL takes 492801

Conclusion

As we can see, from the above runtimes, sequential and MPI implementation have similar performances. However, with OPENCL+MPI implementation, the quicksort function is worst with smaller sized arrays. This is obvious because there is a lot of overhead when we are using OpenCL. The overhead in implementing the parallel sort overwhelms any performance gain that we could have obtained. With increasing array sizes, OpenCL+MPI will perform better, and operations will be much quicker.

The concept behind the decomposition of MPI version of the quicksort is similar to the sequential quicksort. The difference lies in parallelising the sorting function. After partition, higher end will be sorted in one node and lower partition will be sent to a different child node. Each then makes a recursive call and sorts itself. If the array size is too small to have distributed nodes, then it performs normal quicksort which saves a lot of computational power.

With addition of OpenCl, I was having difficulty with running the recursive quicksort function. Hence, I implemented an iterative quicksort using the concept of stacks. A similar concept here when compared to the MPI version. Instead, we are using different buffers to store each partition, reading from the buffer, sorting it and then returning it to an output array.