CSCE 735 Fall 2022

Name: Rohan Chaudhury
UIN: 432001358

HW 4: Parallel Programming with MPI – Hypercube Quicksort

1. (80 points) Complete the MPI-based code provided in qsort_hypercube.cpp to implement the parallel quicksort algorithm for a d-dimensional hypercube with p=2d processors. 60 points will be awarded if the code compiles and executes the following command successfully.
mpirun -np 2 ./qsort_hypercube.exe 4 -1
5 points will be awarded for each of the following tests that are executed successfully.
mpirun -np 4 ./qsort_hypercube.exe 4 -2
mpirun -np 8 ./qsort_hypercube.exe 4 -1
mpirun -np 16 ./qsort_hypercube.exe 4 0
mpirun -np 16 ./qsort_hypercube.exe 20480000 0

Answer:

The submitted zip file contains the parallelized code. All the above-mentioned tests had executed successfully

Following are the execution logs:

```
[rohan.chaudhury@grace1 HW4-735]$ mpirun -np 2 ./qsort_hypercube.exe 4 -1
[Proc: 0] number of processes = 2, initial local list size = 4, hypercube quicksort time = 0.001501
[Proc: 0] Congratulations. The list has been sorted correctly.
[rohan.chaudhury@grace1 HW4-735]$ mpirun -np 4 ./qsort_hypercube.exe 4 -2
[Proc: 0] number of processes = 4, initial local list size = 4, hypercube quicksort time = 0.003985
[Proc: 0] Congratulations. The list has been sorted correctly.
[rohan.chaudhury@grace1 HW4-735]$ mpirun -np 8 ./qsort_hypercube.exe 4 -1
[Proc: 0] number of processes = 8, initial local list size = 4, hypercube quicksort time = 0.004764
[Proc: 0] Congratulations. The list has been sorted correctly.
[rohan.chaudhury@grace1 HW4-735]$ mpirun -np 16 ./qsort_hypercube.exe 4 0
[Proc: 0] number of processes = 16, initial local list size = 4, hypercube quicksort time = 0.006210
[Proc: 0] Congratulations. The list has been sorted correctly.
[rohan.chaudhury@grace1 HW4-735]$ mpirun -np 16 ./qsort_hypercube.exe 20480000 0
[Proc: 0] number of processes = 16, initial local list size = 20480000, hypercube quicksort time = 2.437129
```

[Proc: 0] Congratulations. The list has been sorted correctly.

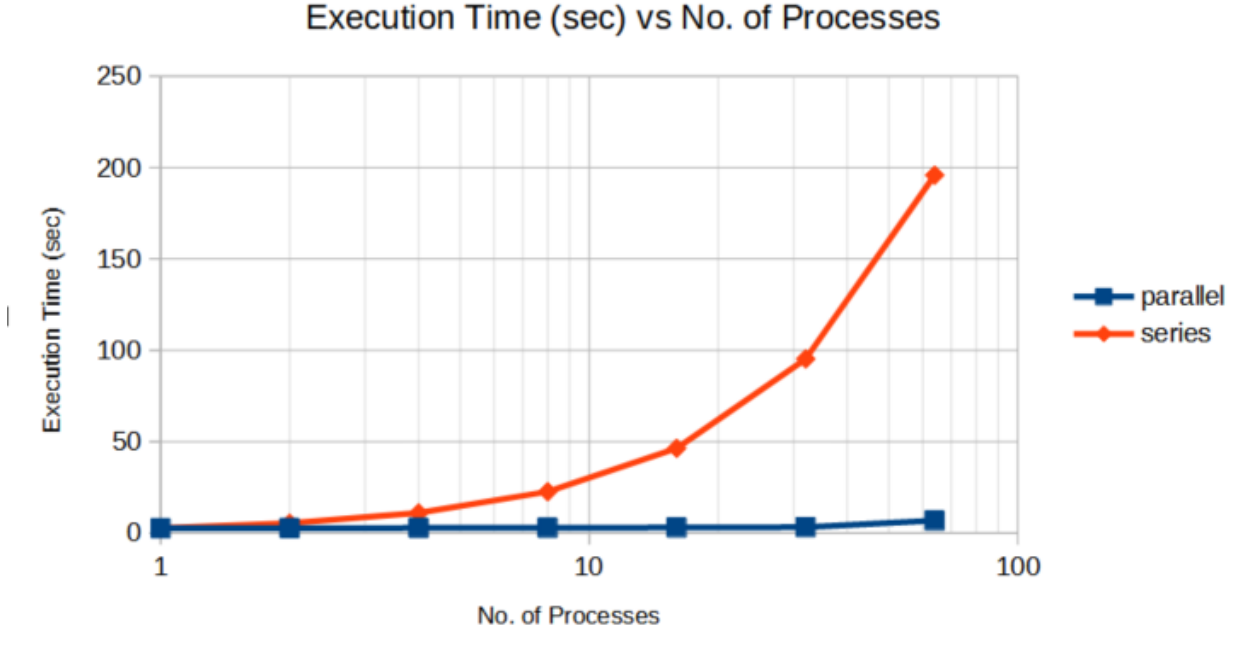
2. (5 points) Weak Scalability Study: Run your code to sort a distributed list of size n×p where n is the size of the local list on each process and p is the number of processes. For your experiments, use n=20,480,000 and p = 1, 2, 4, 8, 16, 32, and 64. Set type=0. Plot the execution time, speedup, and efficiency of your code as a function of p. Use logarithmic scale for the x-axis.

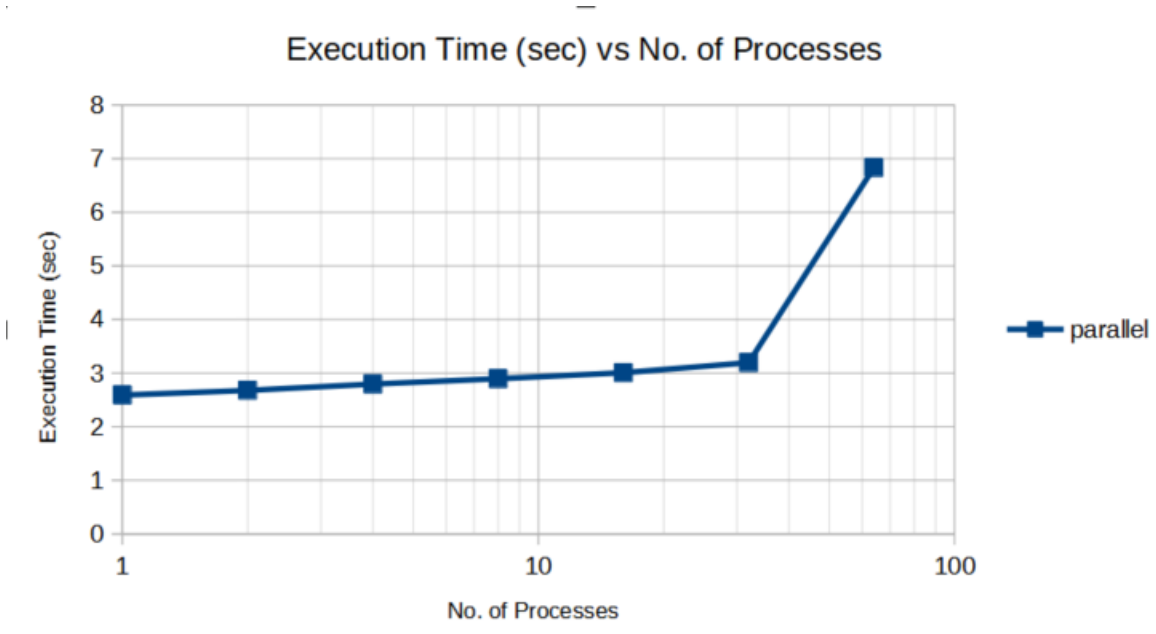
Note that the size of the list to be sorted is proportional to the number of processes p. In order to get speedup for a specific value of p, you need to determine the execution time to sort a list of size n×p with one process. As an example, speedup for p = 4 is the ratio of execution time for a list of size 81,920,000 with one process (T1) to the execution time for a list of size 20,480,000 with 4 processes (T4).

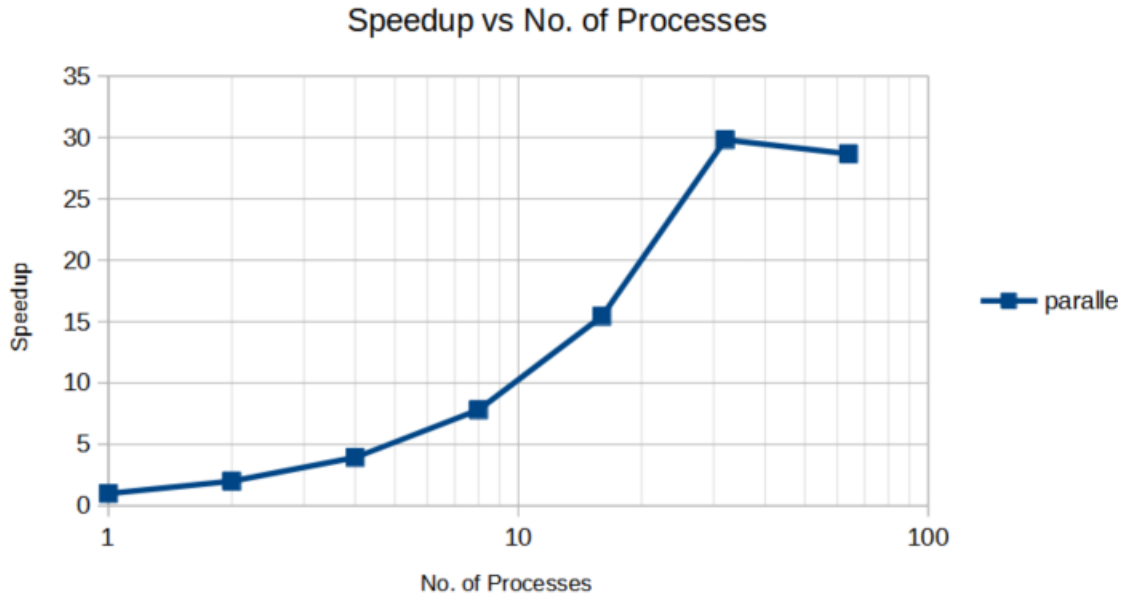
Answer:

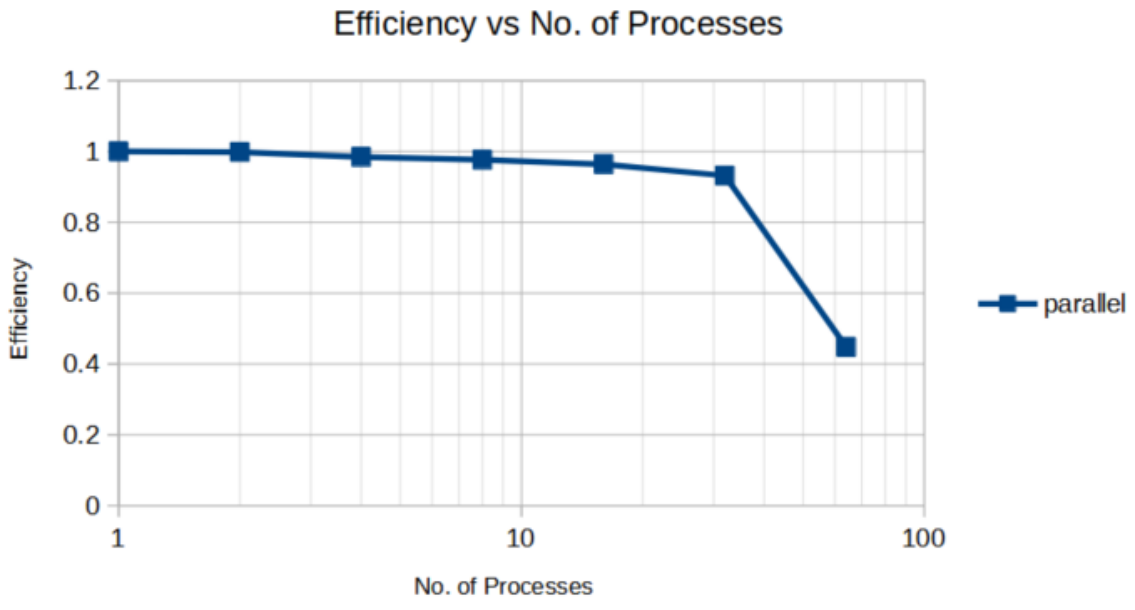
[Proc: 0] number of processes	1	initial local list size	20480000	hypercube quicksort time	2.592332	Speedup	1	Efficiency	1
[Proc: 0] number of processes	2	initial local list size	20480000	hypercube quicksort time	2.680082	Speedup	1.997016136	Efficiency	0.998508068
[Proc: 0] number of processes	4	initial local list size	20480000	hypercube quicksort time	2.794981	Speedup	3.937843585	Efficiency	0.984460896
[Proc: 0] number of processes	8	initial local list size	20480000	hypercube quicksort time	2.895844	Speedup	7.81183724	Efficiency	0.976479655
[Proc: 0] number of processes	16	initial local list size	20480000	hypercube quicksort time	3.00875	Speedup	15.42730104	Efficiency	0.964206315
[Proc: 0] number of processes	32	initial local list size	20480000	hypercube quicksort time	3.19649	Speedup	29.81502116	Efficiency	0.931719411
[Proc: 0] number of processes	64	initial local list size	20480000	hypercube quicksort time	6.832758	Speedup	28.65798057	Efficiency	0.447780946

[Proc: 0] number of processes	1	initial local list size	20480000	hypercube quicksort time	2.595553
[Proc: 0] number of processes	1	initial local list size	40960000	hypercube quicksort time	5.352167
[Proc: 0] number of processes	1	initial local list size	81920000	hypercube quicksort time	11.006198
[Proc: 0] number of processes	1	initial local list size	163840000	hypercube quicksort time	22.621862
[Proc: 0] number of processes	1	initial local list size	327680000	hypercube quicksort time	46.416892
[Proc: 0] number of processes	1	initial local list size	655360000	hypercube quicksort time	95.303417
[Proc: 0] number of processes	1	initial local list size	1310720000	hypercube quicksort time	195.813046




Execution Time (sec) vs No. of Processes
Execution Time (sec)
8
7
6
5
4
3
2
1
0
1
10
100
No. of Processes
parallel


Speedup vs No. of Processes
Speedup
35
30
25
20
15
10
5
0
1
10
100
No. of Processes
paralle

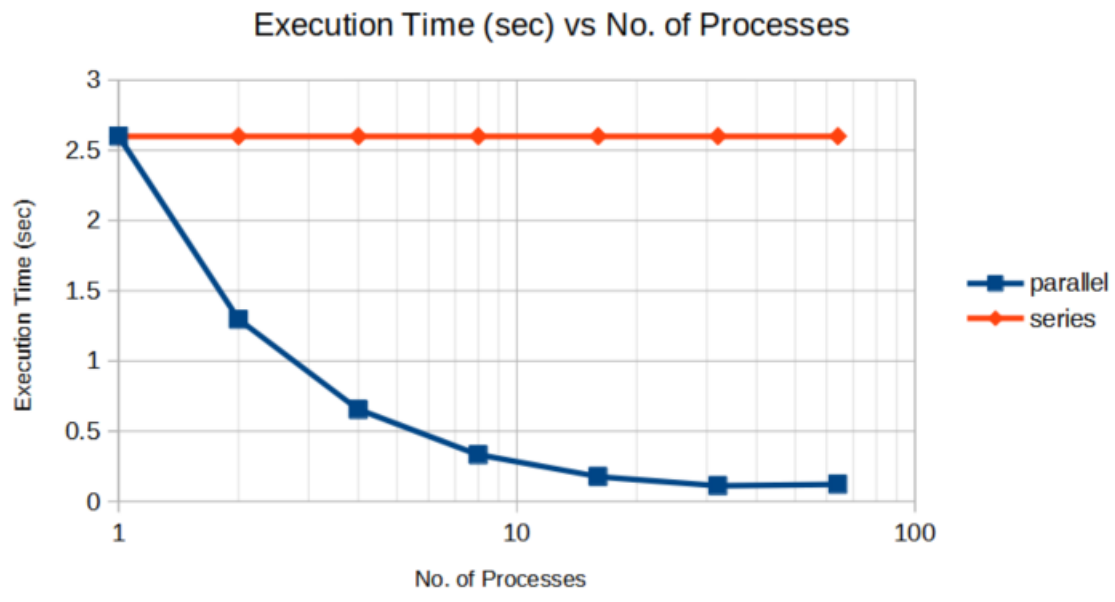


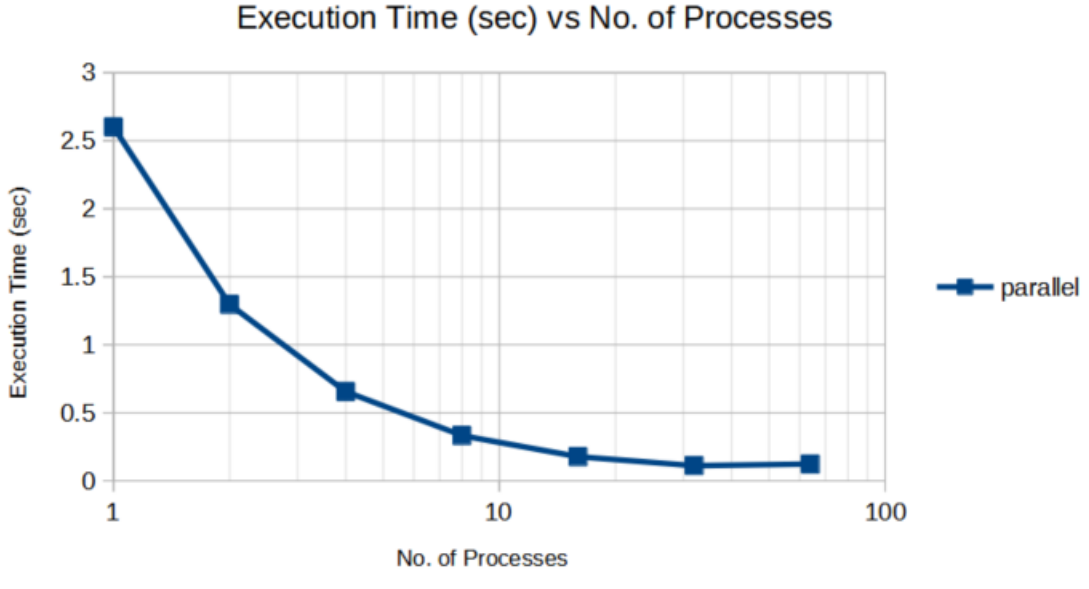
3. (5 points) Strong Scalability Study: Now run your code with n=20,480,000/p where p = 1, 2, 4, 8, 16, 32, and 64. Set type=0. Plot the execution time, speedup, and efficiency of your code as a function of p. Use logarithmic scale for the x-axis.

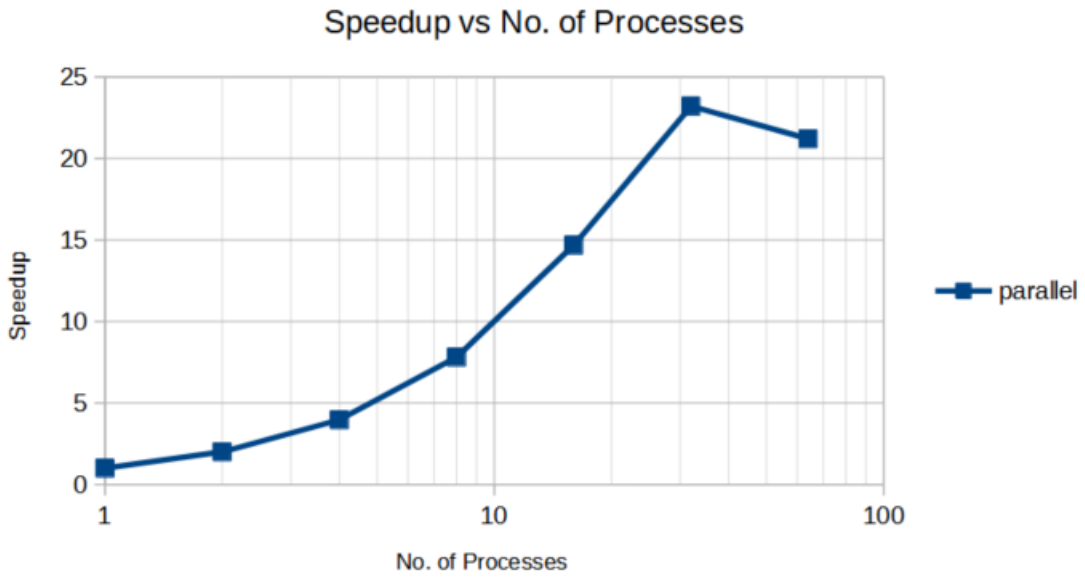
Unlike the weak scalability study, here the size of the list to be sorted remains unchanged at 20,480,000 even as you increase the number of processes. To determine speedup for any p you need to compare the execution time on p processes to the execution time for a list of size 20,480,000 with one process.

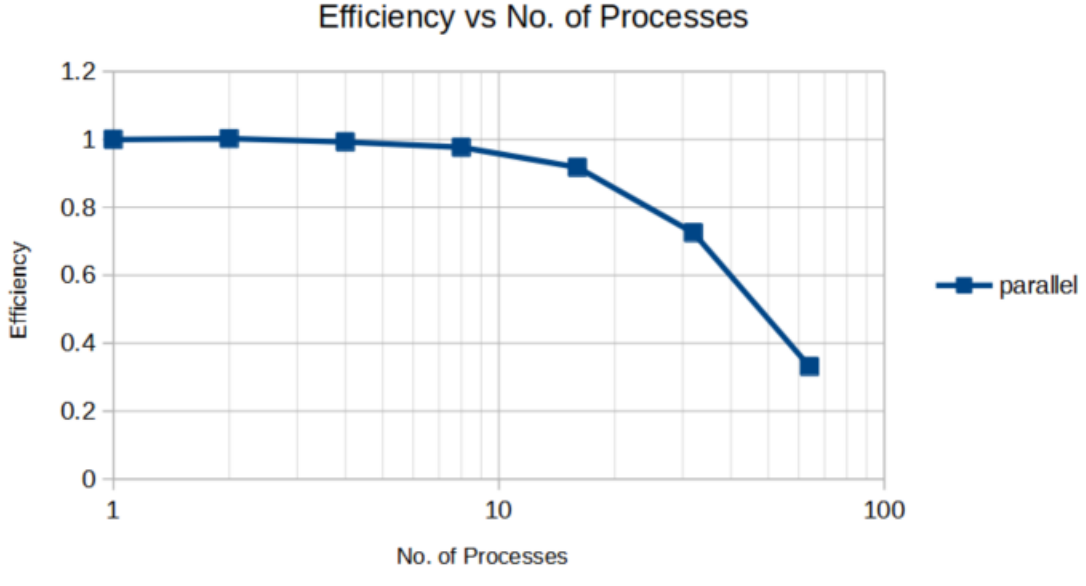
Answer:

[Proc: 0] number of processes	1	initial local list size	20480000	hypercube quicksort time	2.600399	Speedup	1	Efficiency	1
[Proc: 0] number of processes	2	initial local list size	10240000	hypercube quicksort time	1.296899	Speedup	2.00508983351826	Efficiency	1.002544917
[Proc: 0] number of processes	4	initial local list size	5120000	hypercube quicksort time	0.655077	Speedup	3.96960815293469	Efficiency	0.992402038
[Proc: 0] number of processes	8	initial local list size	2560000	hypercube quicksort time	0.332855	Speedup	7.81240780520046	Efficiency	0.976550976
[Proc: 0] number of processes	16	initial local list size	1280000	hypercube quicksort time	0.177117	Speedup	14.6818148455541	Efficiency	0.917613428
[Proc: 0] number of processes	32	initial local list size	640000	hypercube quicksort time	0.112048	Speedup	23.207901970584	Efficiency	0.725246937
[Proc: 0] number of processes	64	initial local list size	320000	hypercube quicksort time	0.122626	Speedup	21.2059351197952	Efficiency	0.331342736


Execution Time (sec) vs No. of Processes
Execution Time (sec)
3
2.5
2
1.5
1
0.5
0
1
10
100
No. of Processes
parallel
series


Execution Time (sec) vs No. of Processes
Execution Time (sec)
3
2.5
2
1.5
1
0.5
0
1
10
100
No. of Processes
parallel





4. (10 points) Modify the code to sort the list in descending order. Submit the modified code as `qsort_hypercube_descending.cpp`. 2 points will be awarded for each of the tests in Problem 1 that are executed successfully. (Note that the `check_list` routine needs to be modified to verify descending order.)

Answer:

The submitted zip file contains the parallelized code for sorting in descending order. All the above-mentioned tests in Question 1 had executed successfully.

Following are the execution logs:

```
[rohan.chaudhury@grace1 HW4-735]$ mpirun -np 2 ./qsort_hypercube_descending.exe 4 -1
```

[Proc: 0] number of processes = 2, initial local list size = 4, hypercube quicksort time = 0.005703

[Proc: 0] Congratulations. The list has been sorted correctly.

[rohan.chaudhury@grace1 HW4-735]$ mpirun -np 4 ./qsort_hypercube_descending.exe 4 -2

[Proc: 0] number of processes = 4, initial local list size = 4, hypercube quicksort time = 0.001457

[Proc: 0] Congratulations. The list has been sorted correctly.

[rohan.chaudhury@grace1 HW4-735]$ mpirun -np 8 ./qsort_hypercube_descending.exe 4 -1

[Proc: 0] number of processes = 8, initial local list size = 4, hypercube quicksort time = 0.002609

[Proc: 0] Congratulations. The list has been sorted correctly.

[rohan.chaudhury@grace1 HW4-735]$ mpirun -np 16 ./qsort_hypercube_descending.exe 4 0

[Proc: 0] number of processes = 16, initial local list size = 4, hypercube quicksort time = 0.006600

[Proc: 0] Congratulations. The list has been sorted correctly.

[rohan.chaudhury@grace1 HW4-735]$ mpirun -np 16 ./qsort_hypercube_descending.exe 20480000 0

[Proc: 0] number of processes = 16, initial local list size = 20480000, hypercube quicksort time = 2.419954

[Proc: 0] Congratulations. The list has been sorted correctly.