Jessica Salguero, UIN: 937005006
CSCE 735, Summer 2025

# HW 4: Parallel Programming with MPI – Hypercube Quicksort

**Question 1**

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

*Question 1 Grace Results*

```
mpirun -np  2 ./qsort_hypercube.exe 4 -1
[Proc: 0] number of processes = 2, initial local list size = 4, hypercube quicksort time = 0.000658
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np  4 ./qsort_hypercube.exe 4 -2
[Proc: 0] number of processes = 4, initial local list size = 4, hypercube quicksort time = 0.032606
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np  8 ./qsort_hypercube.exe 4 -1
[Proc: 0] number of processes = 8, initial local list size = 4, hypercube quicksort time = 0.041254
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np  16 ./qsort_hypercube.exe 4 0
[Proc: 0] number of processes = 16, initial local list size = 4, hypercube quicksort time = 0.004455
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np  16 ./qsort_hypercube.exe 20480000 0
[Proc: 0] number of processes = 16, initial local list size = 20480000, hypercube quicksort time =
2.623379
[Proc: 0] Congratulations. The list has been sorted correctly.
```

Jessica Salguero, UIN: 937005006
CSCE 735, Summer 2025

**Question 2**

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 xaxis.

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).

*Question 2 - Grace Results*

```
mpirun -np 1 ./qsort_hypercube.exe 20480000 0
[Proc: 0] number of processes = 1, initial local list size = 20480000, hypercube quicksort time =
1.790578
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np 2 ./qsort_hypercube.exe 20480000 0
[Proc: 0] number of processes = 2, initial local list size = 20480000, hypercube quicksort time =
2.311683
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np 4 ./qsort_hypercube.exe 20480000 0
[Proc: 0] number of processes = 4, initial local list size = 20480000, hypercube quicksort time =
2.425412
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np 8 ./qsort_hypercube.exe 20480000 0
[Proc: 0] number of processes = 8, initial local list size = 20480000, hypercube quicksort time =
2.485498
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np 16 ./qsort_hypercube.exe 20480000 0
[Proc: 0] number of processes = 16, initial local list size = 20480000, hypercube quicksort time =
2.619511
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np 32 ./qsort_hypercube.exe 20480000 0
[Proc: 0] number of processes = 32, initial local list size = 20480000, hypercube quicksort time =
2.817704
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np 64 ./qsort_hypercube.exe 20480000 0
[Proc: 0] number of processes = 64, initial local list size = 20480000, hypercube quicksort time =
2.828627
[Proc: 0] Congratulations. The list has been sorted correctly.
```

Jessica Salguero, UIN: 937005006
CSCE 735, Summer 2025

## *Question 1 - Weak Scalability T1 Results*

```
T1 for Weak Scalability

p=1, n=20480000
[Proc: 0] number of processes = 1, initial local list size = 20480000, hypercube quicksort time =
1.754402
[Proc: 0] Congratulations. The list has been sorted correctly.

p=2, n=40960000
[Proc: 0] number of processes = 1, initial local list size = 40960000, hypercube quicksort time =
3.607869
[Proc: 0] Congratulations. The list has been sorted correctly.

p=4, n=81920000
[Proc: 0] number of processes = 1, initial local list size = 81920000, hypercube quicksort time =
7.453744
[Proc: 0] Congratulations. The list has been sorted correctly.

p=8, n=163840000
[Proc: 0] number of processes = 1, initial local list size = 163840000, hypercube quicksort time =
15.184951
[Proc: 0] Congratulations. The list has been sorted correctly.

p=16, n=327680000
[Proc: 0] number of processes = 1, initial local list size = 327680000, hypercube quicksort time =
31.241057
[Proc: 0] Congratulations. The list has been sorted correctly.

p=32, n=655360000
[Proc: 0] number of processes = 1, initial local list size = 655360000, hypercube quicksort time =
64.155652
[Proc: 0] Congratulations. The list has been sorted correctly.

p=64, n=1310720000
[Proc: 0] number of processes = 1, initial local list size = 1310720000, hypercube quicksort time =
131.482391
[Proc: 0] Congratulations. The list has been sorted correctly.
```

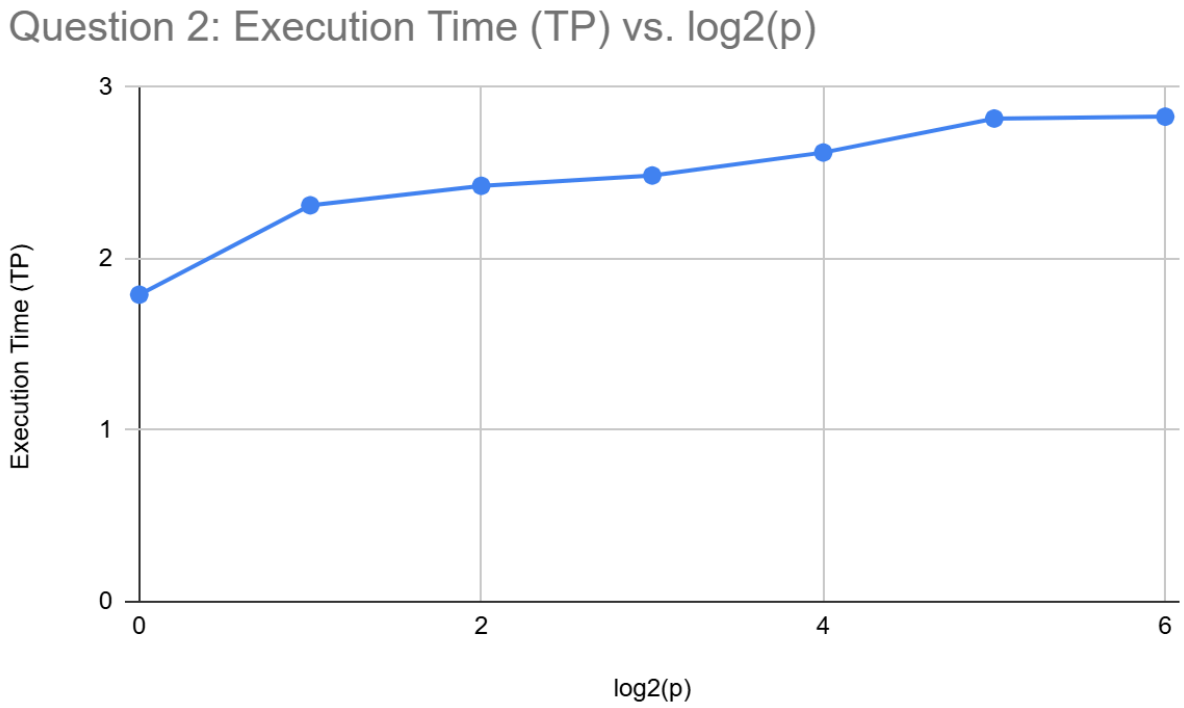
## *Question 2 - Results Table*

| Processors | log2(p) | List Size | TP Execution Time | T1 Execution Time | Speed Up (T1/TP) | Efficiency (SpeedUp/P) |
|---|---|---|---|---|---|---|
| 1 | 0 | 20480000 | 1.790578 | 1.754402 | 0.979796468 | 0.979796468 |
| 2 | 1 | 20480000 | 2.311683 | 3.607869 | 1.560710963 | 0.7803554813 |
| 4 | 2 | 20480000 | 2.425412 | 7.453744 | 3.073186741 | 0.7682966853 |
| 8 | 3 | 20480000 | 2.485498 | 15.184951 | 6.109419923 | 0.7636774904 |
| 16 | 4 | 20480000 | 2.619511 | 31.241057 | 11.9262935 | 0.7453933435 |
| 32 | 5 | 20480000 | 2.817704 | 64.155652 | 22.76876918 | 0.7115240369 |
| 64 | 6 | 20480000 | 2.828627 | 131.482391 | 46.48276036 | 0.7262931307 |

Jessica Salguero, UIN: 937005006
CSCE 735, Summer 2025

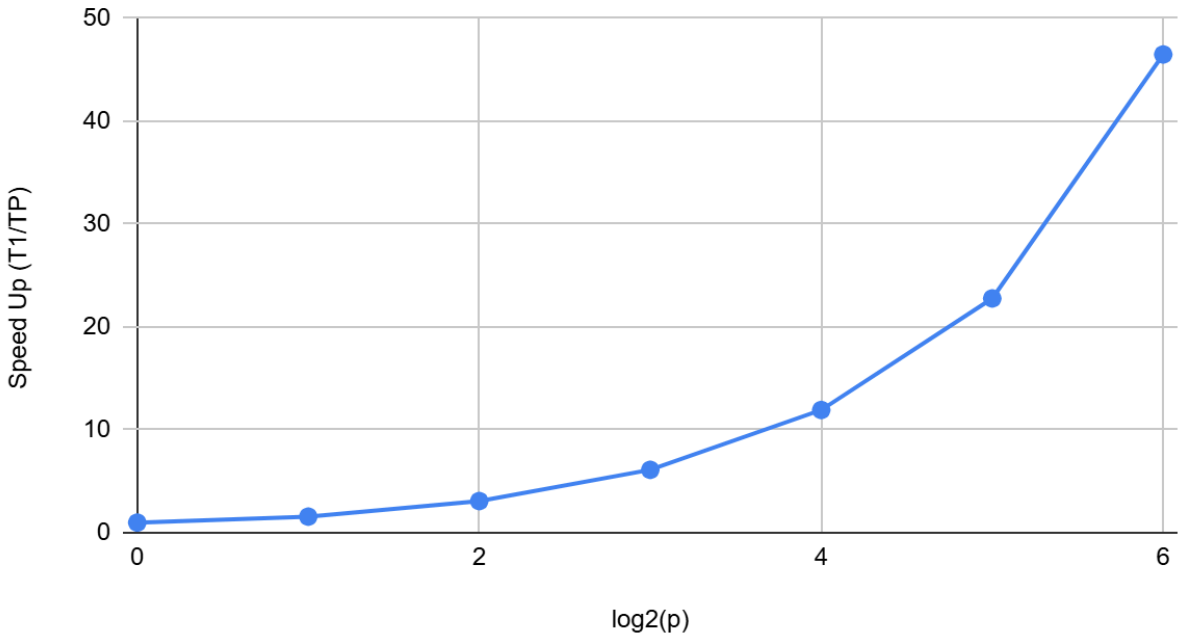
*Question 3 - Graphs*

With this strong scalability test consistently utilizing n=20,480,000, we can see that execution time and speedup have overall upward trajectories; efficiency has an overall downward trajectories. As the number of processors increases from 1 to 64, it is expected that execution time will slightly increase and efficiency will decrease a bit due to the overhead required for processors to communicate with each other. However, speedup still improves as the number of processors increases.

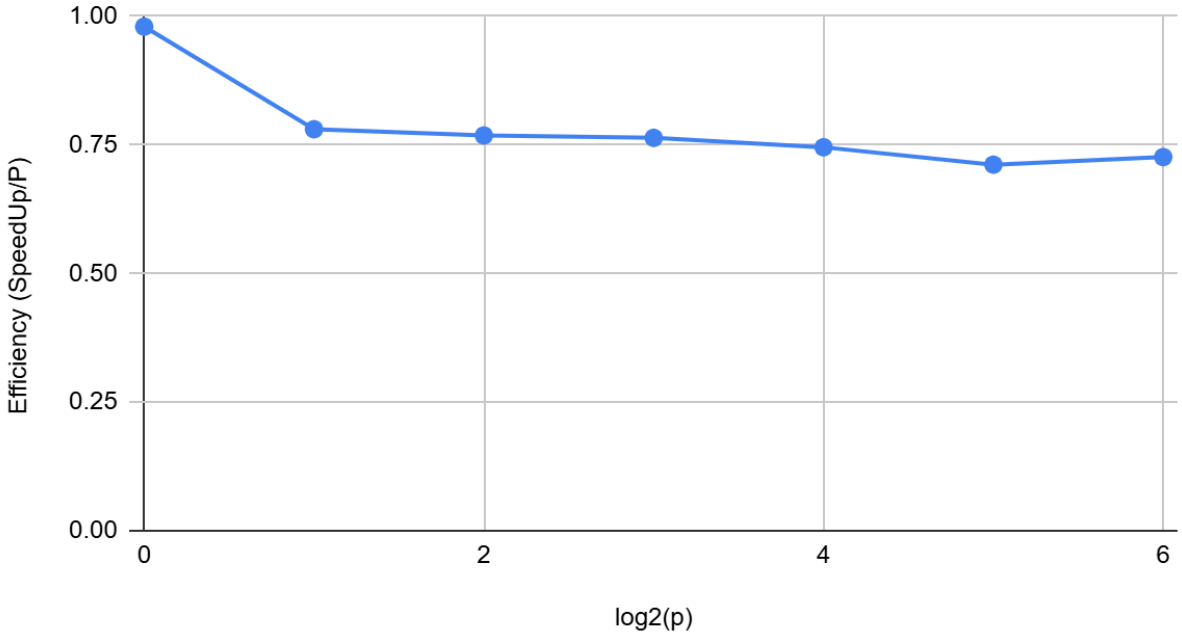


Jessica Salguero, UIN: 937005006
CSCE 735, Summer 2025

## Question 2: Speed Up (T1/TP) vs. log2(p)




## Question 2: Efficiency (SpeedUp/P) vs. log2(p)



Jessica Salguero, UIN: 937005006
CSCE 735, Summer 2025

**Question 3**

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.

*Question 3 - Grace Results*

```
mpirun -np 1 ./qsort_hypercube.exe 20480000 0
[Proc: 0] number of processes = 1, initial local list size = 20480000, hypercube quicksort time =
1.786074
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np 2 ./qsort_hypercube.exe 10240000 0
[Proc: 0] number of processes = 2, initial local list size = 10240000, hypercube quicksort time =
1.121369
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np 4 ./qsort_hypercube.exe 5120000 0
[Proc: 0] number of processes = 4, initial local list size = 5120000, hypercube quicksort time = 0.580270
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np 8 ./qsort_hypercube.exe 2560000 0
[Proc: 0] number of processes = 8, initial local list size = 2560000, hypercube quicksort time = 0.300566
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np 16 ./qsort_hypercube.exe 1280000 0
[Proc: 0] number of processes = 16, initial local list size = 1280000, hypercube quicksort time =
0.161930
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np 32 ./qsort_hypercube.exe 640000 0
[Proc: 0] number of processes = 32, initial local list size = 640000, hypercube quicksort time = 0.398153
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np 64 ./qsort_hypercube.exe 320000 0
[Proc: 0] number of processes = 64, initial local list size = 320000, hypercube quicksort time = 0.082610
[Proc: 0] Congratulations. The list has been sorted correctly.
```

*Question 3 - Strong Scalability T1 Calculations*

```
T1 for Strong Scalability

p=1, n=20480000
[Proc: 0] number of processes = 1, initial local list size = 20480000, hypercube quicksort time =
2.238161
[Proc: 0] Congratulations. The list has been sorted correctly.
```

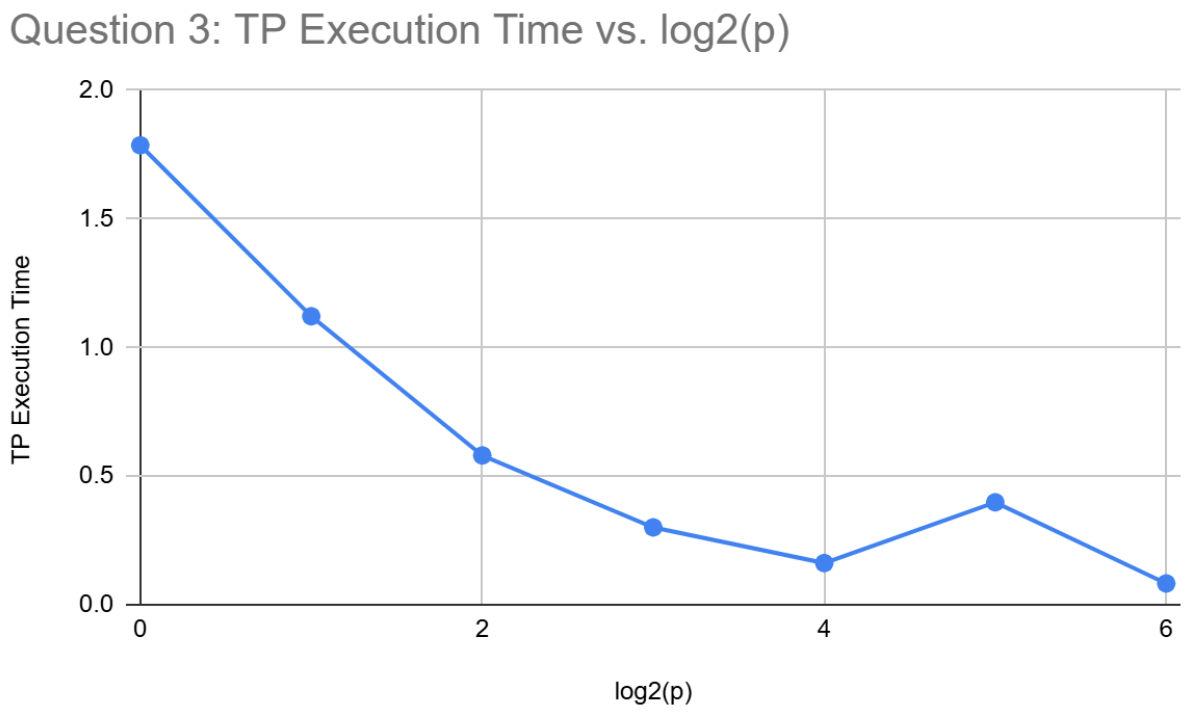
Jessica Salguero, UIN: 937005006
CSCE 735, Summer 2025

*Question 3 - Results Table*

| Processors | log2(p) | List Size | TP Execution Time | T1 Execution Time | Speed Up (T1/TP) | Efficiency (SpeedUp/P) |
|---|---|---|---|---|---|---|
| 1 | 0 | 20480000 | 1.786074 | 2.238161 | 1.253117732 | 1.253117732 |
| 2 | 1 | 10240000 | 1.121369 | 2.238161 | 1.995918382 | 0.997959191 |
| 4 | 2 | 5120000 | 0.58027 | 2.238161 | 3.857102728 | 0.964275682 |
| 8 | 3 | 2560000 | 0.300566 | 2.238161 | 7.446487627 | 0.9308109533 |
| 16 | 4 | 1280000 | 0.16193 | 2.238161 | 13.82178102 | 0.8638613135 |
| 32 | 5 | 640000 | 0.398153 | 2.238161 | 5.621359126 | 0.1756674727 |
| 64 | 6 | 320000 | 0.08261 | 2.238161 | 27.09310011 | 0.4233296892 |

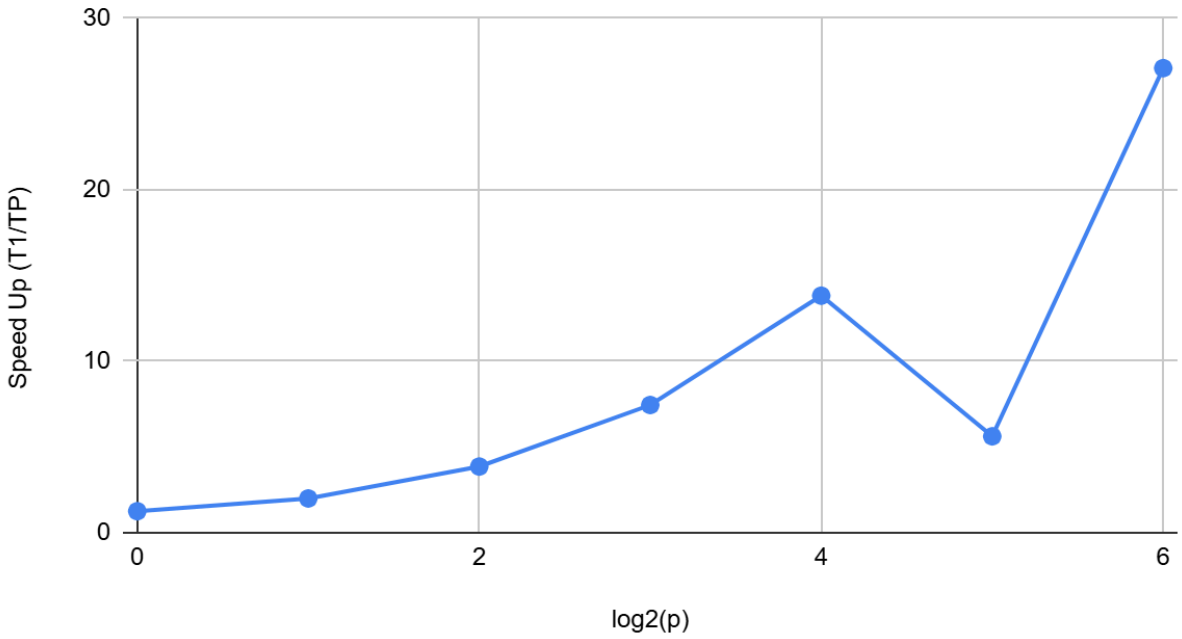
*Question 3 - Graphs*

With this strong scalability test utilizing n=20,480,000/p, we can see that execution time and efficiency have overall downward trajectories; speedup has an overall upward trajectory. As the number of processors increases, speedup is greatly improved while execution time simultaneously decreases a lot. Efficiency likely decreased a bit because of increased overhead associated with more processors needing to communicate, but efficiency had a much larger decrease when tested with more than 16 processors. Overall, the decrease in execution time shows strong scalability.

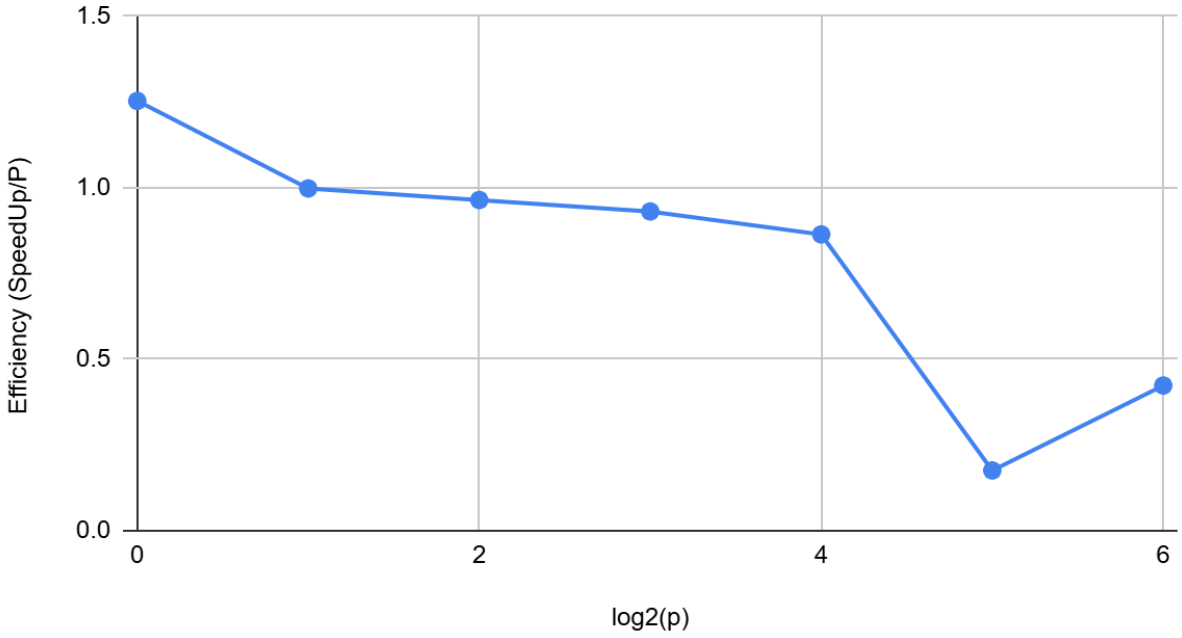


Jessica Salguero, UIN: 937005006
CSCE 735, Summer 2025

## Question 3: Speed Up (T1/TP) vs. log2(p)




## Question 3: Efficiency (SpeedUp/P) vs. log2(p)



Jessica Salguero, UIN: 937005006
CSCE 735, Summer 2025

**Question 4**

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.)

*Question 4 - Grace Results*

```
mpirun -np  2 ./qsort_hypercube_descending.exe 4 -1
[Proc: 0] number of processes = 2, initial local list size = 4, hypercube quicksort time = 0.002191
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np  4 ./qsort_hypercube_descending.exe 4 -2
[Proc: 0] number of processes = 4, initial local list size = 4, hypercube quicksort time = 0.007824
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np  8 ./qsort_hypercube_descending.exe 4 -1
[Proc: 0] number of processes = 8, initial local list size = 4, hypercube quicksort time = 0.009067
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np  16 ./qsort_hypercube_descending.exe 4 0
[Proc: 0] number of processes = 16, initial local list size = 4, hypercube quicksort time = 0.225625
[Proc: 0] Congratulations. The list has been sorted correctly.

mpirun -np  16 ./qsort_hypercube_descending.exe 20480000 0
[Proc: 0] number of processes = 16, initial local list size = 20480000, hypercube quicksort time =
2.152443
[Proc: 0] Congratulations. The list has been sorted correctly.
```