HW 4: Parallel Programming with MPI – Hypercube Quicksort

Mian Qin, UIN: 725006574

1&4, Complete the MPI-based code provided in qsort_hypercube.c and qsort_hypercube_descending.c to sort the list in ascending and descending order.

The source codes are zipped into zipfile. The build script and verify job script (for the five test cases) is also included.

2, Weak Scalability Study: Run your code to sort a distributed list of size np where n is the size of the local list on each process and p is the number of processes. Plot the execution time, speedup, and efficiency of your code as a function of p. Use logarithmic scale for the x-axis.

The experiments is done on **ada cluster**. All results (for both weak and strong scalability studies) are collected through averaging three independent running results (variance is very small). All experiments are using settings of **ptile=4** (4 cores per node).

Fig.1 demonstrated the execution time for weak scalability study on different number of processes. Fig.2 shows the speedup and efficiency on different number of processes for weak scalability study. From Fig. 2 we can see that the scaling is good for processes under 4. With more processes the scaling start to go down (same trend with efficiency). I think this is due to communication overhead across nodes, since we have fixed number of processes per node which is 4 (ptile=4).

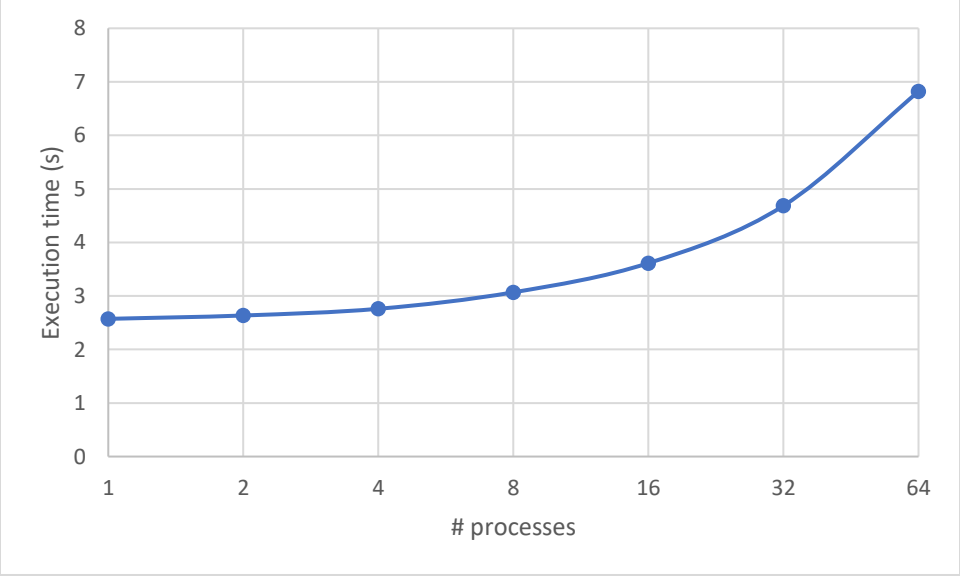

Figure 1 Execution time versus number of processes

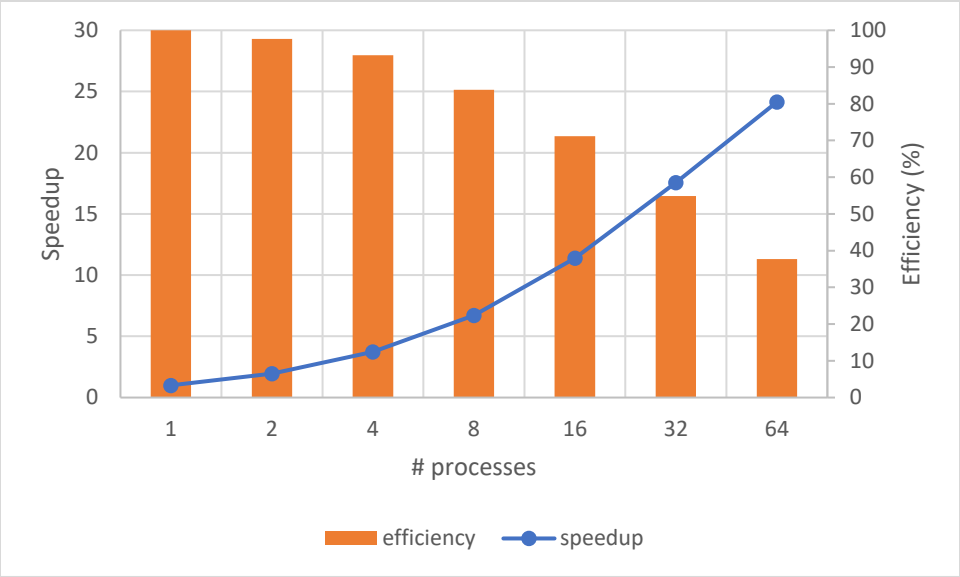

Figure 2 Speedup and Efficiency versus number of processes

3, 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.

Fig.3 demonstrated the execution time for strong scalability study on different number of processes. Fig.4 shows the speedup and efficiency on different number of processes for strong scalability study. From Fig. 4 we can see that the scaling is good for processes under 4. With more processes the scaling start to go down (same trend with efficiency). I think this is due to communication overhead across nodes, since we have fixed number of processes per node which is 4 (ptile=4). Also, the scaling start to get worse for larger number of processes (>=32) for strong scalability study compared to weak scalability study. This is because weak scalability has more work to do and run longer to give more accurate results.

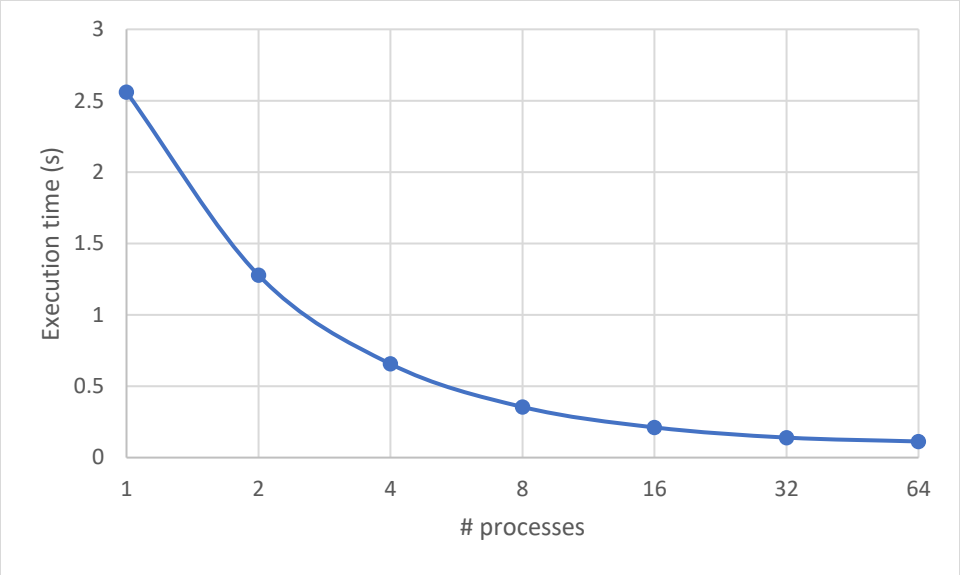

Figure 3 Execution time versus number of processes

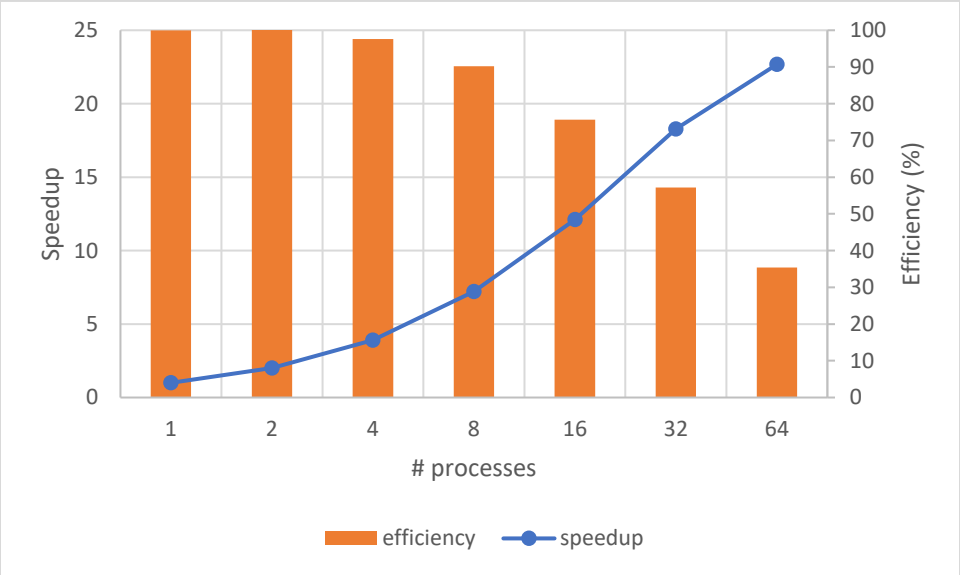

Figure 4 Speedup and Efficiency versus number of processes

From Fig. 2 we can see that before 32 processes, the speedup is almost linear with the increase of number of processes.