The Shellsort, serialized quicksort and standard odd-even transposition sort (Parallel algorithm, given in book 248-250) results are shown in the graph for input size ranging 2^{16}, 2^{20}, 2^{24}. The algorithms result in very long time for 2^{30} and gives error, hence these results are not included.

The serialized quicksort is inherently the O (n log n) algorithm. However, the data supplied to the algorithm has (0,128) range only. For larger data size, there are lot of data with the same number repeated several times. Because of this and the overheads, the running time observed for the quicksort is quadratic. To confirm this behavior, quicksort algorithm was run on data range (0, array size) and it was observed that the algorithm did perform lot quicker with fewer partitions and was more time-efficient. However, the results are not included here.

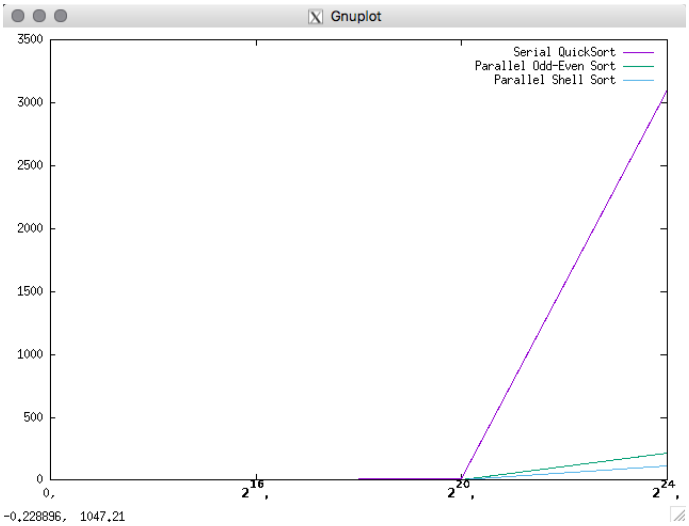
As seen from the results, parallel Shellsort algorithm executes faster compared to the counterparts. Because of the hypercube communications, the results are almost sorted before the odd-even transposition sort phase. This is the reason, phase-2 iterations are quite less compared to odd-even transposition phase (which is n/p). For example, with array size: 2^{16}, number of odd-even transposition iterations is: 5.

Phase 1: Hypercube Communications. This allows us to compare the arrays which are far from each other. During, the following communications take place in case of 8 parallel processes.

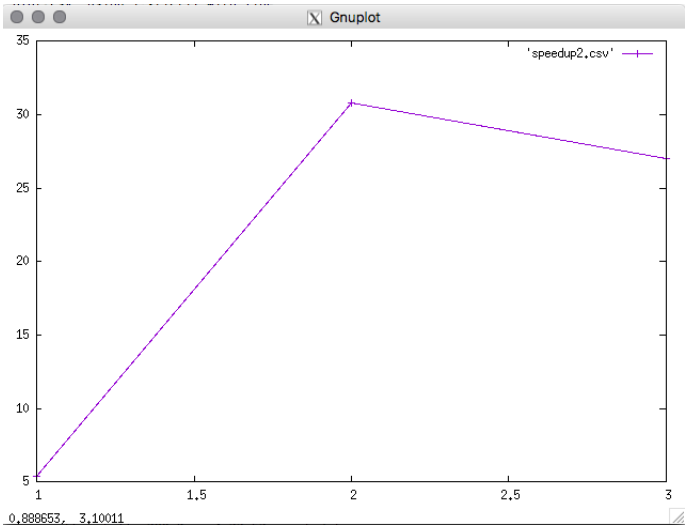
First Iteration: (distance = 4)
(0 <-> 4, 1 <-> 5, 2 <-> 6, 3 <-> 7)
Second Iteration: (distance = 2)
(0 <-> 2, 1 <-> 3, 6 <-> 4, 7 <-> 5)
Third Iteration: (distance = 1)
(0 <-> 1, 2 <-> 3, 4 <-> 5, 6 <-> 7)

However, because of the communications overhead, the MPI Program does take longer than O (n log n) for large array sizes. However, the algorithm performance is still much better compared to quicksort and parallel odd-even sort.

The second graph shows the speed-up of this algorithm compared to quicksort. Ideally, the shell-sort algorithm is p times faster, where p = number of processes. However, because of the nature of data in which quicksort is executing quite slow compared to O (n log n), we see the higher speedup values then expected.



(a) Comparision of different sorting algorithms



(b) Speed up : Quick Sort VS Shell Sort