HYPER QUICK SORT

Qamar Nadeem

Id : 50364371

University at Buffalo The State University of New York

Why Sorting?

- Sorting is one of the most common operations performed by a computer
- Sorted data are easier to manipulate than randomly-ordered data, many algorithms like *Prim's algorithm and Dijkstra's algorithm* require sorted data.
- Sorting makes it possible to search a particular data element in a collection quickly by applying the binary search technique in O(logn)

Application of Sorting in real life

- *Commercial computing.* Government organizations, financial institutions, and commercial enterprises organize much of this information by sorting it. Whether the information is accounts to be sorted by name or number, transactions to be sorted by time or place, mail to be sorted by postal code or address,
- *Numerical computations.* Scientific computing is often concerned with accuracy (how close are we to the true answer?). Accuracy is extremely important when we are performing millions of computations with estimated values such as the floating-point representation of real numbers that we commonly use on computers. Some numerical algorithms use priority queues and sorting to control accuracy in calculations.
- *Operations research.* consider the *load-balancing problem*, where we have M identical processors and N jobs to complete, and our goal is to schedule all of the jobs on the processors so that the time when the last job completes is as early as possible. One method that is known to produce a good schedule is the *longest processing time first* rule, where we consider the jobs in decreasing order of processing time, assigning each job to the processor that becomes available first.

Sequential Quick Sort

- QuickSort is an inplace Divide and Conquer algorithm
- Quick sort doesn’t require any extra storage
- The time taken by QuickSort depends upon the input array and partition strategy.
- Average time complexity is O(nlogn).

Quick sort algorithm

- quick sort follows the below steps:

  Step 1 − Make any element as pivot

  Step 2 − Partition the array on the basis of pivot

  Step 3 − Apply quick sort on left partition recursively

  Step 4 − Apply quick sort on right partition recursively

Why parallel quick sort?

- A sequential sorting algorithm may not be efficient enough when we have to sort a huge volume of data. Therefore, parallel algorithms are used in sorting.
- If we have n numbers of data and have p processor then we could reduce the computational time from O(nlogn) to O((n/p)log(n/p)).

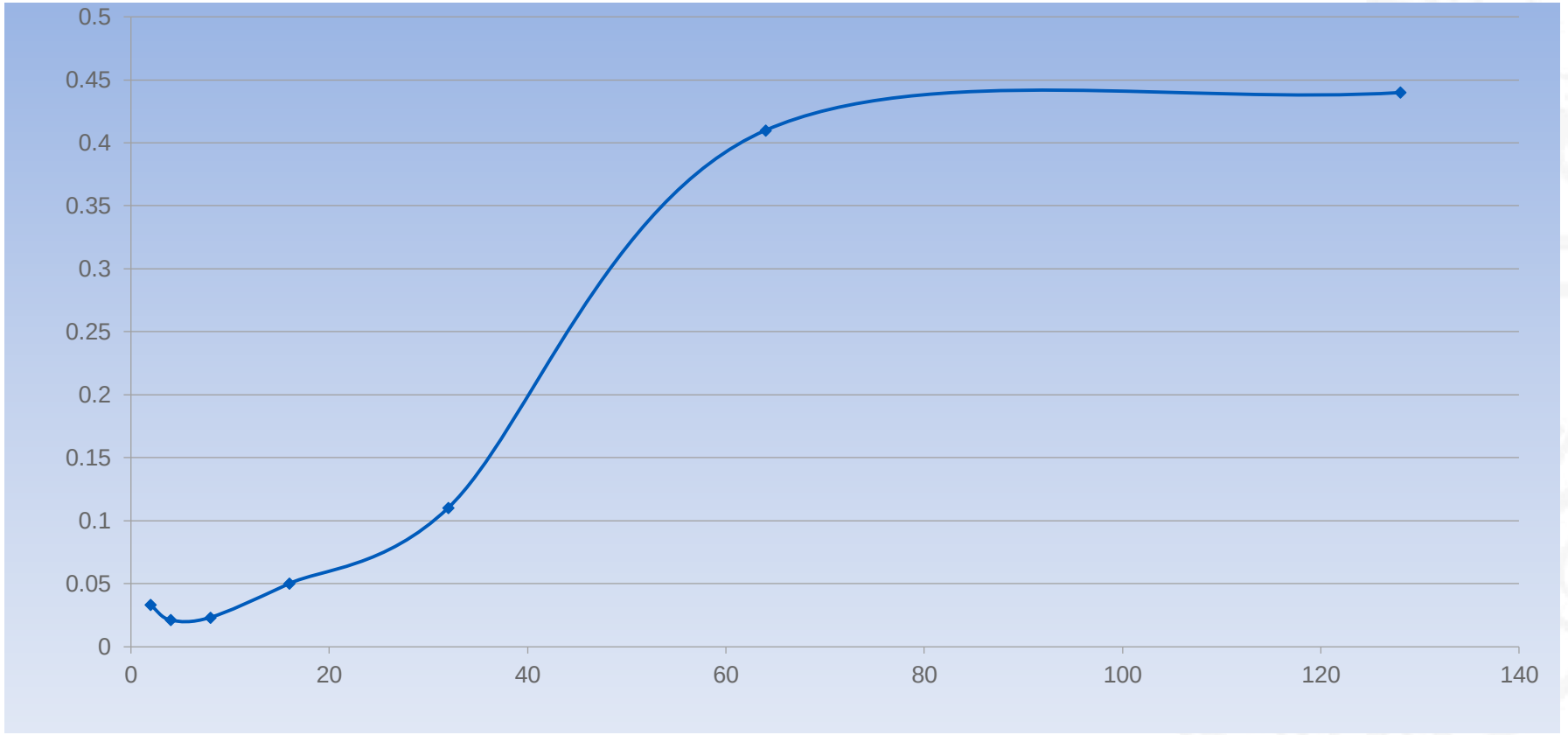
Hyper Quick sort Algorithm

- Scatter the data to p processes.
- Call quicksort algorithm
- Merge the sorted data using hypercube mapping.
- In hypercube compare-exchange operations take place between wires whose labels differ in only one bit. In a hypercube, processes whose labels differ in only one bit are neighbors and sorting and merging performed between neighbour. It stops when process id is root process.

Execution Time vs No. of processors for 500k data set

Processors	Elapsed time (sec)
2	0.033
4	0.021
8	0.023
16	0.05
32	0.11
64	0.41
128	0.44

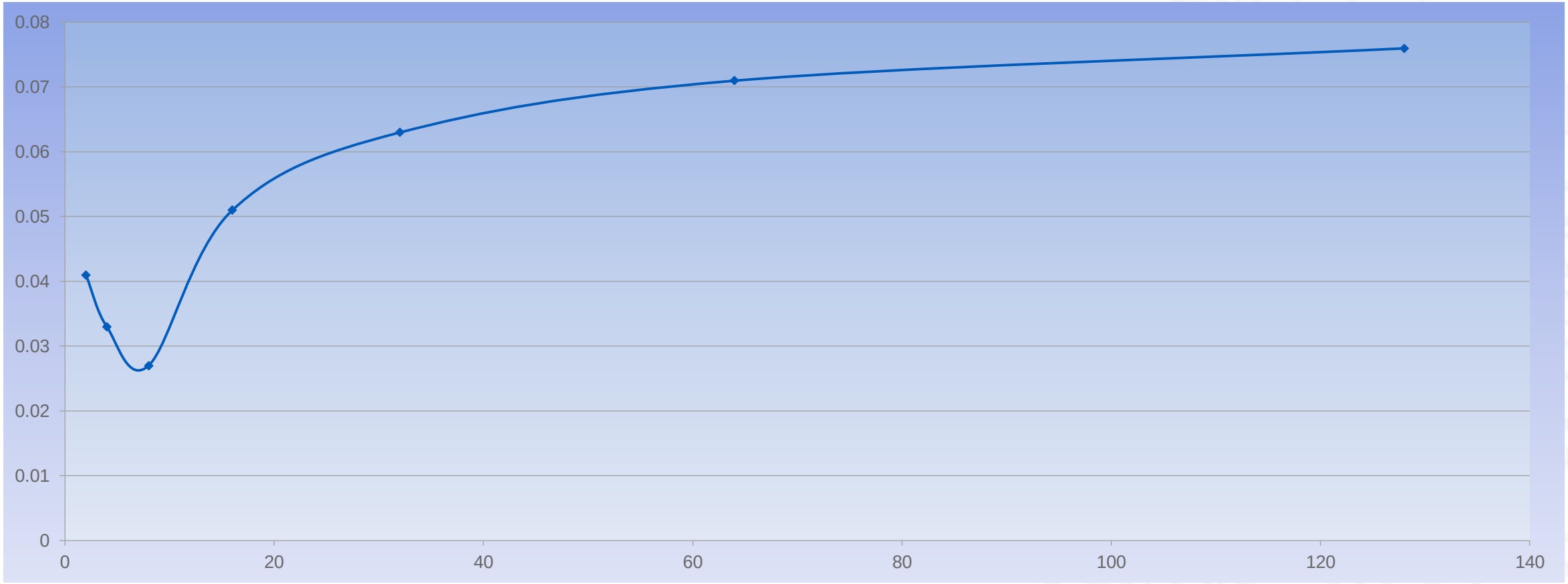
Execution Time vs No. of processors for 500k data set



Execution Time vs No. of Nodes for 500k data set

Nodes	Average run time (sec)
2	0.041
4	0.033
8	0.027
16	0.051
32	0.063
64	0.071
128	0.076

Execution Time vs No. of Nodes for 500k data set



Limitations

- The number of processors has to a be a power of 2.
- Communication overhead increases as the number of nodes increase which reduces the total execution time.

- Reference Algorithms, Sequential and Parallel: A Unified Approach – Russ Miller and Laurence Boxer. 3rd Edition.