Hyper Quick Sort (Parallel Quick Sort)

-

Prashant Srivastava

CSE 633 Spring 2014

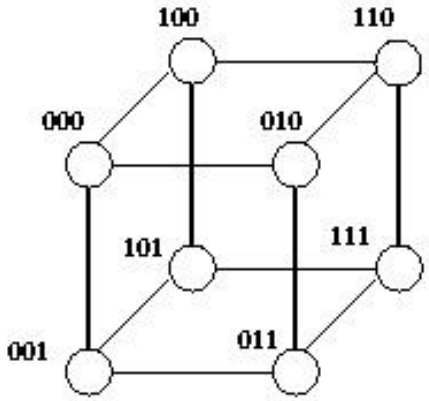
Parallel Quick Sort

Hyper quick sort is an implementation of quick sort on a hypercube.

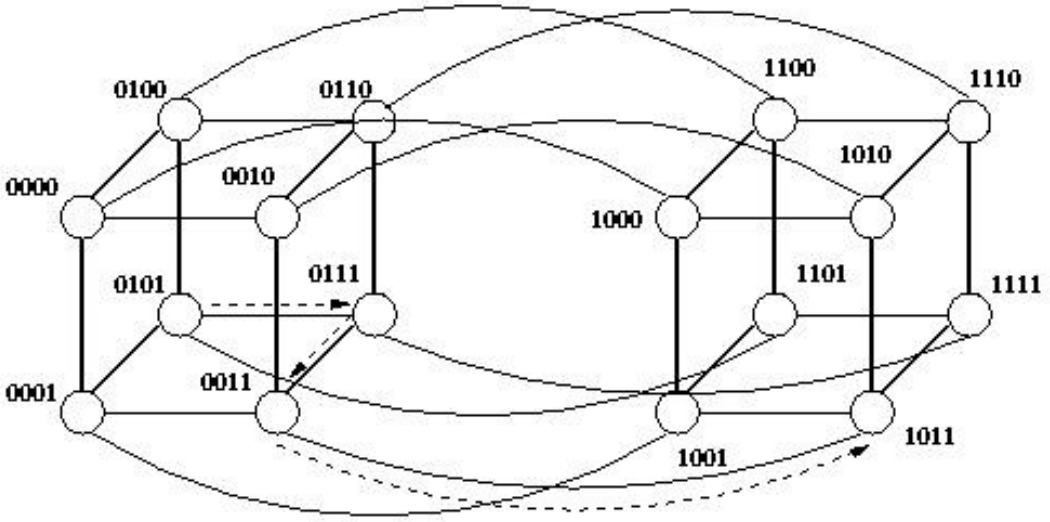
Let the communication network topology be an N-dimensional hypercube (i.e. the number of processors is equal to $p=2^N$).

Hypercube

Formally, *a hypercube of size n* consists of *n* processors indexed by the integers {0,1, . . . , *n* − 1}, where *n* > 0 is an integral power of 2. Processors *A* and *B* are connected *if and only if* their unique $\log_2$ *n*-bit strings differ in *exactly* one position.



3-D hypercube



4-D hypercube

Algorithm 1

- We randomly choose a pivot from one of the processes and broadcast it to every process.
- Each process divides its unsorted list into two lists: those smaller than (or equal) the pivot, those greater than the pivot.
- Each process in the upper half of the process list sends its “low list” to a partner process in the lower half of the process list and receives a “high list” in return.
- Now, the upper-half processes have only values greater than The pivot, and the lower-half processes have only values smaller than the pivot.
- Thereafter, the processes divide themselves into two groups and the algorithm recurses.
- After logP recursions, every process has an unsorted list of values completely disjoint from the values held by the other processes.
  - The largest value on process i will be smaller than the smallest value held by process i + 1.
  - Each process can sort its list using sequential quicksort.

Algorithm 2(My Implementation)

- Each process starts with a sequential quicksort on its local list.
- Now we have a better chance to choose a pivot that is close to the true median.
  - The process that is responsible for choosing the pivot can pick the median of its local list.
- The three next steps of hyper quick sort are the same as in parallel algorithm 1
  - Broadcast
  - Division of “low list” and high list”
  - Swap between partner processes
- The next step is different in hyper quick sort.
  - On each process, the remaining half of local list and the received half-list are merged into a sorted local list.
- Recursion within upper-half processes and lower-half processes.

Expected Case Running Time

$$\Theta\left(N \log N + \frac{d(d+1)}{2} + dN\right).$$

The N log N term represents the sequential running time from Step 2. The $d(d + 1)/2$ term represents the broadcast step used in Step 4. The dN term represents the time required for the exchanging and merging of the sets of elements.

Observations

Log P steps are needed in the recursion.

- The expected number of times a value is passed from one process to another is log P / 2 , that is quite some communication overhead!
- The median value chosen from a local segment may still be quite different from the true median of the entire list.

Although better than parallel quicksort algorithm 1, load imbalance may still arise.

Solution:

- Algorithm 3 – parallel sorting by regular sampling

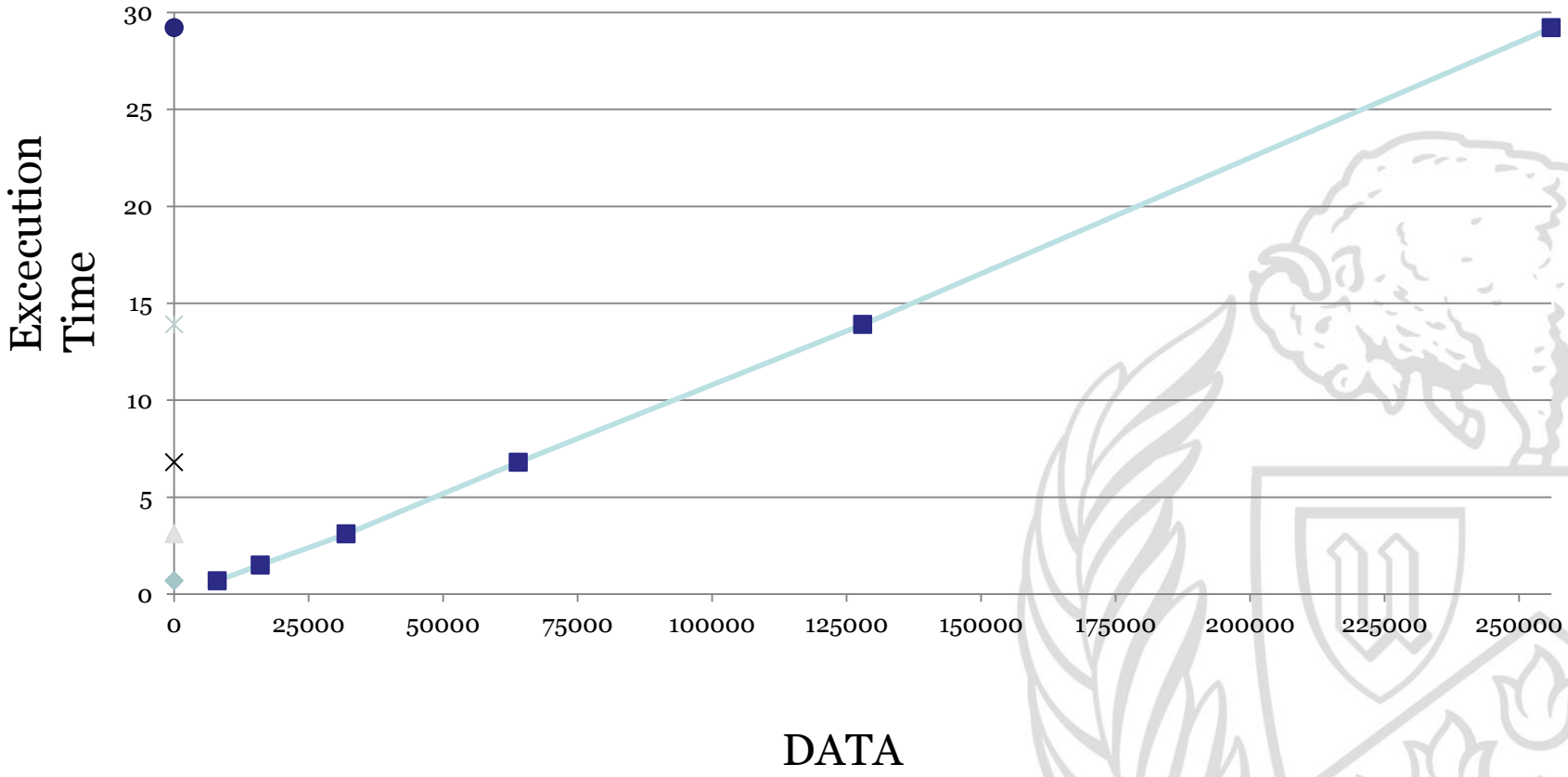
Limitations

The number of processors has to a be a power of 2.

Very High communication overhead.

No. of Processors	Data	Running Time (msec)
1	8000	0.69
1	16000	1.5
1	32000	3.1
1	64000	6.8
1	128000	13.9
1	256000	29.2

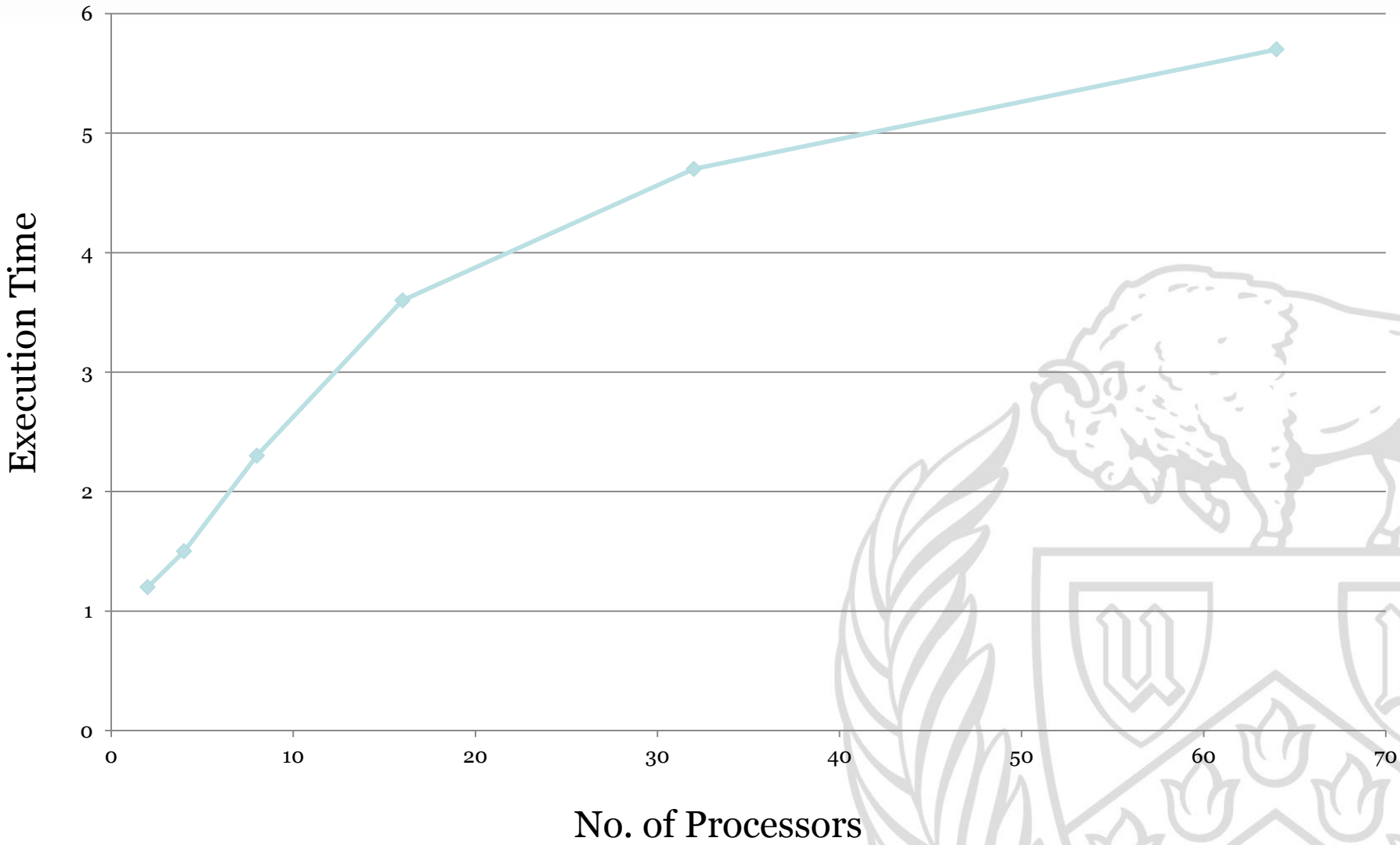
Sequential Sort



Results

No. of Processors	Data/Processor	Running Time(msec)
2	10,000	1.2
4	10,000	1.5
8	10,000	2.3
16	10,000	3.6
32	10,000	4.7
64	10,000	5.7

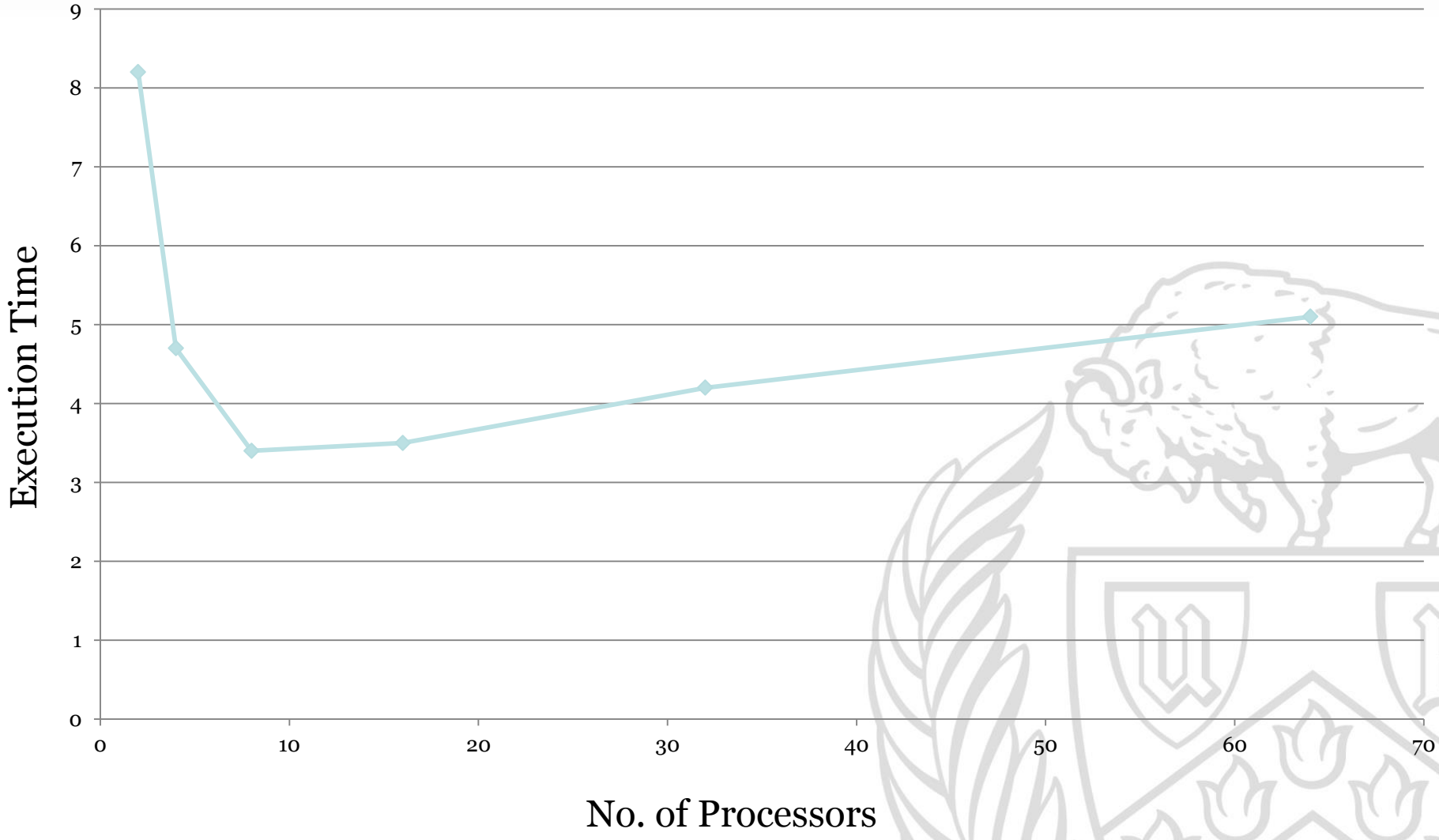
Results



Results

No. of Processors	Data	Running Time (msec)
2	128000	8.8
4	128000	4.8
8	128000	3.4
16	128000	3.5
32	128000	4.2
64	128000	4.9

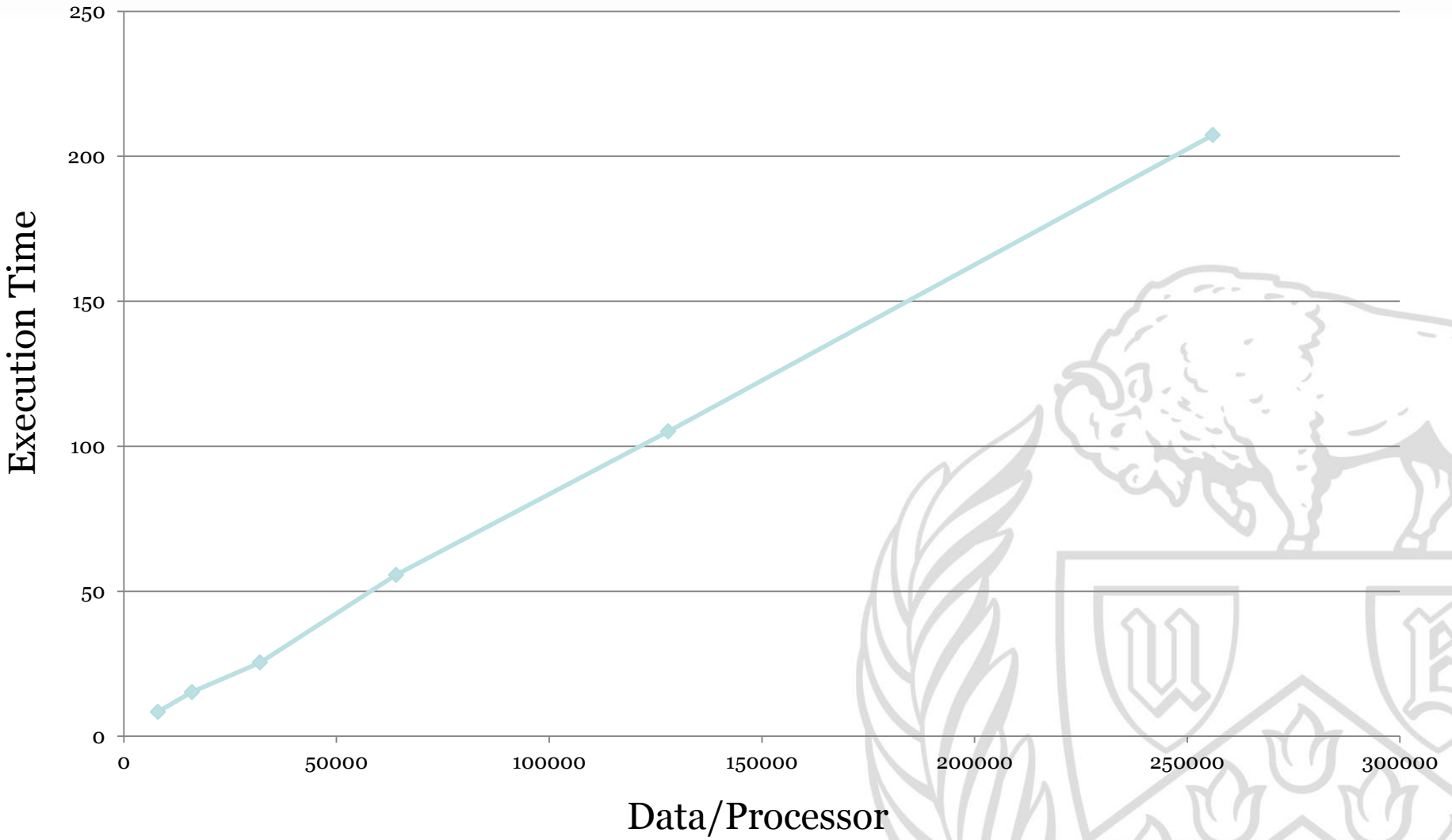
Results


Execution Time
9
8
7
6
5
4
3
2
1
0
0
10
20
30
40
50
60
70
No. of Processors

Results

No. of Processors	Data/Processor	Running Time (msec)
32	8000	8.4
32	16000	15.2
32	32000	25.4
32	64000	55.6
32	128000	105.1
32	256000	207.3

Results


250
200
150
100
50
0
Execution Time
0
50000
100000
150000
200000
250000
300000
Data/Processor

Reference

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

http://www.uio.no/studier/emner/matnat/ifi/INF3380/v10/undervisningsmateriale/inf3380-week12.pdf

THANK YOU