PARALLEL IMPLEMENTATION OF BITONIC SORT – USING MPI

By

Srinath Vikramakumar –
svikrama@buffalo.edu

Guided by:

Dr. Russ Miller and Dr. Matthew Jones

University at Buffalo The State University of New York

Sorting

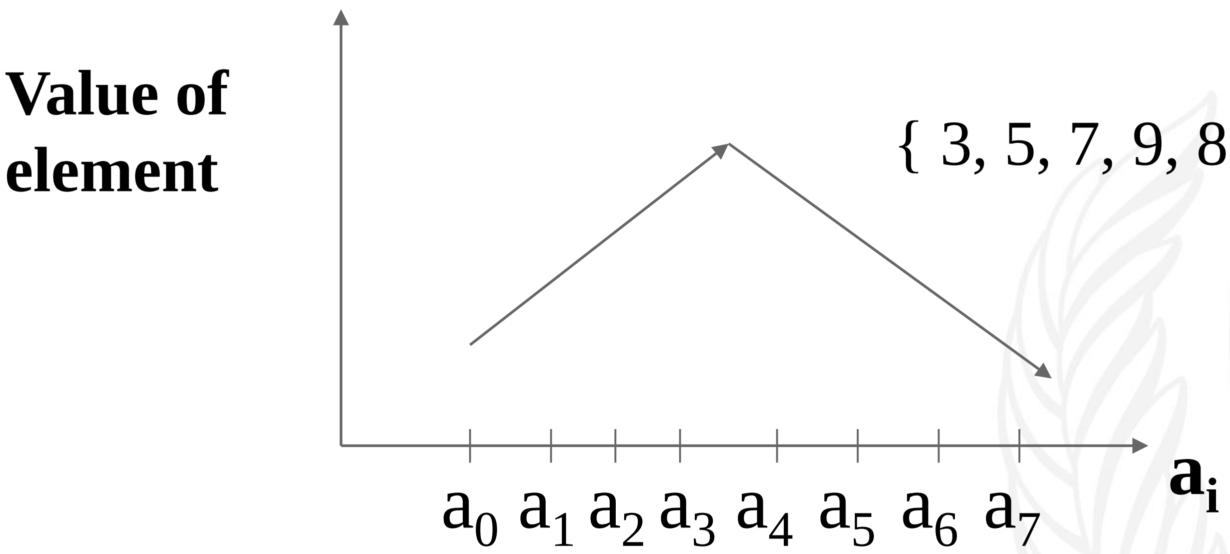
- Arrange an unordered collection of items into a meaningful order.
- The most frequently used orders are numerical order and lexicographical order.
- One of the most commonly used and well-studied kernels.
- Sorting can be *comparison-based* or *noncomparison-based*.
- The fundamental operation of comparison-based sorting is *compare-exchange*.
- The lower bound on any **sequential** comparison-based sort of n numbers is $\Theta(n \log n)$.

Bitonic Sequence

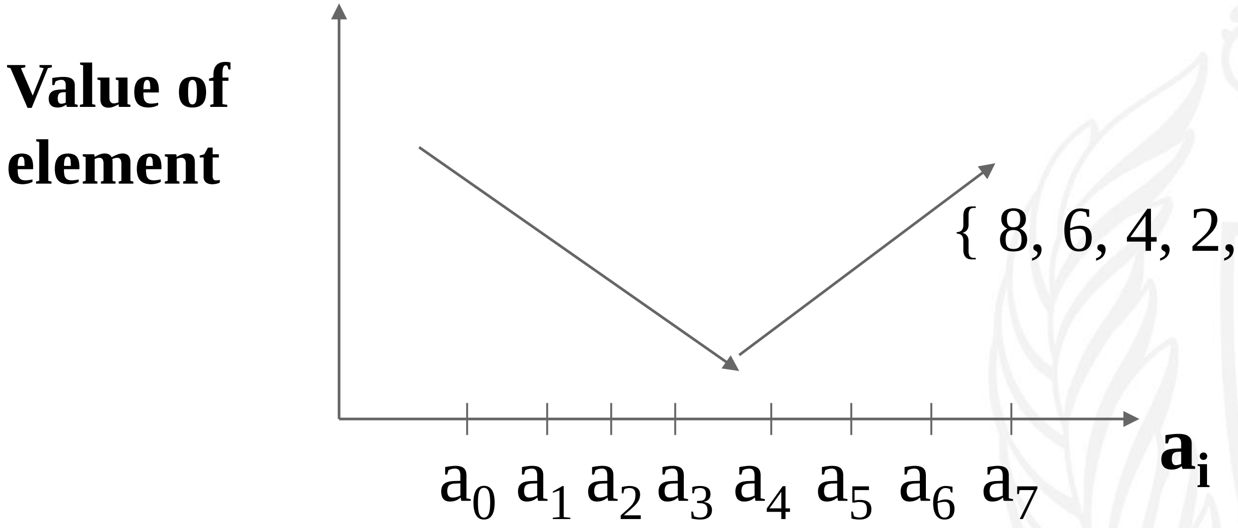
A sequence $a = (a1, a2, . . ., ap)$ of p numbers is said to be *bitonic* if and only if

- $a1 \leq a2 \leq . . . \leq ak \geq . . . \geq ap$, for some k, $1 < k < p$, or
- $a1 \geq a2 \geq . . . \geq ak \leq . . . \leq ap$, for some k, $1 < k < p$, or
- *'a'* can be split into two parts that can be interchanged to give either of the first two cases.

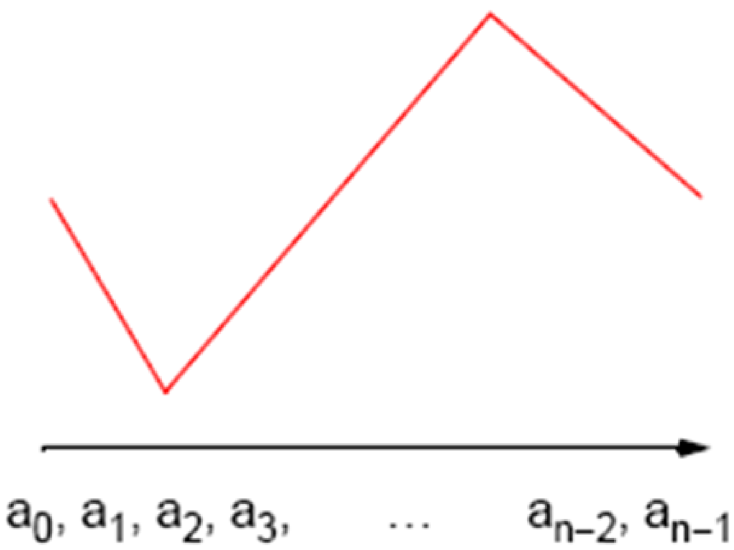
Something like this...



Or this…



This too…

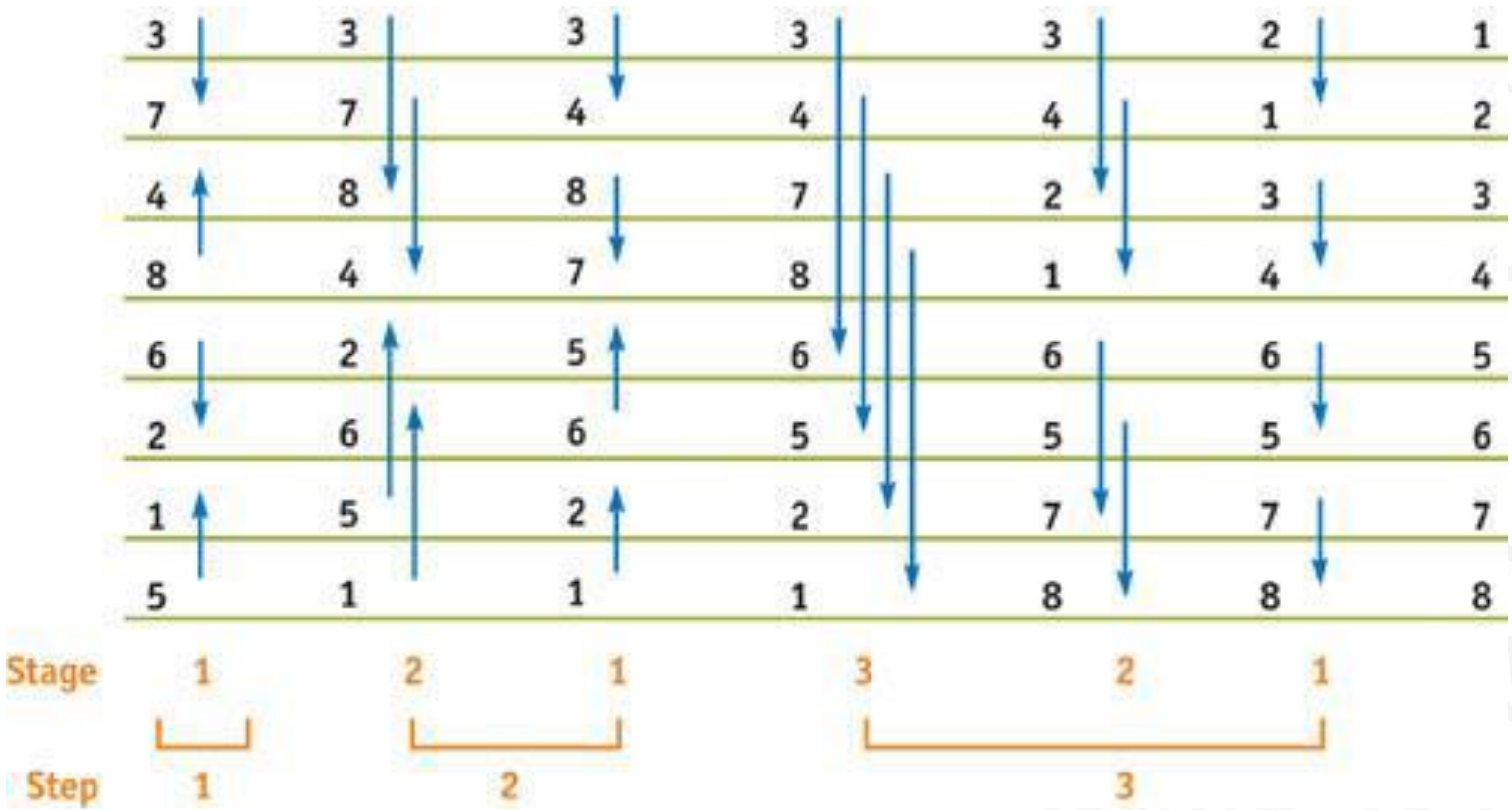


{3,1,2,4,7,8,6,5}

Bitonic Sorting

- To sort an unordered sequence, sequences are merged into larger bitonic sequences, starting with pairs of adjacent numbers.
- By a compare-and-exchange operation, pairs of adjacent numbers formed into increasing sequences and decreasing sequences. Pairs form a bitonic sequence of twice the size of each original sequences.
- By repeating this process, bitonic sequences of larger and larger lengths obtained.
- In the final step, a single bitonic sequence sorted into a single increasing sequence.

Bitonic Sort Example:



Bitonic Sort Efficiency

When (P=n)

$$T_{par}^{bitonic} = \sum_{i=1}^{i=\log n} i = \frac{\log n(\log n + 1)}{2} = O(\log^2 n)$$

Bitonic Sort Efficiency

When (P<<n)

$$T_{par}^{bitonic} = \text{Local Sort} + \text{Parallel Bitonic Merge}$$

$$= \frac{\mathrm{N}}{\mathrm{P}} \log \frac{\mathrm{N}}{\mathrm{P}} + 2\frac{\mathrm{N}}{\mathrm{P}}(1+2+3+\ldots+\log P)$$

$$= \frac{\mathrm{N}}{\mathrm{P}}\{\log \frac{\mathrm{N}}{\mathrm{P}} + 2(\frac{\log P(1+\log P)}{2})\}$$

$$= \frac{\mathrm{N}}{\mathrm{P}}(\log \mathrm{N} - \log P + \log P + \log^2 P)$$

$$T_{par}^{bitonic} = \frac{\mathrm{N}}{\mathrm{P}}(\log \mathrm{N} + \log^2 P)$$

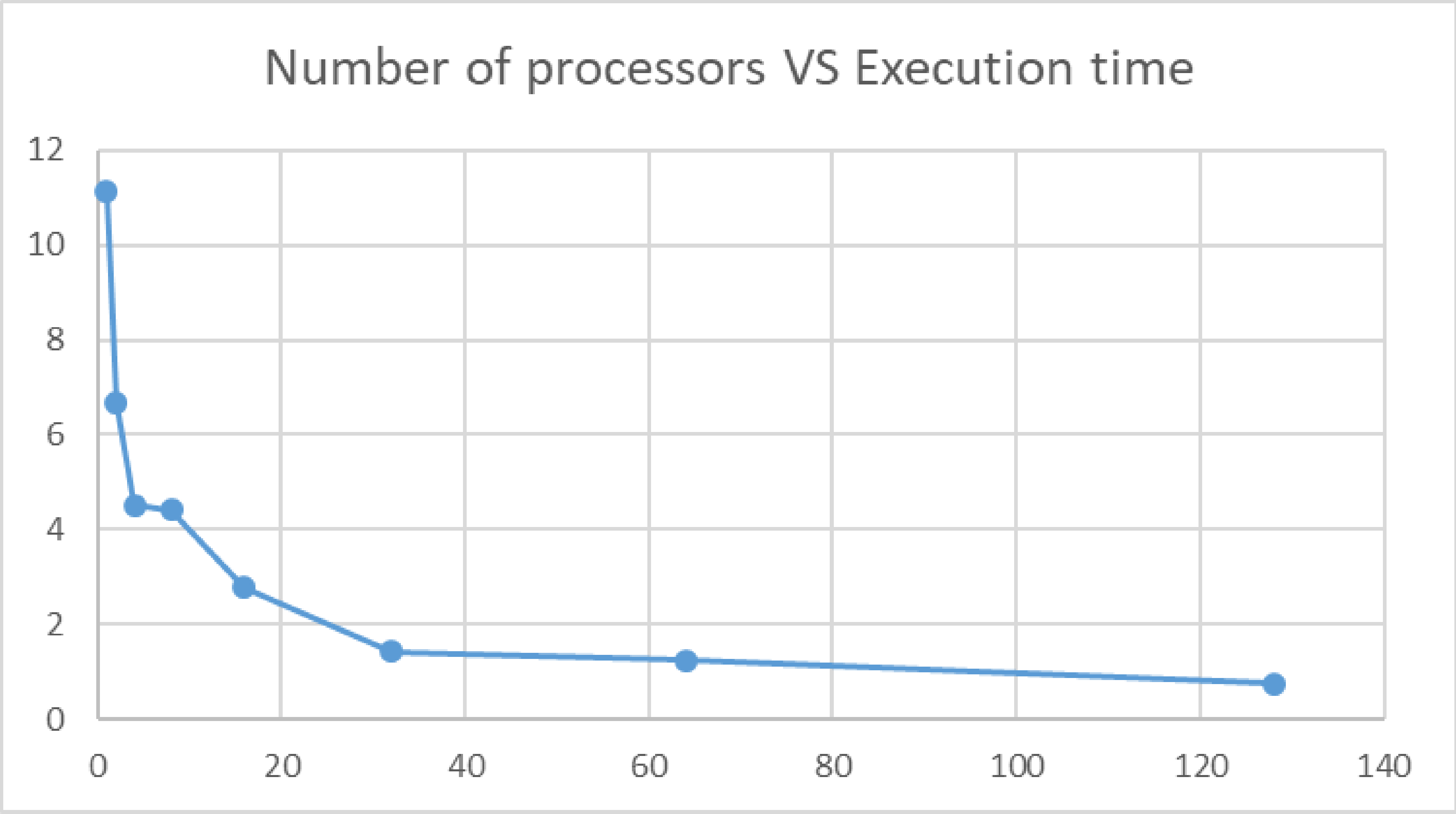
Experiments:

- Keeping the amount of data constant and Increasing number of processors and analyzing the execution time.
- Keeping the number of processors constant and Increasing the amount of data and analyzing the execution time.
- Increasing number of processors and amount of data per processors proportionally and analyzing the execution time.
- Keeping small amount of constant data and increasing the number of processors and analyzing the execution time.
- Keeping the number of processors equal to the number of data and analyzing the execution time.

Number of processors VS Execution time

Constant data size: 32000000

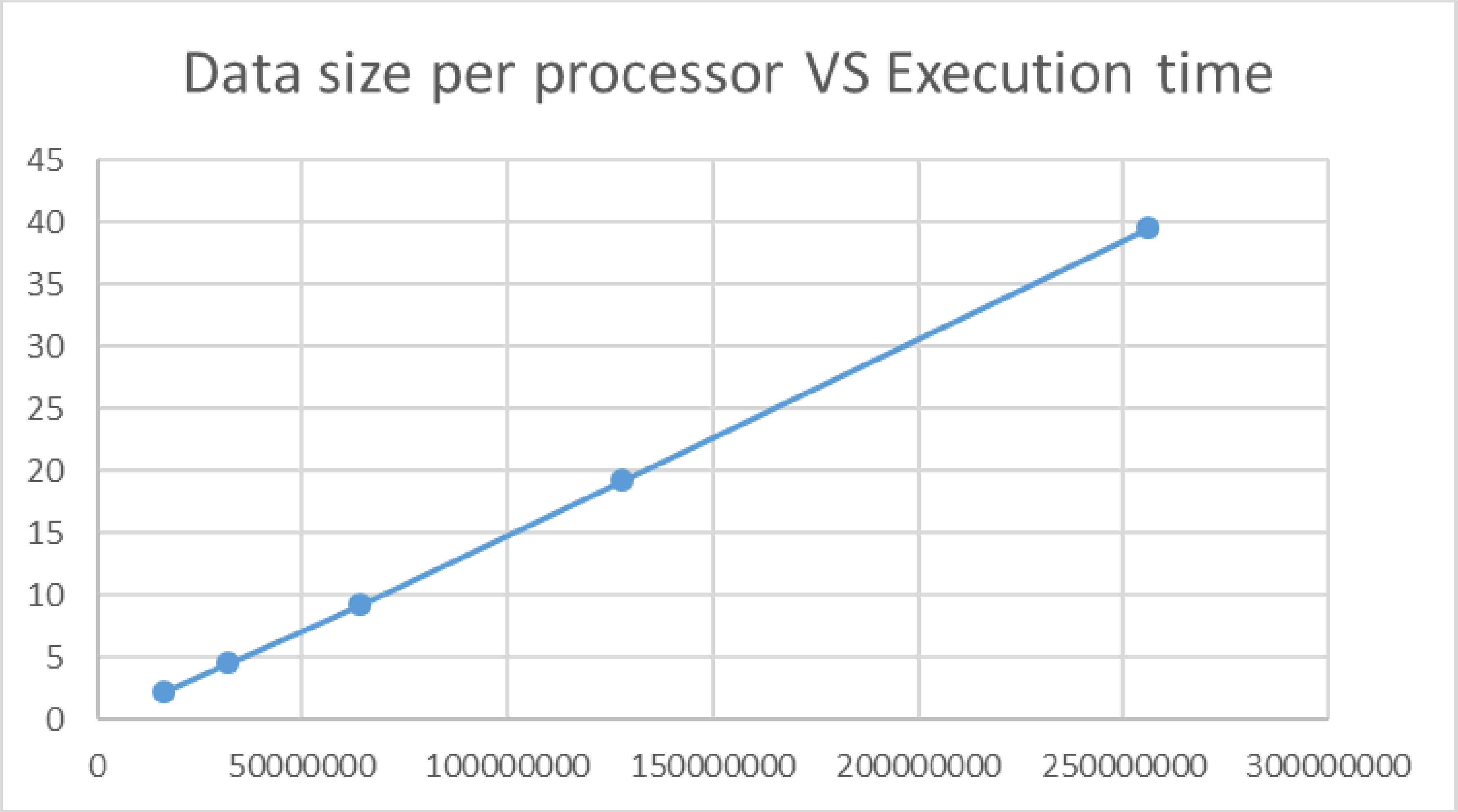
Number of processors	Execution time in seconds
1	11.133358
2	6.656947
4	4.506986
8	4.414858
16	2.765411
32	1.425334
64	1.232986
128	0.750448


Number of processors VS Execution time
12
10
8
6
4
2
0
0
20
40
60
80
100
120
140

Data size VS Execution time

Constant number of processors = 4

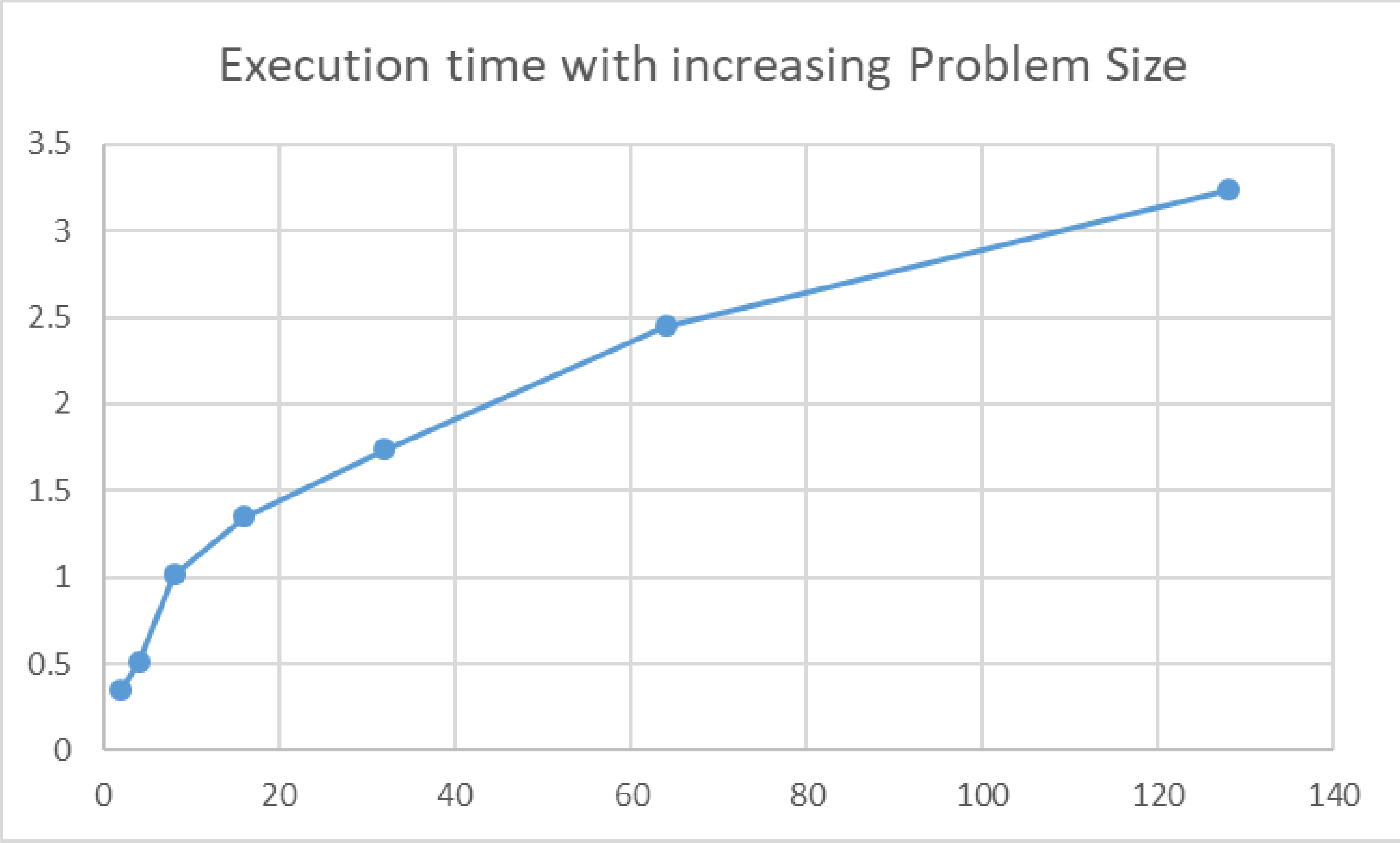
Number of data	Execution time in seconds
16000000	2.151250
32000000	4.491756
64000000	9.171313
128000000	19.184456
256000000	39.466040


Data size per processor VS Execution time
45
40
35
30
25
20
15
10
5
0
0
50000000
100000000
150000000
200000000
250000000
300000000

Number of data VS Number of processors VS Execution time

Number of Processors	Number of Data per processor	Execution time in seconds
2	2000000	0.352784
4	4000000	0.507792
8	8000000	1.013867
16	16000000	1.351265
32	32000000	1.435955
64	64000000	2.448596
128	128000000	3.239658

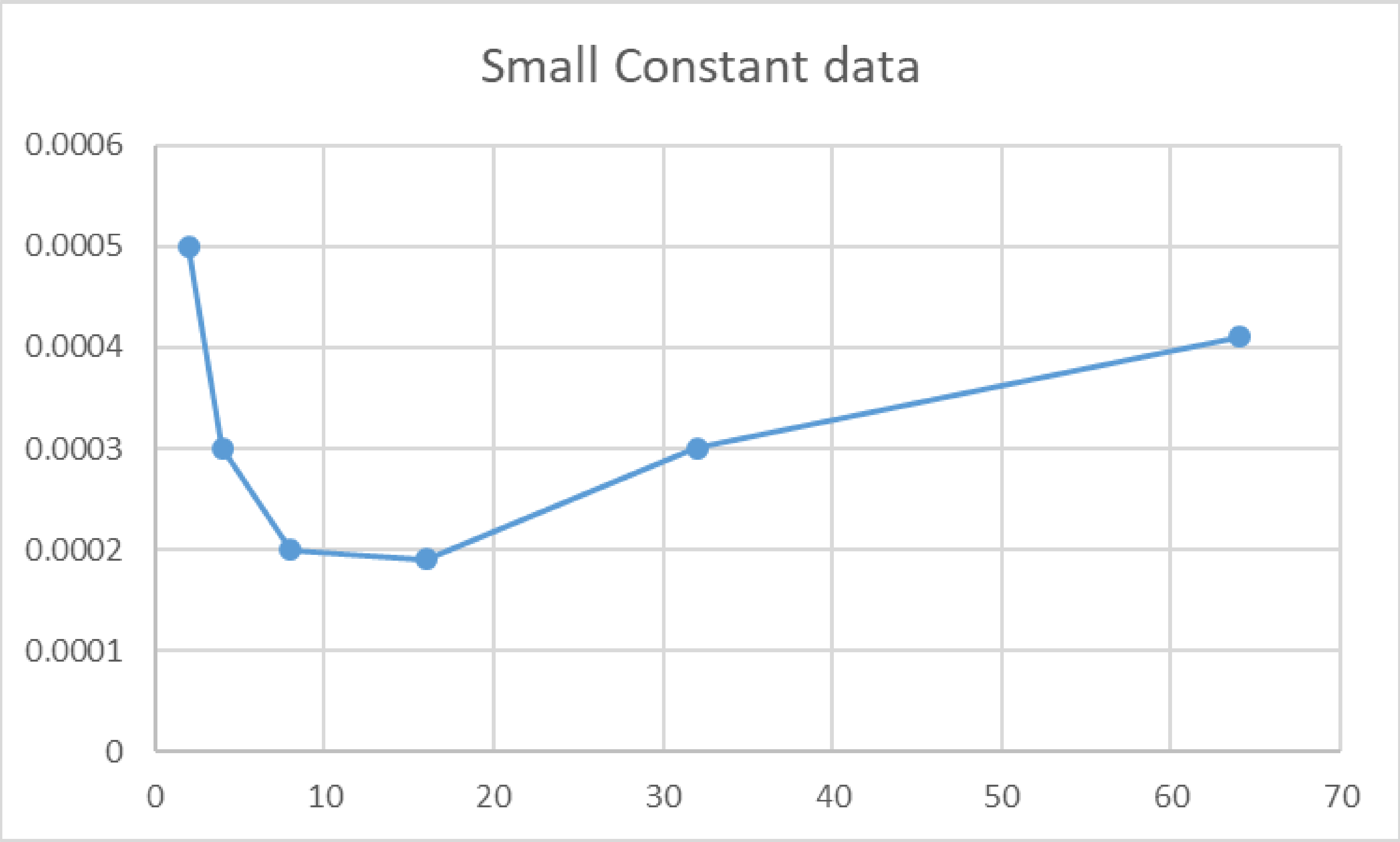
Increasing number of processors as well as number of data items


Execution time with increasing Problem Size
3.5
3
2.5
2
1.5
1
0.5
0
0
20
40
60
80
100
120
140

Number of processors VS Execution time

Small Constant data= 1000

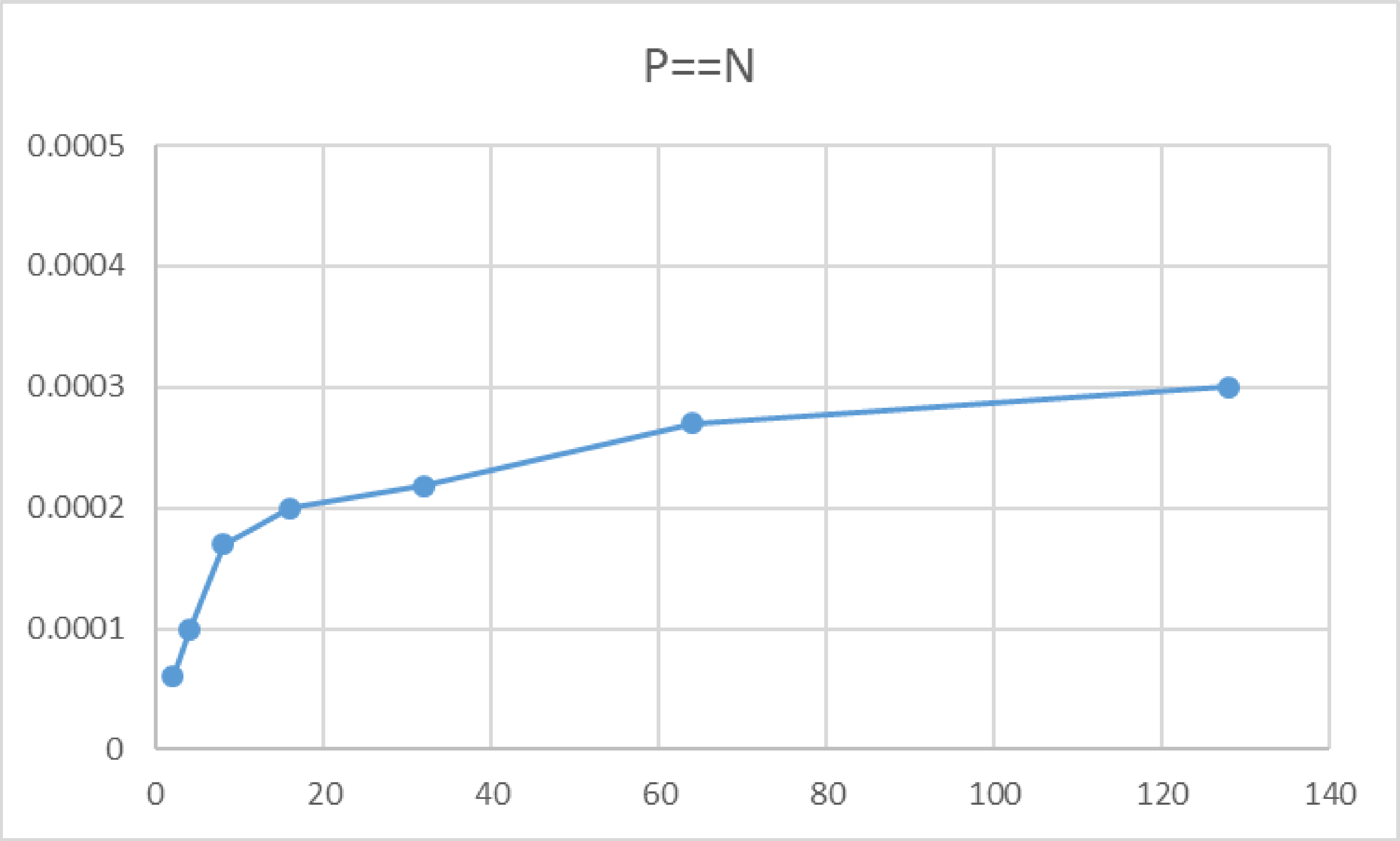
Number of data per processor	Execution time in seconds
2	0.0005
4	0.0003
8	0.0002
16	0.00019
32	0.0003
64	0.00041


Small Constant data
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0
0
10
20
30
40
50
60
70

Number of processors VS Execution time

Number of processors = Data

Number of Data and Number of processors	Execution time in seconds
2	0.00006
4	0.0001
8	0.00017
16	0.0002
32	0.000218
64	0.00027
128	0.0003


P==N
0.0005
0.0004
0.0003
0.0002
0.0001
0
0
20
40
60
80
100
120
140

Future Work

- Compare with other sorting routines.
- Compare distributed memory models(MPI) with shared memory models(OpenMP).
- Analyze performance of GPU's.
- Focus on unsolved problems.

References

- Algorithms Sequential and Parallel: A Unified Approach by Russ Miller and Laurence Boxer
- http://en.wikipedia.org/wiki/Bitonic_sorter
- CCR: Resources and Tutorial Materials by Dr. Matthew Jones

Thank You!!!