Terabyte Sort- Big Data perspective

DR-C.C-CHAN

JASWANTH

BHOOMIKA

DIVYA

Apache Hadoop

- Hadoop is known for its go-to big data analytics engine.
- It is the best framework to be used when there is a large amount of unstructured data.
- Entire input file to be read from disk
- Hadoop uses map/reduce framework to sort the input directory into the output directory
- Sorting is one of the most challenging because there is no reduction of data along the pipeline. Sorting 100 TB of input data requires shuffling 100 TB of data across the network.

Work done on Terabyte Sorting

- In December 1998, IBM sorted a terabyte of data (10 billion 100 byte records) in 17 minutes, 37 seconds using the SP sort program.

- Hadoop Sorts a Petabyte in 16.25 Hours and a Terabyte in 62 Seconds

Steps in Terabyte Sort

- **TeraGen**:
  This gives the output data which is C equivalent for every byte. This data is divided into maps where each map contains predefined number of rows. Gensort is used to assign unique identifier to each map generated.

- **TeraSort:**
  It is a standard map/reduce sort, where <Key, value> pairs are used. Key is the filename and line number and value is the contents of the line.
  Only difference is, for each reduce a custom partitioner which contains a sorted list of N-1 sampled keys that define the key range is used. In particular, all keys such that *sample[i-1] <= key < sample[i]* are sent to reduce *i*.

Steps

- **TeraValidate:**
  It validates the output to be globally sorted by ensuring that the keys are always in a decreasing order for maps.
  The map also generates records with the first and last keys of the file and the reduce ensures that the first key of file *i* is greater that the last key of file *i-1*.
  Any problems are reported as output of the reduce with the keys that are out of order.

Benchmark categories

For each sort benchmark, there are two categories:

Indy

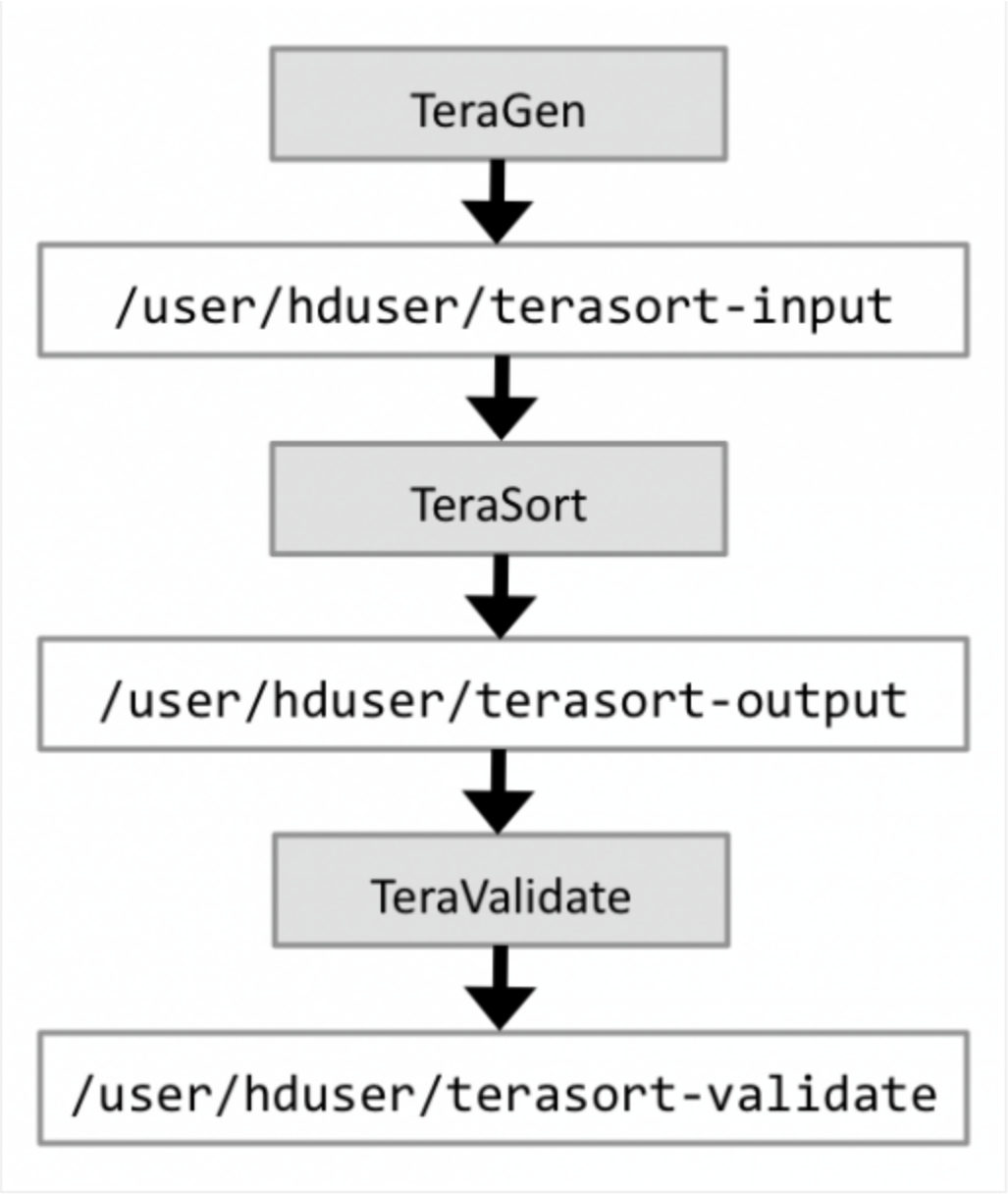
- Need only sort 100-byte records with 10-byte keys.
- The random, dense and uniformly distributed input keys are used both for sorting records and for partitioning records into multiple output files

Benchmark categories

Daytona:

- be able the run continuously for one hour without a system failure.
- be capable of reliably sorting data sets that are larger than the collective main memories of the system.
- For cluster based sort systems that subdivide the input data into partitions each of which is then sorted by a single node, one of the following must be true:
  - the single-node sort can gracefully handle the situation where the partition size does not fit in the main memory of its node; or
  - the partition sizes are guaranteed to fit in the main memory of a node.

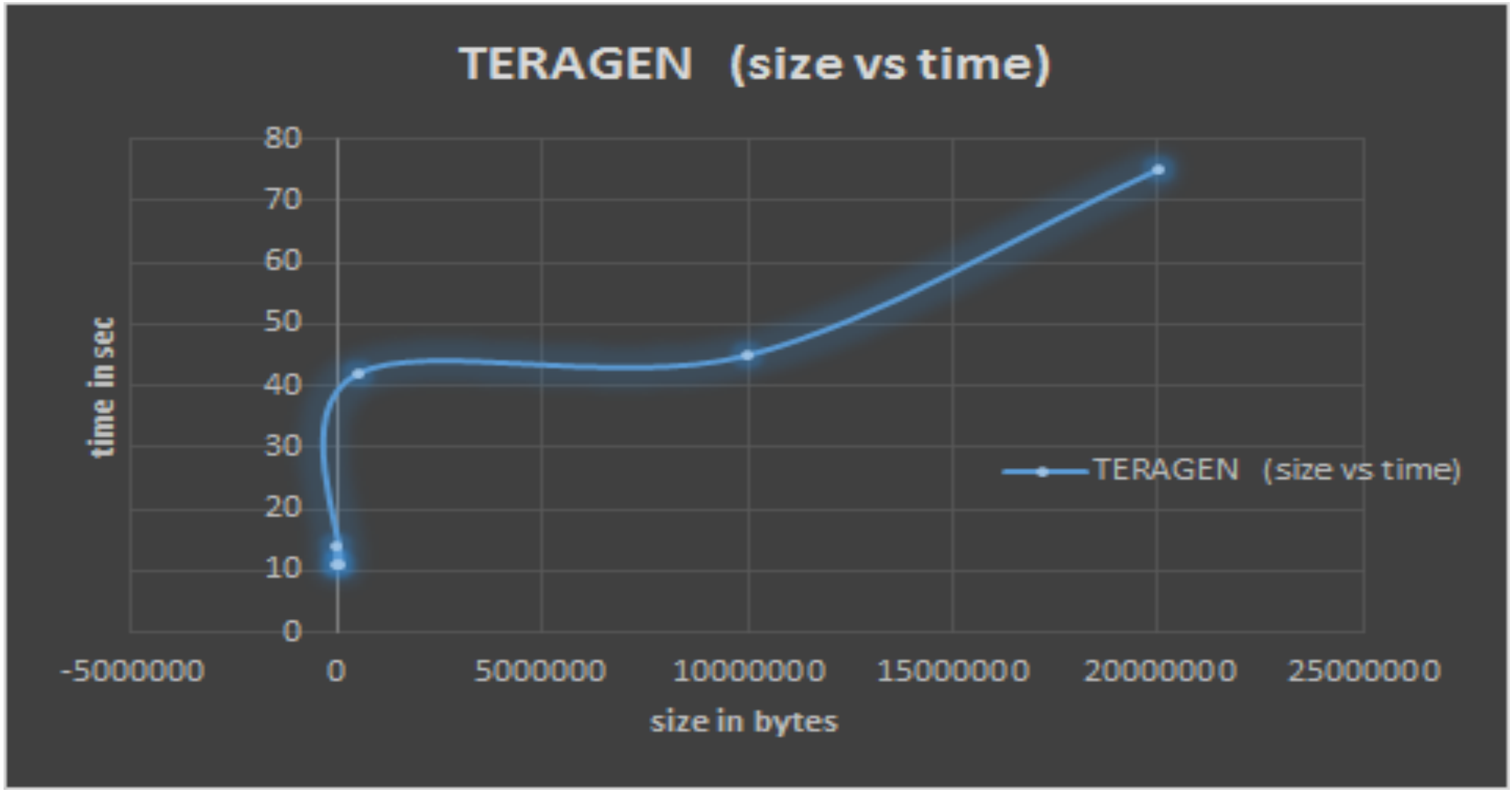
Execution steps



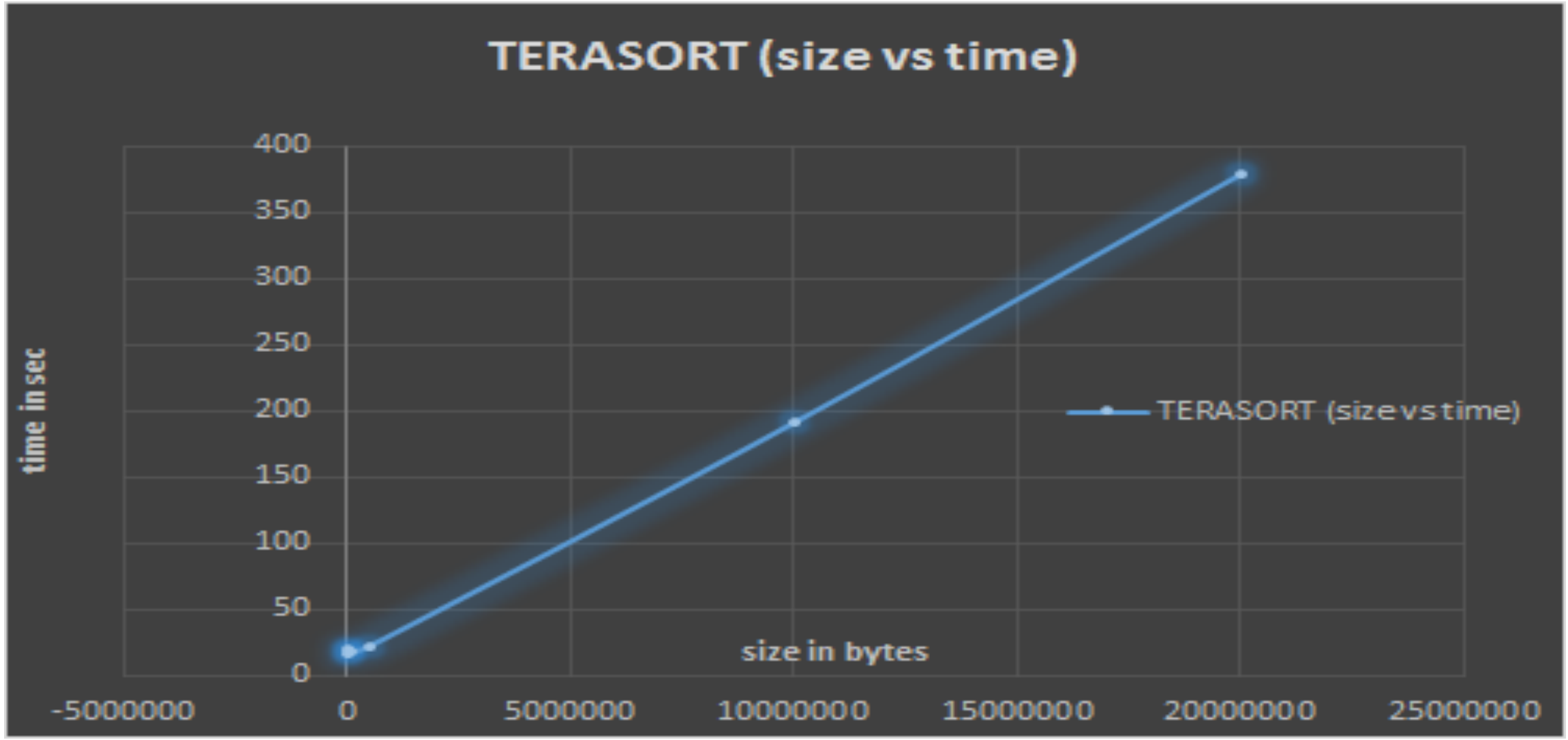
System Configuration

- **Centos OS**
- **Memory RAM 4GB**
- **Processor -no of cores 1**
- **Hard Disk size- 62 GB /later expanded to 100GB**

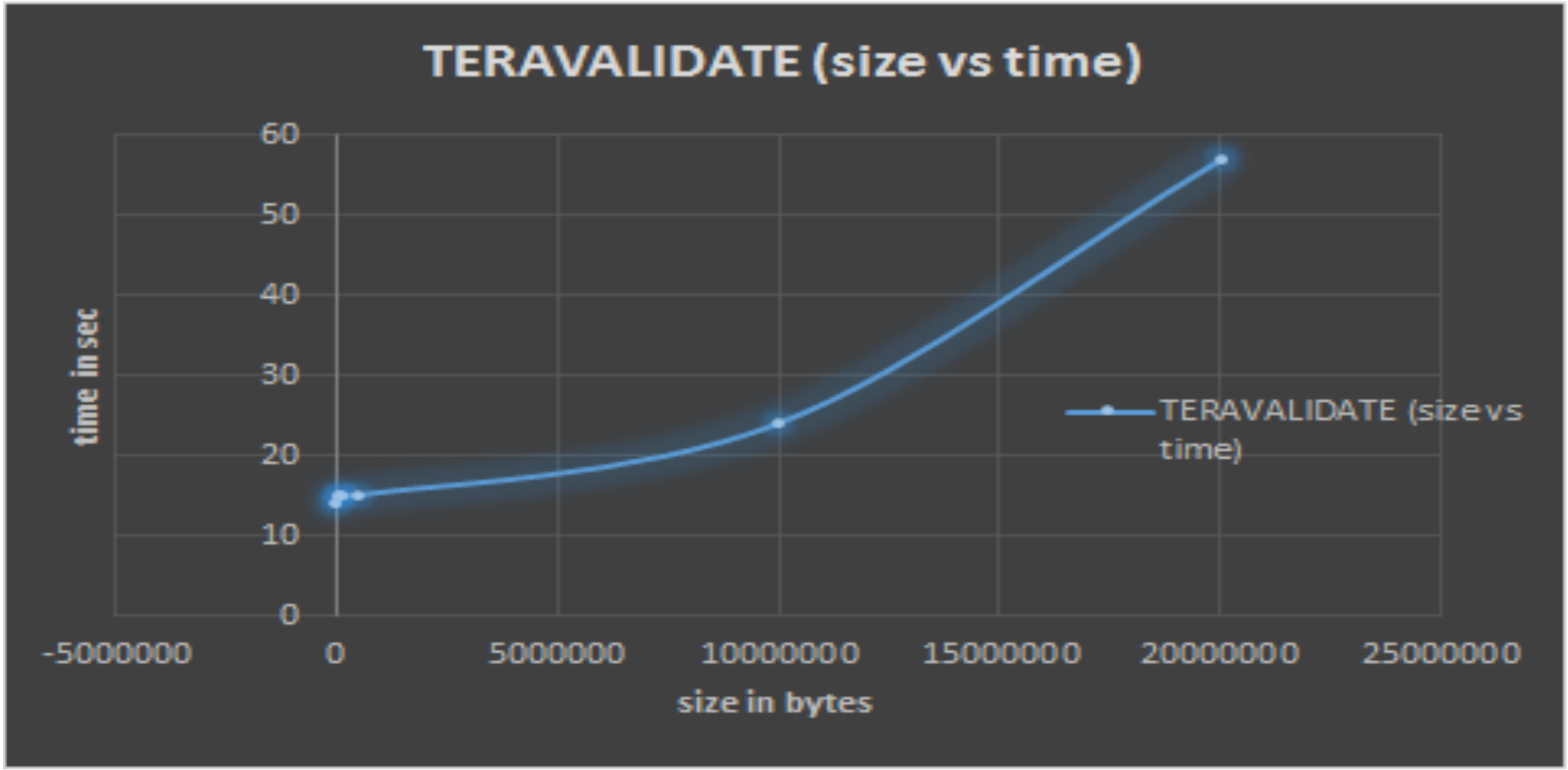
Statistics


TERAGEN (size vs time)
80
70
60
50
40
30
20
10
0
time in sec
-5000000
0
5000000
10000000
15000000
20000000
25000000
size in bytes
TERAGEN (size vs time)

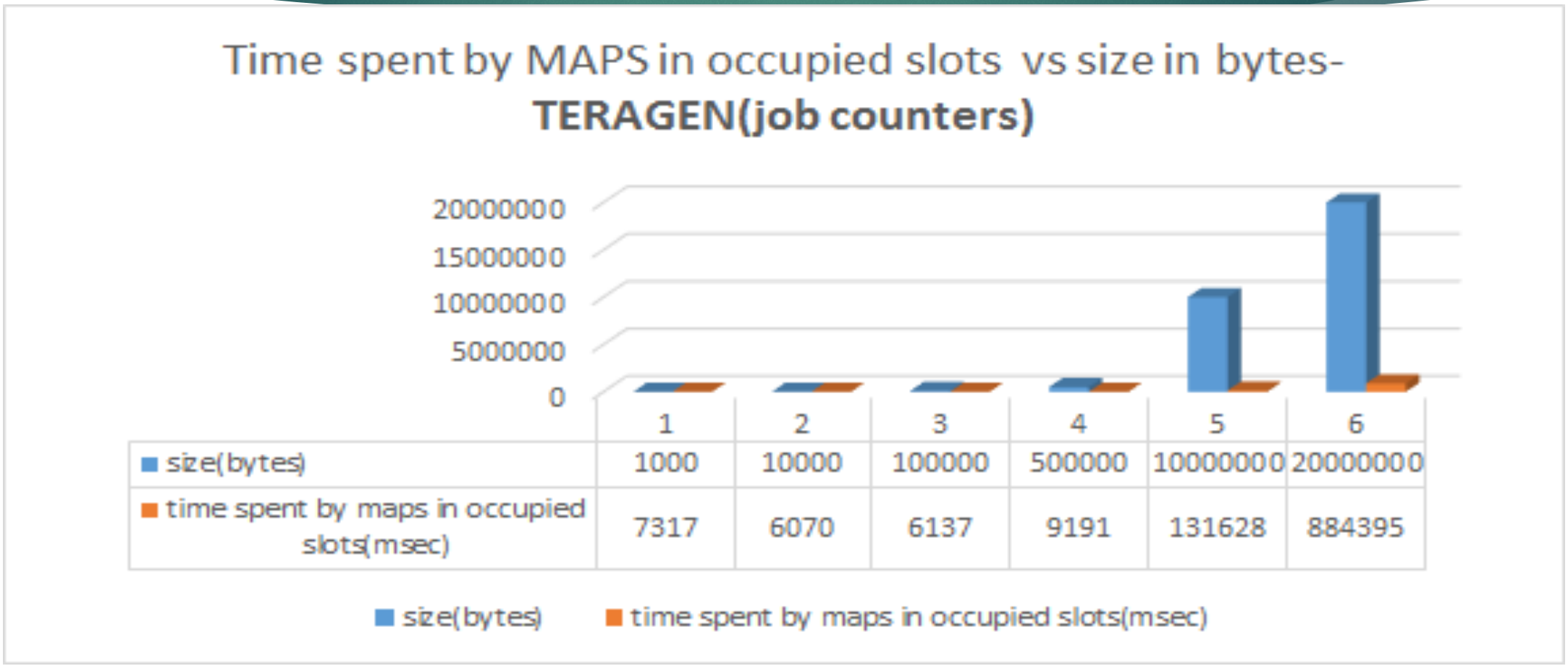
Statistics


TERASORT (size vs time)
400
350
300
250
200
150
100
50
0
time in sec
size in bytes
-5000000
0
5000000
10000000
15000000
20000000
25000000
TERASORT (size vs time)

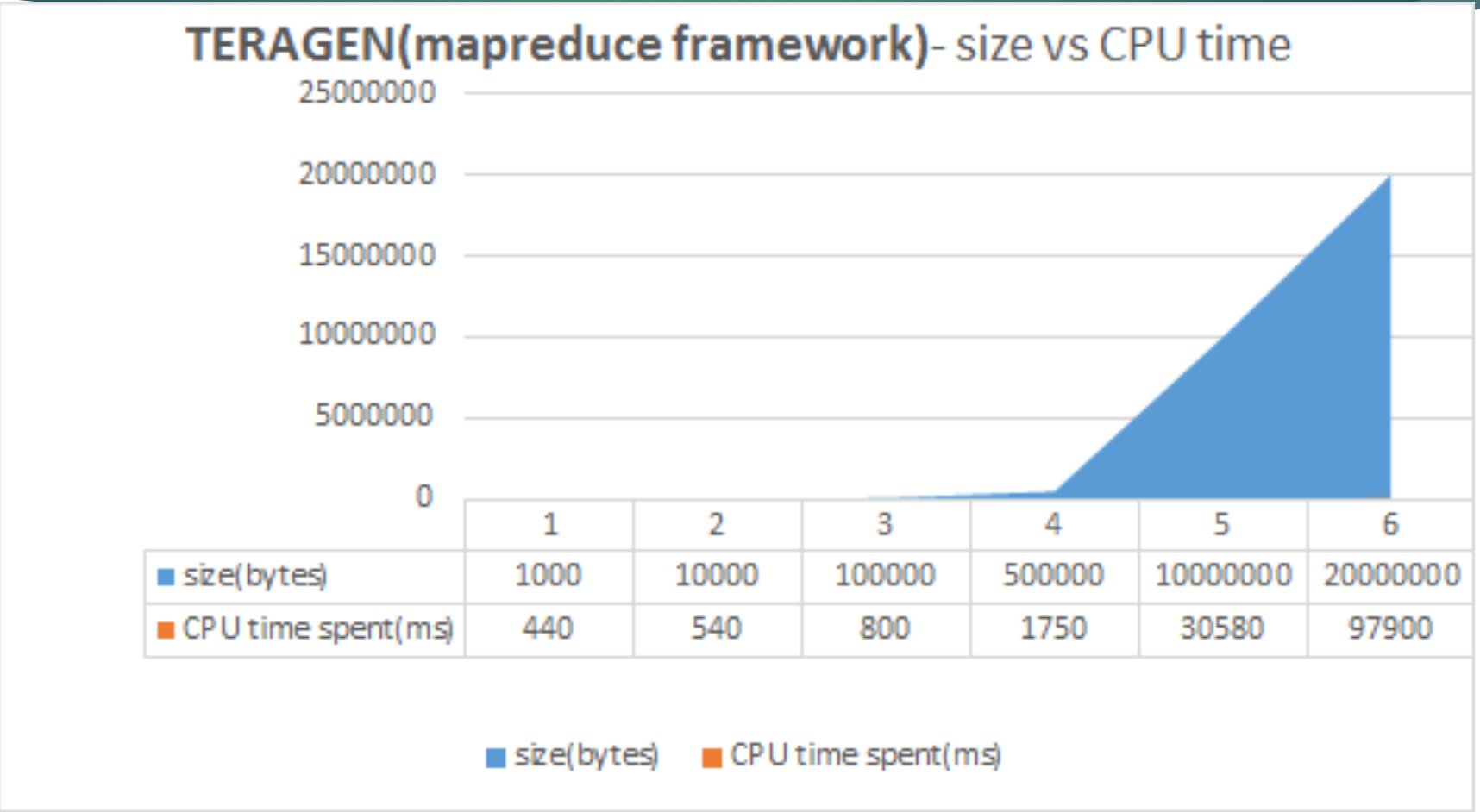
Statistics


TERAVALIDATE (size vs time)
60
50
40
30
20
10
0
time in sec
-5000000
0
5000000
10000000
15000000
20000000
25000000
size in bytes
TERAVALIDATE (size vs time)

Statistics

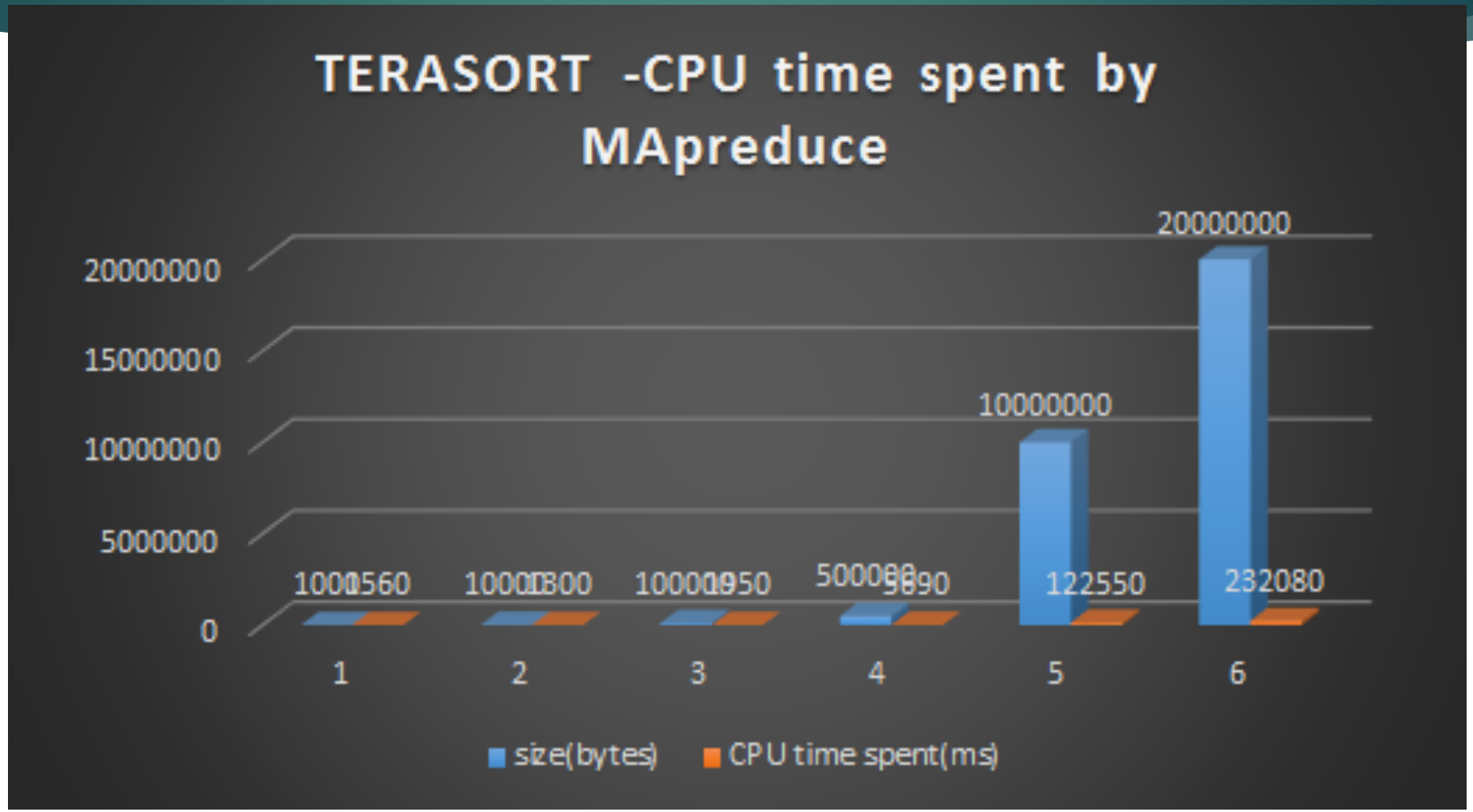

Time spent by MAPS in occupied slots vs size in bytes-
TERAGEN(job counters)
20000000
15000000
10000000
5000000
0
1
2
3
4
5
6
size(bytes)
1000
10000
100000
500000
10000000
20000000
time spent by maps in occupied slots(msec)
7317
6070
6137
9191
131628
884395
size(bytes)
time spent by maps in occupied slots(msec)

Statistics

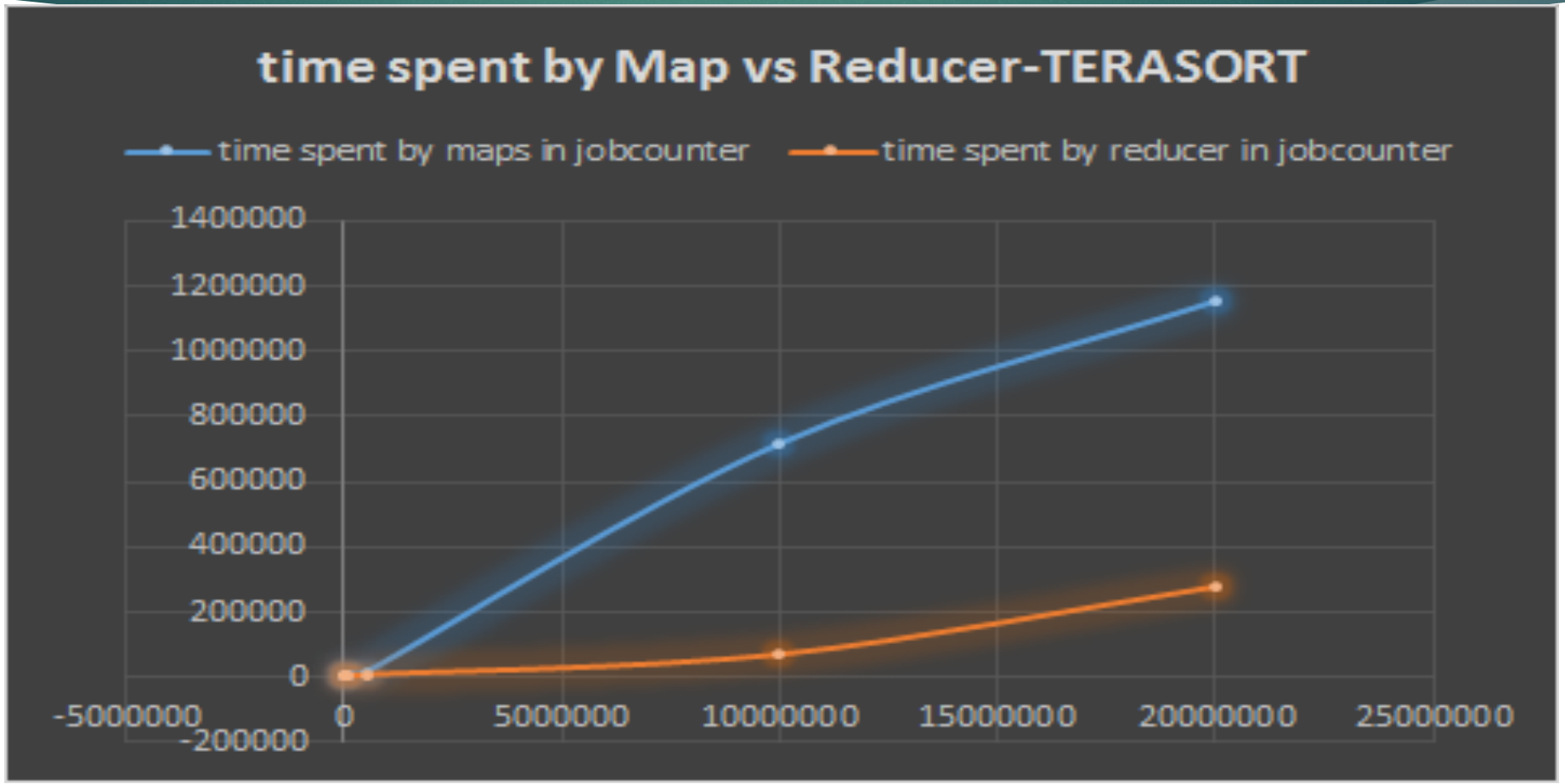


	1	2	3	4	5	6
size(bytes)	1000	10000	100000	500000	10000000	20000000
CPU time spent(ms)	440	540	800	1750	30580	97900

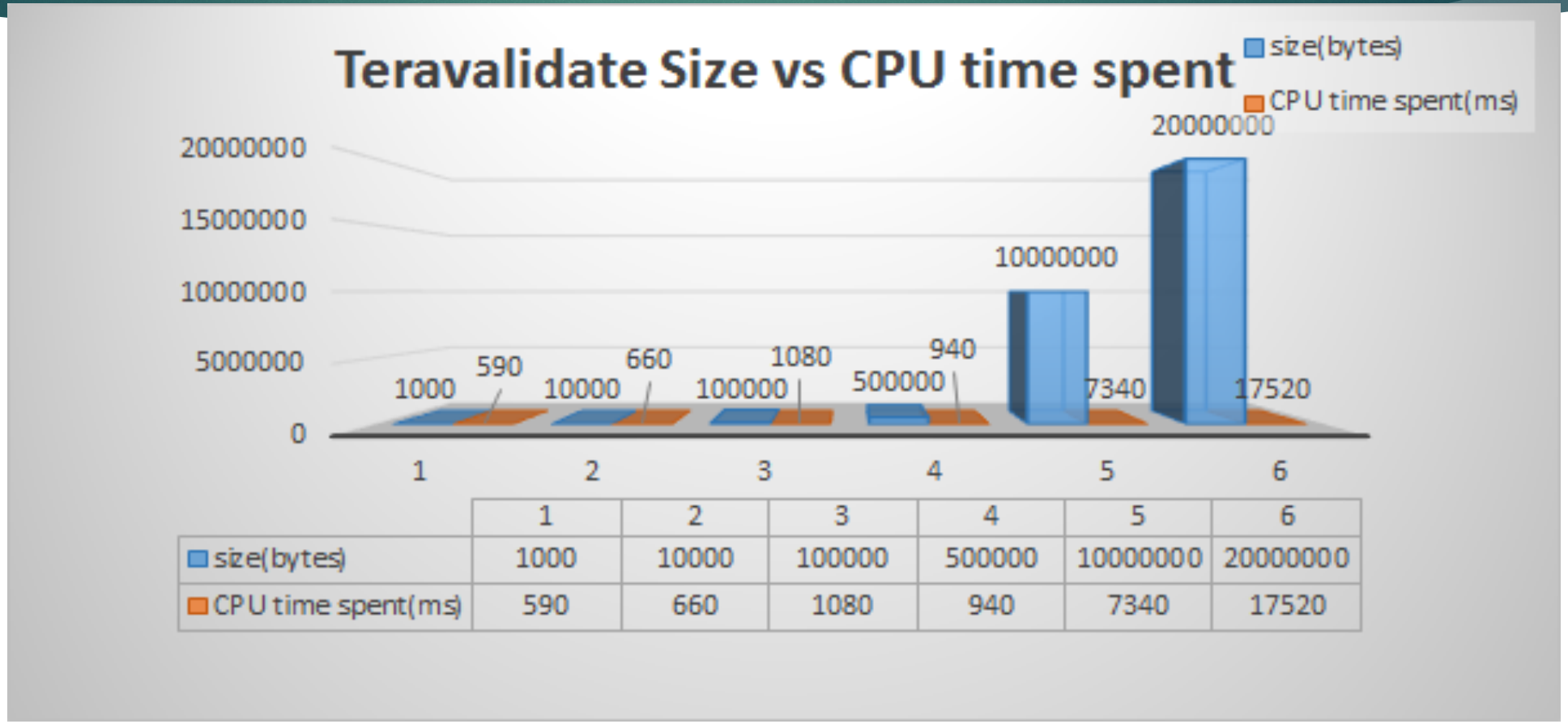
Statistics


TERASORT -CPU time spent by MApreduce
20000000
15000000
10000000
5000000
0
1000560
10000300
10000950
5000000
3690
10000000
122550
20000000
232080
1
2
3
4
5
6
size(bytes)
CPU time spent(ms)

Statistics

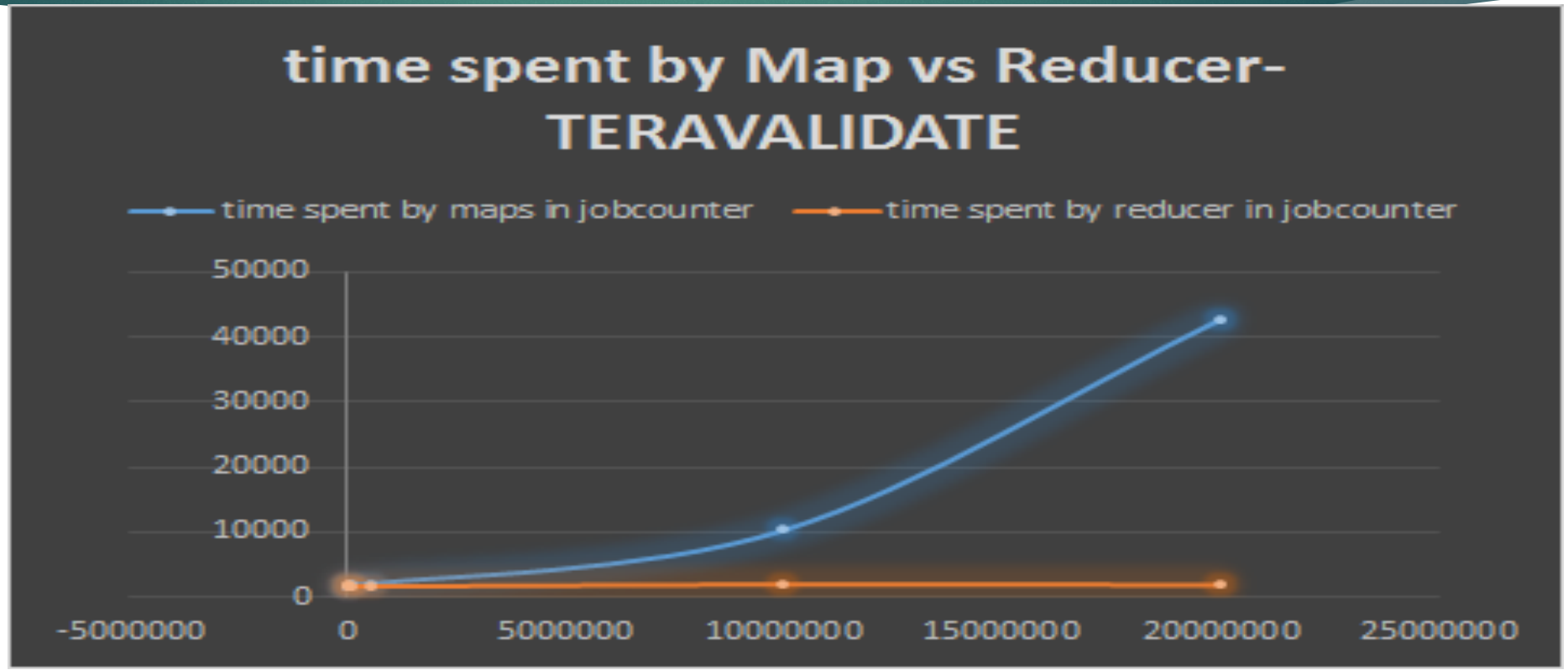

time spent by Map vs Reducer-TERASORT
time spent by maps in jobcounter
time spent by reducer in jobcounter
1400000
1200000
1000000
800000
600000
400000
200000
0
-200000
-5000000
0
5000000
10000000
15000000
20000000
25000000

Statistics

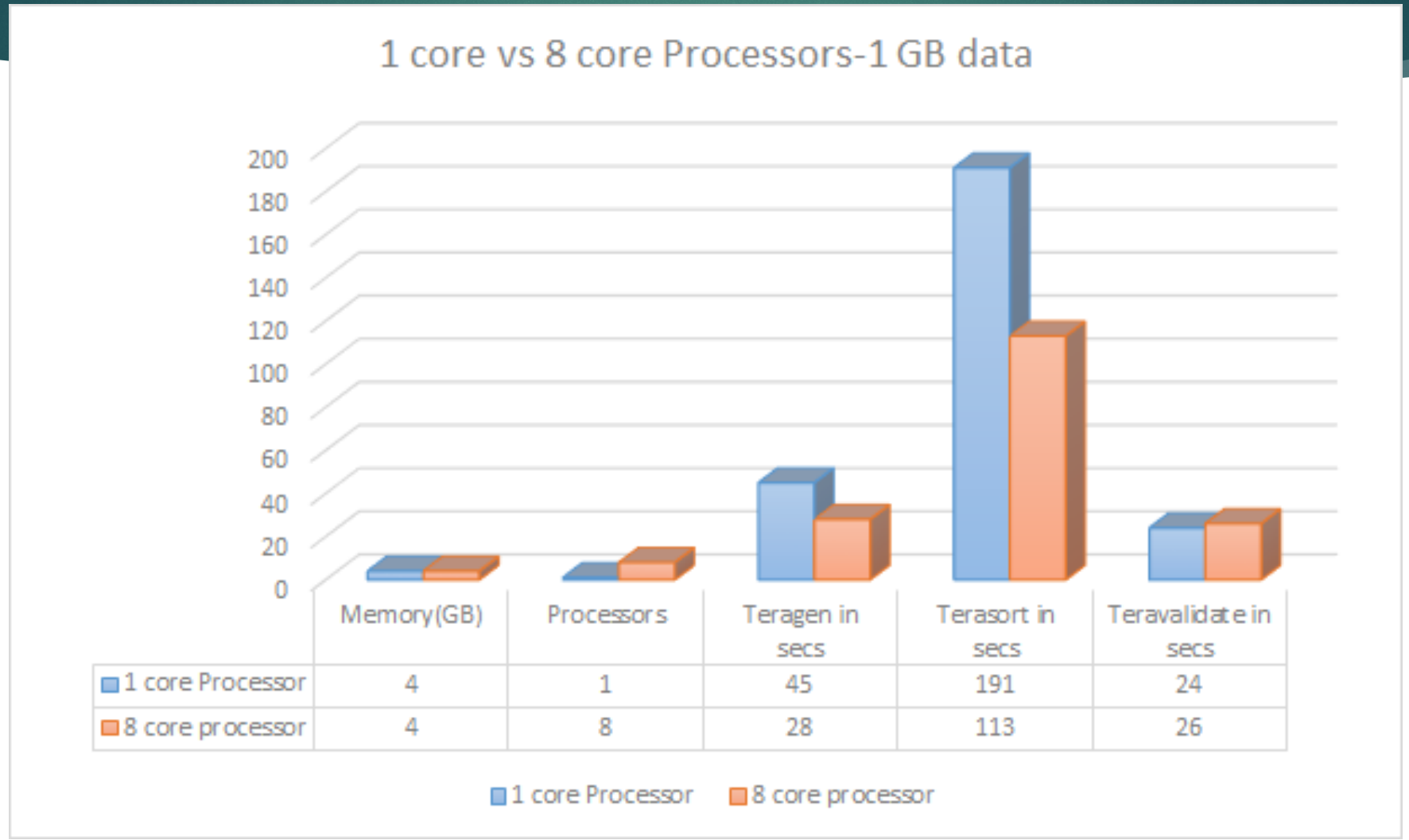


	1	2	3	4	5	6
size(bytes)	1000	10000	100000	500000	10000000	20000000
CPU time spent(ms)	590	660	1080	940	7340	17520

Statistics


time spent by Map vs Reducer-
TERAVALIDATE
time spent by maps in jobcounter
time spent by reducer in jobcounter
50000
40000
30000
20000
10000
0
-5000000
0
5000000
10000000
15000000
20000000
25000000

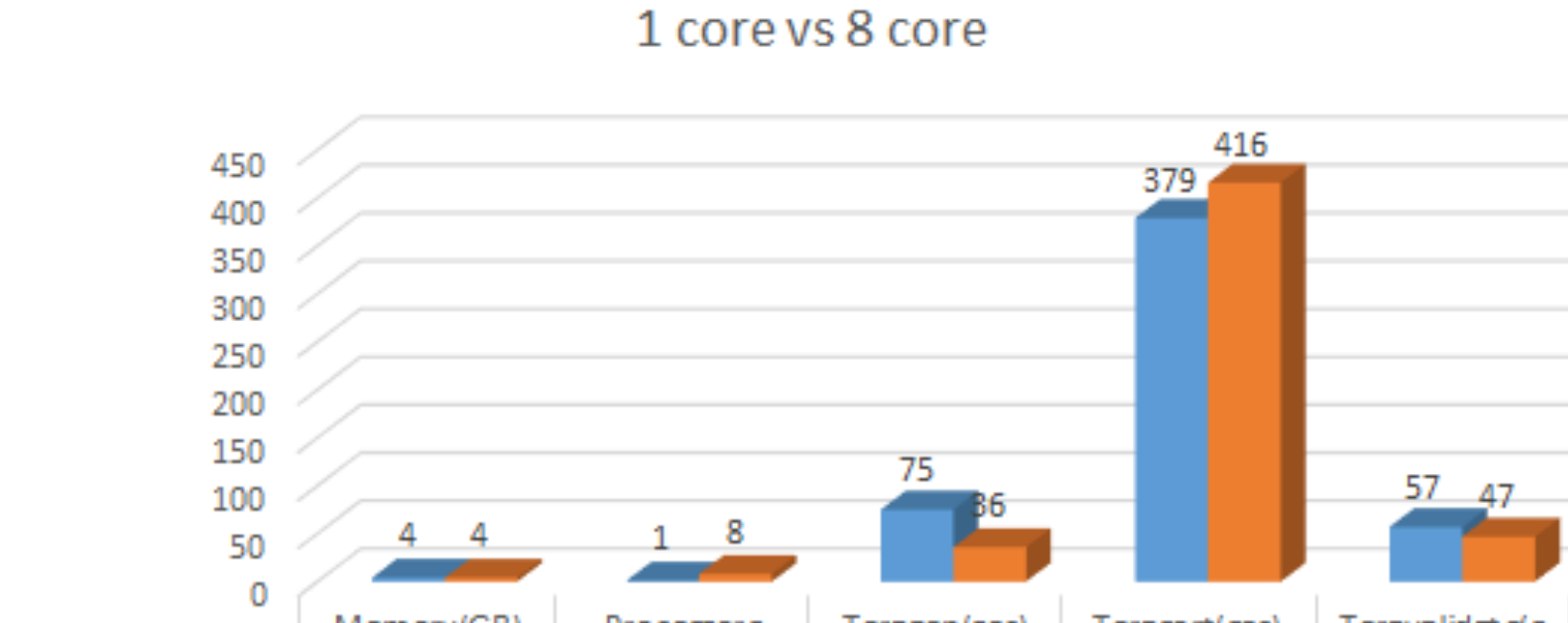
Statistics



	Memory(GB)	Processors	Teragen in secs	Terasort in secs	Teravalidate in secs
1 core Processor	4	1	45	191	24
8 core processor	4	8	28	113	26

Input size:1GB

Statistics



Input size:2GB

References

- https://hadoop.apache.org/docs/r2.4.1/api/org/apache/hadoop/examples/terasort/package-summary.html
- http://sortbenchmark.org/YahooHadoop.pdf
- https://courses.cs.washington.edu/courses/cse490h/08au/lectures/algorithms.pdf