The HiBench Benchmark Suite: Characterization of the MapReduce-Based Data Analysis

Shengsheng Huang, Jie Huang, Jinquan Dai, Tao Xie, and Bo Huang
Intel China Software Center, Shanghai, P.R.China, 200241

Contributions

--The design of HiBench is essential for the community to properly evaluate and characterize the Hadoop framework.

--The quantitative characterization of different workloads in HiBench, including their data access patterns, system resource utilizations and HDFS bandwidth

--The evaluation of different deployment (both hardware and software) choices using HiBench, in terms of speed, throughput and resource utilizations

TABLE I
CONSTITUENT BENCHMARKS

Category	Workload
Micro Benchmarks	Sort
	WordCount
	TeraSort
Web Search	Nutch Indexing
	PageRank
Machine Learning	Bayesian Classification
	K-means Clustering
HDFS Benchmark	EnhancedDFSIO

--The first seven are directly taken from their open source implementations

--the last one is an enhanced version of the DFSIO benchmark that extended to evaluate the aggregated bandwidth delivered by HDFS

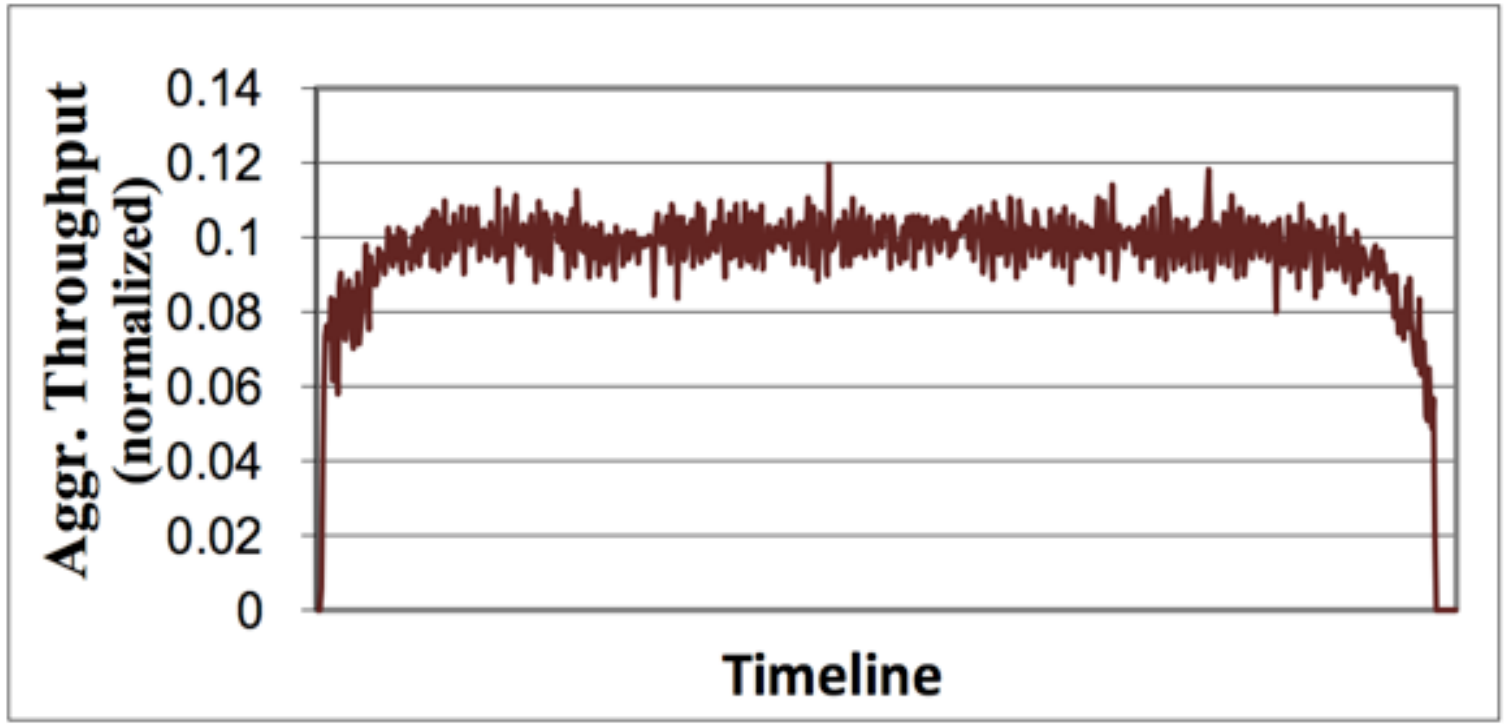

Fig. 1 Normalized aggregated throughput curve of Enhanced DFSIO (write)

-- the original DFSIO program only computes the average I/O rate and throughput of each map task

-- the Enhanced DFSIO workload computes the aggregated HDFS throughput by averaging the throughput value of each time slot in the steady period

Evaluation and Characterization

-- evaluate and characterize the Hadoop framework using HiBench
  --speed (i.e., job running time)
  --throughput (i.e., the number of tasks completed per minute)
  --aggregated HDFS bandwidth
  --system resource (e.g., CPU, memory and I/O) utilizations
  --data access patterns

-- four slaves in the Hadoop

Hadoop Workload Characterization--data access patterns

Workload	Job	Job/ Map Input	Map Output (Combiner Input)	Shuffle Data (Combiner Output)	Job/ Reduce Output
Sort	Sort	60G	60G	60G (no combiner)	60G
Word Count	WordCount	60G	85G	3.99G	1.6G
TeraSort	TeraSort	1T	139G [b]	139G (no combiner)	1T
Nutch Indexing	Nutch Indexing	8.4G [b]	26G	26G (no combiner)	6.3G
PageRank[a]	Dangling-Pages	1.21G	81.7K	672B	30B
	Update-Ranks	1.21G	5.3G	5.3G (no combiner)	1.21G
	SortRanks	1.21G	86M	86M (no combiner)	167M
Bayesian Classification	Feature	1.8G	52G	39.7G	35G
	Tfldf	35G	26G	22.8G	13.3G
	Weight-Summer	13.3G	16.8G	8.7G	8.2G
	Theta-Normalizer	13.3G	5.6G	163K	8.6K
K-means Clustering [c]	Centroid-Computing	66G	67G	303K	4.6K
	Clustering	66G	66.3G	no combiner	no reducer

-- summarizes the data access patterns of MapReduce model for all workloads

-- Enhanced DFSIO that has no relevant input and output

Hadoop Workload Characterization--speed

Workload	Job	Avg. Map Task vs. Avg. Reduce Task Execution Time Ratio[a]	Map Stage vs. Reduce Stage Execution Time Ratio[b]
Sort	Sort	2.00%	33.36%
WordCount	WordCount	55.98%	94.67%
TeraSort	TeraSort	0.26%	70.71%
Nutch Index	Nutch Indexing	6.22%	31.16%
PageRank	DanglingPages	15.36%	124.12%
	UpdateRanks	49.90%	66.81%
	SortRanks	5.74%	49.50%
Bayesian Classicification	Feature	14.33%	57.93%
	Tfldf	7.86%	75.57%
	WeightSummer	19.32%	72.64%
	ThetaNormalizer	29.37%	87.75%
K-means Clustering	CentroidComputing	5.67%	102.58%
	Clustering	no reduce tasks	no reduce tasks

-- summarizes the ratio of map execution time vs. reduce execution time of each workload

-- Map Stage is defined as the period between when the first Map Task starts and when the last Map Task ends. The Reduce Stage is defined similarly

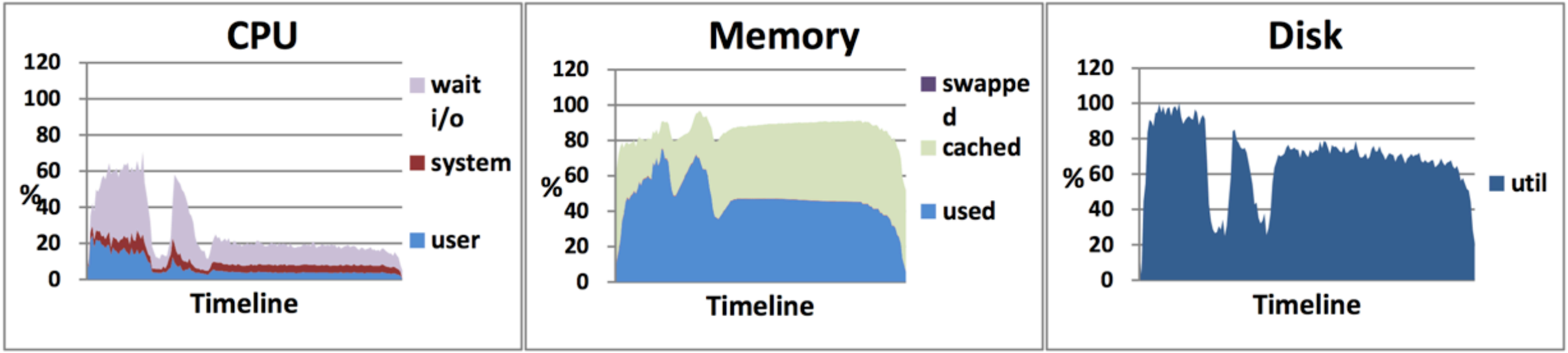

Fig. 2 System Utilization of Sort

-- Sort workload transforms data from one representation to another, the shuffle data and the job output of Sort are of the same size as the job input

-- Sort workload is mostly I/O bound, having moderate CPU utilization and heavy disk I/O

-- considering the large amount of shuffle data, it is expected to have network I/O bottlenecks in the shuffle stage when running on large (multi-rack) clusters

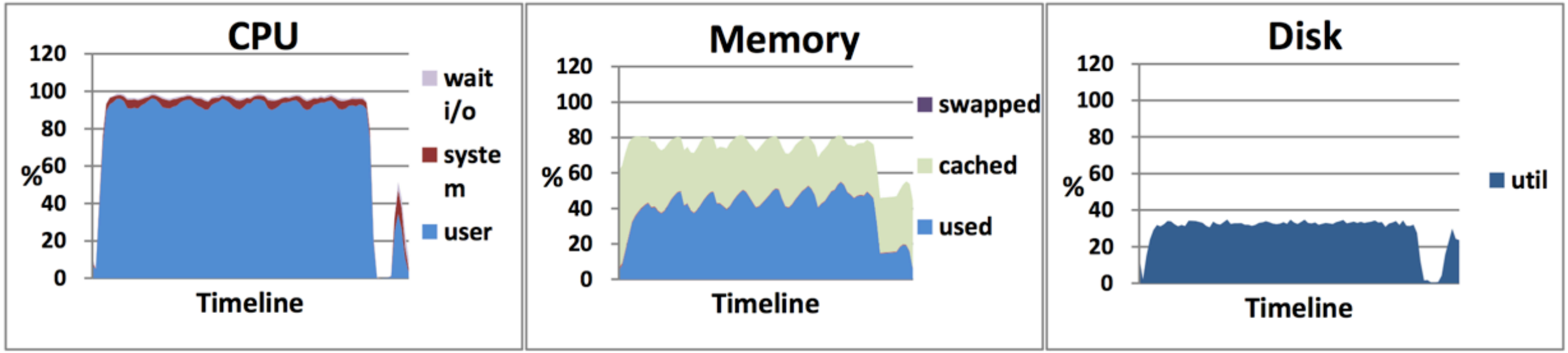

Fig. 3 System Utilization of WordCount

--WordCount workload extracts a small amount of interesting data from a large data set, the shuffle data and the job output of WordCount are much smaller than the job input

--WordCount workload is mostly CPU bound (especially during the map stage), having high CPU utilization, light disk/network I/O

--its behavior is expected to remain somewhat the same even on larger clusters

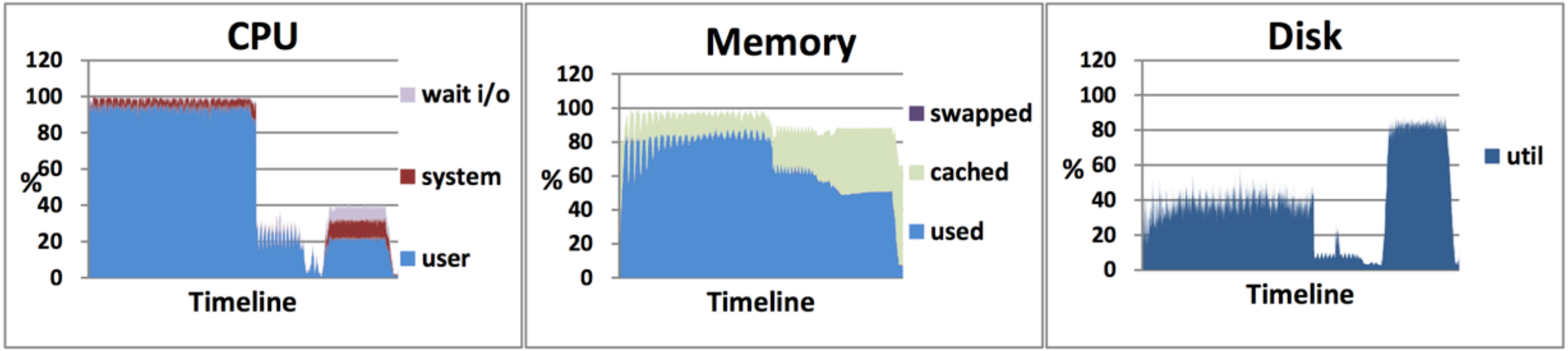

Fig. 4 System Utilization of TeraSort

--TeraSort workload is similar to Sort and therefore is I/O bound in nature

--However, compressing shuffle data (i.e., map output) in the experiment so as to minimize the disk and network I/O during shuffle

--TeraSort have very high CPU utilization and moderate disk I/O during the map stage and shuffle phases, and moderate CPU utilization and heavy disk I/O during the reduce phases

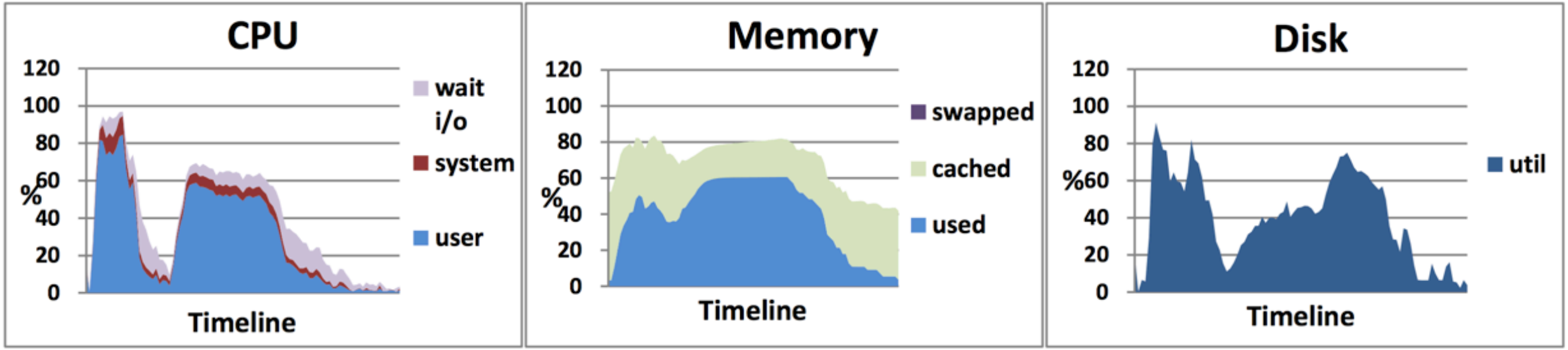

Fig. 5 System Utilization of Nutch Indexing

--Nutch Indexing is CPU bound during map stage and more disk I/O bound (with about 60% CPU utilizations) in the reduce stage

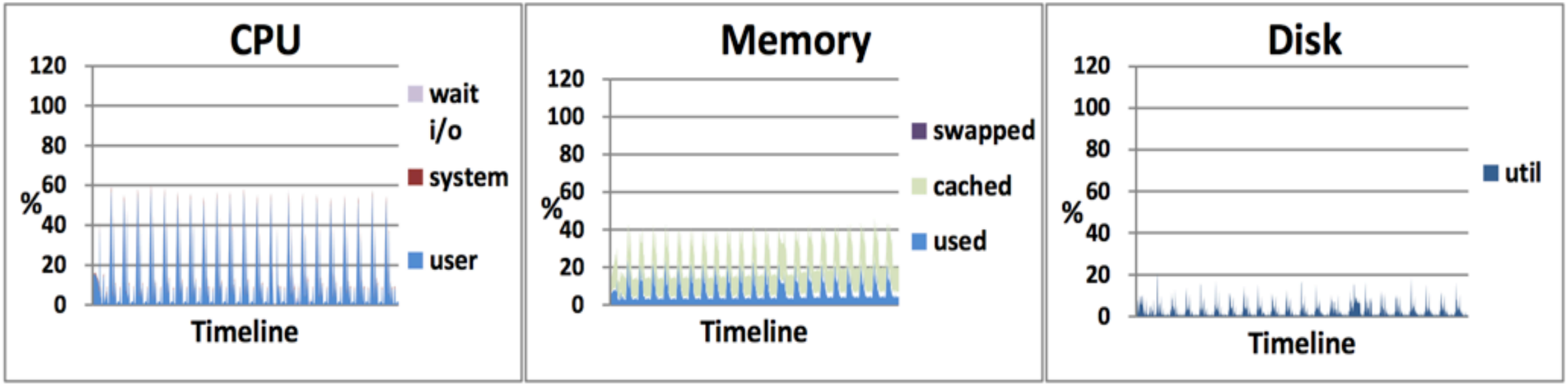

Fig. 6 System Utilization of PageRank

--PageRank workload spends most of its time on the iterations of several jobs, and these jobs are generally CPU bound, with low to medium disk I/O and memory utilizations

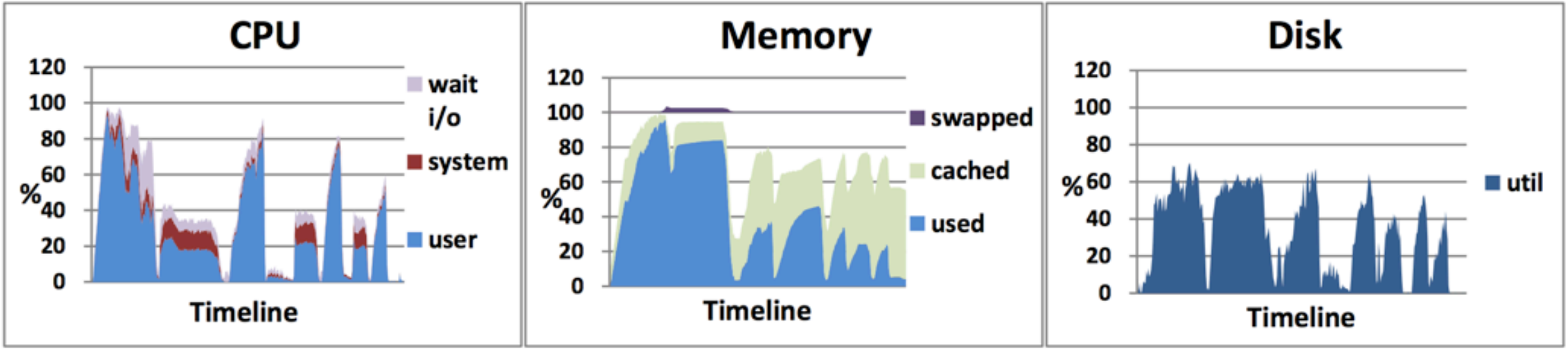

Fig. 7 System Utilization of Mahout Bayesian Classification

--Bayesian Classification workload contains four chained Hadoop jobs, and the first job is the most time consuming (taking about half of the total running time in the experiment)

-- the four jobs are all mostly disk I/O bound, except that the map tasks of the first job also have high (over 80%) CPU utilizations

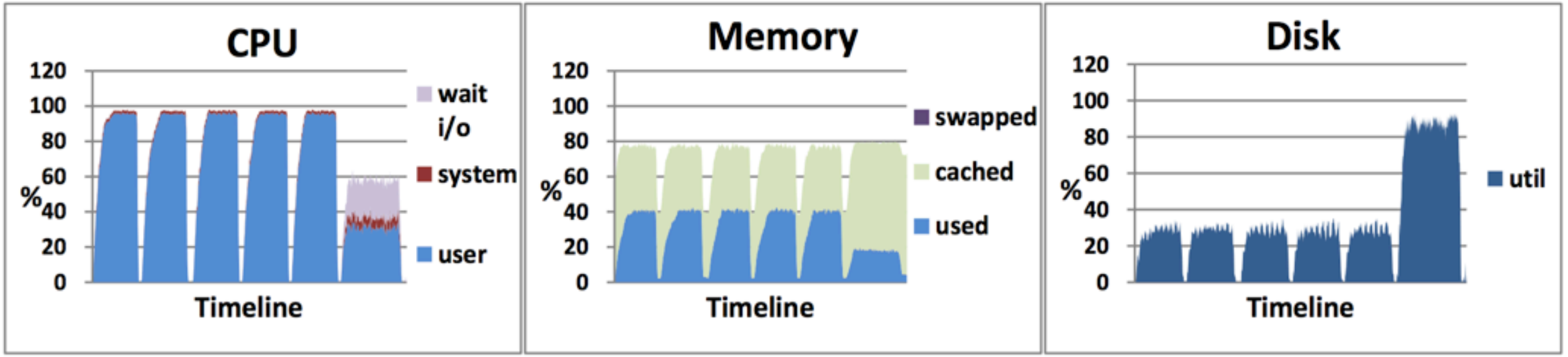

Fig. 8 System Utilization of Mahout K-Means Clustering

-- is mostly CPU bound, because its combiner swallows most of the map outputs and consequently the shuffle data are relatively small

-- the clustering job is I/O bound, mostly due to the output to HDFS of the map tasks (as there are no reduce tasks in the clustering job)

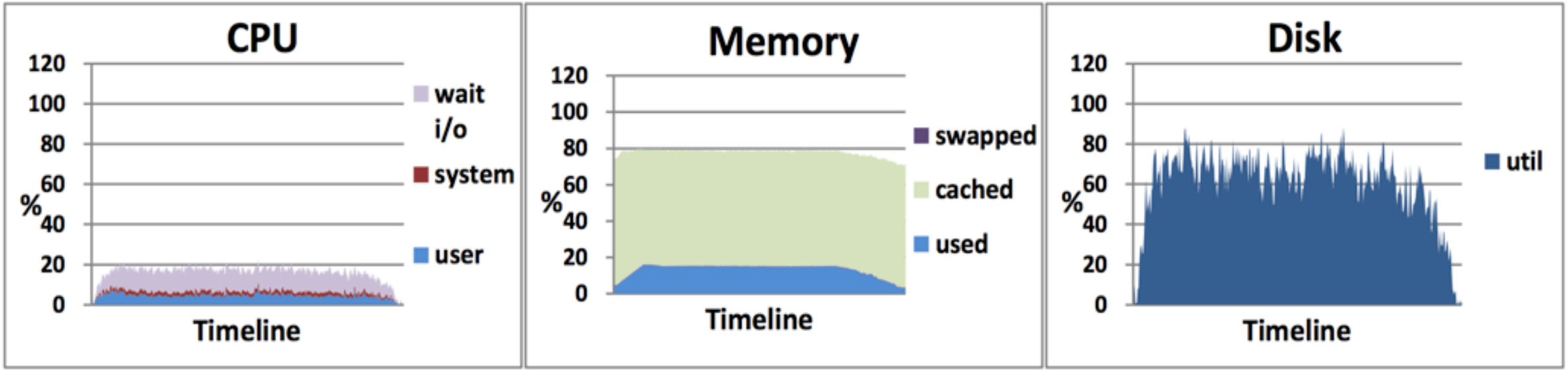

Fig. 9 System Utilization of DFSIO (read)

--Enhanced DFSIO workload only tests the HDFS throughput, it is completely disk I/O bound

--This experiment tuned the workloads to keep these data in memory as much as possible, and consequently, as shown in Fig. 2 – 9, all of them have high memory consumptions (over 80% memory utilization), except PageRank that performs a lot of computations on a relatively small data set

Conclusion:

--Hadoop workload usually has very heavy disk and network I/O, unless the map output size is (or can be combined to be) very small (e.g., as in the case of WordCount and K-means Clustering)

--The best performance (total running time) of Hadoop workloads is usually obtained by accurately estimating the size of the map output, shuffle data and reduce input data, and properly allocating memory buffers to prevent multiple spilling (to disk) of those data

TABLE V
TEST CONFIGURATION OF ENHANCED DFSIO BENCHMARKING

Workload Configuration	Size of Each File to Be Read/Written (MB)	Total Number of Files to Be Read/Written
Write – sameLoadPerCluster	360	256
Write – sameLoadPerNode	360	16 x number of slave nodes
Write – sameLoadPerMap	360	16 x number of map slots per node
Read - sameLoadPerCluster	720	256
Read - sameLoadPerNode	720	16 x number of slave nodes
Read - sameLoadPerMap	720	16 x number of map slots per node

--examine the results of Enhanced DFSIO workload and the original DFSIO benchmark, using the *same-load-per-cluster*, *same-load-per-node* and *same- load-per-map* configurations, and for different numbers of slaves and different *map slots* (i.e. the maximum number of parallel map tasks that can run on each server)

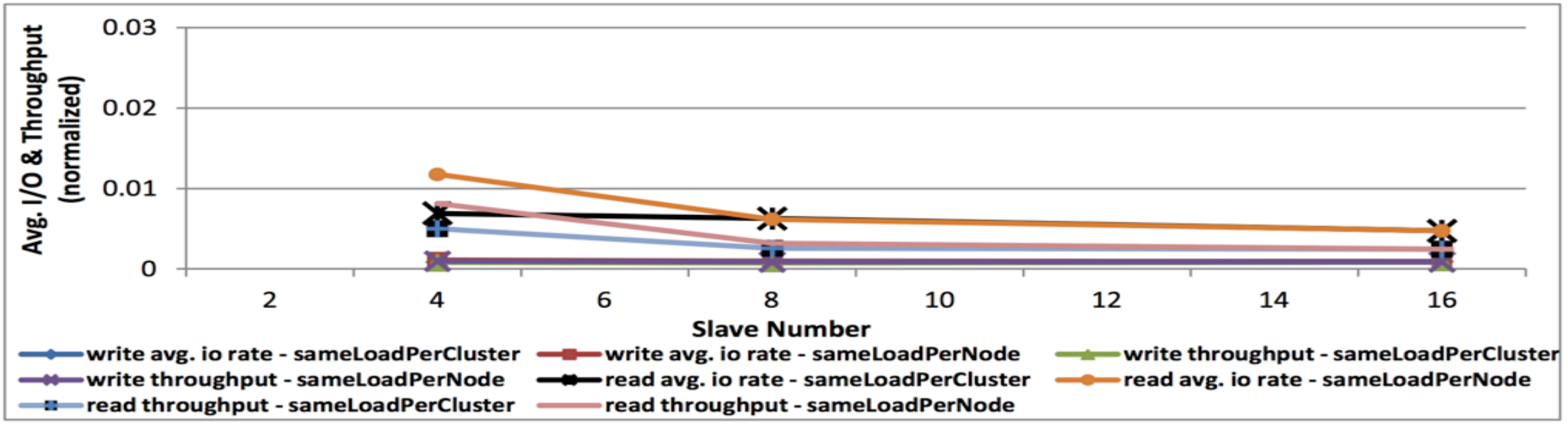

Fig. 10 DFSIO: normalized average I/O rate & throughput as function of slave number

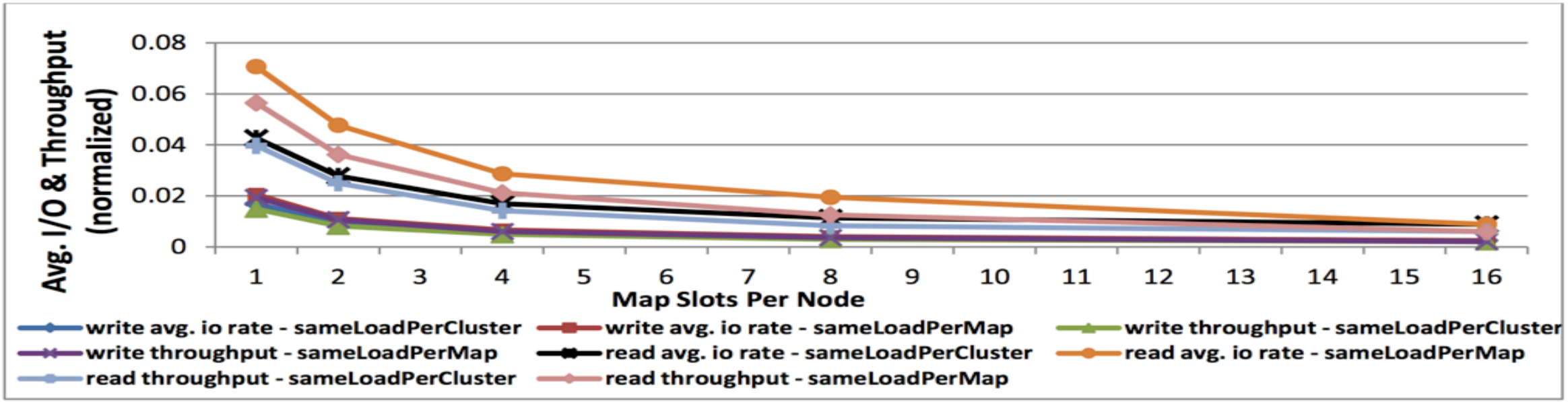

Fig. 11 DFSIO: normalized average I/O rate & throughput as function of map slots per node

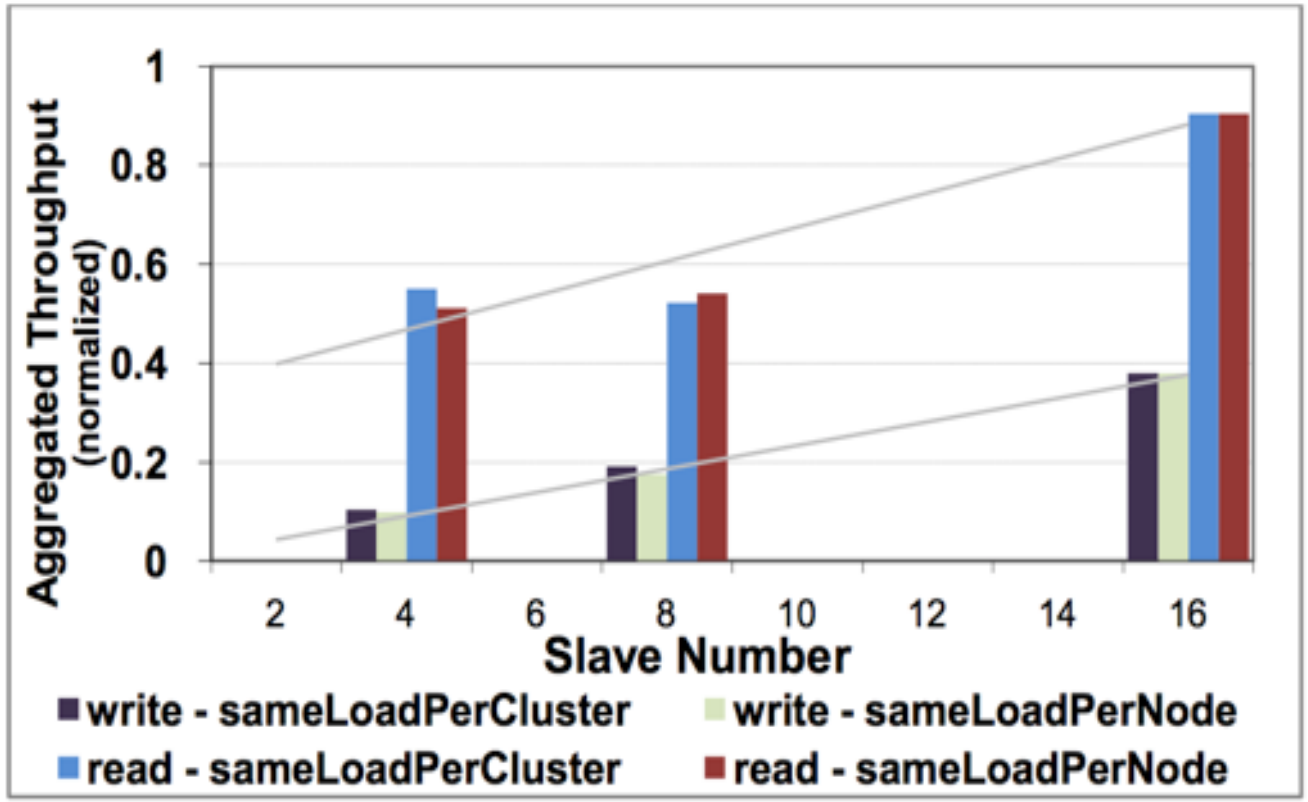

Fig. 12 Enhanced DFSIO: normalized aggregated throughputs as function of slave number

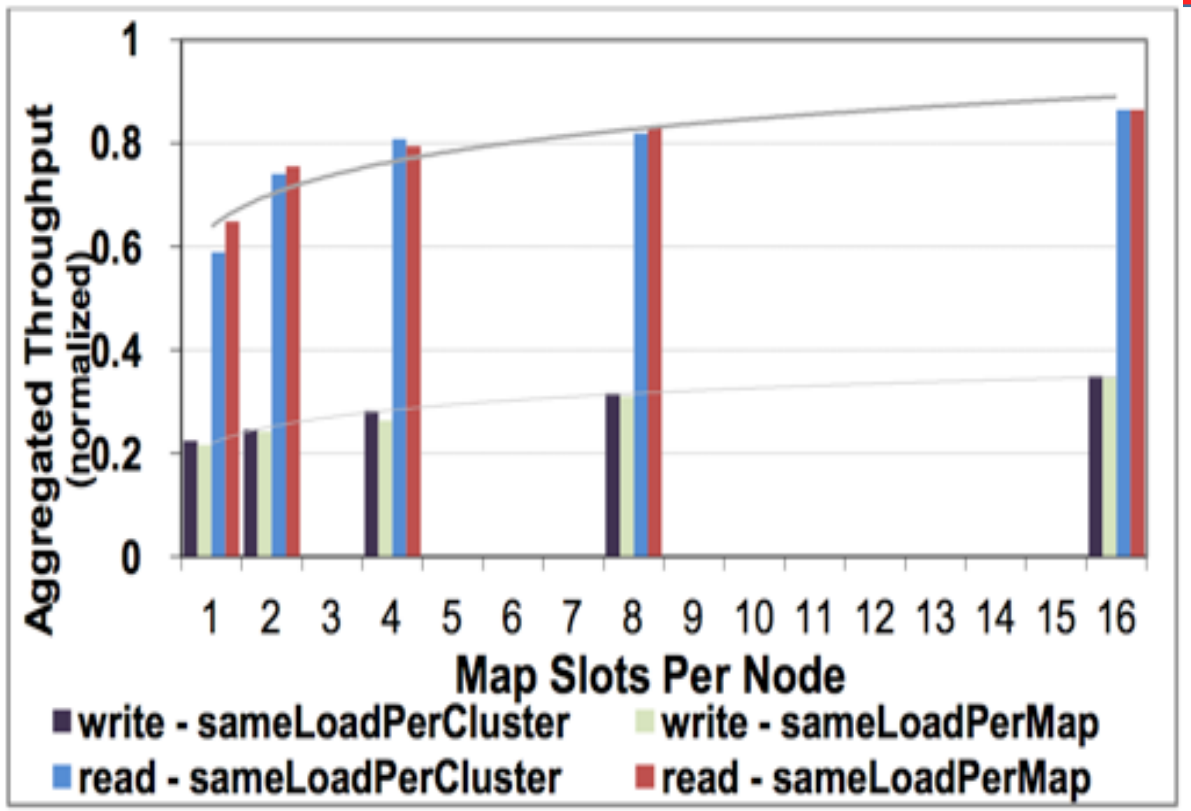

Fig. 13 Enhanced DFSIO: normalized aggregated throughputs as function of map slots per node

--Conclusion:DFSIO is not appropriate for the evaluation of the aggregated bandwidth delivered by HDFS, which is the design goal of HDFS and GFS

--Both workloads measure the number of bytes read/written at the application level; however, in HDFS write operations are buffered and performed asynchronously

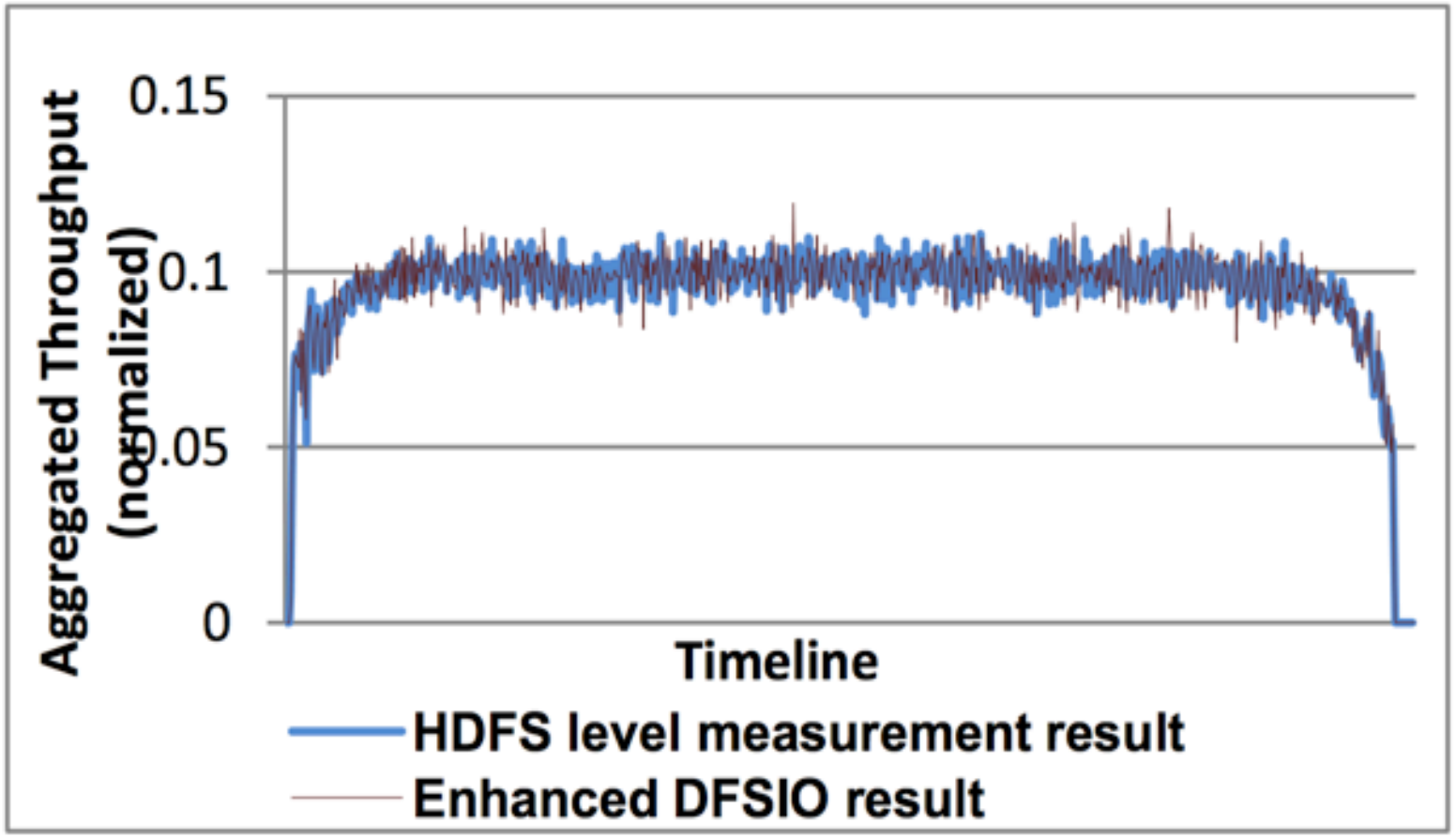

Fig. 14 Normalized aggregated throughput of Enhanced DFSIO vs. HDFS level measure result

--compares the aggregated throughput measured at the application level by Enhanced DFSIO and measured at the HDFS level (i.e., when the bytes to be written are acknowledged by HDFS), and those two results are almost the same

--Conclusion: the aggregated throughput measured at the application level by the Enhanced DFSIO can be used to properly evaluate the aggregated throughput of the HDFS cluster

--*Comparison of processor platforms*

--*Comparison of Hadoop versions*

--Hadoop 0.19.1 and Hadoop 0.20.0

--Map stage in v0.20.0 is slower (and uses more memory) than v0.19.1, but the overall job running time is about the same or even slightly faster in v0.20.0

--In Hadoop 0.19.1, most of reduce tasks cannot start until after all the map tasks are done. in Hadoop 0.20.0, the JobTracker can schedule multiple tasks in a heartbeat ,so former performance issue is no long there