2014 9th IEEE International Conference on Networking, Architecture, and Storage

# Performance Characterization of Hadoop and DataMPI Based on Amdahl's Second Law

Fan Liang [1,2], Chen Feng [1,2], Xiaoyi Lu [3], Zhiwei Xu [1]

[1]*Institute of Computing Technology, Chinese Academy of Sciences*
[2]*University of Chinese Academy of Sciences*
[3]*Department of Computer Science and Engineering, The Ohio State University*
{liangfan, fengchen, zxu}@ict.ac.cn, luxi@cse.ohio-state.edu

***Abstract*—Amdahl's second law has been seen as a useful guideline for designing and evaluating balanced computer systems for decades. This law has been mainly used for hardware systems and peak capacities. This paper utilizes Amdahl's second law from a new angle, i.e., evaluating the influence on systems performance and balance of the application framework software, a key component of big data systems. We compare two big data application framework software systems, Apache Hadoop and DataMPI, with three representative application benchmarks and various data sizes. System monitors and hardware performance counters are used to record the resource utilization, characteristics of instructions execution, memory accesses, and I/O rates. These numbers are used to reveal the three runtime metrics of Amdahl's second law: CPU speed (GIPS), memory capacity (GB), and I/O rate (Gbps). The experiment and evaluation results show that a DataMPI-based big data system has better performance and is more balanced than a Hadoop-based system.**

***Keywords*-Big data, Amdahl's second law, Hadoop, DataMPI**

## I. Introduction

Data explosion is becoming an irresistible trend with the development of Internet, social network, e-commerce, etc. Over the last decade, there have been emerging a lot of systems and frameworks for big data, such as MapReduce [1], Hadoop [2], Dyrad [3], Yahoo! S4 [4], Spark [5] and so on. Evaluation and comparison of these systems are becoming an important issue. Most of the previous work focuses on the performance of the big data systems, and pays less attention to system balance. In the traditional computer system area, much work has been done to help us analyze the architecture balance, such as Amdahl's second law [6].

Amdahl's second law states that for every one instruction per second of processing speed (CPU speed), a balanced computing system needs to provide one bit per second of I/O rate and one byte of main memory capacity. The Amdahl number of a system is its minimum of *I/O rate/CPU speed* and *memory capacity/CPU speed*. The Amdahl's second law has been a useful guideline for designing and evaluating balanced computer systems for decades. To save system cost, many supercomputing systems select an Amdahl number as low as 0.001, resulting in unbalanced designs. Recently, researchers are applying this law to design more balanced data intensive computer systems. For instance, the GrayWulf and the DataScope systems for scientific data processing are designed to have an Amdahl number between 0.5 to 1, and eBay's data analytics system has an Amdahl number over 0.83 [7].

Traditional applications of Amdahl's second law mainly consider balanced hardware systems and peak capacities, as shown in Figure 1. In data intensive computing systems, the application frameworks, also play an important role to affect system balance and performance. In this paper, we compare and analyze two big data computing application framework software systems, Apache Hadoop and DataMPI [8], to see their influence on system balance. Our experiment uses three representative application benchmarks with different data sizes on the same hardware machines. The three benchmarks are: Sort, an I/O-intensive MapReduce application; WordCount, a CPU-intensive MapReduce application; and K-means, an iterative application. The sizes of the datasets are selected to cover both in-memory and out-memory cases.

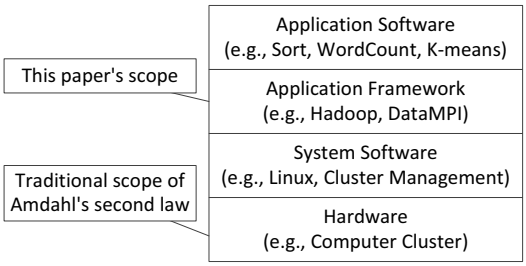



Fig. 1. The Architecture Stack of Big Data System

To understand the system execution patterns and the insights of Amdahl's second numbers, which are defined in section III-C, we characterize the CPU and memory behaviors by monitoring various architecture metrics such as cycles, instructions, first-/second-/last-level processor cache miss rates, instruction/data TLB miss rates, branch misprediction ratio, and resource utilization by monitoring the CPU usage, memory footprint, disk I/O bandwidth and network I/O bandwidth.

Our contributions in this paper are:

- We give a preliminary evaluation of two big data systems, Hadoop and DataMPI. By profiling the system execution characteristics and execution patterns, we observe DataMPI can execute more efficiently than Hadoop.
- Based on the resource utilization and architecture metrics, we observe that the Amdahl's second law is suitable to

978-1-4799-4087-5/14 $31.00 © 2014 IEEE
DOI 10.1109/NAS.2014.39

the big data systems.
- Based on the Amdahl I/O number and Amdahl memory number, we observe the balance of DataMPI-based big data system is better than that of Hadoop-based system.

The rest of this paper is organized as follows. Section II discusses background and related work. Section III states our experimental methodology. The evaluation results and analysis are given in Section IV. Section V concludes this paper.

## II. Background and Related Work

### A. Big Data Systems

This paper focuses on two big data systems: Apache Hadoop and DataMPI.

Apache Hadoop [2] is an open-source implementation of the MapReduce programming model, which has been widely used in various areas and applications, such as log analytics, machine learning, search engine, etc. Because Hadoop spills the intermediate data on disks, it often has high I/O operations.

DataMPI [8] is a key-value pair based communication library which extends MPI for Hadoop-like big data computing systems. By extending the MPI operations, DataMPI can adopt the state-of-the-art technologies in high performance computing (HPC) to solve big data computing problems.

### B. System Characterization

System characterization studies are helpful to identify the systems' bottlenecks and figure out solutions for optimizing performance. Much work has been done on different areas, such as networking [9], [10], storage [11], [12], and cluster resource management [13].

There have been many previous efforts analyzing the characteristics of MapReduce. Lu et. al. [14] discuss the improvement potentials of high-performance communication primitives over Hadoop. Stonebraker et. al. [15] compare the MapReduce and parallel databases. Jiang et. al. [16] offer an in-depth study of the MapReduce performance over traditional MapReduce applications. Mishra et. al. [17] give an insight into workloads characterization based on the CPU and memory behaviors. Authors in [18], [19], [20] show that the workload behavior at the granularity of job and task based on the statistical profile trace. Chang et. al. [21] use Amdahl's metrics to evaluate database systems under different workloads and propose some suggestions on designing balanced data analytics systems.

### C. Amdahl's Second Law

Gene Amdahl has set up many rules of thumb for computer architects since the 1960s which have been applied for many years [22]. Bell et. al. stated that in the era of data intensive computation, Amdahl's second law should be applied to build balanced system [23]. Szalay et. al. [24] presented a system, GrayWulf, for data intensive computation based on the design of balanced system. Considering the power consumption, authors in [25] proposed an alternative architecture comprising the Amdahl-balanced blades for energy-efficient computation. Cohen et. al. [26] studied the interplay between the processor and network interconnection based on Amdahl's second law.

Previous work focuses on the hardware to build a balanced system. In this paper, we give a uniform evaluation of two big data application frameworks to show the differences of their execution behaviors based on Amdahl's second law.

## III. Experimental Methodology

### A. Workloads

We choose WordCount, Sort, and K-means from BigDataBench [27] as our workloads.

*1) WordCount:* WordCount is a common operation in big data computing, which counts the appearance times of each word in the input dataset. WordCount is a CPU-intensive MapReduce workload. In most cases, the data sizes of communication and output are much smaller than the input size.

*2) Sort:* Sort is another common operation of big data computing, which sorts the input data based on the keys. Sort is an I/O-intensive MapReduce workload.

*3) K-means:* K-means is a popular algorithm for cluster analysis in data mining, which partitions input objects into $k$ clusters by assigning each object to the cluster with the nearest mean. Unlike WordCount and Sort, K-means is an iterative workload, which means during the execution, the output of each iteration will be used as the input data of the next iteration.

The three workloads are commonly used and have various computation and communication characteristics, which are helpful to explore the patterns of different systems.

### B. Benchmark Setup

All the workloads based on the Hadoop and DataMPI systems conform to the BigDataBench specification. For micro-benchmarks, the input datasets are generated by *Big Data Generator Suite (BDGS)* [27]. For the K-means benchmark, we use BDGS to create the RAW datasets and convert them to vector points as the input data. The data model used in BDGS is *Amazon Movie Reviews*.

We use four dataset scales for our micro-benchmark evaluations, mini, small, medium, large. Working with mini dataset, the efficiency is heavily affected by the system native overhead. For small dataset, the intermediate data can be totally buffered in memory by DataMPI systems. And the data size should be larger than that of mini dataset. For medium dataset, less than 70 percent of the intermediate data will be buffered in memory by DataMPI, the rest of data must be spilt to disk. Hadoop will use system cache to buffer data similarly. For large dataset, more than 70 percent of the intermediate data must be spilt to disk.

We define the sizes of mini, small, medium and large dataset as 1 GB, 8 GB, 32 GB, 96 GB, respectively.

### C. Metrics Selection

For analyzing the characteristics of the workloads, we monitor the systems with resource metrics and architecture metrics. When doing experiments with one workload over

one system, we record the real-time resource metrics, which can help us understand the runtime status and the resource consumption of the systems. These metrics include:

- CPU Utilization: this is a measure of the amount of work handled by all the CPU in one computing node. The CPU utilization can be classified to different types. We focus on the total and iowait utilization. Iowait indicates the overhead of CPU spent on waiting for I/O operations to complete.
- Disk I/O Bandwidth: the I/O bandwidth of disk is defined as the data transmit rate (read/write) by the whole system when running the workload.
- Network I/O Bandwidth: the I/O bandwidth of network is defined as the amount of data transmitted per second over network.
- Memory Footprint: the memory footprint refers to the peak memory that is touched by the workload at once. It indicates the minimum amount of memory required by the workload to reverse the data or instructions while running.

The architecture metrics that are of interest include:

- CPI (Cycles per Instruction): this is a measure of the efficiency of the instructions executing on the hardware. A smaller value indicates instructions are running with fewer wait stalls or instruction dependencies.
- CPU Speed: this is the average processor execution speed on artificial instruction sequences when running the realistic workloads. It can be measured as the instructions per second (IPS) or giga instructions per second (GIPS).
- Cache Misses per Instruction: the processor of the testbed has three levels of cache hierarchy. Each core has its own L1 and L2 caches. All cores share the last level cache (LLC) in a socket. The first two levels of caches are smaller, but faster than the last level. LLC miss will conduct a reference to the DRAM which will have a latency of several hundred nanoseconds as penalty [28] in our testbed. LLC miss is an important indicator reflecting the locality when executing workloads over a certain system.
- Instruction and Data Translation Lookaside Buffer (TLB) Misses per Instruction: TLBs are used for caching the page table entries. The smaller miss rate indicates a higher access locality.
- Branch Misses per Instruction: branch prediction helps modern processors improve the instruction pipeline execution. Branch misprediction makes the hardware remove the incorrect executed micro operations, then decode and insert the correct path instructions to the execution pipeline, which degrades the performance.

The above metrics discussed provide information about the spatial and temporal pattern. Another issue for systems is about their balance. The performance of a system is limited by the slowest component. Balanced systems can provide data to CPU at a reasonable speed to prevent from experiencing delays. To determine whether or not the system is balanced, Amdahl has set up a number of laws:

- Amdahl Memory Law: One byte of memory per instruction per second. This law is known as Amdahl memory number. The number indicates whether or not there is sufficient memory to provide to the computation. This number can be calculated by using Equation 1.

$$Amdahl\ Memory\ Number = \frac{Memory\ Size\,(GB)}{CPU\ Speed\,(GIPS)} \quad (1)$$

- Amdahl I/O Law: One bit of I/O per second per instruction per second. This law is known as Amdahl I/O number. The number could be calculated by dividing the bandwidth by the instruction rate, as shown in Equation 2.

$$Amdahl\ I/O\ Number = \frac{Bandwidth\,(bit/sec)}{CPU\ Speed\,(IPS)} \quad (2)$$

The numbers of Amdahl's second law include the Amdahl memory number and Amdahl I/O number. To observe the memory behavior, we calculate the Amdahl maximum memory number and Amdahl average memory number. The Amdahl maximum memory number can be calculated by dividing the peak memory touched by the CPU speed during the workload execution and the Amdahl average memory number can be calculated by dividing the average memory used by the CPU speed during the workload execution.

### D. Experiment Methodology

To capture the micro-architecture level numbers, *perf* [29], a Linux profiling subsystem, is used to record the hardware performance counters, which are supported by modern processors. We collect kinds of events whose event number and unmask value can be found in the Intel Developer's Manual [30]. We also use *dstat* [31], a versatile resource statistics tool, to record the resource utilization. We use *perf* and *dstat* to monitor the overall system during the whole lifetime of each workload. To minimize the effect of the sampling tools on the monitor results, we bind the *perf* and *dstat* to execute on one core, and isolate the other cores to run workloads. Before collecting the counters and resource data of each test, we perform a ramp up period.

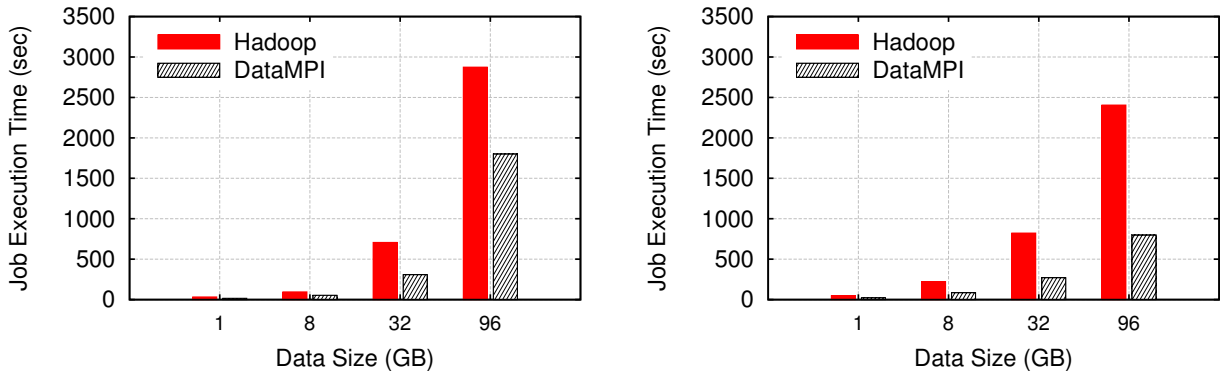
## IV. Results

### A. Hardware Configuration

We use a cluster composed of five nodes interconnected by a one Gigabit Ethernet switch as our testbed. Each node is equipped with two Intel Xeon E5620 CPU processors (2.4 GHz) with disabling the hyper-threads. In each CPU, there are four physical cores with private L1 and L2 caches and a shared LLC cache. Each node has 16 GB DDR3 RAM with 1333 MHz and one 2 TB SATA disk with 7200 RPM.

### B. Cluster Environments

The operation system used is CentOS 6.5 (Final) with kernel version 2.6.32-431.el6.x86-64. The software used includes BigDataBench 2.2, DataMPI 0.6, Hadoop 1.2.1 and JDK 1.7.0_25. The MPI implementation is MVAPICH2-2.0b. The cluster deploys 1 master node and 4 slave nodes. For a fair




(a) SORT (b) WORDCOUNT

Fig. 2. Performance comparison of different benchmarks with various datasets over Hadoop and DataMPI

comparison, we assign the same degree of parallelism among Hadoop and DataMPI and the same JVM heap size for each Hadoop/DataMPI task. Each slave node is configured with 4 Map/Reduce slots for Hadoop, 4 O/A tasks for DataMPI. We assign 3 GB JVM heap for each task. The HDFS block size is configured as 256 MB. In this section, we first give the resource utilization and architectural analysis of different workloads, then discuss the balance of the two systems based on the Amdahl's second law.

### C. Profile of Resource Utilization

We execute Sort, WordCount and K-means benchmarks over the two systems with varied datasets and record the resource utilization. When executing Sort and WordCount benchmarks, we observe DataMPI has up to 67% performance improvement compared to Hadoop, as shown in Figure 2.

For the limit of memory space, we show the resource utilization when processing medium dataset. All of the resource utilization numbers in the figures are averaged across four slave nodes. The execution times of WordCount over Hadoop and DataMPI are 823 sec and 274 sec, respectively. For the Sort benchmark, the execution times are 707 sec and 308 sec, respectively.

When running WordCount (WC), the total CPU utilizations of Hadoop and DataMPI are stable. Figure 3(a) and Figure 3(d) show DataMPI has 30% higher CPU utilization and 3 times disk bandwidths compared to Hadoop. This is because DataMPI has advanced overlapping of computation/communication/merge operations, which accelerates the processing progress of WordCount. More computation tasks will be finished earlier than Hadoop. Thus, we see a higher CPU utilization and I/O bandwidth of DataMPI for the WordCount benchmark. Figure 3(j) shows Hadoop and DataMPI have little data transmitted through network, because Hadoop and DataMPI load the input data according to the locality and combine the intermediate data in the local tasks to several kilobytes of data before the network transmission. Figure 3(g) shows all of the two systems have used similar amount of memory resource during execution.

When running Sort (ST), as Figure 3(b) shows, the average total CPU utilization and CPU iowait percentage of Hadoop are 41% and 26%, while those of DataMPI are 57% and 23%. The CPU iowait percentage of Hadoop is larger than that of DataMPI, while CPU utilization percentage of Hadoop is smaller than that of DataMPI. This also indicates DataMPI is more efficient than Hadoop at computation because of the pipeline design. Figure 3(e) shows the average disk bandwidths of Hadoop and DataMPI are 51.4 MB/sec and 59.5 MB/sec. We should notice that Hadoop will reserve intermediate data on disk at the end of the Mapper tasks for preparing the shuffle execution in the Reduce tasks. Different from Hadoop, DataMPI will buffer the intermediate data in memory and spill part of the buffered data on disk when the memory is not sufficient. DataMPI O tasks will prepare the intermediate data in memory for A tasks processing. The total amount of I/O data on disks generated by Hadoop (36.1 GB) is nearly as twice much as that of DataMPI (19.2 GB). Figure 3(k) shows the total amount of network data transmitted across the 4 slave nodes of the two systems are nearly equal. The average network bandwidth of DataMPI is 40.5 MB/sec, which is almost twice larger than that of Hadoop (17.2 MB/sec). That means DataMPI can provide better communication efficiency.

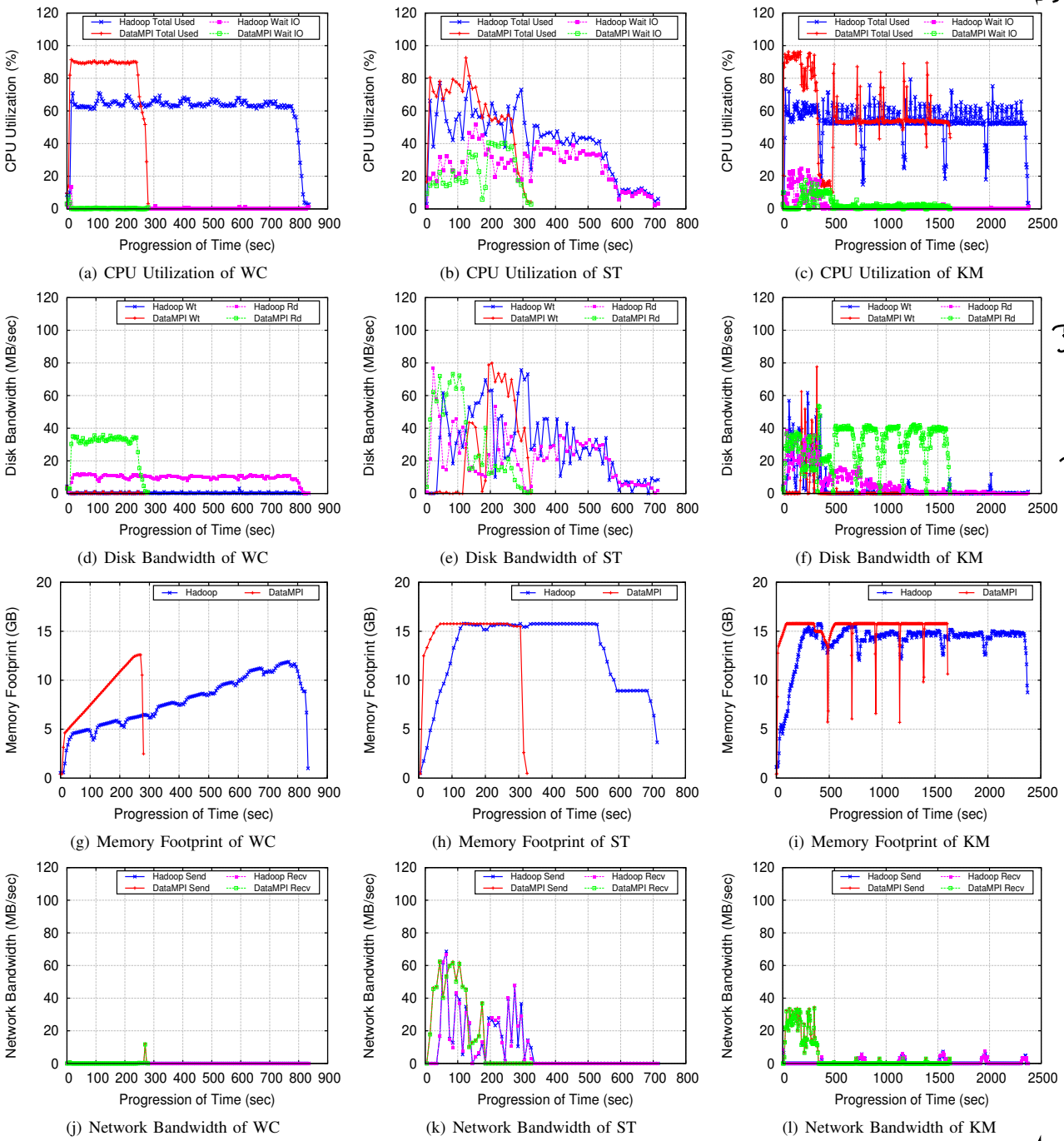
The above evaluations show DataMPI costs less time than Hadoop when executing the same benchmarks, while the CPU utilization is relatively higher than that of Hadoop. It is contributed to the pipeline design in DataMPI which overlaps the computation/communication/merge operations and accelerates the execution progress.

We evaluate the two systems using K-means (KM), which is a typical iterative workload. We record the total execution time of the calculation composed of data stage-in and five iterations. The performance of DataMPI has about 30% improvement than that of Hadoop.

### D. Hardware Performance Counters

Figure 4 shows the architecture metrics when executing Sort, WordCount benchmarks with different datasets.

*1) Instruction Execution:* It is observed that with the same workload over the same system, the CPI varies slightly while




(a) CPU Utilization of WC (b) CPU Utilization of ST (c) CPU Utilization of KM

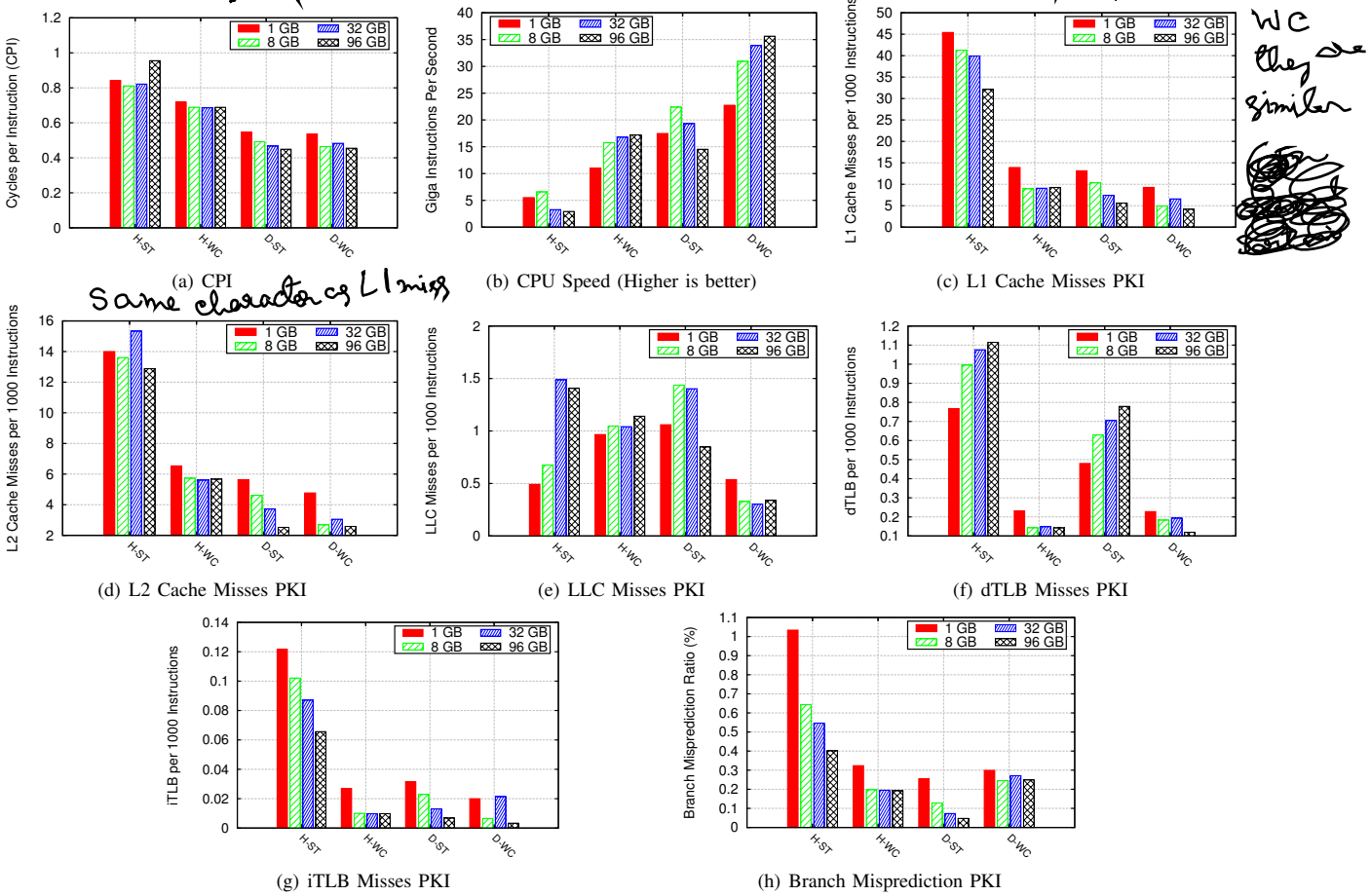
(d) Disk Bandwidth of WC (e) Disk Bandwidth of ST (f) Disk Bandwidth of KM

(g) Memory Footprint of WC (h) Memory Footprint of ST (i) Memory Footprint of KM

(j) Network Bandwidth of WC (k) Network Bandwidth of ST (l) Network Bandwidth of KM

Fig. 3. Resource utilization of different benchmarks with medium dataset over Hadoop and DataMPI

increasing the volume of the input data. For a better comparison, we calculate the average CPI of one workload over one system when processing different datasets. Figure 4(a) shows when running DataMPI, the average CPIs of Sort and WordCount are 0.49 and 0.48, which are smaller than those of Hadoop (0.86, 0.7). The CPI could be limited by the pipeline stall and instruction dependencies. The smaller CPI a system has, the better performance it achieves. Besides, Figure 4(b) shows when running DataMPI, the average CPU speeds of Sort and WordCount are 18 GIPS and 30 GIPS, which are much larger than those of Hadoop (4 GIPS, 14 GIPS). Both of the metrics indicate DataMPI can leverage the CPU to execute instructions more efficiently than Hadoop.




(a) CPI (b) CPU Speed (Higher is better) (c) L1 Cache Misses PKI

(d) L2 Cache Misses PKI (e) LLC Misses PKI (f) dTLB Misses PKI

(g) iTLB Misses PKI (h) Branch Misprediction PKI

Fig. 4. Architectural profiling of different benchmarks with various datasets over Hadoop and DataMPI. 'H-' stands for executing Hadoop and 'D-' stands for executing DataMPI. 'ST' is short for Sort and 'WC' is short for WordCount. Except the CPU speed, in the other metrics, the smaller number is better.

*) In general Sort shows larger misses than WC because of its Randomness. DataMPI shows smaller misses than Hadoop

*2) Memory Hierarchy:* According to the Xeon CPU developer manual, the CPUs on the slaves of our testbed have three tiers of cache. For better understanding the data presentation and execution locality, we record the numbers of L1 cache misses, L2 cache misses, LLC (L3 cache) misses, instruction TLB (iTLB) misses and data TLB (dTLB) misses per thousand instructions (in short for MPKI).

Figure 4(c) shows the average L1 cache MPKIs of Sort over Hadoop, DataMPI are 39, 9, which are larger than those of WordCount (11, 7). Figure 4(d) shows the average L2 cache MPKIs of Sort over Hadoop, DataMPI are 14, 4, which are larger than those of WordCount (6, 3). Figure 4(e) shows the average LLC MPKIs of Sort over Hadoop, DataMPI are 1.0, 1.3, which are slightly larger than those of WordCount (1.0, 0.3). This is because sorting operation of Sort workload requires many random accesses around a large memory space and the access pattern does not utilize the CPU cache efficiently. Besides, we can observe DataMPI achieves smaller numbers of L1 cache/L2 cache/LLC MPKIs than those of Hadoop when running the same workloads. It implies DataMPI can utilize the CPU cache more efficiently than Hadoop. This is because DataMPI has designed a partition list based buffer management scheme which is applying to the relaxed all-to-all communication pattern.

Figure 4(f) and Figure 4(g) show the TLB behaviors. The average dTLB MPKIs of Sort, WordCount over Hadoop are 1.0, 0.2, while those of DataMPI are 0.6, 0.2. The Sort workload has more complex data access pattern which leads to larger dTLB MPKI numbers. From the Sort workload, the iTLB MPKIs of Hadoop (0.09) on the average are larger than those of DataMPI (0.02). The low iTLB MPKIs of DataMPI could be caused by the less dependency of the third-party libraries and light-weight software stack.

*3) Branch Misprediction:* Figure 4(h) shows the branch misprediction ratio of various workloads over the two systems. The average branch misprediction ratios of Sort, WordCount over Hadoop are 0.66%, 0.25%, which are larger than those of DataMPI (0.13%, 0.25%). This indicates that DataMPI has a simpler execution pattern.

### *E. Numbers of Amdahl's Second Law*

According to our testbed, each node has 8 processors and each processor can issue 4 instructions per cycle, the CPU

TABLE I
THE THEORETICAL VALUES OF EACH NODE IN OUR TESTBED

| Metrics | Ideal Value | Hardware Value |
|---|---|---|
| CPU Speed (GIPS) | 76.8 | 76.8 |
| Disk Bandwidth (Gbps) | 76.8 | 1.2 |
| Network Bandwidth (Gbps) | 76.8 | 1 |
| Memory Size (GB) | 76.8 | 16 |

TABLE II
AMDAHL MEMORY NUMBER AND AMDAHL I/O NUMBER OF DIFFERENT WORKLOADS OVER HADOOP AND DATAMPI

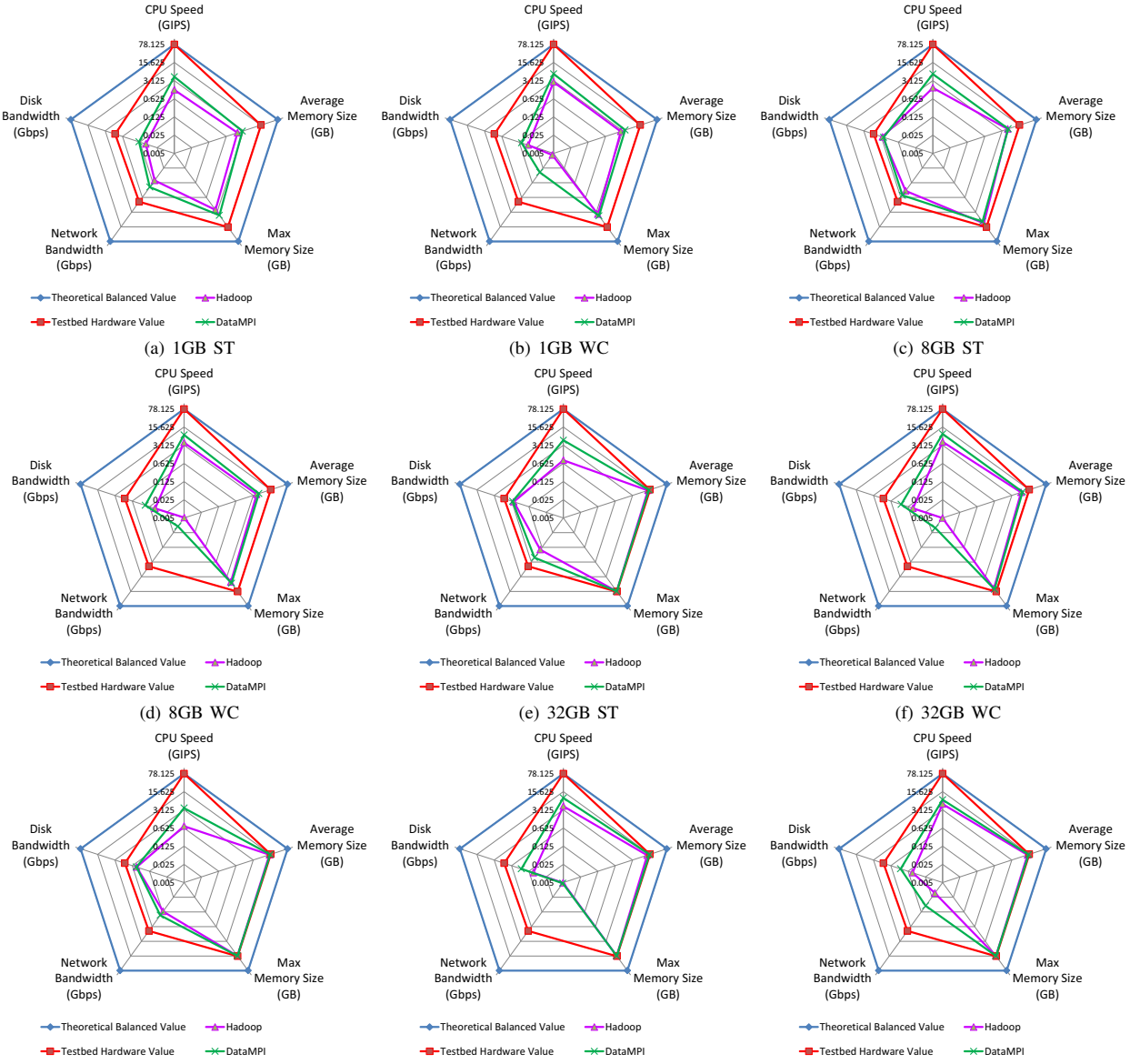
| Metrics | Data Size (GB) | H-ST | H-WC | H-KM | D-ST | D-WC | D-KM |
|---|---|---|---|---|---|---|---|
| Execution Time (sec) | 1 | 33 | 49 | | 17 | 22 | |
| | 8 | 83 | 221 | | 52 | 86 | |
| | 32 | 633 | 833 | 2377 | 266 | 274 | 1673 |
| | 96 | 2569 | 2441 | | 1705 | 765 | |
| CPU Speed (GIPS) | 1 | 1.38 | 2.7525 | | 4.37 | 5.6875 | |
| | 8 | 1.645 | 3.935 | | 5.6 | 7.7325 | |
| | 32 | 0.82 | 4.1975 | 5.2225 | 4.8275 | 8.475 | 7.58 |
| | 96 | 0.7325 | 4.305 | | 3.635 | 8.9075 | |
| AVG Memory (GB) | 1 | 1.7975 | 2.61 | | 2.8175 | 3.8875 | |
| | 8 | 5.32 | 4.335 | | 5.575 | 5.275 | |
| | 32 | 13.155 | 7.445 | 13.6 | 14.6325 | 8.49 | 14.965 |
| | 96 | 14.18 | 12.2625 | | 15.2125 | 15.12 | |
| MAX Memory (GB) | 1 | 2.45 | 3.9375 | | 4.4475 | 4.67 | |
| | 8 | 10.7225 | 5.65 | | 8.79 | 6.3625 | |
| | 32 | 15.4175 | 11.3325 | 15.4 | 15.4175 | 12.395 | 15.4175 |
| | 96 | 15.445 | 15.415 | | 15.445 | 15.4125 | |
| I/O Bandwidth (MB/sec) | 1 | 21.1875 | 7.4375 | | 42.1225 | 17.395 | |
| | 8 | 106.11 | 9.9325 | | 123.3325 | 24.8725 | |
| | 32 | 83.74 | 9.9125 | 12.36 | 118.8875 | 31.545 | 39.3175 |
| | 96 | 66.7175 | 10.205 | | 75.5125 | 31.415 | |
| Amdahl AVG Memory Number (byte per ins per sec) | 1 | 1.397 | 1.019 | | 0.692 | 0.734 | |
| | 8 | 3.472 | 1.183 | | 1.069 | 0.732 | |
| | 32 | 17.243 | 1.904 | 2.796 | 3.254 | 1.076 | 2.120 |
| | 96 | 20.781 | 3.059 | | 4.493 | 1.822 | |
| Amdahl MAX Memory Number (byte per ins per sec) | 1 | 1.904 | 1.536 | | 1.093 | 0.882 | |
| | 8 | 6.999 | 1.541 | | 1.685 | 0.883 | |
| | 32 | 20.209 | 2.898 | 3.166 | 3.429 | 1.570 | 2.184 |
| | 96 | 22.631 | 3.845 | | 4.562 | 1.858 | |
| Amdahl I/O Number (bit per ins) | 1 | 0.129 | 0.023 | | 0.081 | 0.026 | |
| | 8 | 0.541 | 0.021 | | 0.185 | 0.027 | |
| | 32 | 0.857 | 0.020 | 0.020 | 0.207 | 0.031 | 0.044 |
| | 96 | 0.764 | 0.020 | | 0.174 | 0.030 | |

clock rate of each processor is 2.4 GHz. Theoretically, the peak CPU speed in our testbed is 76.8 GIPS. According to the Amdahl's second law, a well-balanced node should provide 76.8 GB memory and 76.8 Gbps I/O bandwidth. However, the real I/O bandwidth and memory size of each node in our testbed are much smaller than the ideal numbers, as shown in Table I. In Figure 5, we plot the ideal numbers as the blue line, and the actual hardware numbers as the red line. According to Equation 1 and Equation 2, the Amdahl memory number and Amdahl I/O number of each node in our testbed are 0.2 and 0.028, respectively.

We analyse the system balance of Hadoop and DataMPI based on Amdahl's second law. We record the execution time, CPU speed, average memory footprint, max memory footprint, average I/O bandwidth and calculate the Amdahl max memory number, Amdahl average memory number and Amdahl I/O number. The results are shown in Table II. When running Sort with different datasets, the Amdahl average memory number of Hadoop ranges from 1.4 to 20.8, while that of DataMPI ranges from 0.7 to 4.5. When running WordCount, the Amdahl average memory number of Hadoop ranges from 1.0 to 3.1, while that of DataMPI ranges from 0.7 to 1.8. DataMPI achieves smaller Amdahl memory numbers than Hadoop when running the same workload. This means, compared to Hadoop, DataMPI averagely demands less memory to execute one instruction and indicates DataMPI is more balanced in terms of CPU and memory.

The Amdahl I/O number is calculated based on the total amount of the system I/O, which includes disk I/O and network I/O. It reflects how many bits of system I/O are handled per instruction per second. We also observe when running WordCount, the Amdahl I/O number of Hadoop (0.02) and DataMPI (0.03) are approximately the same. When running Sort, the Amdahl I/O number of Hadoop ranges from 0.129 to 0.857, while that of DataMPI ranges from 0.081 to 0.207. The Amdahl I/O numbers of DataMPI are smaller than those of Hadoop, which indicates the balance between I/O and CPU of DataMPI is better than that of Hadoop.

Figure 5 shows the CPU speed, disk I/O bandwidth, network I/O bandwidth and memory size used by Hadoop and DataMPI




(a) 1GB ST (b) 1GB WC (c) 8GB ST

(d) 8GB WC (e) 32GB ST (f) 32GB WC

(g) 96GB ST (h) 96GB WC (i) 32GB KM

Fig. 5. Comparison of CPU speed, disk bandwidth, network bandwidth and memory usage between Hadoop and DataMPI

when running Sort, WordCount and K-means with different datasets. Results show the CPU speed and I/O bandwidth of DataMPI are larger than those of Hadoop. Meanwhile, the execution time of each workload over DataMPI is less than that over Hadoop. This indicates DataMPI can leverage the resources more efficiently than Hadoop. This explains why DataMPI is more balanced.

## V. Conclusion

This paper presents a performance characterization of two big data application frameworks, Hadoop and DataMPI, utilizing Amdahl's second law. We experiment on a five-node cluster using three benchmarks, Sort, WordCount and K-means, with 1 GB, 8 GB, 32 GB and 96 GB of data. The evaluation results show that:

- Measured by job execution time, DataMPI has 37-67% performance improvement over Hadoop.
- For every instruction executed per second, DataMPI requires a main memory capacity of 0.8-4.5 bytes, while Hadoop requires a main memory capacity of 1.5-22.6 bytes. Averagely, DataMPI has smaller main memory footprint than Hadoop.
- For every instruction executed per second, DataMPI

requires an I/O bandwidth of 0.02-0.21 bit per second, while Hadoop requires an I/O bandwidth of 0.02-0.85 bit per second. Overall, DataMPI demands a smaller I/O bandwidth than Hadoop.

Our preliminary research results show the smaller the number of Amadhl's second law achieves, the more efficiently the system performs. That is, to achieve the same performance, the smaller the resources are used, the better. Overall, a DataMPI-based big data computing system is more balanced than a Hadoop-based system.

## VI. Acknowledgments

We are very grateful to Lu Chao for his help of this work. This work is supported in part by the Strategic Priority Program of Chinese Academy of Sciences (Grant No. XDA06010401), the Guangdong Talents Program (Grant No. 201001D0104726115), the Hi-Tech Research and Development (863) Program of China (Grant No. 2013AA01A209, 2013AA01A213).

## References

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the ACM*, vol. 51, pp. 107–113, 2008.

[2] "The Apache Hadoop Project," http://hadoop.apache.org.

[3] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks," *ACM SIGOPS Operating Systems Review*, vol. 41, pp. 59–72, 2007.

[4] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari, "S4: Distributed Stream Computing Platform," in *Proceedings of the 10th IEEE International Conference on Data Mining Workshops (ICDMW '10)*, 2010.

[5] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, and I. Stoica, "Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing," in *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI '12)*, 2012.

[6] G. Amdahl, "Computer Architecture and Amdahl's Law," *Computer*, vol. 46, no. 12, pp. 38–46, 2013.

[7] A. Szalay, K. Church, C. Meneveau, A. Terzis, and S. Zeger, "MRI: The Development of Data-Scope A Multi-Petabyte Generic Data Analysis Environment for Science," Patent.

[8] X. Lu, F. Liang, B. Wang, L. Zha, and Z. Xu, "DataMPI: Extending MPI to Hadoop-like Big Data Computing," in *Proceedings of the 28th International Parallel and Distributed Processing Symposium (IPDPS '14)*, 2014.

[9] D. Ersoz, M. Yousif, and C. Das, "Characterizing Network Traffic in a Cluster-based, Multi-tier Data Center," in *Proceeding of the 27th International Conference on Distributed Computing Systems (ICDCS '07)*, 2007.

[10] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center TCP (DCTCP)," in *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM '10)*, 2010.

[11] F. Wang, Q. Xin, B. Hong, S. A. Brandt, E. L. Miller, D. D. E. Long, and T. T. McLarty, "File System Workload Analysis for Large Scale Scientific Computing Applications," in *Proceedings of the 12th NASA Goddard/21st IEEE Conference on Mass Storage Systems and Technologies (MSST '04)*, 2004.

[12] L. N. Bairavasundaram, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, G. R. Goodson, and B. Schroeder, "An Analysis of Data Corruption in the Storage Stack," *Transactions on Storage*, vol. 8, pp. 1–28, 2008.

[13] A. Iamnitchi, S. Doraimani, and G. Garzoglio, "Workload Characterization in a High-energy Data Grid and Impact on Resource Management," *Cluster Computing*, vol. 12, pp. 153–173, 2009.

[14] X. Lu, B. Wang, L. Zha, and Z. Xu, "Can MPI Benefit Hadoop and MapReduce Applications?" in *Proceedings of the 40th International Conference on Parallel Processing Workshops (ICPPW '11)*, 2011.

[15] M. Stonebraker, D. Abadi, D. J. DeWitt, S. Madden, E. Paulson, A. Pavlo, and A. Rasin, "MapReduce and Parallel DBMSs: Friends or Foes?" *Communications of the ACM*, vol. 53, pp. 64–71, 2010.

[16] D. Jiang, B. C. Ooi, L. Shi, and S. Wu, "The Performance of Mapreduce: An in-depth Study," *in Proceedings of the VLDB Endowment*, vol. 3, pp. 472–483, 2010.

[17] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das, "Towards Characterizing Cloud Backend Workloads: Insights from Google Compute Clusters," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, pp. 34–41, 2010.

[18] Y. Chen, S. Alspaugh, and R. H. Katz, "Design Insights for MapReduce from Diverse Production Workloads," DTIC Document, Tech. Rep., 2012.

[19] S. Kavulya, J. Tan, R. Gandhi, and P. Narasimhan, "An Analysis of Traces from a Production MapReduce Cluster," in *Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid '10)*, 2010.

[20] Z. Ren, X. Xu, J. Wan, W. Shi, and M. Zhou, "Workload Characterization on a Production Hadoop Cluster: A Case Study on Taobao," in *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC '12)*, 2012.

[21] J. Chang, K. T. Lim, J. Byrne, L. Ramirez, and P. Ranganathan, "Workload Diversity and Dynamics in Big Data Analytics: Implications to System Designers," in *Proceedings of the 2nd Workshop on Architectures and Systems for Big Data (ASBD '12)*, 2012.

[22] J. Gray and P. Shenoy, "Rules of Thumb in Data Engineering," in *Proceedings of the 16th International Conference on Data Engineering (ICDE '00)*, 2000.

[23] G. Bell, J. Gray, and A. Szalay, "Petascale Computational Systems," *Computer*, vol. 39, pp. 110–112, 2006.

[24] A. S. Szalay, G. Bell, J. Vandenberg, A. Wonders, R. Burns, D. Fay, J. Heasley, T. Hey, M. Nieto-SantiSteban, A. Thakar, *et al.*, "Graywulf: Scalable Clustered Architecture for Data Intensive Computing," in *Proceedings of the 42nd Hawaii International Conference on System Sciences (HICSS '09)*, 2009.

[25] A. S. Szalay, G. C. Bell, H. H. Huang, A. Terzis, and A. White, "Low-power Amdahl-balanced Blades for Data Intensive Computing," *ACM SIGOPS Operating Systems Review*, vol. 44, pp. 71–75, 2010.

[26] D. Cohen, F. Petrini, M. D. Day, M. Ben-Yehuda, S. W. Hunter, and U. Cummings, "Applying Amdahl's Other Law to the Data Center," *IBM Journal of Research and Development*, vol. 53, pp. 683–694, 2009.

[27] L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia, Y. Shi, S. Zhang, C. Zheng, G. Lu, K. Zhan, X. Li, and B. Qiu, "BigDataBench: a Big Data Benchmark Suite from Internet Services," in *Proceedings of the 20th IEEE International Symposium On High Performance Computer Architecture (HPCA '14)*, 2014.

[28] D. Levinthal, "Performance Analysis Guide for Intel Core i7 Processor and Intel Xeon 5500 Processors (2008-2009)," Tech. Rep.

[29] A. C. de Melo, "Performance Counters on Linux," in *Linux Plumbers Conference*, 2009.

[30] *Intel® 64 and IA-32 Architectures Software Developers Manual*, Intel, 2014.

[31] "Dstat: Versatile Resource Statistics Tool," http://dag.wiee.rs/home-made/dstat/.