Understanding Behavior Trends of Big Data Frameworks in Ongoing Software-Defined Cyber-Infrastructure

Shouwei Chen, Ivan Rodero
Rutgers Discovery Informatics Institute (RDI^2), Rutgers University
Piscataway, New Jersey
{shouwei.chen,irodero}@rutgers.edu

ABSTRACT

As data analytics applications become increasingly important in a wide range of domains, the ability to develop large-scale and sustainable platforms and software infrastructure to support these applications has significant potential to drive research and innovation in both science and business domains. This paper characterizes performance and power-related behavior trends and tradeoffs of the two predominant frameworks for Big Data analytics (i.e., Apache Hadoop and Spark) for a range of representative applications. It also evaluates system design knobs, such as storage and network technologies and power capping techniques. Experimental results from empirical executions provide meaningful data points for exploring the potential of software-defined infrastructure for Big Data processing systems through simulation. The results provide better understanding of the design space to build multi-criteria application-centric models as well as show significant advantages of software-defined infrastructure in terms of execution time, energy and cost. It motivates further research focused on in-memory processing formulations regarding systems with deeper memory hierarchies and software-defined infrastructure.

1 INTRODUCTION

The proliferation of digital data provides new opportunities in all areas of science, engineering, and industry. About 2.5 quintillion bytes of data [2] is generated every day through the Internet. However, the increasing volume and rate of data [26], along with the associated costs in terms of latency and energy, quickly overpower and limit data analytics applications' ability to leverage this data in an effective and timely manner. The co-design process enables scientists to reason about the rich design spaces available in software and hardware, which is fundamental for constructing the next generations of cyber-infrastructure. While system architecture trends include larger core counts, deeper memory hierarchies (e.g., larger amounts of non-volatile memory), and constrained power budgets, application formulations for Big Data are trending toward in-memory processing solutions. Nevertheless, as current solutions for Big Data analysis pipelines require complex solutions involving different specialized platforms and configurations depending on application requirements, it is not clear how to effectively realize and optimize them in these ongoing architectures. Further, ongoing processor architectures, non-volatile technologies such as Intel Optane NVMe, and the advances in integrated silicon photonics promise systems capable for delivering off-node non-volatile memory latency and bandwidth comparable to PCIe-based in-node access [23], which is essential for realizing actual software-defined infrastructures. As a result, exploring key co-design issues in the scope of Big Data analytics has become a critical concern.

The goal of our current research is understanding system behavior and the tradeoffs associated with the use of different architectural designs and processing frameworks for different classes of relevant applications under different constrains. This provides the foundations to develop models that can fundamentally enable Big Data analytics on ongoing cyber-infrastructure based on software-defined infrastructure (SDI). As opposed to other research efforts that investigate balanced systems for a range of analytics applications [9], this research is aimed at understanding what the optimal design choices are, given a multi-criteria approach and under different constraints (e.g., power budget). This paper is focused on understanding these behaviors and tradeoffs for two of the main distributed processing systems for Big Data analytics: Apache Hadoop and Spark, both of which are currently the most widely used open source parallel programing frameworks for Big Data analytics.

Current data analysis workflows may require different types of analytics, where some are more appropriate for batch-oriented processing (e.g., Hadoop), micro-batch processing (e.g., Spark), or near real-time processing (e.g., Storm, Flink, Heron). However, there is an increasing interest from both scientific and industry communities to move to in-memory approaches for a broader range of analytics. Power requirements to run workloads in-memory may have different resource utilization patterns than more I/O-bounded approaches. Existing work [9] has shown that the performance of storage devices used in Hadoop deployments impacts the execution time of data and compute-intensive applications, and that the execution of specific graph-based workloads is more energy efficient with Spark (i.e., Spark GraphX) than with Hadoop (i.e., Hadoop Giraph) [19]. However, there is still a gap in the study between Hadoop and Spark behaviors and tradeoffs related to performance, energy efficiency, power requirements, and design knobs like storage devices and power capping strategies. This paper bridges that gap by providing a comprehensive study of representative workloads for Hadoop and Spark using different storage technologies and design choices, with a concentration on power-related issues, and explores the potential of software-defined infrastructure for

BDCAT'17, , December 5–8, 2017, Austin, Texas, USA
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5549-0/17/12...$15.00
https://doi.org/10.1145/3148055.3148079

Big Data processing frameworks, with a concentration on the non-volatile deep memory hierarchy.

The results from our empirical experiments confirms expected behaviors (e.g., Spark not only provides better energy efficiency than Hadoop, but it is also more efficient than Hadoop, even under power capping, and NVRAM and SSD significantly decrease CPU wait time); even so, this provides meaningful data points that can be used for building multi-criteria application-centric models for Big Data co-design and provides insights for conducting simulations of these processing frameworks using software-defined infrastructures. The contributions of this paper are summarized as follows: (1) we provide a comprehensive characterization of performance and energy/power behaviors and tradeoffs of Hadoop and Spark using different technologies, which is not available in existing literature; (2) we study using power capping techniques in Spark deployments for operating under power constraints while meeting performance goals; (3) we identify a number of factors that play an important role in Hadoop and Spark's performance, power, and energy efficiency; and (4) we explore for first time the potential of software-defined infrastructure for Big Data processing frameworks in terms of execution time, energy and cost.

The rest of this paper is organized as follows: Section 2 provides the literature review, which is augmented in Section 3, which provides further background and outlines the targeted tradeoffs. Section 4 describes the evaluation methodology used for obtaining the experimental results presented in Section 5. Finally, Section 6 concludes this work and describes future directions to which this work can be extended.

2 RELATED WORK

A large body of literature in this area is focused on MapReduce's workloads and runtime instead of hardware/software co-design issues. A comprehensive stud of a MapReduce workload analyzed a ten-month workload trace from the Yahoo! M45 supercomputing cluster [18]. However, most of existing studies focus on benchmarks instead of real production workloads [11, 20, 32]. Other work has focused on specific issues, such as job and task run times [11, 15, 18, 20, 32], Map vs. Reduce tasks [18, 32], CPU and memory demand [12], I/O and data locality [7, 32], and cluster utilization, failures, and energy consumption [7, 18]. Models for MapReduce workloads have also been developed [1, 5, 11, 18, 32]; however, their primary focus is on job completion times. Furthermore, different MapReduce simulators have been developed [5, 10, 14, 28, 32] that mainly focus on simulating the execution of synthetic workloads. However, combining macro- and micro-models into simulations for exploring a co-design process via characterization is still challenging but one of the long-term goals of this research.

Other recent research efforts, such as the Aloja project [3], aim to explore upcoming hardware architectures for Big Data processing and reduce the Total Cost of Ownership (TCO) of running Hadoop clusters. Aloja's approach is to create a comprehensive open public Hadoop benchmarking repository based on empirical executions. It allows for comparisons between not only software configuration parameters, but also current hardware (e.g., SSDs, Infiniband networks). Our work also addresses this problem with empirical experimentation to extract models, but it is focused on memory hierarchy and power/energy-related issues (e.g., power capping knobs) with the ultimate goal of targeting future system architectures and application formulations via co-design.

Existing literature has also addressed the optimization of energy efficiency at the cluster level for Hadoop MapReduce [17] by dividing the cluster into two zones: a "hot" zone with frequently used data on higher performance processors and a "cold" zone for low-frequency access data with a large amount of disks. Goiri et al. [13] introduced GreenHadoop, which is powered via solar array and uses the electrical grid as backup. Lang et al. [21] came up with the All-In-Strategy (AIS), which toggles nodes on or off based on the amount of Hadoop jobs in the queue. Amur et al. [4] presents the power-proportional distributed file system (Rabbit) that divides the nodes of a cluster into primary nodes (for primary replicas) and secondary nodes (for other replicas), which also provides a higher level of fault tolerance. Chen et al. [6] implemented Berkeley Energy Efficient MapReduce (BEEMR), an energy-efficient MapReduce workload manager motivated by the empirical analysis of real-life MapReduce with Interactive Analysis (MIA) traces at Facebook. BEEMR classifies jobs into either an interactive zone, a full-power-ready state and batch zone, and a low-power state in order to optimize energy efficiency.

Dynamic Voltage and Frequency Scaling (DVFS) has been used to improve energy efficiency. Tiwai et al. [31] addressed CPU frequency tuning based on application type to decrease energy consumption. Wirtz et al. [33] compared three different CPU frequency policies for Hadoop: 1) a fixed frequency for all cores during execution, 2) a maximum CPU frequency for map and reduce functions and a minimum CPU frequency otherwise, and 3) an adjustment to the CPU frequency while satisfying performance requirements. Li et al. [22] proposed temperature-aware power allocation (TAPA) to reduce energy consumption and Shadi et al. [16] recently explored DVFS usage in Hadoop clusters.

Current research also addresses hardware and data optimization to improve energy efficiency. Chen et al. [8] studied the energy consumption for Hadoop applications in three dimensions: the number of nodes, the number of HDFS replicas and different HDFS block sizes, and data compression methods that may improve energy efficiency [7]. Yigitbas et al. [34] proposed an Intel Atom processor-based Hadoop cluster for better energy efficiency than an Intel Sandy Bridge processor-based Hadoop cluster with I/O-bound MapReduce workloads. Luo et al. [25] evaluated CPU frequency, memory mode, and different storage parameters for compute intensive, storage intensive, and I/O intensive applications.

The literature summarized above can be complemented with research on MapReduce schedulers [12, 27, 29, 30] and existing work on MapReduce frameworks for many-core systems (e.g., the Intel Xeon Phi platform) focusing on SIMD support and performance issues [24]. A comprehensive characterization of different big data processing frameworks along multiple dimensions and technologies is not available in existing literature and it is needed for our study that targets software-defined infrastructure. Instead of focusing on a specific technology solution or optimization, this paper provides insight and methodology for exploring the design space. At the best of our knowledge, there is no existing work studying the potential of software-defined infrastructure for big data processing systems.

3 TARGETED TRADEOFFS IN BIG DATA SYSTEMS

The overarching goal of this work includes developing a methodology to construct models to understand and explore the design space, which includes building a framework to evaluate different classes of Big Data analytics in systems with ongoing architecture, such as deeper memory hierarchies and software-defined infrastructures. In order to build such a framework, this paper studies key aspects of existing data analytics frameworks (e.g., Hadoop and Spark - Apache is omitted from this point).

In order to design models to realize this vision, it is required to characterize a comprehensive set of data-centric benchmark applications in terms of performance, energy, and power behaviors on an instrumented platform to best understand their resource requirements and identify possible performance/power tradeoffs. Such a comprehensive characterization is not currently available in existing literature.

The ultimate goals of our research are, on the one hand, using this characterization to build models and develop heuristics/meta-heuristics that will consider different criteria and constraints including but not limited to: performance (e.g., response time or quality of service), capital costs (e.g., the infrastructure available, such as number of servers, cores, or memory), operational costs (e.g., energy consumption), and the power budget. Such an approach is expected to be multi-dimensional and multi-criteria. Parameter examples follow: (1) hardware choices (e.g., core count, memory size, I/O and network bandwidth), (2) data processing system (e.g., batch vs. micro-batch), (3) virtualization (e.g., bare-metal vs. containers vs. VMs), (4) processing framework (e.g., Hadoop vs. Spark), (5) programming language (e.g., Scala vs. Python), etc. On the other hand, we aim at developing resource provisioning and scheduling approaches for big data workloads in systems based on ongoing software-defined infrastructure. In this scenario, the problem becomes more challenging as it spans across different dimensions (multi-dimensional knapsack-like problem, i.e., NP-hard).

This paper is focused on understanding the behaviors and tradeoffs of the two primary open source processing frameworks for Big Data analytics: Hadoop and Spark. Data movement is one of the main bottlenecks for these data processing frameworks, as their applications typically require very heavy read and write operations during processing. While Hadoop supports the MapReduce programming model using a storage-centric approach, Spark is based on in-memory processing (through Resilient Distributed Datasets - RDDs) and requires much less access to storage. These two processing frameworks are shown in Figure 1, which provides an overall classification of some of the most widely used open source distributed data processing frameworks.

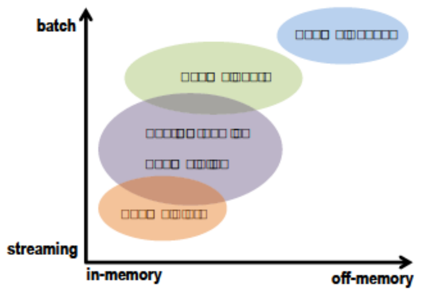


Figure 1: Classification of the most extended (Apache-based) distributed processing back-ends for big data analytics

The characterization in this paper considers two fundamental parameters: (i) energy/power tradeoffs, and (ii) storage technology (i.e., memory hierarchy). While in-memory processing systems are expected to be faster than storage-based systems for running a Big Data processing workload, the power required to run this workload in-memory is expected to be higher if a larger pool of resources are needed to handle in-memory data. There are also clear issues related to power requirements for a more I/O-bounded approach due to lower CPU and memory utilization over time.

The tradeoff between required power and energy consumption in this context requires investigation. For example, in the scenario depicted in Figure 2, the energy cost of running a workload using Hadoop could be higher than using Spark; however, the fastest option (i.e., Spark) might not be viable due to the power budget constraints or availability of servers needed to handle RDDs in memory. The figure (top) also shows that power capping can be used as a mechanism to manage these possible tradeoffs. Power capping has also been considered to better understand the possible tradeoffs between Hadoop and Spark. Since the memory hierarchy/storage technology is expected to significantly impact performance and other metrics, Section 5 investigates different storage hierarchies (ranging from hard disk to PCIe-based non-volatile memory devices) and power capping using RAPL. However, the use of power capping strategies in software-defined infrastructures remains part of our future work.

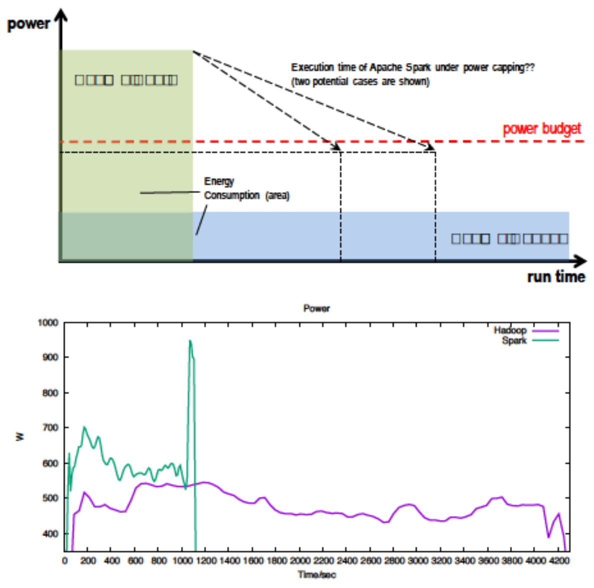


Figure 2: Possible (top) and observed (bottom) run time and power consumption behavior of a data analytics workload run with Hadoop and Spark. The real execution of the bottom is obtained using Grep (see Section 4 for more details)

4 EVALUATION METHODOLOGY

While the evaluation focused on software-defined infrastructures in systems with non-volatile memory technology is based on simulations, the empirical executions were conducted on the NSF-funded research instrument "Computational and dAta Platform for Energy efficiency Research" (CAPER). CAPER is an eight-node cluster based on SuperMicro SYS-4027GR-TRT system with a flexible configuration. The servers have two Intel Xeon Ivy Bridge E5-2650v2 (16 cores/node) and the configuration used in this work includes 128GB DRAM, 1TB Flash-based NVRAM (Fusion-io IoDrive-2), 2TB SSD and 4TB hard disks (as a RAID with multiple spindles, as recommended by best practices) and both 1GbE and 10GbE network connectivity. This platform mirrors key architectural characteristics of high-end system, which will allow us to extrapolate our models to larger systems and make projections. Further information is available at [1]. In addition to server level power measurement mechanisms, it supports RAPL (Running Average Power Limit)-based metering to provide CPU-centric power measurements at a sampling rate at processor level up to 20Hz. RAPL also provides power capping capabilities by setting power limitations on the processor package or DRAM.

We have configured the big data processing frameworks as a baseline using commonly used and balanced configurations without optimizations (e.g., it doesn't feature DC/OS layer such as Apache Mesos). The specific characteristics of the system configuration are described as follows. (1) Hadoop version 2.7.1 was deployed using YARN. One server was configured as NameNode and seven servers as DataNodes for the HDFS file system. HDFS uses 128MB blocks with 3 replicas for each block. Hadoop was configured to run 32 containers per node, and there is at least 2GB memory for each container. The memory of JVM heap size, Map Task and Reduce Task are set to 4GB, per task, and (2) Spark version 1.5.1 was deployed using YARN. Like Hadoop, one server was configured as NameNode and seven servers as DataNode for the HDFS file system. One server as Master and seven servers as Slaves were configured. For each Spark application 7 executors were configured (i.e., one per node), using 8 cores each. The JVM memory was set to 20GB and 64GB for Spark Driver and Executor, respectively.

A comprehensive set of representative workloads were selected, including Grep, K-Means, and WordCount for both Hadoop and Spark. TeraSort, PageRank, and Connected Components were used for Spark to understand and characterize Spark behaviors in more detail. Tables 1 and 2 show the workloads that we used with their typical characteristics and utilized data sets, respectively. Grep and WordCount are data intensive and one-pass-type workloads. K-Means is typically compute intensive and an iterative workload. In order to further investigate the impact of storage technologies in Spark, Terasort, PageRank, and Connected Components were selected. Different metrics were collected for each of the workloads, including energy consumption, power requirements, execution time, and resource utilization (e.g., CPU utilization, RAM memory pressure - via LLC miss rate, and I/O throughout).

[1]http://nsfcac.rutgers.edu/GreenHPC/caper/

Table 1: Hadoop and Spark Workloads

Workload	Description	Type
Grep	extracts matching strings from text files and counts how many time they occurred	IO-bound, one pass
Word Count	reads text files and counts how often words occur	IO-bound, one pass
K-Means	K-Means classifier	CPU-bound, iterative
Terasort	samples the input data and uses map/reduce to sort the data into a total order	Network-bound
PageRank	measures the importance of each vertex in a graph	CPU-bound, iterative
Connected Components	labels each connected component of the graph with the ID of its lowest-numbered vertex	CPU-bound, iterative

Table 2: Hadoop and Spark Datasets

Input Dataset	Workload
PUMA Wikipedia	Grep, Word Count
Friendster social network	PageRank, Connected Components (65×10^6 Nodes, 1.8×10^9 Edges)
Hadoop TeraGen	TeraSort
BigDataBench K-Means	K-Means

5 EXPERIMENTAL RESULTS

The experimentation presented in the following sub-sections is focused on following three main issues: (1) understanding performance, energy and power behaviors of the workload for both Apache Hadoop and Apache Spark using different workloads and classes of storage devices and interconnects, (2) exploring the potential of power capping for Apache Spark to control power requirements, and (3) understanding the potential of software-defined infrastructures in systems with non-volatile memory technology.

5.1 Characterizing Behavior Patterns of Big Data Processing Frameworks

This sub-section first explores and discusses how different Big Data processing frameworks impact performance, power, energy, and resource utilization using Grep, K-Means, and WordCount workloads. Hadoop and Spark workloads are both configured to run using HDD as the storage device for the HDFS setup, which is its baseline and standard configuration.

Figure 3 (top) presents the energy consumption of Grep, K-Means, and WordCount for both Hadoop and Spark. The results show that executions with Hadoop consume about 3.2 times, 3.1 times, and 2.2 times more energy than Spark for Grep, K-Means, and WordCount, respectively. Figure 3 (bottom) shows that the execution time of Grep, K-Means, and WordCount using Hadoop is 5%, 2.4 times, and 23% longer than using Spark, respectively, which indicates that executions with Spark consume less energy than with Hadoop - and not only because Spark executions are shorter. The results also show that the I/O throughput with Spark is 7% and 15% higher than Hadoop for Grep and WordCount, on average. Since Grep and WordCount are one-pass-type workloads, Hadoop and Spark have similar sizes of shuffle data, where Hadoop spends more time waiting for data reading and writing when compared to Spark.

As shown in Table 3, the CPU utilization during the execution of Grep, K-Means, and WordCount using Hadoop is longer (e.g., up to 1.8 times for Grep) than the execution of these benchmark applications using Spark. The results indicate that both execution time and CPU utilization with Hadoop are higher than with Spark; as a result, Spark is more energy efficient than Hadoop. Note that

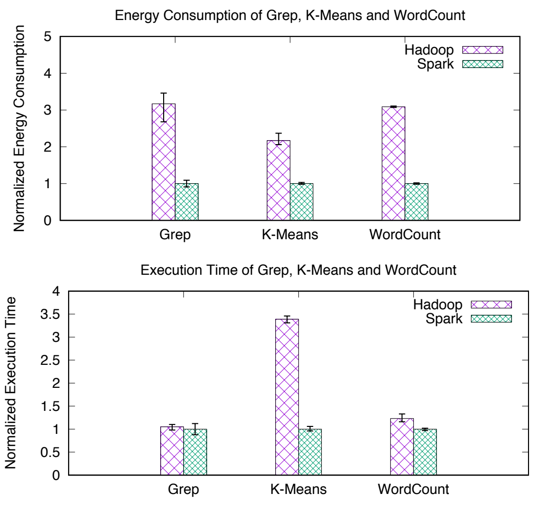


Figure 3: Normalized energy consumption (top) and normalized execution time (bottom) of Grep, K-Means and WordCount using Hadoop and Spark

Table 3: CPU utilization and power consumption of Grep, K-Means, and WordCount execution using Hadoop and Spark

Workload	Framework	AVG Power (W)	Max Power (W)	CPU Util AVG (%)
Grep	Hadoop	667	1,031	61
	Spark	222	979	22
K-Means	Hadoop	625	1,346	40
	Spark	687	938	37
WordCount	Hadoop	1,015	1,261	75
	Spark	573	1,221	48

these quantitative results are with a system configuration using HDD and 1G network connectivity.

Figures 4 and 5 show that CPU utilization and power consumption using Spark is much lower than using Hadoop; however, the I/O throughput using Spark is higher than using Hadoop. This behavior suggests that Spark is capable of delivering higher efficiencies when running workloads than Hadoop under the same constraints and system configuration. The results provided in the following sections will show that Spark provides higher resource utilization efficiencies with other storage and network configurations, and therefore, higher overall efficiency, which is consistent with existing literature.

5.1.1 ***Understanding the Impact of Storage Technology***. This sub-section explores the impact of the storage technology on performance, energy consumption, and other relevant metrics using Grep, K-Means, WordCount, and TeraSort. Figures 6 and 7 show that energy consumption for executions using NVRAM is lower than when using HDD or SSD for all application workloads with both Hadoop and Spark, as expected. The energy consumption of Grep executions using HDD is higher compared to executions using an SSD or NVRAM, especially for Spark (i.e., 40% and 2 times higher energy consumption with Hadoop and Spark, respectively). However, the difference in execution time across storage technology

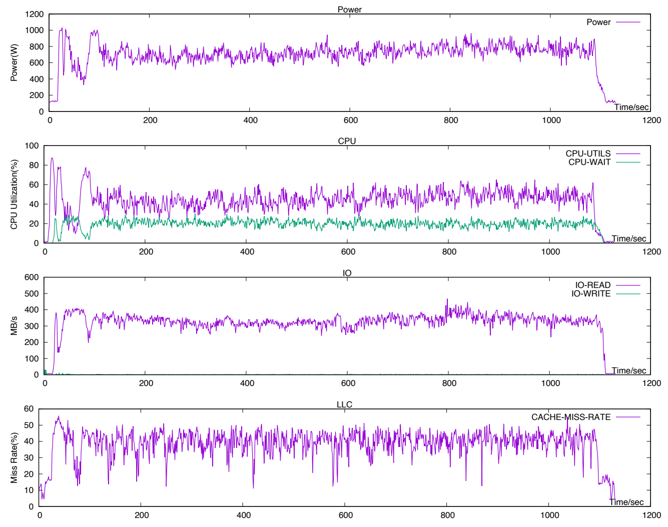


Figure 4: Resource utilization and power consumption of Grep using Hadoop

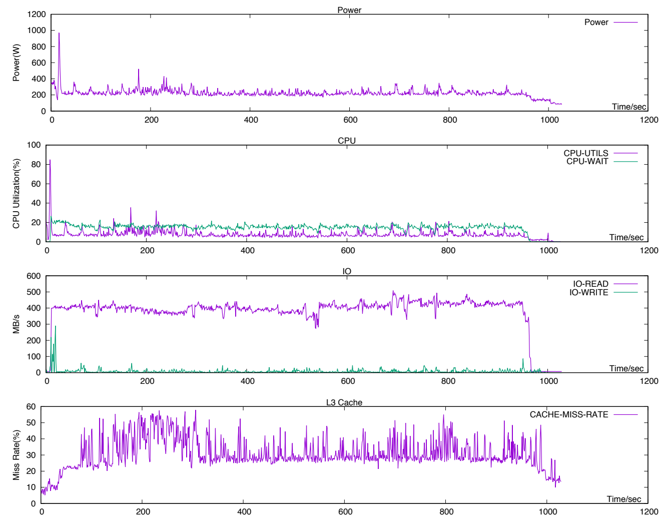


Figure 5: Resource utilization and power consumption of Grep using Spark

configurations is much higher than the difference in energy consumption (e.g., 1.28 times longer execution time using Hadoop and HDD with respect to NVRAM vs. 40% increased energy consumption). Overall, the difference between execution time and energy increase is significantly higher with Spark. As shown in Table 4, the CPU wait percentage is significantly higher using HDD compared to an SSD and NVRAM (i.e., up to 33.5% with Hadoop). Overall, the CPU wait percentage is higher using Spark (i.e., 65.3%, 32.6%, and 6.8% for HDD, SSD, and NVRAM). These results indicate that using NVRAM reduces CPU wait time for both Hadoop and Spark; consequently, it provides higher energy efficiency.

As K-Means is iterative and not an I/O-bound workload, its executions using the different storage choices are similar in terms

Table 4: Resource utilization of Grep, K-Means, WordCount and Terasort with Hadoop and Spark using HDD, SSD and NVRAM

Workload-Framework	Storage device	Time (s)	AVG Power (W)	Max Power (W)	CPU-UTIL (%)	CPU-WAIT (%)	I/O (MB/s)	LLC Miss Rate (%)
Grep-Hadoop	HDD	1,082	667	1,031	66.7	33.3	335	33
	SSD	480	1,108	1,294	99.6	0.4	695	37
	NVRAM	474	1,093	1,283	100.0	0.0	902	37
Grep-Spark	HDD	1,030	222	979	34.7	65.3	385	29
	SSD	319	326	993	67.4	32.6	1,079	30
	NVRAM	140	551	1,213	93.2	6.8	3,213	29
K-Means-Hadoop	HDD	4,237	625	1,346	99.0	1.0	48	17
	SSD	4,344	594	1,300	99.8	0.2	68	19
	NVRAM	4,415	574	1,329	100.0	0.0	70	18
K-Means-Spark	HDD	1,249	687	938	97.6	2.4	30	23
	SSD	1,287	636	945	100.0	0.0	38	25
	NVRAM	1,220	677	913	100.0	0.0	73	24
WordCount-Hadoop	HDD	1,705	1,015	1,261	92.1	7.9	381	42
	SSD	1,509	1,115	1,287	99.6	0.4	424	43
	NVRAM	1,477	1,099	1,274	99.7	0.3	469	43
WordCount-Spark	HDD	1,390	573	1,221	65.9	34.1	407	39
	SSD	595	1,003	1,262	97.6	2.4	835	43
	NVRAM	565	1,032	1,296	100.0	0.0	1,113	43
Terasort-Hadoop	HDD	3,191	565	1,096	66.5	33.5	918	31
	SSD	2,361	746	1,278	80.7	19.3	1,292	34
	NVRAM	1,896	857	1,250	88.7	11.3	1,626	34
Terasort-Spark	HDD	5641	322	932	52.7	47.3	591	25
	SSD	2,375	508	920	76.3	23.7	1,166	35
	NVRAM	1,798	639	981	95.6	4.4	1,941	43

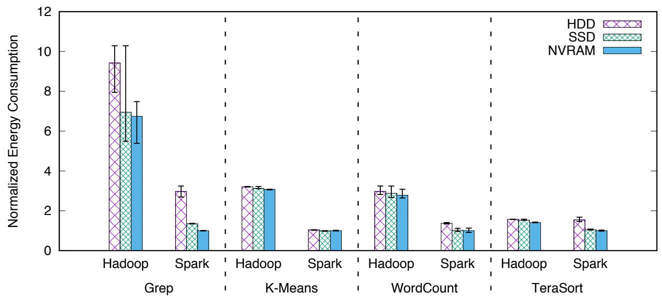


Figure 6: Energy Consumption of Grep, Kmeans, WordCount and Terasort using HDD, SSD and NVRAM with Hadoop and Spark

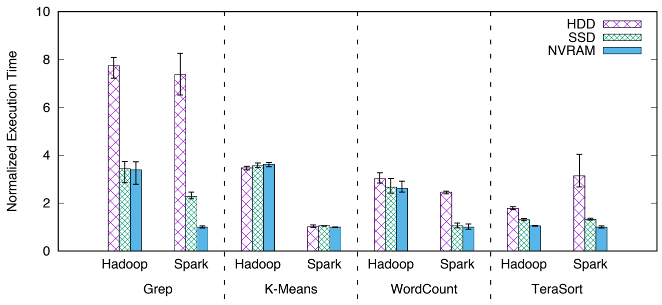


Figure 7: Execution Time of Grep, Kmeans, WordCount and Terasort using HDD, SSD and NVRAM with Hadoop and Spark

of execution time and energy. However, executions using Spark are much shorter and consume much less energy than executions using Hadoop. Another factor in Spark is RDD caching. Specifically, the first iteration of K-Means scans all data into RDD caching, which means the calculation of subsequent iterations are based on the cached RDD. Since Spark reads are from RDDs, K-Means is not constrained by I/O throughout using Spark.

Figures 6 and 7 show that WordCount executions with HDD are 15% longer and consume 7% more energy than executions with NVRAM using Hadoop. However, the execution time and energy consumption of WordCount executions with SSD and NVRAM are similar. As shown in Table 4, the I/O throughput observed in executions with HDD is 10% and 19% lower than executions with an SSD and NVRAM. The CPU wait time using HDD is higher than using an SSD and NVRAM, which results in higher energy consumption and longer execution time. In the case of Spark, WordCount executions using HDD are significantly longer (up to 1.46 times) and consume more energy than executions using an SSD and NVRAM. As observed in Table 4, the I/O throughout of executions using HDD with Spark is lower than the executions with an SSD or NVRAM. Similarly, the CPU wait time of executions using HDD is much higher than the executions using an SSD and NVRAM. As a result, the power required for executions using HDD are approximately half of the power required in executions using an SSD and NVRAM.

Figures 6 and 7 show that TeraSort executions using HDD and SSD consume significantly more energy than executions using NVRAM, especially with Spark (up to 68.3%). Spark does not support a simultaneous read and write function, therefore, if the storage and/or network are not fast enough, the shuffling phase will consume a lot of time. As TeraSort is an I/O-bounded workload, it has a heavy shuffle phase. Consequently, Hadoop provides similar or superior performance than Spark in executions using HDD. Figures 8 and 9 show TeraSort behavior patterns with Hadoop and Spark using NVRAM. The figures clearly show different CPU and I/O patterns, which result in different power consumption profiles.

The results discussed above show tradeoff between power budget, execution time and energy consumption and indicate that, overall, Spark provides higher performance and lower energy consumption. The rest of this sub-section concentrates on further understanding the impact of the different storage choices using the following three Spark workloads:

TeraSort is a popular sorting workload for benchmarking Big Data frameworks. As shown at the bottom of Table 4, TeraSort executions

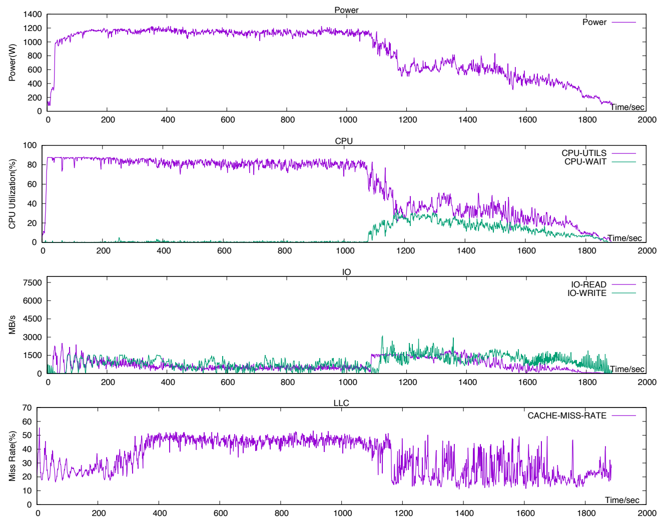


Figure 8: Resource utilization and power consumption of TeraSort with Hadoop using NVRAM

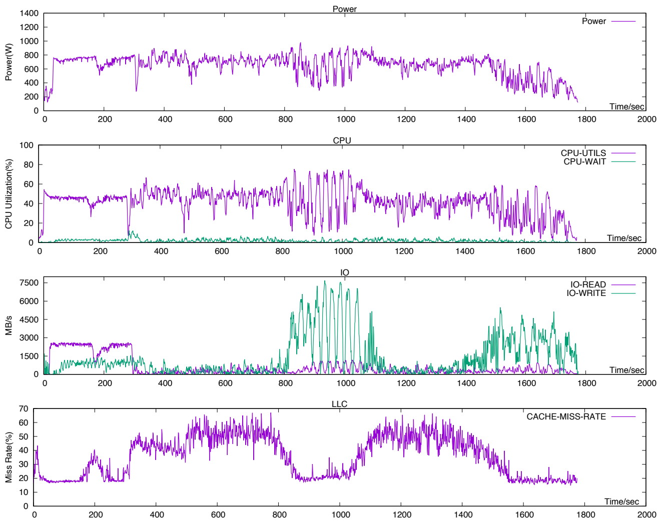


Figure 9: Resource utilization and power consumption of TeraSort with Spark using NVRAM

are heavily influenced by the storage technology. As TeraSort is both I/O- and CPU-bounded, the CPU wait percentage is lower with superior storage technologies (i.e., NVRAM).

PageRank is a graph algorithm proposed by Google to rank web pages by the number and quality of links to a page. Five iterations of the workload were used in each execution. In contrast to TeraSort, Table 5 shows that the CPU wait percentage is similar and almost null for the different storage technologies. However, the energy consumption is most efficient using NVRAM (6.8% and 4.6% lower than HDD or SSD, respectively).

Connected Components is also an iterative graph processing workload. It computes the connected component of each vertex and returns a graph with the vertex value containing the lowest vertex ID in the connected component containing that vertex. The

Table 5: Resource utilization of PageRank execution with Spark using HDD, SSD and NVRAM

Storage	Energy (KJ)	Time (s)	AVG Power (W)	Max Power (W)	CPU-Util (%)	CPU-Wait (%)	IO (MB/s)
HDD	715	2,921	321	759	99.5	0.5	22.0
SSD	628	2,212	284	747	99.6	0.4	25.0
NVRAM	657	2,137	308	830	99.6	0.5	31.6

Table 6: Resource utilization of Connected Components execution with Spark using HDD, SSD and NVRAM

Storage	Energy (KJ)	Time (s)	AVG Power (W)	Max Power (W)	CPU-Util (%)	CPU-Wait (%)	IO (MB/s)
HDD	893	3,587	256	760	99.6	0.4	12.8
SSD	888	3,558	249	739	99.6	0.4	17.2
NVRAM	839	2,952	287	755	99.6	0.4	24.9

results shown in Table 6 indicate that Connected Components and PageRank have very similar behavior patterns.

5.1.2 *Understanding the Impact of the Network.*

This subsection briefly discusses how network bandwidth impacts performance, power, energy, and resource utilization with Hadoop and Spark using Grep, WordCount, K-Means, TeraSort, PageRank, and Connected Components (CC). The network is configured to use either 1Gb Ethernet or 10Gb Ethernet interfaces.

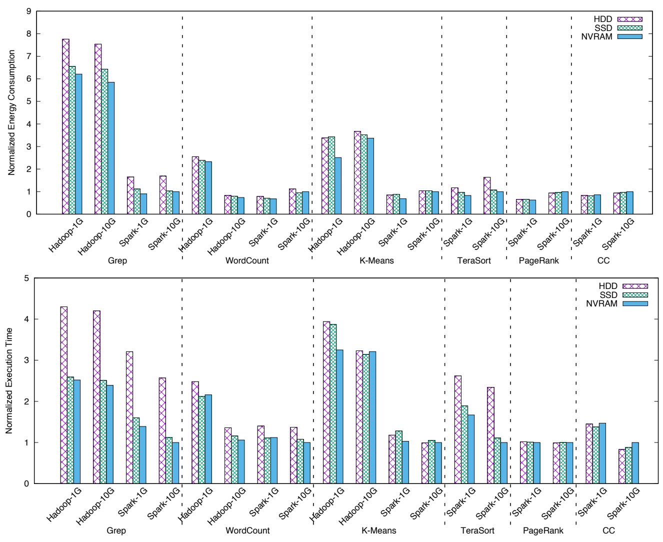


Figure 10: Normalized energy consumption (top) and execution time (bottom) of Grep, WordCount, K-Means, TeraSort, PageRank and Connected Components using HDD, SSD and NVRAM with Hadoop and Spark

Figure 10 shows that behavior patterns are workload-dependent and that in general, the energy consumption is higher (or similar) for executions using the 10G network compared to the energy consumption of executions using the 1G network with both Hadoop and Spark; however, the execution time using the 10G network is shorter (or similar) than executions using the 1G network.

The results discussed above are consistent with the expected behaviors; however they provide an understanding of different design choices based on different workload profiles and optimization goals (e.g., performance, power, and cost), which we use for understanding the potential of software-defined infrastructure in the context of big data processing frameworks. For example, using a 10G network is worthwhile for Spark when high performance is needed and neither power nor budget are constrained; however, when performance degradation can be tolerated and power is not heavily constrained, 10G is not worth the cost when using Hadoop. This example represents a class of data-intensive one-pass workloads that can be heavily influenced by the storage technology used.

5.2 Exploring the Impact of Power Capping

This section studies the potential of power capping via RAPL for balancing the power and performance tradeoffs while also considering system designs with HDD, SSD, and NVRAM storage options. This section considers Grep and WordCount, as they are high-power workloads (see Table 4). We used 25W, 40W, 55W, 70W, and 95W CPU power caps. Table 7 shows the Energy Delay Product (EDP) for both Grep and WordCount with Hadoop and Spark using HDD, SSD and NVRAM. We use the EDP metric as it shows different groups of combinations with different behavior trends and incorporate both energy consumption and execution time. ED^2P is not used as this paper is not focused on studying the CPU voltage level or propagation of delay.

Overall, the results show that RAPL is effective in reducing the energy consumption for all three workloads. Table 7 also presents the resource utilization including power consumption for different RAPL settings. The main purpose of power capping is running a workload within a given power budget (i.e., maximum power). Table 7 shows that when CPU package power is capped from 95W to 25W the reduction of maximum power is (on average) 27%, 29% and 32% for all combinations using HDD, SSD and NVRAM, respectively. It is worth noting that the average energy reduction when capping the CPU package power from 95W to 25W is higher when using NVRAM than when using HDD or SSD, i.e., 18.9%, 20.1%, and 25.1% execution time increase (on average) when using HDD, SSD and NVRAM, respectively. However, the execution time increase when capping the CPU package power from 95W to 25W is lower when using HDD than when using SSD or NVRAM, i.e., 0.97%, 5.22%, and 7.72% execution time increase (on average) when using HDD, SSD and NVRAM, respectively.

5.3 Exploring the Potential of SDI

In this sub-section we explore the potential of software-defined infrastructure for existing big data processing frameworks through simulation using the information obtained in the characterization presented above. We have developed a simulation framework focused big data workload allocation to resources. These workloads are composed of big data applications for both Hadoop (Grep, WordCount, TeraSort and K-Means) and Spark (Grep, WordCount, TeraSort, K-Means, Connected Component and PageRank) frameworks. As the storage technology used in the big data frameworks running these applications significantly impact different key metrics such as execution time and energy consumption, the workload allocation algorithm focuses on storage issues in the system design.

In order to simulate the execution of the workloads in software-define infrastructure and traditional infrastructures (i.e., with fixed amount of accessible storage resources), we: (1) assume that the datacenter is composed of multiple big data deployments (clusters) for running big data workloads, and (2) fix the total amount of storage available in the datacenter (i.e., total amount of HDD, SSD and NVRAM). Under these assumptions, we consider two scenarios:

- **non-SDI** (traditional): The storage available in one cluster is fixed and can be used only by applications running in that cluster.
- **SDI**: All storage available in the datacenter is available to all applications running in any cluster.

In this initial approximation, we also assume that the latency and bandwidth to off-node and in-node storage devices are similar. Our ongoing work includes the exploration of the design space (e.g., interconnect capabilities required for realizing effective software-defined infrastructure) by introducing different latency and bandwidth limitations to off-node storage device access.

Table 7: Power, I/O utilization and EDP of Grep and WordCount with Hadoop and Spark using HDD, SSD and NVRAM

Workload-PlatForm	RAPL Power Cap	Storage Device	AVG Power (W)	Max Power (W)	IO (MB/s)	EDP ($/10^6$)	Storage	AVG Power (W)	Max Power (W)	IO (MB/s)	LLC Miss Rate	Storage Device	AVG Power (W)	Max Power (W)	IO (MB/s)	EDP ($/10^6$)
Grep Hadoop	25W		508	797	333	628.1		751	879	681	210.5		727	860	803	198.9
	40W		522	975	334	639.5		746	977	678	205.7		730	978	815	200.2
	55W	HDD	614	1,014	337	737.3	SSD	769	1,070	696	204.5	NVRAM	750	1,044	826	202.7
	70W		666	1,040	338	783.9		1,069	1,237	764	249.1		1,033	1,179	903	235.6
	95W		667	1,031	335	784.2		1,108	1,294	695	256.5		1,093	1,283	902	246.2
WordCount Hadoop	25W		712	850	345	2,155.9		731	872	396	1830.5		711	839	412	1,722.7
	40W		716	995	346	2,184.2		734	992	399	1,811.4		714	977	420	1,750.5
	55W	HDD	782	1,032	350	2,266.8	SSD	784	1,048	408	1861.5	NVRAM	752	1,000	450	1,811.8
	70W		1,030	1,222	380	2,605.9		1,070	1,217	437	2,388.6		1,023	1,210	475	2,374.3
	95W		1,015	1,261	381	2,952.5		1,115	1,287	424	2,538.6		1,099	1,274	469	2,397.4
Grep Spark	25W		213	743	387	221.1		327	747	1,211	32.6		449	881	3,140	8.9
	40W		213	980	387	224.6		320	979	1,199	32.4		474	1,063	3,443	7.9
	55W	HDD	215	953	387	223.0	SSD	328	975	1,188	33.0	NVRAM	483	1,101	3,255	8.9
	70W		223	979	383	232.6		332	988	1,195	33.4		559	1152	3,384	9.9
	95W		222	979	385	235.4		326	993	1,079	33.1		551	1,213	3,213	10.8
WordCount Spark	25W		457	831	399	886.7		733	894	877	292.6		716	896	1,002	272.9
	40W		491	985	402	950.6		730	987	884	286.7		712	951	1,012	271.7
	55W	HDD	556	1,016	404	1,066.6	SSD	756	1,070	894	292.0	NVRAM	741	1,122	1,055	271.7
	70W		578	1,180	398	1,094.5		1,006	1,271	927	345.9		985	1,198	1,126	316.1
	95W		573	1,221	407	1,107.0		1,003	1,262	835	354.8		1,032	1,296	1,113	329.5

Our simulation study requires key data points, such as the workloads' execution time and energy consumption using different storage device technology. The meaning of the parameters used in the simulation are described as follows:

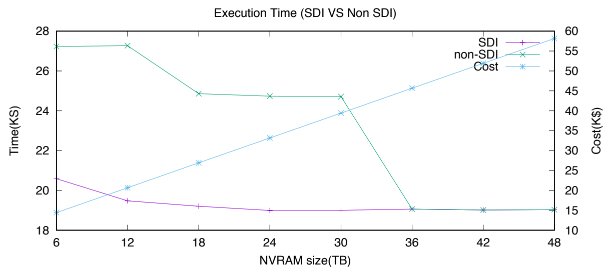
- W: Randomly generated workload with 100 application instances
- S_{WL}: Workload required storage capacity
- $S_{HDD}, S_{SSD}, S_{NVRAM}$: Available HDD, SDD and NVRAM capacity, respectively
- $T_{HDD}, T_{SSD}, T_{NVRAM}$: Workload execution time using HDD, SSD and NVRAM, respectively
- $E_{HDD}, E_{SSD}, E_{NVRAM}$: Energy consumption using HDD, SSD and NVRAM, respectively
- $C_{HDD}, C_{SSD}, C_{NVRAM}$: Energy consumption using HDD, SSD and NVRAM devices, respectively
- $T/E/C_{SDI}, T/E/C_{non-SDI}$: Execution Time, Energy Consumption and Cost, using SDI and non-SDI configurations, respectively

In the simulations, the datacenter contains 10 clusters, each composed of 8 nodes, which is the configuration used in the characterization presented above. The total size of HDD, SSD and NVRAM for the overall datacenter are set to 37 TB, 10.8 TB and 6-48 TB, respectively. The cost (C) refers to the capital cost of different technologies (e.g., NVRAM vs. HDD), which is part of TCO (Total Cost of Ownership). Default values are based on standard pricing for enterprise storage at \$0.4882/GB, \$0.5859/GB and \$1.0417/GB for HDD, SSD and NVRAM, respectively. Algorithm 1 presents the workload allocation algorithm, which by default prioritizes NVRAM as the first choice, the second choice is SSD and last choice is HDD. We follow this approach to understand the tradeoff between response time and energy efficiency and cost, which is a key issue in datacenter design and deployment.

Figure 11 shows the tradeoff between cost and execution time and energy consumption using different NVRAM size (i.e., different investment choices) using non-SDI and SDI scenarios. The results

Algorithm 1: Workloads Allocation Algorithm.

```
1 Function Storage Device Priority ;
  Input  : $W$, $S_{WL}$,
           $S_{HDD}$, $S_{SSD}$, $S_{NVRAM}$, $T_{HDD}$, $T_{SSD}$, $T_{NVRAM}$,
           $E_{HDD}$, $E_{SSD}$, $E_{NVRAM}$, $C_{HDD}$, $C_{SSD}$, $C_{NVRAM}$
  Output : $T_{SDI}$, $E_{SDI}$, $C_{SDI}$, $T_{non-SDI}$, $E_{non-SDI}$, $C_{non-SDI}$
   1: start time;
   2: while $W$ is not empty do
   3:     for $i = 1; i <= 10; i++$ do
   4:         if cluster $i$ is empty then
   5:             if $S_{NVRAM} >= S_{WL}$ then
   6:                 push next workload into NVRAM;
   7:             else if $S_{SSD} >= S_{WL}$ then
   8:                 push next workload into SSD;
   9:             else
  10:                 push next workload into HDD;
  11:             end if
  12:         end if
  13:     end for
  14: end while
  15: end time;
  return ($T_{SDI}$, $E_{SDI}$, $C_{SDI}$, $T_{non-SDI}$, $E_{non-SDI}$, $C_{non-SDI}$);
```



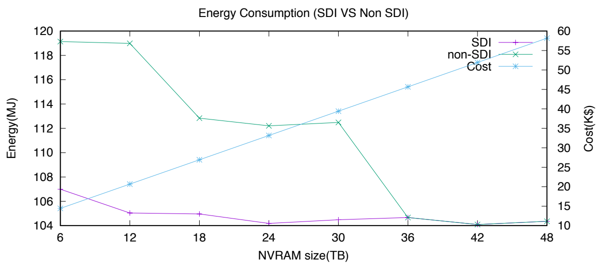
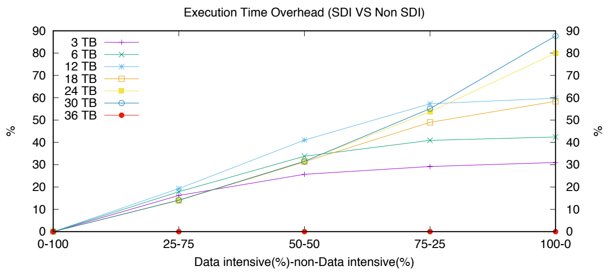


Figure 11: Execution time (top) and energy (bottom) vs. total storage cost for SDI and non-SDI scenarios



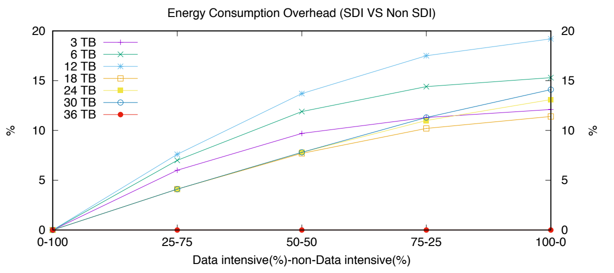


Figure 12: Execution time (top) and energy (bottom) overheads of non-SDI scenarios with respect to SDI

show that both execution time and energy consumption in the SDI scenario are significantly lower than the non-SDI scenario with up to 36TB of NVRAM. While larger NVRAM sizes provides better performance/energy (up to 24-30TB) in the SDI scenario, these improvements are at a significant cost increase.

As opposed to CPU-bound workloads, data-intensive workloads are highly impacted by the storage technology used. In order to understand this issue in SDI and non-SDI scenarios, we classify the simulated applications into two types: (1) Data intensive (Grep, WordCount and TeraSort) and 2) non-Data intensive (K-Means, PageRank and Connected Component) and generate workloads with different proportions of these types of applications (from 0% to 100%). Figure 12 shows the execution time and energy overhead of the non-SDI scenario with respect to the SDI one. In addition to the tradeoffs shown, the results indicate that the execution time of data-intensive workloads in the SDI scenario can get up to 87.7% shorter than in the non-SDI scenario. However, with 36TB of NVRAM the execution time and energy are similar in both SDI and non-SDI scenarios, which is consistent with the results shown in Figure 11. These results clearly show the potential of software-defined infrastructure for big data processing frameworks.

6 CONCLUSIONS AND FUTURE WORK

This paper provided a detailed evaluation of performance, power and resource utilization behaviors trends of Hadoop and Spark using a relevant set of Big Data benchmarks and different technology choices. The experimental evaluation supports the argument that NVRAM is a solid candidate for supporting in-memory analytics in ongoing architectures with deeper memory hierarchies. The experimental evaluation also showed that the network bandwidth impacts more significantly the performance in Spark workloads than in Hadoop's ones. This work also proposed using power capping (i.e., RAPL) and evaluated options for enabling data analytics under power constraints while meeting performance and other goals. The experimental evaluation showed that Spark is also more efficient than Hadoop under power capping. Finally, simulation-based experimentation showed the significant advantages (upper bound) of software-defined infrastructures for existing Big Data processing frameworks.

The results from this work provide meaningful data points to build multi-criteria application-centric models for Big Data co-design and motivate further research focused on in-memory processing systems with deeper memory hierarchies and different design options and constraints for software-defined infrastructures (e.g., 400G MSA vs. 400/800G embedded optics vs. PCIe 5.0). Power capping techniques and workload scheduling in software-defined infrastructures remain part of future work. Our future work also includes understanding the tradeoffs of using other applications formulations and streaming-based processing frameworks such as Apache Storm and Apache Flink.

ACKNOWLEDGMENTS

This work is supported in part by National Science Foundation via grants numbers ACI-1464317 and CNS-1305375, and was conducted as part of the Rutgers Discovery Informatics Institute (RDI^2).

REFERENCES

[1] 2009. Rumen: A tool to extract Job Characterization Data from Job Tracker Logs. https://www.top500.org/lists/2016/06//. (2009).

[2] 2012. IBM:What Is Big Data: Bring Big Data to the Enterprise. http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html/. (2012).

[3] 2014. ALOJA, Benchmark Repository and Performance Analysis Tool. (2014).

[4] Hrishikesh Amur, James Cipar, Varun Gupta, Gregory R Ganger, Michael A Kozuch, and Karsten Schwan. 2010. Robust and flexible power-proportional storage. In *Proceedings of the 1st ACM symposium on Cloud computing*. ACM.

[5] Kelvin Cardona, Jimmy Secretan, Michael Georgiopoulos, and Georgios Anagnostopoulos. 2007. A grid based system for data mining using MapReduce. In *Seventh IEEE International Conference on Grid Computing*. Citeseer, 33.

[6] Yanpei Chen, Sara Alspaugh, Dhruba Borthakur, and Randy Katz. 2012. Energy efficiency for large-scale mapreduce workloads with significant interactive analysis. In *Proc. of the 7th ACM european conference on Computer Systems*. 43–56.

[7] Yanpei Chen, Archana Ganapathi, and Randy H Katz. 2010. To compress or not to compress-compute vs. io tradeoffs for mapreduce energy efficiency. In *Proceedings of the first ACM SIGCOMM workshop on Green networking*. 23–28.

[8] Yanpei Chen, Laura Keys, and Randy H Katz. 2009. Towards energy efficient mapreduce. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-109* (2009).

[9] Arghya Kusum Das, Seung-Jong Park, Jaeki Hong, and Wooseok Chang. 2015. Evaluating different distributed-cyber-infrastructure for data and compute intensive scientific application. In *Big Data (Big Data), IEEE Intl. Conf. on*. 134–143.

[10] Chris Douglas and Hong Tang. 2010. Gridmix3 Emulating Production Workload for Apache Hadoop. (2010).

[11] Archana Ganapathi, Yanpei Chen, Armando Fox, Randy Katz, and David Patterson. 2010. Statistics-driven workload modeling for the cloud. In *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*. IEEE, 87–92.

[12] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. 2011. Dominant Resource Fairness: Fair Allocation of Multiple Resource Types.. In *NSDI*, Vol. 11. 24–24.

[13] Íñigo Goiri, Kien Le, Thu D Nguyen, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. 2012. GreenHadoop: leveraging green energy in data-processing frameworks. In *Proceedings of the 7th ACM european conference on Computer Systems*. ACM, 57–70.

[14] Suhel Hammoud, Maozhen Li, Yang Liu, Nasullah Khalid Alham, and Zelong Liu. 2010. MRSim: A discrete event based MapReduce simulator. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh Intl. Conf. on*, Vol. 6. 2993–2997.

[15] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D Joseph, Randy H Katz, Scott Shenker, and Ion Stoica. 2011. Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center.. In *NSDI*, Vol. 11. 22–22.

[16] Shadi Ibrahim, Tien-Dat Phan, Alexandra Carpen-Amarie, Houssem-Eddine Chihoub, Diana Moise, and Gabriel Antoniu. 2016. Governing energy consumption in hadoop through cpu frequency scaling: An analysis. *Future Generation Computer Systems* 54 (2016), 219–232.

[17] Rini T Kaushik and Milind Bhandarkar. 2010. Greenhdfs: towards an energy-conserving, storage-efficient, hybrid hadoop compute cluster. In *Proceedings of the USENIX annual technical conference*. 109.

[18] Soila Kavulya, Jiaqi Tan, Rajeev Gandhi, and Priya Narasimhan. 2010. An analysis of traces from a production mapreduce cluster. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*. IEEE, 94–103.

[19] Kashif Nizam Khan, Mohammad Ashraful Hoque, Tapio Niemi, Zhonghong Ou, and Jukka K Nurminen. 2016. Energy efficiency of large scale graph processing platforms. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. ACM, 1287–1294.

[20] Kiyoung Kim, Kyungho Jeon, Hyuck Han, Shin-gyu Kim, Hyungsoo Jung, and Heon Y Yeom. 2008. Mrbench: A benchmark for mapreduce framework. In *Parallel and Distributed Systems, 2008. ICPADS'08. 14th IEEE Intl. Conf. on*. 11–18.

[21] Willis Lang and Jignesh M Patel. 2010. Energy management for mapreduce clusters. *Proceedings of the VLDB Endowment* (2010).

[22] Shen Li, Tarek Abdelzaher, and Mindi Yuan. 2011. Tapa: Temperature aware power allocation in data center with map-reduce. In *Green Computing Conference and Workshops (IGCC), 2011 International*. IEEE, 1–8.

[23] Ling Liao. 2017. Intel Silicon Photonics: from Research to Product. *IEEE Components, Packaging and Manufacturing* (2017).

[24] Mian Lu, Lei Zhang, Huynh Phung Huynh, Zhongliang Ong, et al. 2013. Optimizing the mapreduce framework on intel xeon phi coprocessor. In *Big Data, IEEE International Conference on*. 125–130.

[25] Liang Luo, Wenjun Wu, Dichen Di, Fei Zhang, Yizhou Yan, and Yaokuan Mao. 2012. A resource scheduling algorithm of cloud computing based on energy efficient optimization methods. In *Intl. Green Computing Conference (IGCC)*. 1–6.

[26] Clifford Lynch. 2008. Big data: How do your data grow? *Nature* (2008).

[27] AC Murthy. 2011. The hadoop map-reduce capacity scheduler. *URL http://developer. yahoo. com/blogs/hadoop/posts/2011/02/capacity-scheduler* (2011).

[28] Arun C Murthy. 2009. Mumak: Map-Reduce Simulator. *MAPREDUCE-728, Apache JIRA* (2009).

[29] Jorda Polo, David Carrera, Yolanda Becerra, Malgorzata Steinder, and Ian Whalley. 2010. Performance-driven task co-scheduling for mapreduce environments. In *IEEE Network Operations and Management Symposium-NOMS 2010*. 373–380.

[30] Thomas Sandholm and Kevin Lai. 2010. Dynamic proportional share scheduling in hadoop. In *Job Scheduling Strategies for Parallel Processing*. 110–131.

[31] Nidhi Tiwari, Umesh Bellur, Santonu Sarkar, and Maria Indrawan. 2016. Identification of critical parameters for MapReduce energy efficiency using statistical Design of Experiments. In *Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International*. IEEE, 1170–1179.

[32] Guanying Wang, Ali R Butt, Prashant Pandey, and Karan Gupta. 2009. Using realistic simulation for performance analysis of mapreduce setups. In *Proc. of the 1st ACM workshop on Large-Scale system and application performance*. 19–26.

[33] Thomas Wirtz and Rong Ge. 2011. Improving MapReduce energy efficiency for computation intensive workloads. In *Green Computing Conference and Workshops (IGCC), 2011 International*. IEEE, 1–8.

[34] Nezih Yigitbasi, Kushal Datta, Nilesh Jain, and Theodore Willke. 2011. Energy efficient scheduling of mapreduce workloads on heterogeneous clusters. In *2nd International Workshop on Green Computing Middleware*.