Hadoop Workloads Characterization for Performance and Energy Efficiency Optimizations on Microservers

Maria Malik, Katayoun Neshatpour, Setareh Rafatirad, Houman Homayoun

Abstract—*The traditional low-power embedded processors such as Atom and ARM are entering into the high-performance server market. At the same time, big data analytics applications are emerging and dramatically changing the landscape of data center workloads. Emerging big data applications require a significant amount of server computational power. However, the rapid growth in the data yields challenges to process them efficiently using current high-performance server architectures. Furthermore, physical design constraints, such as power and density have become the dominant limiting factor for scaling out servers. Numerous big data applications rely on using Hadoop MapReduce framework to perform their analysis on large-scale datasets. Since Hadoop configuration parameters as well as system parameters directly affect the MapReduce job performance and energy-efficiency, joint application, system and architecture level parameters tuning is vital to maximize the energy efficiency for Hadoop-based applications. In this work, through methodical investigation of performance and power measurements, we demonstrate how the interplay among various Hadoop configuration parameters, as well as system and architecture level parameters affect not only the performance but also the energy-efficiency across various big data applications. Our results identify trends to guide scheduling decision and key insights to help improving Hadoop MapReduce applications performance, power and energy-efficiency on microservers.*

Index Terms—Application Characterization, Hadoop MapReduce, big data, Microservers, Energy-Efficiency, Performance, Power and Performance Tuning Parameters

1. Introduction

Low power is one of the main constraints for the design of battery-operated embedded systems. However, this design objective has come into attention for high performance and data center systems as well. The main reasons are power constraint of the processor and physical constraint of the chip as the semiconductor industry has reached its physical scaling limits. In fact, continuous increase in the number of transistors on a chip has led to the so-called "dark silicon" phenomena, where the power density does not allow all the transistors to turn on simultaneously. There is a large body of research on harnessing dark silicon or maximizing performance under power constraints [1, 2, 4, 5]. Cost and environmental reasons are other motivations to govern energy-efficient and low power design. As a consequence, hardware design companies have considered energy efficiency as one of the main design concerns and have provided mechanisms to ease developing green applications. Intel provides RAPL interface which enables the software developers to measure and control the power consumption at different domain, including core, package, DRAM and embedded graphic [6]. ARM has introduced big.LITTLE technology, which allows migrating applications between simple and complex cores based on workload demands. IBM has employed low power little cores in BlueGene/Q to increase power efficiency [7]. As it is evident by these latest developments, the paradigm shift has been occurring from the performance centric to energy-efficient centric design methodologies in the industry.

The energy demand of data centers that support MapReduce model is increasing rapidly [8, 9], which is the main obstacle for their scalability. Moreover, since energy consumption in data centers contributes to major financial burden [10] and prolongs break-even point (when a data center makes a profit), designing energy-efficient data centers is becoming very important [23]. Current server designs, based on commodity high-performance processors are not an efficient way to deliver green computing in terms of performance/watt. Therefore, the embedded processors that are designed and developed based on energy efficiency metrics are finding their way in server architectures [3]. Microservers employ embedded low power processors as the main processing unit. These platforms are shown to be a promising solution to enhance energy-efficiency and reduce cost in data centers. They follow the System-on-Chip (SoC) approach to render the CPU, I/O and networking components fully integrated onto a single chip.

Several companies and academics have developed cluster architectures based on ARM or Intel Atom cores. An example is FAWN (Fast Array of WimpyNodes) [11], which composed of a large number of embedded and efficient Intel Atom cores where each core is low power dissipating only a few watts of power. X-Gene platform [12] developed by Applied Micro is another example of a server-class SoC which is designed for cloud and enterprise servers based on ARM v8 64-bit core architecture. HP low-power Moonshot servers [13] also uses ARM and Atom embedded cores on a single rack. Due to the wide adoption of x86-based architectures in servers, in this paper we

- *Maria Malik is with the Department of ECE, George Mason University, Fairfax, VA. E-mail: mmalik9@gmu.edu.*
- *Katayoun Neshatpour is with the Department of ECE, George Mason University, Fairfax, VA. E-mail: katayoun.neshatpour@gmail.com*
- *Setareh Rafatirad is with the Department of IST, George Mason University, Fairfax, VA. E-mail: srafatir@gmu.edu.*
- *Houman Houmayoun is with the Department of ECE, George Mason University, Fairfax, VA. E-mail: hhomayou@gmu.edu.*

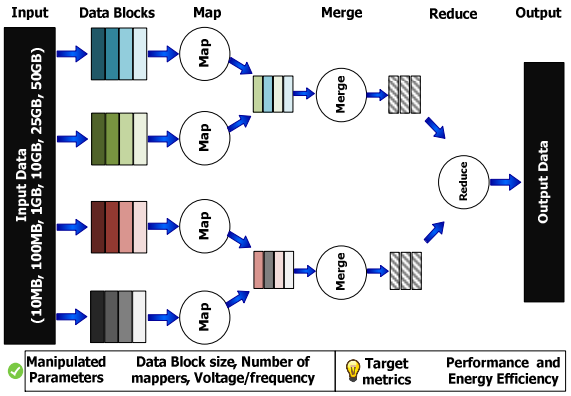

Figure 1. A simple conceptual view of Hadoop data flow

choose Atom to study, as it has a low power embedded micro-architecture with high-performance x86 ISA.

The world of big data is changing constantly and producing a large amount of data that creates challenges to process them using existing solutions. Big data applications heavily rely on deep machine learning and data mining algorithms, running complex database software stack with significant interaction with I/O and OS [43]. The Apache Hadoop framework, a defacto standard for analytics, assists the processing of large datasets in a distributed computing environment. Numerous big data applications rely on using the Hadoop MapReduce framework to perform their analysis on large-scale datasets. Several research works have reported the performance analysis of Hadoop MapReduce applications on high performance servers such as Xeon [14, 15]. However, the important research question is whether low-power embedded architectures are suited to process big data and in particular MapReduce applications efficiently. To understand this, in a recent work [16], we evaluated big data applications on two distinct server architectures; high-performance Xeon server and low-power embedded Atom server. Our results demonstrate that while big Xeon core provides high performance and more energy-efficiency for traditional CPU applications compared to little core, it is not power efficient to process big data applications. The results further show that the little core is more energy-efficient than big core in almost all studied applications, and in particular for compute-intensive applications. Overall, low power embedded architectures can provide significant energy-efficiency for processing big data analytics applications compared to conventional big high performance core.

There have been several works on characterizing Hadoop MapReduce applications [17, 18, 19, 20], or optimizing them for performance or power [15, 21]. Most of these works either mainly focus on performance optimization [22], ignoring energy-efficiency, or mainly deployed on high performance big Xeon core. In addition, given that the performance and power of Hadoop MapReduce applications is sensitive to various tuning parameters at application (application type, data size per node), system (HDFS block size, number of mappers running simultaneously per microserver node) and architecture levels (operating voltage and frequency of core), it is important to understand the role of these parameters and the interplay among them for energy-efficiency optimizations. While prior work mainly ignored the interplay among these tuning parameters, in this work we characterize Hadoop applications across a wide range of tuning parameters to understand the interplay effect and accordingly the optimization opportunities on microservers for enhancing their energy efficiency.

Contributions: To the best of our knowledge this is the first paper that comprehensively analyzes the behavior of the emerging big data applications running in Hadoop MapReduce environment on microserver with respect to various system, application and architecture levels tuning parameters and the interplay among them. This analysis will assist guiding the scheduling decisions and help optimizing for performance, power and energy-efficiency improvements. Grounded in empirical analysis, our main contributions are:

- We analyze the impact of various tuning parameters at system-level (number of mappers running simultaneously per microserver node, HDFS block size), application-level (application type and input data size) and architectural-level (operating voltage and frequency) on the performance, power and energy efficiency for various Hadoop micro-benchmarks and real-world applications.
- We analyze how the interplay of various tuning parameters at application, system, and architecture levels affects the power and performance sensitivity of the Big data applications.
- We analyze the execution time breakdown of various phases of Hadoop micro-benchmarks. To guide power optimization using frequency scaling, we further analyze how the performance of these phases is sensitive to the operating frequency.
- We evaluate real time system resources measurement including CPU utilization and memory footprint to understand the runtime behavior and resource consumption of Hadoop micro-benchmarks when varying system, architecture and application level tuning parameters.

Consequently, we make the following major observation:

- The speedup obtained when increasing the number of available cores on microserver node outweighs the power overhead associated with increasing the number of cores. This indicates that utilizing the maximum number of available cores per node achieves the best energy-efficiency across all studied applications.
- While utilizing all available cores on each microserver node provides the maximum energy-efficiency across all studied applications, concurrent fine-tuning of frequency and HDFS block size reduces the reliance on the maximum number of cores. We can achieve a competitive energy-efficiency with fewer number of cores compared to the maximum number of cores by simultaneously fine tuning the HDFS block size and the operating frequency of the system. This helps freeing up cores on each node to accommodate scheduling co-runner applications in a cluster computing environment.
- Hadoop I/O bound applications can be scheduled at lower processor operating frequency on microserver

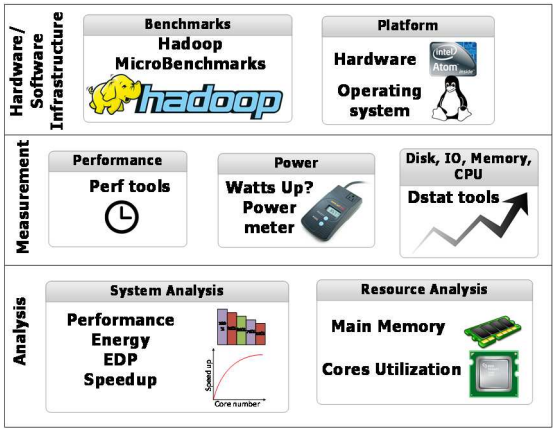

Figure 2. Methodology

to save power. Performance loss can be compensated to a significant extent by increasing the number of mappers, and therefore the number of cores, with a small impact on total power consumption.

2. Hadoop Framework and Tuning Parameters

Apache Hadoop is an open-source Java-based framework of MapReduce implementation. It assists the processing of large datasets in a distributed computing environment and stores data in highly fault-tolerant distributed file system, HDFS. Figure 1 shows a simple conceptual view of steps involve in Hadoop MapReduce. When an application is submitted for scheduling, Hadoop splits its input data into a fixed data blocks where each block is assigned to a map task. A map task transforms the input data into intermediate key-value pairs. These generated intermediate values are transferred from the mappers to the appropriate reducers in the merge stage. Shuffle and sort of key-values are done in this stage. As different subset of intermediate key-value pairs are assigned to each reducer, the reducers consolidate data into the final output. There are a number of parameters that directly impact the MapReduce application performance and energy-efficiency. In this work, we study various parameters including the number of mappers, operating voltage and frequency of the core, HDFS block size, and the size of data per node that can be tuned by the user, scheduler or the system and are impacting the energy-efficiency.

2.1 Application Diversity

Hadoop cluster hosts a variety of big data applications running concurrently. We have included four micro-benchmarks in our study, namely WordCount-WC, Sort-ST, Grep-GP and TeraSort-TS. We have selected these micro-benchmarks as they are kernels in many big data applications representing diverse behavior [19]. These micro-benchmarks stress-test different aspects of a microserver cluster [19]. We have also included two real-world applications namely Naïve Bayes -NB and Collaborative Recommendation Filtering-CF) in our study by incorporating mahout library [42]. Table 1 shows Hadoop micro-benchmarks and real-world applications for this study along with their particular domain and data type.

2.2 Interdependent Tuning Parameters

We have studied the impact of the system, application, and architectural level performance and power tuning parameters including the HDFS block size (32MB, 128MB, 256MB, 512MB, 1024MB), input data size of the application (10MB, 100MB, 1GB, 10GB, 25GB and 50GB), number of mappers that run simultaneously on a single node (1, 2, 4 and 8), and frequency settings (1.2GHz, 1.6GHz, 2.0GHz, 2.4GHz) to evaluate how these parameters affect energy efficiency of big data applications on microserver. Moreover, we thoroughly analyze the impact of these parameters on memory system and processor utilization.

3. Measurement And Methodology

The methodology in which our experiments are conducted is presented in Figure 2. Our methodology is divided into three major steps.

3.1 Hardware/software infrastructure

We conduct our study on Intel Atom C2758 server that has 8 processing cores per node and two levels of cache hierarchy shown in table 2. The operating system is Ubuntu 13.10 with Linux kernel 3.11. All experiments are performed on eight-node Atom server with Hadoop 1.2.1. It is important to note that while network overhead in general is influencing the performance of studied applications and therefore the characterization results, for big data applications, as shown in a recent work [24], a modern high speed network introduces only a small 2% performance overhead. We therefore used a high speed 1 Gbit/s network to avoid making it a performance bottleneck. For this study we have selected parameters that are tunable at user, scheduler, application or system levels [41]. There could be certainly more parameters for performance and power tuning, however, this paper attempts to provide an in-depth understanding of how concurrent tuning of these highly accessible and easy tunable parameters at various levels can significantly impact the performance and energy

Table 1: Studied Hadoop Applications

Type of Benchmark	Application Domain	Workloads	Data Source	Software Stacks
Micro Benchmark	I/O - CPU testing micro program	WordCount (WC)	Text	Hadoop 1.2.1
		Sort (ST)	Table	
		Grep (GP)	Text	
		TeraSort (TS)	Table	
Real world Application	Social Network	Collaborative Filtering (CF)	Text	Hadoop 1.2.1, Mahout 0.6
	E-commerce	Classification (NB)		

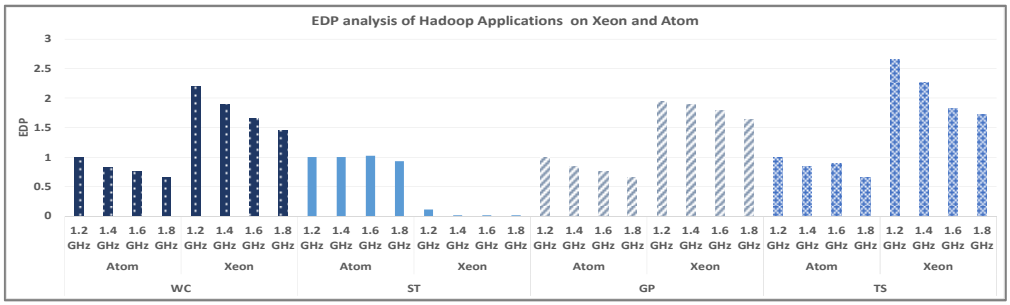

Figure 3: EDP analysis of Hadoop applications on Xeon and Atom with frequency scaling

efficiency.

3.2 Measurement

We use Perf [39] to capture the performance characteristics of the studied applications. Perf is a Linux profiler tool that records hardware performance counters data. Perf exploits Performance Monitoring Unit (PMU) in the processor to measure performance as well as other hardware events at turn-time. For measuring power consumption, Wattsup PRO power meter [40] measures and records power consumption at one second granularity. The power reading is for the entire system, including core, cache, main memory, hard disks and on-chip communication buses. We have collected the average power consumption of the studied applications and subtracted the system idle power to estimate the power dissipation of the core. The same methodology is used in [25], for power and energy analysis. Idle power is measured using Watts up power meter when the server is not running any application and is in the idle state (note that most of power consumption of the off-chip memory subsystems are due to leakage). Dstat [26] is used for main memory, disk and CPU utilization analysis. Dstat is a system-monitoring tool, which collects various statistics of the system.

3.3 Results analysis

The resource utilizations including CPU utilization and memory footprint are saved at run-time in CSV file and then processed by R, an environment for statistical analysis. MapReduce execution breakdown, including setup, map, reduce and clean up phases is obtained through parsing the log files of Hadoop framework. The main analysis of this work includes performance, EDP, MapReduce execution time breakdown, CPU utilization and main memory footprint.

4. Energy efficiency Analysis on Xeon vs Atom

In this section, we present energy efficiency analysis of the studied applications when changing the frequency on two very distinct microarchitectures; Intel Xeon- conventional approach to design a high-performance server and Intel Atom- microserver that advocates the use of a low-power core. Figure 3 and Figure 4 show the EDP results on Atom and Xeon. For each workload, the EDP values are normalized to the EDP result on Atom at the lowest frequency of 1.2GHz and with 512MB HDFS block size. The low power characteristics of the Atom results in a lower EDP on Atom compared to Xeon for most applications with the exception of the Sort. This is due to the fact that the performance gap (in terms of execution time) for the I/O bound benchmarks is very large between Atom and Xeon. Since EDP is the function of the execution time and power, the total EDP on Xeon is lower for the Sort benchmark. In addition, the results show that increase in the frequency reduces the total EDP. While increasing the frequency increases the power consumption, it reduces the execution time of the application and consequently the total EDP.

In addition, we carry out a sensitivity analysis of EDP ratio of the applications on Xeon to Atom. Figure 4 presents the EDP change with respect to the HDFS block size for a frequency of 1.8GHz. The results show that increasing HDFS block size increase the EDP gap between Atom and Xeon. Since in Atom, the performance bottleneck exists in the memory subsystem, improving memory subsystem performance by increasing HDFS block size enhances its performance more significantly compared to Xeon, and reduces the performance gap between the two architectures.

Overall, Atom has shown to be significantly more sensitive to tuning parameters. Therefore, the performance gap between the two architectures can be reduced significantly through fine-tuning of the system and architectural parameters on Atom, allowing maximum energy efficiency.

5. Experimental Results and Analysis

In this section, we discuss the power and performance characteristics of Hadoop micro-benchmarks and real-world applications on Atom microserver with respect to the Hadoop configuration parameters.

5.1 Execution Time Analysis

Figure 5 (represented as a bar graph) shows the execution time of the studied Hadoop applications with respect to the number of mapper slots (cores), HDFS block size and operating frequency with the fixed input data size of 10GB per node for Hadoop micro-benchmarks and real-world applications, respectively For instance, 10GB input data

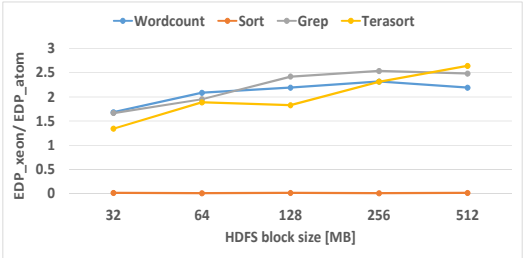

Figure 4: EDP ratio of Hadoop applications on Xeon to Atom at various HDFS block size

size per node presents 80GB input data size processed by application in an 8-node cluster. Hadoop exploits cluster-level infrastructure with many nodes for processing big data applications, however, the experimental data should be collected at the node level to understand how various optimizations and scheduling decisions affects the performance, architectural parameters and energy-efficiency at the node level. Across almost all studied applications, the HDFS block size of 32MB has the highest execution time. Small HDFS block size generates large number of map tasks [number of map task = Input data size /HDFS block size] that increases the interaction between master and slave node. The performance improves significantly with the increase in the HDFS block size. This behavior is consistent across all studied applications when the number of mapper slots is less than 4. With few number of mapper slots (Mapper 2 and Mapper 1), large HDFS block size generates adequate number of map task to keep all mapper slots (cores) in the system busy and reduces the interaction between master and slave node. On the other hand, medium HDFS block size of 256MB and 512MB are more preferable for large number of mapper slots (cores) as it generates more number of map tasks to run simultaneously with a fast execution time per map task. In contrast to Sort, other applications such as WordCount, Grep, and Terasort show a parabolic behavior at large number of mapper slots/cores and achieve the minimum execution time at 256MB or 512MB HDFS block size. Sort optimal HDFS block size is 1024MB whereas WordCount optimal block size is 256MB with the maximum number of mappers' slots/cores. Similar to recent work [19], we observe that Terasort shows hybrid characteristics. Map phase of Terasort is CPU-bound and Reduce phase is I/O-bound, therefore unlike Sort, Terasort optimal HDFS block size is 512MB. Moreover, Grep also illustrates hybrid characteristics with a 512MB optimal HDFS block size. Grep consists of two separate phases; search phase and sort phase running in sequence. Search phase is compute-bound that counts how many times a matching string occurs and sort phase is I/O-bound that matches strings with respect to their frequency.

Table 2: Experimental Microserver Platform

Hardware Type	Parameter	Value
Motherboard	Model	Super micro A1SRM-2758F
CPU (*BW= Bandwidth)	Model	Intel Atom C2758
	# Core	8
	Hyper-Threading	No
	Base Frequency	1.9 GHz
	Turbo Frequency	No
	TDP	20 W
	L1 Cache	24 KB
	L2 Cache	4 * 1024 KB
	Memory Type	DDR3 1600 MHz
	Max. Memory BW*	25.6 GB/s
	Max. Memory Channels	Dual Channel
Disk (HDD)	Model	Seagate ST1000DM003-1CH1
	Capacity	1000 GB
	Speed	7200 RPM
Network Interface Card	Model	ST1000SPEXD4
	Speed	1000 Mbps

The parabolic behavior of WordCount, Grep, and Terasort with respect to HDFS block size can be explained as follows: Small HDFS block size introduces a large number of map tasks that generates more interaction between master and slave nodes. These interactions are necessary to request the HDFS block location information. On the other hand, large HDFS block size reduces the slave node interaction with the master node. Additionally, with a large block size, small metadata is required to be stored on the master node that can be placed in the memory which is faster to access. Conversely, storing large chunk size of data on a node can create performance bottleneck if the application requires accessing the same data recursively. This explains the parabolic behavior in the compute-bound and hybrid applications.

In addition, we have studied the impact of CPU operating frequency to understand how Hadoop applications are sensitive to processor frequency scaling. The results show that Sort application is least sensitive to the frequency, compared to other applications. For this application when CPU frequency is reduced to half, the performance only drops by 20%. Sort is an I/O bound benchmark, which spends most of the execution time requesting data and waiting for I/O operations to complete.

Figure 6 (bar graph) presents the execution time of the studied real-world applications for 1, 2, 4 and 8 number of mapper slots. Based on the micro-benchmark results, we run the real-world applications with the HDFS block size of 64MB – 1024MB as 32MB HDFS block size has the maximum execution time. Similar to micro-benchmarks, CF and NB applications have shown significant reduction in the execution time when changing the HDFS block size and number of mappers. The optimal HDFS block size for CF and NB is 256MB. Both of these applications are compute-bound applications as they have a higher CPU utilization compared to the traditional CPU and parallel benchmarks [16]. Additionally, by increasing the frequency from 1.2GHz to 2.4GHz, we observe a 34.4% - 56.6% and 54.4% - 61.1% reduction in the execution time at the maximum number of mappers with the increase in the HDFS block size in CF and NB, respectively.

Although, the optimal HDFS block size for the peak performance is closely decided by the application type, extensive experimental search to determine the best HDFS size can be avoided by assigning 256MB block size for compute-bound and 1024MB for I/O- bound applications as an optimal choice to get close to the maximum achievable performance.

5.2 Energy-Efficiency Analysis

EDP is a fair metric to compare various architectures, or even the impact of changing optimization knobs in an architecture. EDP (or PxDxD) represents a trade-off between power and performance. Without EDP and just using energy metric for comparison, we can simply reduce the voltage and frequency in an architecture, and reduce its energy, however at a cost of lowering the performance (increased execution time). Therefore, performance along

with energy is important to find out the impact of optimization parameters. Therefore, In order to characterize the energy efficiency, we evaluate Energy Delay Product (EDP) metric to investigate trade-off between power and performance when tuning Hadoop and processor parameters, as shown in Figure 5 (represented as line graph). We observe that the increase in the number of mappers running simultaneously equal to the number of available cores, minimizes the EDP. Worst EDP is reported with one mapper, while 8 mappers give the best EDP by effectively utilizing all available cores. The margin of EDP improvement becomes smaller with the increase in the number of mappers.

The general observation is that the optimal energy efficiency is achieved when we utilize all available cores. In other words, the performance improvement achieved by adding more cores outweighs the power overhead associate with additional cores. However, the important observation is that we can reduce the reliance on the maximum number of available cores by fine-tuning the system and architecture parameters (discussed later in detail). In section 5.4, we present the speedup improvement of each benchmark when increasing the number of mappers. The EDP trend is consistent with the execution time trend showing that in I/O-bound applications, the maximum energy efficiency is achieved with the largest HDFS block size, however compute-bound and hybrid applications achieve optimal EDP at 256MB and 512MB, respectively. Moreover, we have conducted the analyses of frequency

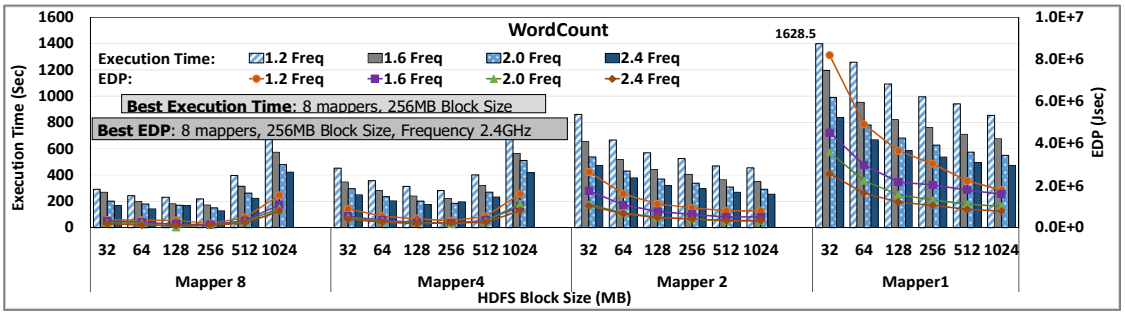

Figure 5(a): Execution Time and EDP of WordCount with various mappers, HDFS block size and operating frequencies

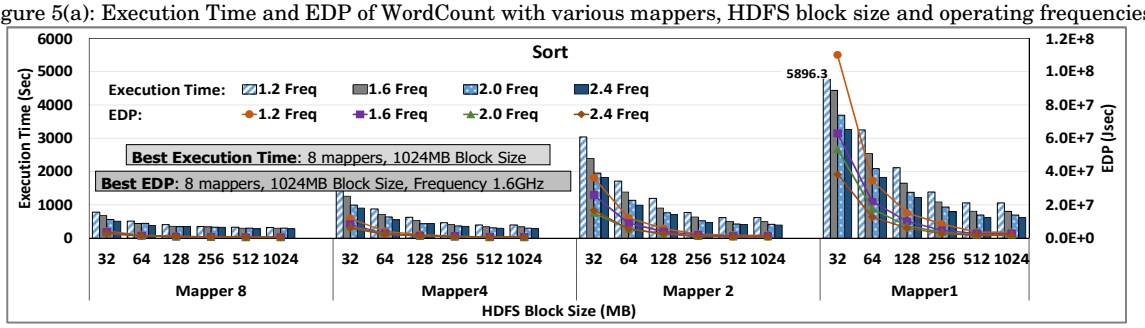

Figure 5(b): Execution Time and EDP of Sort with various mappers, HDFS block size and operating frequencies

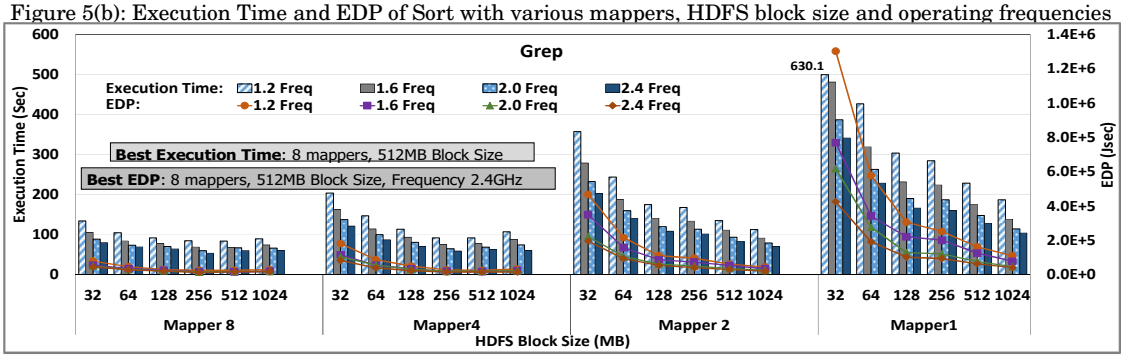

Figure 5(c): Execution Time and EDP of Grep with various mappers, HDFS block size and operating frequencies

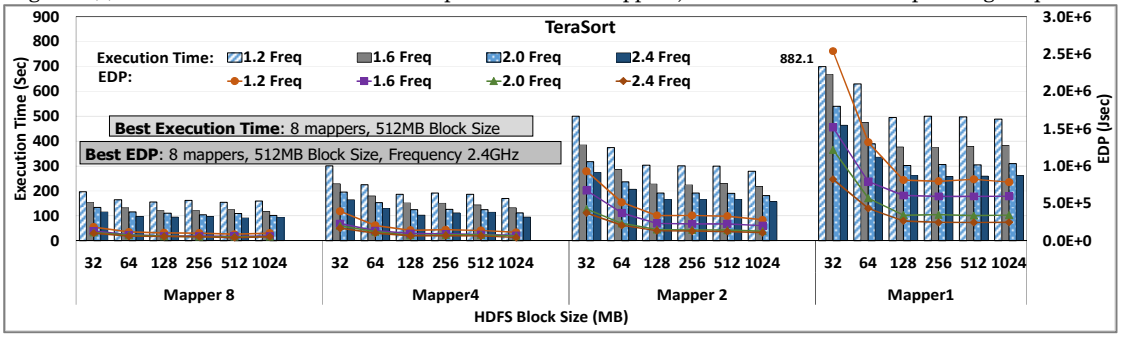

Figure 5(d): Execution Time and EDP of Terasort with various mappers, HDFS block size and operating frequencies

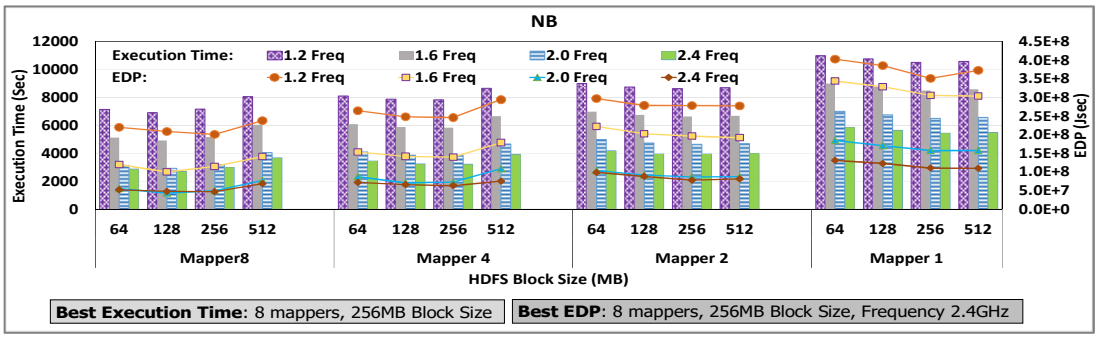

Figure 6(a): Execution Time and EDP of NB with various mappers, HDFS block size and operating frequencies

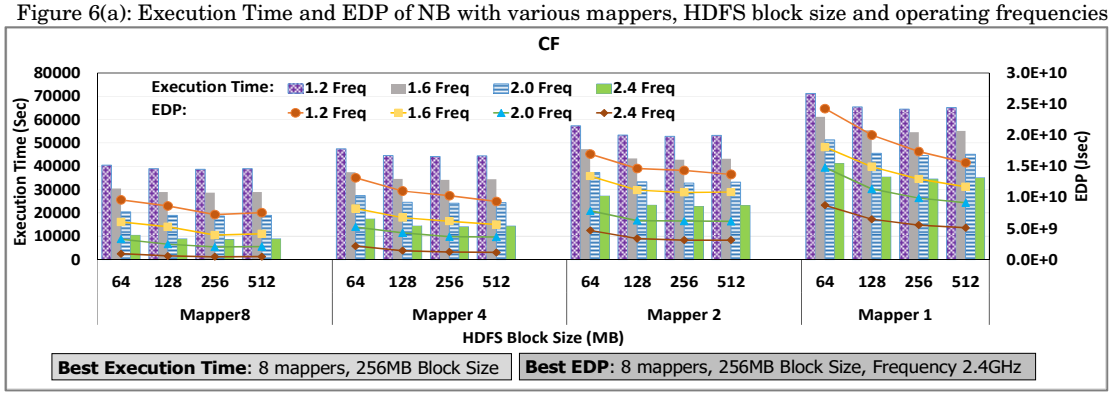

Figure 6(b): Execution Time and EDP of CF with various mappers, HDFS block size and operating frequencies

scaling on the EDP results. Energy efficiency is maximized at the highest frequency of 2.4GHz in all the studied applications with an exception of Sort. Sort operating at a frequency of 1.6GHz provides the maximum energy efficiency as opposed to 2.4GHz frequency. As discussed earlier, Sort is an I/O bound application that spends a significant amount of execution time reading data from and writing to HDFS. This behavior makes the performance of Sort almost insensitive to the operating frequency.

As mentioned earlier the interesting observation is regarding the tuning of the HDFS block size and frequency for various number of mappers. The results show that by simultaneously fine-tuning the HDFS block size and operating frequency, we can reduce the number of mappers and yet be as energy-efficient as with the maximum number of mappers. For example, Grep of 512 MB block size and 2.4 GHz frequency with 2 and 4 mappers achieves higher or similar energy-efficiency compared to the maximum number of mappers. This indicates that in the absence of available cores, for instance due to co-scheduling of other jobs on the server, with fewer mapper we can fine-tune frequency and HDFS block size and still be energy-efficient competitive with more number of cores/mappers.

There are several works that attempt to find which applications should co-schedule simultaneously on a CMP on high performance servers. [27] proposes a methodology to build models that predicts application execution time and energy consumption due to contention in shared cache and memory resources when co-located applications run simultaneously on the same node. This work analyzes the co-location interference effects on execution time and energy dissipation caused by resources shared among the cores in a multicore processor for the HPC application simulated on the high-performance server, Xeon. [28] proposes the energy-aware thread-to-core scheduling policy for heterogeneous multicore processor. This study attempts to spread the shared resources contention uniformly across available cores by predicting the future thread behavior from the study of memory and performance demands of individual threads to maximize the energy efficiency. Bubble-up [29], a characterization and profiling methodology, predicts the performance degradation between pairwise application co-locations.

It is also important to note that most prior research showed promising results by co-scheduling applications, however, using SPEC and HPC applications on the high-performance servers. Our work targets microservers and highlights the fact that Hadoop-based big data applications can also be co-scheduled onto one node by concurrent fine-tuning of frequency and HDFS block size and still remain as energy-efficient as using maximum number of cores.

Figure 6 (line graph) presents the EDP analysis of the real-world applications when utilizing 1, 2, 4 and 8 cores (number of mappers). We have analyzed the effect of the HDFS block size and frequency scaling on the EDP. The results show that the most energy-efficient HDFS block size for compute bound applications - CF and NB is 256MB. The trend for these two real-world applications is similar to what we have already observed in micro-benchmarks where optimal HDFS block size for the compute bound applications is 256MB. Additionally, CF and NB provide the best EDP at the maximum frequency. The margin of EDP improvement becomes smaller with the increase in the HDFS block size at the maximum frequency.

5.3 MapReduce Phase Breakdown Analysis

There are several tasks involved in an end-to-end Hadoop MapReduce environment. The main tasks are map, reduce, shuffle, sort, setup and clean up. The first phase is

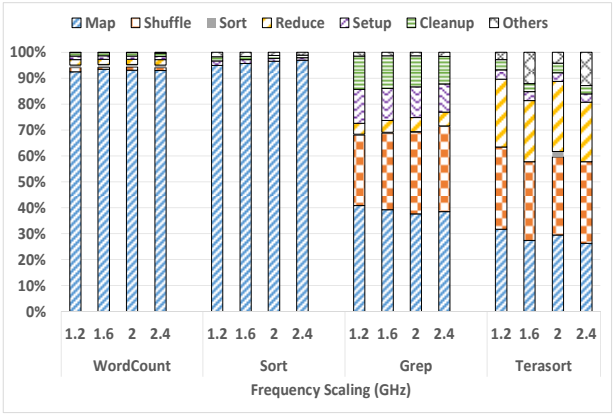

Figure 7: MapReduce normalized execution time breakdown

the map that executes the user defined map tasks on the entire input data. During this phase, the input data is divided into fixed-size blocks called splits and is converted into the <key,value> format. In the second phase, all <key, value> pairs of a particular key are sent to a single reduce task. To do so, shuffling is done to transfer intermediate data from mappers to the reducers. Shuffle phase starts shortly after the first map finishes, and does not complete until all the map tasks are done. Later on, sort phase occurs that sort <key,values> pairs to provide the correct form of mappers to the reducers. Sort phase finishes after the shuffle phase ends. Setup and cleanup are other major phases of big data processing in Hadoop. The setup reads parameters from the configuration object and does all the bookkeeping before the map task execution starts. The setup time of JVM is included in the setup phase of MapReduce application. Map and Reduce phases are the computational intensive portion of the application. The cleanup frees all of the resources that have allocated during execution and flush out any intermediate variable.

In Figure 7, we present the normalized execution break-down of MapReduce phases for the studied micro-benchmarks when we change frequency for 512MB HDFS block size and eight mappers. Note that for Sort benchmark, there is no reduce task. For Grep benchmark, which includes two separate phases (i.e., searching and then sorting the results), the setup and cleanup contribute to a significant portion of execution time.

Phase analysis is essential to profiling and characterizing the application behavior. In Figure 8, we have analyzed the performance of various phases of MapReduce application to analyze the frequency impact on various phases of MapReduce application, while tuning parameters at the application, system, and architecture levels. Our results show that reduce phase of Grep and Map phase of the sort application are less sensitive to the frequency as these phases are I/O intensive in nature. Therefore, running these phases at lower frequencies provides significant opportunity for reducing the power consumption with a negligible performance degradation.

In Figure 8, we illustrate the impact of frequency scaling on each phase of Hadoop MapReduce normalized to its corresponding phase running at minimum frequency, namely 1.2GHz. The trend of decreasing in execution time as the operating CPU frequency increases is consistent with the results in Figure 5. Figure 8 shows that setup and cleanup phase of micro-benchmarks are frequency sensitive. Since the computation intensive part of the micro-benchmarks lies on the map and reduce phase, it is critical to understand how sensitive they are to frequency scaling. Having no reduce phase, Sort application spends most of its execution time in the map phase. Interestingly, map phase in Sort is insensitive to the operating frequency as this phase spends a significant amount of execution time reading data to and from the HDFS. One can execute such phase at a lower frequency to save power. Another observation is regarding Grep reduce phase which shown to be less sensitive to the frequency. This is due to the fact that Grep benchmark consists of two independent steps that are Grep searching and Grep Sorting, the latter step is I/O bound. Consequently, unlike WordCount and TeraSort, the reduce phase of Grep exhibits a different behavior; reducing the CPU frequency by half, from 2.4 GHz to 1.2 GHz only results in an 18% reduction in the execution time, therefore, providing significant opportunity for reducing power consumption.

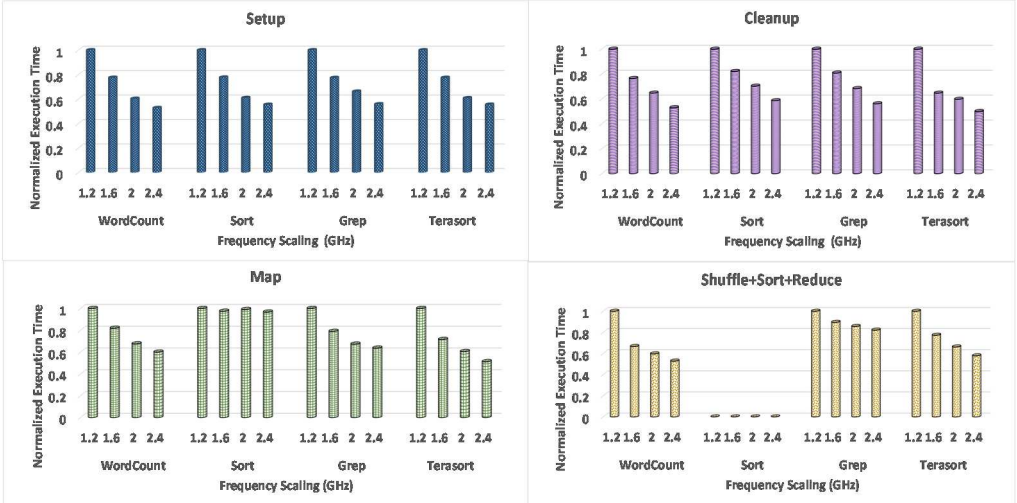

Figure 8: MapReduce Phases normalized execution time to minimum frequency at various frequencies

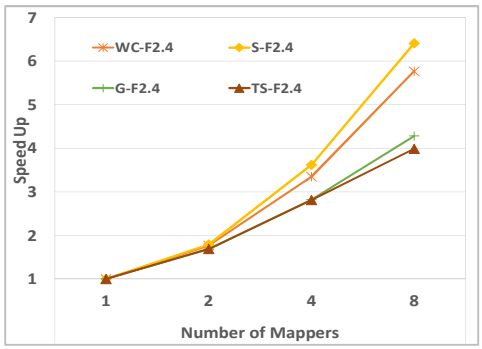

Figure 9(a): Speed up with 32 MB HDFS block size at maximum frequency 2.4GHz

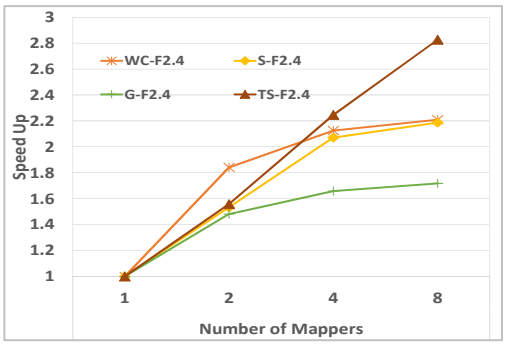

Figure 9(b): Speed up with 512 MB HDFS block size at maximum frequency 2.4GHz

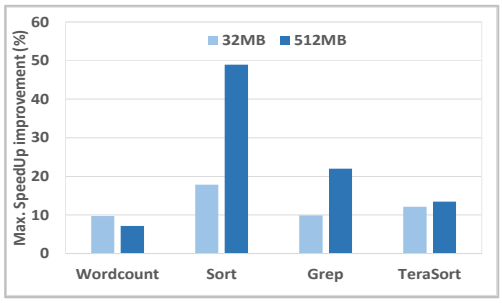

Figure 10: Maximum speed up improvement when cores operates at the minimum frequency normalized to the performance at maximum frequency

5.4 Speedup

In this section, we analyze performance improvement as the number of mappers increase with 32MB and 512MB HDFS block size. The results are presented in Figure 9(a)-(b). All the values are normalized to the execution time of the application with one mapper. At the maximum frequency, the speedup of 6.40, 5.76, 4.28 and 3.98 is achieved for Sort, WordCount, Grep and TeraSort, respectively. As Figure 9(b) shows, the increase in the HDFS block size reduces the speedup gains (2.18, 2.2, 1.71 and 2.82, respectively). It is important to observe that TeraSort benchmark attain more speedup gain at 512MB than 32MB of HDFS block size. In other words, tuning the HDFS block size not only changes the execution time but also affects the speedup gain of big data applications, as the number of mappers' increases.

We also evaluate how the frequency scaling affects the speedup achieved for the studied benchmarks. Figure 10 shows that when the frequency is reduced from 2.4GHz to 1.2GHz, the speedup gain increases. For example, when the frequency reduces to the minimum, the speedup achieved in Figure 9(a) is increased to 7.55, 6.32, 4.70 and 4.47, respectively. The speedup improvement is more when the HDFS block size is larger. In other words, when the application is operating at a lower frequency to save power, we can compensate the performance loss to some extend by increasing the number of mappers. This is the case, in particular for Sort benchmark, as at the minimum frequency the speed up improvement is almost 50% compared to the maximum frequency for 512MB HDFS block size. Consequently, the most efficient configuration for this benchmark is with 8 mappers, 1.6GHz frequency and large HDFS block size as it is shown in Figure 5(b).

5.5 Input Data Size Sensitivity Analysis

In this section, we study the impact of input data size on power and performance. We conduct the data sensitivity analysis of Hadoop applications with the dataset of 10MB, 100MB, 1GB, 10GB, 25GB and 50GB per node. In a distributed framework like Hadoop, the input data is divided into data block and assigned to each node. Although Hadoop exploits cluster-level infrastructure with many nodes for processing big data applications, to understand the impact of various parameters and how their interplay impacts EDP, single node characteristics analysis is required. The number of mappers is fixed at 8 with the default HDFS block size (64MB) and governor is set as on-demand. The results show that the execution time is proportional to the input data size. Power consumption also increases slightly as the size of input data increases. However, the power and performance sensitivity to the input

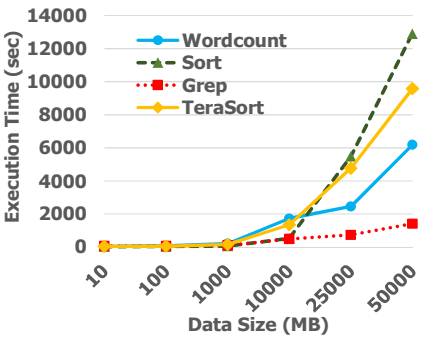

Figure 11(a): Execution time of Hadoop micro-benchmarks with various data sizes

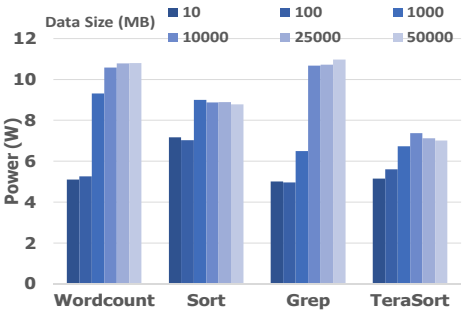

Figure 11(b): Power of Hadoop micro-benchmarks with various data sizes

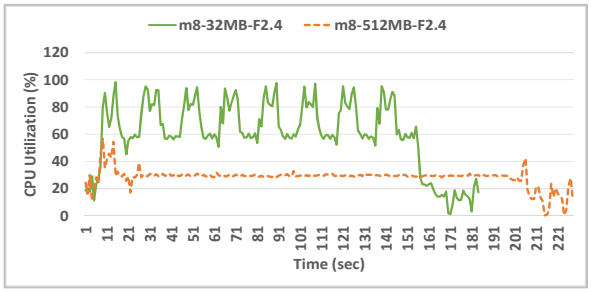

Figure 12(a): CPU utilization trace of WordCount for HDFS block size comparison

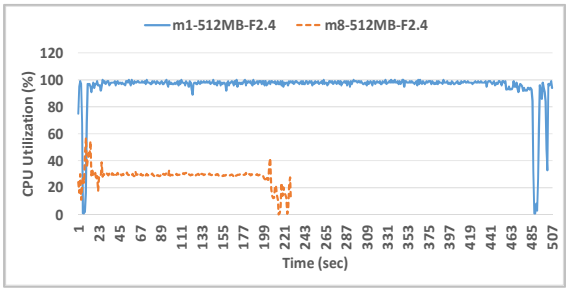

Figure 12(b): CPU utilization trace of Wordcount for number of mappers comparison

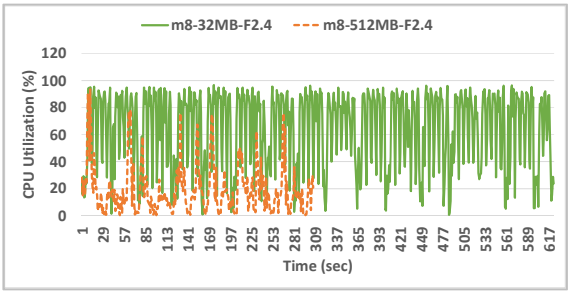

Figure 13(a): CPU utilization trace of Sort for HDFS block size comparison

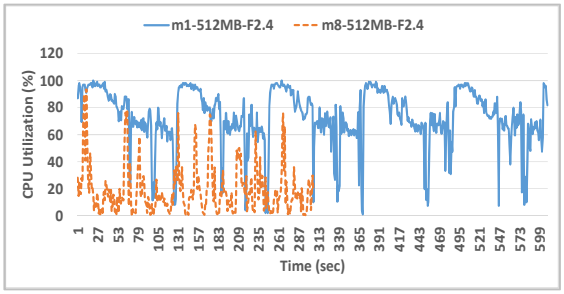

Figure 13(b): CPU utilization trace of Sort for number of mappers comparison

data size is different across various applications. The execution time and power results are shown in Figure 11(a)-(b), respectively. WordCount, which is a compute bound benchmark is less sensitive to the input data size, whereas, Sort execution time is shown to be highly affected by the input data size. With 10MB and 100MB input size there is not a significant variation in power consumption across all bench-marks, however, with larger data sizes the power consumption varies more noticeably and suddenly increases. For Sort, the power consumption is more than other studied benchmarks when the input data size is small (i.e., 10MB and 1000MB). However, when the input size is large (i.e., 25GB and 50GB) the power consumption of WordCount and Grep becomes larger than Sort.

5.6 System Resources Profiling and Utilization

In this section, we present the real time system resources profiling (CPU utilization and memory footprint) to understand the runtime behavior and resource utilizations of Hadoop micro-benchmarks. Real-world applications have not been included in the system resource utilization study, as we have observed that they have similar behavior as compute-bound micro-benchmarks. In this set of experiments, we study the following parameters: number of mappers (1 and 8), HDFS black size (32MB and 512 MB) and operating frequency (1.2GHz and 2.4 GHz).

5.6.1 CPU Utilization Analysis

Table 3 presents CPU utilization of Hadoop micro-benchmarks that include overall CPUuser, CPUidle and CPUiowait utilization. We use the dstat profiling tool that classifies CPU utilization into different types including user, idle, wait, system, hardware interrupt and software interrupt. To evaluate the CPU utilization of an application under test, we have selected user, idle and wait parameters. CPUuser utilization present the amount of time when the core is busy working on the user application and is not idle (CPUidle utilization) or stalled due to I/O (CPUiowait utilization). The CPU utilization trace is generated per second and the reported values are the average utilization of all cores over the total execution time.

CPUuser utilization decreases when the number of mappers increases. In the I/O bound application-Sort-CPU spends most of its execution time waiting for IO request. We have observed a similar trend in Table 3 where Sort has low CPUuser utilization and high CPUiowait readings compared to other applications. In WordCount, with HDFS block size of 32MB, the average CPUuser utilization decreases to 60%. However, with 512MB HDFS block size the utilization reduces to 28%. (CPUidle is 70%). This is mainly due to the fact that large HDFS block size is under-utilizing the number of active cores.

To illustrate the benchmark behavior, we have pres-

Table 3: CPU Utilization (%)

		WC	ST	GP	TS
m1_32MB_F1.2	user	96.29	93.52	93.16	89.92
	Idle	0.30	0.02	1.43	0.08
	iowait	0.04	0.00	0.07	0.56
m1_512MB_F1.2	user	96.54	81.40	88.96	86.17
	Idle	0.57	0.05	3.81	0.85
	iowait	0.02	0.48	0.13	0.70
m8_32MB_F1.2	user	61.38	66.16	56.64	47.23
	Idle	36.70	17.56	40.48	47.56
	iowait	0.18	1.48	0.22	0.81
m8_512MB_F1.2	user	28.12	33.95	23.11	34.21
	Idle	70.80	34.13	74.08	59.55
	iowait	0.37	22.99	0.80	1.44
m1_32MB_F2.4	user	96.21	93.52	92.32	88.81
	Idle	0.51	0.04	2.27	0.38
	iowait	0.08	0	0.08	1.04
m1_512MB_F2.4	user	95.32	73.44	80.22	85.02
	Idle	1.36	0.36	11.05	0.85
	iowait	0.78	6.72	1.03	0.97
m8_32MB_F2.4	user	60.15	63.48	49.97	47.77
	Idle	37.68	25.15	46.24	44.83
	iowait	0.16	6.7	0.98	2.97
m8_512MB_F2.4	user	28.3	19.17	19.53	30.38
	Idle	70.07	44.08	75.49	60.18
	iowait	0.95	31.52	3.09	5.21

ented the timeline based CPUuser utilization of WordCount and Sort benchmark with respect to the number of mappers and the HDFS block size in Figure 12(a)-(b) and Figure 13(a)-(b), respectively. In WordCount, as the HDFS block size increases from 32MB to 512 MB, the traces show a stable CPUuser utilization averaged at 60% and 25% with an exception that 32MB finishes earlier than the 512 MB. In contrast, CPUuser utilization reaches to almost 96% on average with single mapper (see Figure 12(b)). Sort benchmark also shows a similar trend. However, for this benchmark, the execution time with 512MB is less than 32MB HDFS block size. The average CPUuser utilization for Sort with maximum block size is only 19%. Moreover, this benchmark shows large variation in the CPUuser utilization and stays below 15% for a considerable amount of time. The WordCount benchmark with one mapper almost keeps the CPU busy for nearly the entire duration of the application. Given that WordCount extracts a small amount of data from a large set of data, its job output is much smaller than the job input. Consequently, the WordCount is CPU bound having high average CPU utilizations.

Increasing the operating frequency results in a slight reduction in the CPU utilization except for WordCount, which is more compute bound benchmark compared to others. Moreover, changing the block size almost does not change the utilization with a single mapper, however, it leads to considerable CPU utilization reduction when the number of mappers increases. This behavior exhibits that all cores are not actively working most of the time with the largest HDFS block size. The underutilized cores are waiting for I/O, being synchronized with other cores or waiting until the results of other cores produced.

It is noteworthy that when the number of mappers increases, the average CPUuser utilization reduces from 73%-96% to 19%-60% with respect to the HDFS block size. To explain this behavior, we have analyzed the resource stalls introduced at the back-end of processor pipeline using Intel Vtune [30]. The back-end contains record buffer (ROB) and reservation stations (RS). When the ROB or RS becomes full, the back-end stalls and does not accept any new instruction. We have observed that with the increase in the number of mappers, ROB stalls do not change significantly, however, the RS stalls increase from 0.2% to 15%. RS stalls occur when processor is waiting for inputs and resources to be available. This behavior illustrates that with the increase in the number of mappers, shared resources in the memory hierarchy including the cache, shared memory, and DRAM become the bottlenecks that results in a low CPUuser utilization. In other words, most of the time the cores are idle and dissipating leakage power. Such large idle time motivates employing the Dynamic Power Management (DPM) techniques [31] for big data applications when running large number of mappers. The low core utilization indicates a significant potential to mitigate leakage power dissipation.

5.6.2 Main Memory Footprint

In this section, we present the analyses of memory

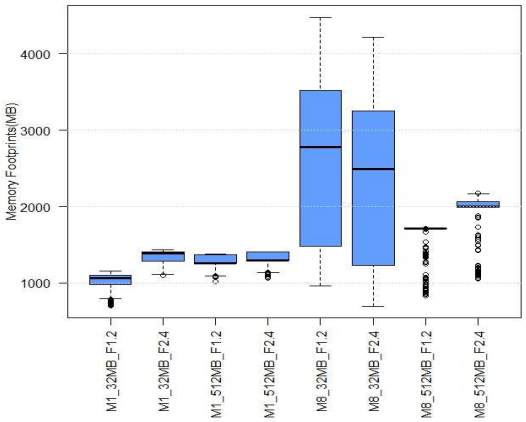

Figure 14(a): Memory Footprints (MB) of WordCount

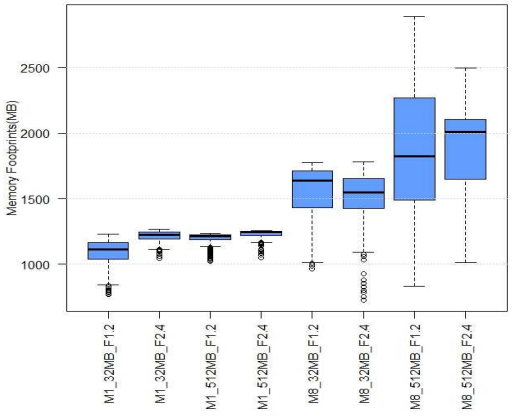

Figure 14(b): Memory Footprints (MB) of Sort

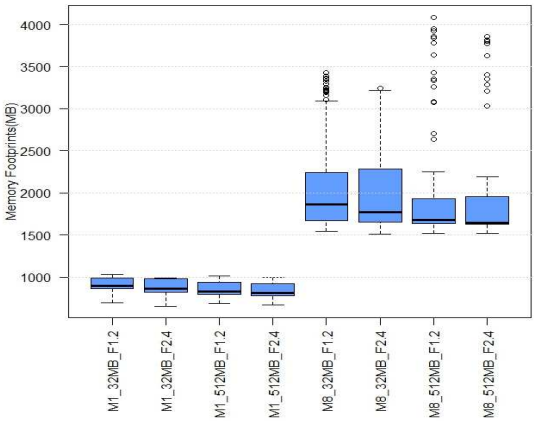

Figure14(c): Memory Footprints (MB) of Grep

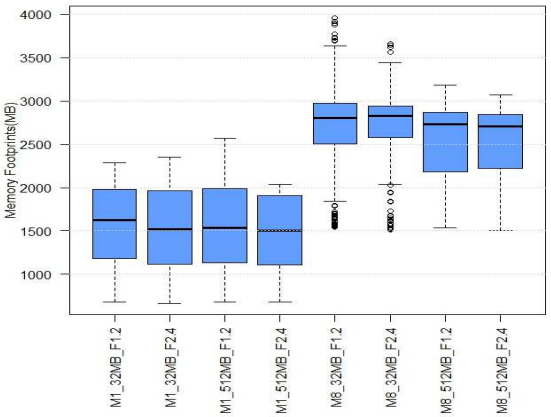

Figure14(d): Memory Footprints (MB) of TeraSort

footprints results. Figure 14(a)-(d) illustrates how much stress the memory (in MB) experiences while running the studied benchmarks. The increase in the number of mappers indicates that multiple cores are processing the benchmark, which eventually put more stress on the memory subsystem. We have observed 19% to 120 % increase in average memory footprint with the increase in the mappers from one to eight. Variation in the memory footprint is minor with changing the frequency. As the HDFS block size varies, minor changes are observed in the average memory footprint for most of the cases with the exception of WordCount.

6. Discussion

In this section, based on the results and discussions through-out the paper, the key findings are presented as follows:

- The speedup obtained when increasing the number of available cores on a microserver node outweighs the power overhead associated with increasing the number of cores, making a configuration that uses the maximum number of available cores per node the most energy-efficient across all studied applications. Unlike microservers, for traditional high performance server the power consumption increase, as the number of mappers' increases, outweighs the performance gains. Therefore, microservers introduces a new trade-offs to process the Big data applications for maximum energy-efficiency.
- Increasing the number of mappers/cores, improves performance and reduces the CPU utilization. In all studied cases using maximum number of cores produces best results in terms of both performance and energy-efficiency. It was also observed that if the number of mappers exceeds available cores, mapper tasks are buffering which potentially reduces the performance and impact the energy-efficiency.
- Although utilizing all available cores on each microserver node provides maximum energy-efficiency across all studied applications, concurrent fine-tuning of frequency and HDFS block size reduces the reliance on the maximum number of cores, and instead make a configuration with fewer number of cores to be energy-efficient competitive with the maximum number of cores. This helps freeing up cores on each node to accommodate scheduling incoming applications in a cloud-computing environment.
- Tuning the block size significantly affects the performance and energy-efficiency of the system. I/O bound Hadoop applications provide the optimal execution time and EDP with the largest HDFS block size. Default HDFS block size of 64MB is not optimal, neither for power nor for the performance.
- The speed up improvement is more when the HDFS block size is larger. I/O bound applications can run at a lower frequency to save power. Performance loss can be compensated to a significant extend by increasing the number of mappers.
- Increasing the number of mappers and the number of active cores result in drastic reduction in average core utilization. In other words, with more number of mappers most of the times the cores are becoming idle and dissipate leakage power. This motivates employing Dynamic Power Management (DPM) techniques [31] for big data applications when running large number of mappers.
- Default Hadoop configuration parameters are not optimal for maximizing the performance and energy-efficiency. With fine tuning the Hadoop parameters along with the system configurations, a significant gain in performance and energy-efficiency can be achieved.

7. Related Work

Recently, there have been a number of efforts to understand the behavior of big data and cloud scale applications by benchmarking and characterizing them, to find out whether state-of-the-art high performance server platforms are suited to process them efficiently. The most prominent big data benchmarks, includes CloudSuite, HiBench, BigDataBench, LinkBench and CloudRank-D which mainly focus on the applications' characterization on high performance servers [17, 18, 19, 20, 32]. CloudSuite benchmark was developed for Scaleout cloud workloads. HiBench is a benchmark suite for Hadoop MapReduce. The BigDataBench was released recently and includes online service and offline analytics for web service applications. LinkBench is a real-world database benchmark for social network application. CloudCmp [33] use a systematic approach to benchmark various components of the cloud to compare cloud providers. These works analyze the application characterization of big data applications on the Hadoop platform, but they do not discuss the Hadoop configuration parameters for energy efficiency.

Many recent works have investigated the energy efficiency in the Hadoop system; Examples are energy-efficient storage for Hadoop [8, 9], energy aware scheduling of MapReduce jobs [34] and GreenHadoop [35]. However, the focus of these works is on the reduction of operating cost of data centers for energy efficiency. Our study is different as it focuses on tuning Hadoop parameters to improve the performance and energy efficiency. The impact of Hadoop configuration parameters is discussed briefly in [17] and [36] but they have not studied the impact of frequency scaling and its interplay on Hadoop specific parameters such as HDFS block size and the number of mappers for optimizing the energy efficiency. [21] has focused on the resource utilization for performance and energy efficiency on Amdahl blades running Hadoop. However, they have studied only two applications with default Hadoop configuration parameters. Our study illustrates that default Hadoop configuration parameters (like HDFS block size of 64 MB) are not optimal for maximizing performance and energy efficiency. In [15], authors analyzed the performance and throughput with the scale-up and scale-out cluster environment to figure out which cluster configuration is suitable for Hadoop Mapreduce jobs. Additionally, they have presented the optimization applied to Hadoop like concurrency, network, memory and reduce-phase optimization on the high performance server; Xeon.

However, this work has not discussed the power and energy efficiency. [22] presents a study of a Hadoop cluster for processing big data on ARM servers. Authors have evaluated three different hardware configurations to understand the limitations and constraints of the cluster. The energy usage and total cost of ownership for MapReduce applications has been analyzed on the Xeon and ARM big.LITTLE architecture in [37]. They have not evaluated the impact of the Hadoop configuration parameters for performance and energy efficiency. HDFS block size is one of the key design parameters and vital to the performance and power optimization. Additionally, this works does not discuss the interplay of system, architectural and applications parameters nor study the resource profiling that is essential to understand the runtime behavior and resource utilization of the Hadoop applications. The work in [38] is the closest to our work as they conduct a study of microserver performance for Hadoop applications. However, their main focus is on the assessment of five different hardware configuration clusters for performance, energy dissipation and cost. In contrast, our work explores Hadoop configuration parameters such as number of mappers, HDFS block size and data input size as well as a system parameter (frequency scaling) for the performance and energy efficiency on microserver.

Our work is different from all above work as it primarily focuses on various Hadoop configuration parameters that directly affect the MapReduce job performance, power and energy efficiency on emerging x86 based low power cores microservers and help to understand the interplay of the Hadoop system, architecture and application parameters to achieve the maximum performance and energy efficiency improvement.

8. Conclusions

In this paper, we present a comprehensive analysis of the impact of Hadoop system configuration parameters, as well as application and architecture level parameters, and the interplay among them on performance and energy-efficiency of various real-world big data applications running on Atom microserver, a recent trend in server design which advocates the use of low-power small cores to address the power and energy-efficiency challenges. We showed that performance and energy efficiency of big data applications are highly sensitive to various Hadoop configuration parameters, as well as system and architecture level parameters, demonstrating that the baseline Hadoop as well as system configurations are not necessarily optimized for a given benchmark and data input size.

Through performance and power measurements and analysis on Atom microserver, first, we showed that increasing the number of mappers that run simultaneously along with increasing the number of active cores help to maximize energy efficiency. Second, our analysis showed that the overall energy efficiency is highly decided by the HDFS block size and is different for each benchmark, demonstrating that the default configuration parameters are not optimal. Third, we have explored the impact of scaling the operating frequency of the compute node for the performance and energy efficiency. Our results show that big data applications become less sensitive to frequency with large number of mappers. Lastly, we conducted the data size sensitivity analysis of Hadoop micro-benchmarks. Results illustrate that the performance and power of compute bound applications are less sensitive to the input data size as compared to I/O bound applications. The results indicate that when not all cores are available, for instance due to co-scheduling of other jobs on the server, with fewer mapper/cores we still can be as energy-efficient and competitive with a case when maximum cores/mappers are available by fine-tuning several parameters such as core frequency and HDFS block size. In addition, the results showed that increasing the number of mappers/active cores result in a noticeable reduction of average CPU utilization, which indicates the potential of using power management techniques when the number of mappers/available cores is at maximum.

We believe that the analyses provided in this work and the trends identified help guiding the scheduling decision to better utilize microserver resources by jointly tuning the application, system and architecture level parameters that influence the performance and energy efficiency.

Acknowledgments

This work is supported by the Nation Science Foundation under grant no. CNS 1526913.

References

[1] A. Venkat, et al., "Harnessing ISA diversity: Design of a heterogeneous-ISA chip multiprocessor." In ISCA, 2014

[2] B. Raghunathan, et al., "Cherry-picking: exploiting process variations in dark-silicon homogeneous chip multi-processors." In DATE, 2013

[3] M. Malik, et. al., "Big Data on Low Power Cores Are Low Power Embedded Processors a good fit for the Big Data Workloads?", in ICCD, 2015

[4] M. B. Taylor, "Is dark silicon useful?: harnessing the four horsemen of the coming dark silicon apocalypse." In DAC, 2012

[5] T. S. Muthukaruppan, et al., "Hierarchical power management for asymmetric multi-core in dark silicon era." In DAC, 2013

[6] Intel, Intel 64 and IA-32 Architecture Software Development Manual, http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html, Aug 2012

[7] P. Boyle, "The bluegene/q supercomputer." PoS LATTICE2012 20

[8] R. T. Kaushik, et al., "Greenhdfs: towards an energy-conserving, storage-efficient, hybrid hadoop compute cluster." In Proceedings of the USENIX annual technical conference, p. 109. 2010

[9] R. T. Kaushik, et al., "Evaluation and analysis of greenhdfs: A self-adaptive, energy-conserving variant of the hadoop distributed file system." In CloudCom, 2010

[10] L. A. Barroso, et al., "The datacenter as a computer: An introduction to the design of warehouse-scale machines." Synthesis lectures on computer architecture 8, 2013

[11] V. Vasudevan, et al., "Energy-efficient cluster computing with FAWN: Workloads and implications." In ICEECN, 2010

[12] X-Gene™, https://www.apm.com/products/data-center/x-gene-family/x-gene/

[13] Moonshot System, http://www8.hp.com/us/en/products/servers/moonshot/

[14] M. Zaharia, et al., "Improving MapReduce Performance in Heterogeneous Environments." In OSDI, 2008.

[15] R. Appuswamy, et al., "Scale-up vs Scale-out for Hadoop: Time to rethink?." In SoCC, 2013
[16] M. Malik, et al., "System and architecture level characterization of big data applications on big and little core server architectures." In Big Data, 2015
[17] C. Luo, et al., "Cloudrank-d: benchmarking and ranking cloud computing systems for data processing applications." Frontiers of Computer Science 6, 2012
[18] M. Ferdman, et al., "Clearing the clouds: a study of emerging scale-out workloads on modern hardware." In ACM SIGPLAN, 2012
[19] S. Huang, et al.,"The HiBench benchmark suite: Characterization of the MapReduce-based data analysis." In ICDEW, 2010
[20] T. G. Armstrong et al., "LinkBench: a database benchmark based on the Facebook social graph." In ACM SIGMOD, 2013
[21] Da Zheng, Alexander Szalay, and Andreas Terzis. "Hadoop in Low-Power Processors." arXiv preprint arXiv:1408.2284 (2014)
[22] C. Kaewkasi, et al., "A study of big data processing constraints on a low-power hadoop cluster." In ICSEC, 2014
[23] Maria Malik, Katayoun Neshatpour, Tinoosh Mohsenin, Avesta Sasan, Houman Homayoun, "Big vs Little Core for Energy-Efficient Hadoop Computing", in DATE, 2016
[24] K. Ousterhout, et al. "Making Sense of Performance in Data Analytics Frameworks." NSDI. Vol. 15. 2015.
[25] E. Blem, et al., "Power struggles: Revisiting the RISC vs. CISC debate on contemporary ARM and x86 architectures." In HPCA, 2013
[26] Dstat http://lintut.com/dstat-linux-monitoring-tools/
[27] D. Dauwe, et al., "HPC node performance and energy modeling with the co-location of applications," The Journal of Supercomputing, 72(12), 2016, pp.4771-4809
[28] R. Nishtala, et al., "Energy-aware thread co-location in heterogeneous multicore processors" In EMSOFT, 2013.
[29] J. Mars, et al., "Bubble-up: Increasing utilization in modern warehouse scale computers via sensible co-locations," In MICRO, 2011
[30] Intel Vtune, https://software.intel.com/en-us/intel-vtune-amplifier-xe?language=de 2016
[31] L. Benini, et al., "A survey of design techniques for system-level dynamic power management." In VLSI, 2000
[32] W. Gao, et al. "Bigdatabench: a big data benchmark suite from web search engines." ASBD 2013 in conjunction with ISCA 2013
[33] A. Li, et al., "CloudCmp: comparing public cloud providers." In ACM SIGCOMM conf. on Internet measurement. ACM, 2010
[34] N. Yigitbasi, et al., "Energy efficient scheduling of mapreduce workloads on heterogeneous clusters." In GCM. ACM, 2011
[35] Í. Goiri, et al., "GreenHadoop: leveraging green energy in data-processing frameworks." Proc. of the 7th ACM EuroSys, 20122
[36] Z. Guo, et al., "Investigation of data locality in mapreduce." In CCGRID, 2012
[37] D. Loghin, et al., "A performance study of big data on small nodes."Proceedings of the VLDB Endowment 8, no. 7 (2015)
[38] A. Anwar, et al., "On the use of microservers in supporting hadoop applications." In CLUSTER, 2014
[39] Perf https://perf.wiki.kernel.org/index.php/Main_Page
[40] WattsUpPro meter https://www.wattsupmeters.com/
[41] M. Malik, et. al., "Characterizing Hadoop Applications on Microservers for Performance and Energy Efficiency Optimizations", in ISPASS, 2016
[42] Apache Mahout: http://mahout.apache.org/
[43] K. Neshatpour, et. al., "Accelerating Machine Learning Kernel in Hadoop Using FPGAs", in CCGRID, 2015

Maria Malik is currently working towards the Ph.D. degree in Electrical and Computer Engineering department, at George Mason University, VA. She has received the M.S. degree in Computer Engineering from the George Washington University, DC and B.E. degree in Computer Engineering from the Center of Advanced Studies in Engineering, Pakistan. Her research interests are in the field of Computer Architecture with the focus of performance characterization and energy optimization of big data applications on the high performance servers and low-power embedded servers, accelerating machine learning kernels, parallel programming languages and parallel computing.

Katayoun Neshatpour is a PhD student at the department of Electrical and Computer Engineering at George Mason University. She is a recipient of the three-year Presidential Fellowship and a 1-year supplemental ECE department scholarship. Advised by Dr. Homayoun and co-advised by Dr. Sasan, her PhD research is on Hardware Acceleration of Big data applications, with a focus on the implementation of several machine learning algorithms in Apache Hadoop and efficient implementation of convolutional neural networks. Katayoun got her Master's degree from Sharif University of Technology, where she worked on the VLSI implementation of a MIMO detector applied to the LTE.

Setareh Rafatirad is an Assistant Professor of the IST department at George Mason University. Prior to joining George Mason, she spent four years as a Research Assistant at UC Irvine. Prior to that, she worked as a software developer on the development of numerous industrial application systems and tools. As a known expert in the field of Data Analytics and Application Design, she has published on a variety of topics related to big data, and served on the panel of scientific boards. Setareh received her PhD degree from the Department of Information and Computer Science at the UC Irvine in 2012. She was the recipient of 3-year UC Irvine CS department chair fellowship. She received her MS degree from the Department of Information and Computer Science at the UC Irvine in 2010.

Houman Homayoun is an Assistant Professor of the ECE department at George Mason University. He also holds a joint appointment with the Computer Science department. Prior to joining GMU, he spent two years at the UC San Diego, as NSF Computing Innovation (CI) Fellow awarded by the CRA and CCC. Houman is currently leading a number of research projects, including the design of heterogeneous architectures for big data and non-volatile logics to enhance design security, which are funded by National Science Foundation (NSF), General Motors Company (GM) and Defense Advanced Research Projects Agency (DARPA). Houman received his PhD degree from the Department of Computer Science at the UC Irvine in 2010, an MS degree in computer engineering in 2005 from University of Victoria, and his BS degree in electrical engineering in 2003 from Sharif University of technology.