Big vs Little Core for Energy-Efficient Hadoop Computing

Maria Malik[1], Katayoun Neshatpour[1], Tinoosh Mohsenin[2], Avesta Sasan[1], Houman Homayoun[1]
[1]Department of Electrical and Computer Engineering, George Mason University,
{mmalik9, kneshatp, asasan, hhomayou}@gmu.edu
[2]Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, {tinoosh}@umbc.edu

Abstract- The rapid growth in the data yields challenges to process data efficiently using current high-performance server architectures such as big Xeon cores. Furthermore, physical design constraints, such as power and density, have become the dominant limiting factor for scaling out servers. Heterogeneous architectures that combine big Xeon cores with little Atom cores have emerged as a promising solution to enhance energy-efficiency by allowing each application to run on an architecture that matches resource needs more closely than a one-size-fits-all architecture. Therefore, the question of whether to map the application to big Xeon or little Atom in heterogeneous server architecture becomes important. In this paper, we characterize Hadoop-based applications and their corresponding MapReduce tasks on big Xeon and little Atom-based server architectures to understand how the choice of big vs little cores is affected by various parameters at application, system and architecture levels and the interplay among these parameters. Furthermore, we have evaluated the operational and the capital cost to understand how performance, power and area constraints for big data analytics affects the choice of big vs little core server as a more cost and energy efficient architecture.

Keywords—Heterogeneous Architectures; Hadoop; Big Data; Energy and Cost Efficiency; Big and Little Cores; Scheduling

I. INTRODUCTION

The steep increase in the volume of data imposes significant limitations on data centers to process Big Data applications using existing hardware solutions. Big Data applications require computing resources and storage subsystems that can scale to manage massive amounts of diverse data. Additionally, Big Data applications have fundamentally different microarchitectural behavior than traditional applications highlighted in recent work [2, 13, 16, 21]. This new set of characteristics necessitates a change in the direction of server-class microarchitecture to improve their computational efficiency [2, 13]. Moreover, physical design constraints (power and area density) have become the dominant limiting factor for scaling out data centers [2-5]. Consequently, current server designs, based on commodity homogeneous processors, are not the most efficient in terms of performance/watt and area to process Big Data applications [5, 8]. All these factors are shifting the hardware design paradigm, in particular for big data and server class architectures, from the performance centric to energy-efficient centric design methodology. A key challenge here is to achieve a favorable trade-off between power, performance and area cost. Therefore, we believe this is the right time to identify the right computing platform for Big Data analytics processing that can provide a balance between processing capacity, cost efficiency, and energy efficiency.

To address the energy-efficiency challenges, heterogeneous architectures have emerged as a promising solution to enhance energy efficiency by allowing each application to run on a core that matches resource needs more closely than a one-size-fits-all core [6, 14, 17]. A heterogeneous chip architecture integrates cores with various micro-architectures (in-order, out-of-order, varying cache and window sizes, etc.) to provide more opportunities for efficient workload mapping in order to explore a better match for the application among various components to improve power efficiency [1].

To explore the choice of server architecture for Big Data, in this paper, we present a comprehensive analysis of the performance and energy-efficiency measurements for Hadoop MapReduce-based applications on two very distinct micro-architectures; Intel Xeon- conventional approach to design a high-performance server and Intel Atom- advocates the use of a low-power core to address the dark silicon challenge facing servers [9]. Moreover, given that Hadoop MapReduce has distinct phases of execution, it is important to understand the characteristics of various phases on big and little core architectures to find out which phase is best suited for which architecture. Thus, we further study how the choice between big and little core changes across various phases of MapReduce tasks. Given that the choice of big vs little core can be impacted by various tuning knobs, in this paper we also study the impact of application, system and architecture parameters and the interplay among them on performance and energy-efficiency and the choice of big vs little cores.

The characterization analysis presented in this paper helps guiding scheduling decision in future cloud-computing environment equipped with heterogeneous server architectures. In such a heterogeneous environment with diverse cores, the scheduling decision needs to be driven not only by user expected performance (delay) but also by energy as well as chip cost. For this reason, we have also performed the Energy-Delay^X Product (ED^XP) analysis -to evaluate the trade-off between power and performance- and ED^XAP -a recently introduced figure of merit for heterogeneous architectures to include the chip area as an indication of cost [11] - to understand how performance, power, and area constraints for Big Data analytics affects the choice of big vs. little core server as a more efficient architecture. ED^XAP metric includes both an operational cost component (energy) as well as a capital cost component (area). We experiment this through a case study demonstrating how scheduling decisions for a heterogeneous architecture combining X and Y number of Xeon and Atom cores can be improved significantly in terms of energy efficiency as well as cost.

This paper makes the following key contributions:

- *Analyzing the performance, energy efficiency and cost efficiency of various phases of MapReduce on big and little cores across a large range of tuning parameters at application (application type), system (HDFS block size) and architecture (operating voltage and frequency of core) levels to find out how the choice between big and little cores is affected by these parameters.*
- *Demonstrating how the characterization results help scheduling decision to improve operational and capital cost in heterogeneous big+little core architectures.*

978-3-9815370-8-6/17/$31.00 ©2017 IEEE

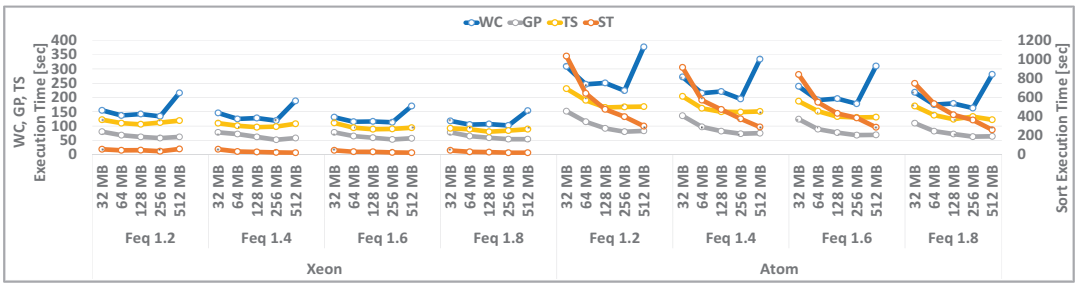

Figure 1: Execution time of Hadoop micro-benchmarks with respect to various HDFS block size and frequency scaling

II. Measurement Tools and Methodology

We conduct our study on two state-of-the-art servers; Intel Xeon and Intel Atom. Intel Xeon E5 enclosed with two Intel E5-2420 processors that include six aggressive processor cores with three-level of the cache hierarchy. Intel Atom C2758 has 8 processor cores and a two-level cache hierarchy. To have a fair comparison between the two architectures, we used same DRAM system of 8GB for the applications under test in both architectures. Additionally, we have set the number of mappers/active-cores on each big and little core to 6 per node. We have studied four micro-benchmarks that are used as kernels in many Big Data applications, namely Wordcount-WC, Sort-ST, Grep-GP and TeraSort-TS in this paper. We have also included real world applications (Naïve Bayes –NB and FP-Growth- FP). All experiments are performed on a 3-nodes Xeon and a 3-nodes Atom server. To calculate the dynamic power of the server, we have used the similar methodology explained in [16] with Wattsup PRO power meter.

III. Experimental Results and Analysis

In this section, we discuss the application (application type), system (HDFS block size) and architecture (operating voltage and frequency of core) tuning parameters and evaluate how these parameters affect the performance, energy-efficiency and the choice of the big vs little cores. We have conducted the HDFS block size sensitivity analysis (32MB, 64MB, 128MB, 256MB, and 512MB) at different operating frequencies (1.2GHz, 1.4GHz, 1.6GHz and 1.8GHz) for Hadoop micro-benchmarks and real world applications at 1GB and 10GB of data per node, respectively.

A. Performance Analysis

In this section, we discuss and analyze the execution time and sensitivity analysis of each benchmark based on the HDFS block size and the core operating frequency.

1) *Application Execution Time*

Figure 1 shows the execution time results. For graph visibility, *Sort* performance results are presented on the secondary axis as compared to the other applications that are on the primary axis. Note that *Sort* benchmark has no reduce phase. The first observation is that as expected, the execution time of all the workloads is expectedly lower on big cores, compared to little cores. Moreover, increasing the frequency and HDFS block size enhances the performance on both architectures. The results for the *Sort* benchmark show that the sensitivity of the execution time, to the HDFS block size and frequency, on Atom is significantly more than Xeon. On Xeon, the execution time is slightly enhanced with increasing size of HDFS block size upto 256MB. Further increase in the HDFS block size has a negligible effect on the execution time. The reason for this behavior is that the large HDFS block size increases the amount of data processed by each task and can result in more I/O operations per task. For example, if map task has to handle more than one spill (spilling occurs when there is not enough memory to fit all the mapper output), more read/write operations will be required to merge the mapper output and dispatch it for the reduce phase. Additionally, large HDFS block size means fewer blocks with long tasks and therefore less parallelism. Moreover, the variation in the execution time with respect to HDFS block size is more significant on Atom. On both Atom and Xeon, increasing the frequency reduces the execution time, as expected. However, the rate of decrease in the execution time is more significant on Atom. In fact, the execution time is proportional to the inverse of IPC and inverse of frequency. Considering the fact that Xeon has a high processing capacity (issue width of 4) and can hide the memory subsystem misses more effectively than Atom, Xeon is less sensitive to memory latency resulting in Atom being more frequency-sensitive than Xeon. This behavior illustrates that Xeon can operate at the lower frequency without significant performance loss. An interesting observation is that with the large HDFS block size, the sensitivity of the execution time to the frequency is reduced as the large HDFS block size increases the I/O read/write operations. Thus, instead of operating the core at a higher frequency, we can operate it at a lower frequency while selecting an HDFS block size that is sufficiently large, which reduces the performance sensitivity to frequency and therefore reduces the power as well.

Terasort, unlike sort, is a hybrid workload. It incorporates the reduce phase and significantly enhances the execution time of the benchmark that results in the reduction of the performance gap between Xeon and Atom. While for *Sort,* big core shows better capability in hiding large cache misses and I/O accesses compared to Atom, in *Terasort*, only a moderate I/O accesses and cache misses occurs. Therefore, *Terasort* does not require the large bandwidth of big core superscalar pipeline to hide these latencies. This explains the smaller performance gap between the two cores for *Terasort.* Moreover, the sensitivity of the execution time on Atom to HDFS block size and frequency is reduced. However, the variations of the execution time with respect to frequency and HDFS block size follow a similar trend as Sort on both machines.

For compute-bound application- Wordcount, the trend is slightly different from I/O intensive application-Sort. The Wordcount execution time decreases with the increasing frequency on both machines, as is the case with Sort. However, while increasing the HDFS block size to 256MB decreases the

execution time, further increase in HDFS block size (e.g. 512MB) increases the execution time significantly. Results show that the performance gap of 2X between Atom and Xeon architecture can be reduced to 1X through fine-tuning of the system (HDFS block size) and architectural parameters (Frequency), allowing higher energy efficiency. Moreover, the performance gap between Xeon and Atom shows to be lower for WordCount (compute-intensive) compared to Sort (I/O-intensive). This can be explained similarly as was discussed for Terasort. The Grep also shows hybrid characteristics and follow the same trend as Terasort. Grep consists of two separate phases; search and sort running in sequence. Compute bound applications do not show performance improvement beyond 256MB, however, considering I/O applications are benefiting from higher CPU processing capacity, we have observed a better performance at the higher block size (512MB) for these applications and in particular on Xeon.

Figure 2 presents the execution time analysis of the real world applications. HDFS block size is one of the key parameters to improve the workload performance. In Hadoop micro-benchmarks, HDFS block size of 32 MB has the highest execution time as a small HDFS block size generates large number of map tasks [number of map tasks = Input data size /HDFS block size] that increases the interaction between master and slave node. Based on this observation, we have considered 64MB the smallest HDFS block size for the real world applications throughout the paper. Results are consistent with the micro-benchmarks, illustrating that default HDFS block size (64MB) is not optimal to achieve the maximum performance improvement. The HDFS block sizes of upto 256MB reduce the execution time. However, further increase in HDFS has a negligible effect on the execution time. Considering NB and FP are both compute-intensive applications, 256MB is the optimal choice to achieve the maximum performance.

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 block size can be avoided by considering 256MB block size for the compute bound and 512MB for other applications as an optimal choice for performance.

2) Sensitivity Analysis

Overall, the results show that while for I/O intensive MapReduce applications Xeon has a clear performance advantage, the gap between Xeon and Atom reduces significantly for compute-intensive applications. Moreover, the results suggest that Atom is significantly more sensitive to frequency and HDFS block size. Therefore, the performance gap between Atom and Xeon architecture can be reduced significantly through fine-tuning of the system and architectural parameters on Atom, allowing maximum energy efficiency, as will be discussed later. Also, the results suggest that the optimal HDFS block size for the maximum performance is closely decided by the application type and fine tuning this parameter reduces the dependence on the highest operating frequency.

B. Energy-Efficiency Analysis

In this section, we analyze the energy-delay-product (EDP) of the studied applications while changing the frequency. Figure 3 and Figure 4 show the EDP results on Atom and Xeon for the entire application. Figure 5 and Figure 6 present the map and

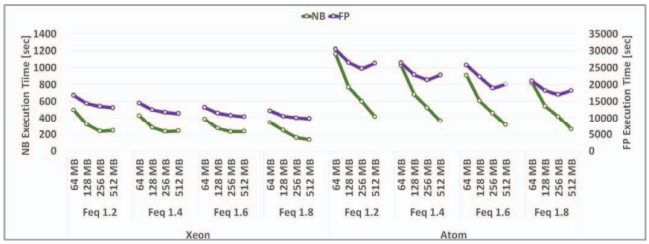

Figure 2: Execution time [sec] of real world applications with respect to various HDFS block size and frequency scaling

reduce phases of all the studied applications. In order to make fair comparisons, for each workload, the EDP values are normalized to the EDP on Atom at the lowest frequency of 1.2GHz and with 512MB HDFS block size.

1) EDP of the entire application

The major observation for the EDP is that for most applications, the low power characteristics of the Atom results in a lower EDP on Atom compared to Xeon, with the exception of the *Sort* benchmark. This is due to the fact that, the performance gap (in terms of execution time) for the I/O intensive benchmark is very large between Atom and Xeon. Since the EDP is a function of the execution time and the power, the total EDP on Xeon is lower for the sort benchmark. Moreover, Figure5 and Figure 6 also show that across all studied applications, the 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.

2) Map Reduce phase analysis

The results show that map phase follows similar trend as the entire application in terms of EDP; as frequency increases, the EDP for map phase reduces. Also, the most energy-efficient core is Atom for the map phase. However, for the reduce phase, a different trend is observed. Increasing the frequency does not always reduce the EDP. For instance, for NB and GP an opposite

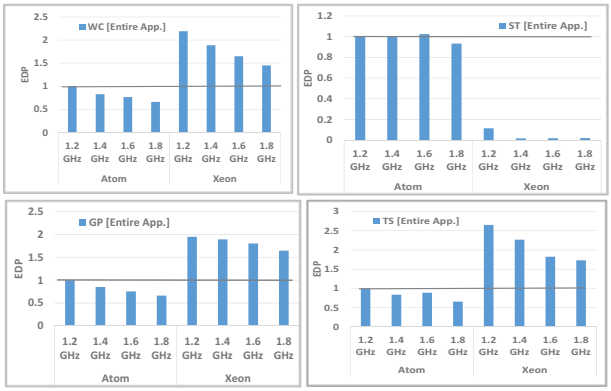

Figure 3: EDP analysis of entire Hadoop micro-benchmark (a) WordCount (b) Sort (c) Grep (d) Terasort on big and little core with frequency scaling

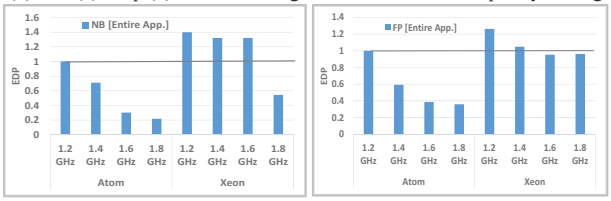

Figure 4: EDP analysis of entire Real World Application (a) NB (b) FP on big and little core with frequency scaling

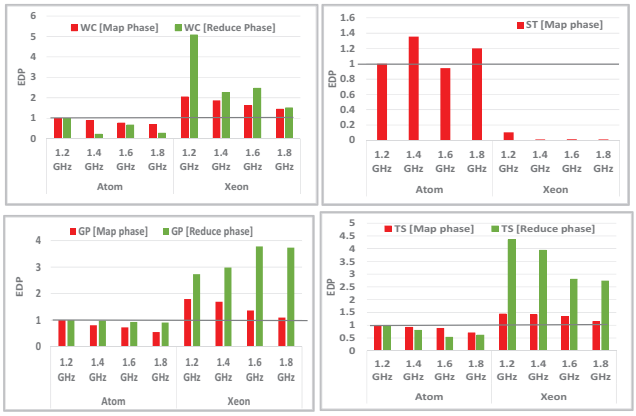

Figure 5: EDP analysis of Map and Reduce phase of (a) WordCount (b) Sort (c) Grep (d) Terasort on big and little core with frequency scaling

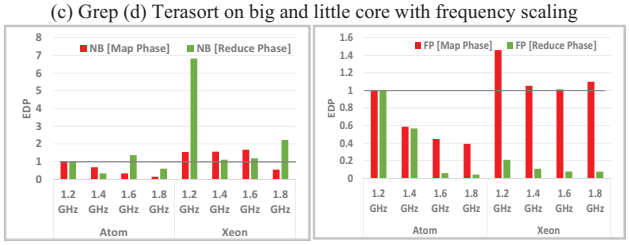

Figure 6: EDP analysis of Map and Reduce phase of Real World applications (a) NB (b) FP on big and little core with frequency scaling

trend is observed. This is mainly due to the fact that reduce phase, unlike map phase is memory intensive as it requires significant communication with memory subsystem. Also comparing Atom and Xeon running at the same frequency, while map phase prefers Atom almost all applications, reduce phase prefers Xeon in several cases; examples are NB and GP.

3) Sensitivity Analysis

We carry out a sensitivity analysis of the EDP ratio of the applications on Xeon to Atom. The motivation is to compare the EDP gap between Xeon and Atom for various tuning parameters. Figure 7 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 increases 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 compared to Xeon, and reduces the performance gap between the two architectures.

C. Scheduling

In the previous sections, we analyzed the execution time and the energy-efficiency of MapReduce benchmarks across a wide range of application, system and architecture levels parameters on Xeon and Atom cores. These analyses will help guide the scheduling optimization decisions in a heterogeneous architecture, as we will show, through several case studies. Assume that we have X number of Xeon cores and Y available number of Atom cores available for scheduling. While from the user perspective, improving performance and getting the MapReduce jobs done faster is the goal which is mainly accomplished by allocating the maximum number of available big Xeon cores to the task, from the cloud computing provider perspective, the choice of X and Y is influenced by not only the performance but also the cost including the operational cost as well as the capital cost. Operational cost is proportional to the energy and the capital cost is proportional to the chip area of the core. In that perspective, Atom cores are clearly a preferred choice. Assuming Atom and Xeon architectures with 2, 4, 6 and 8 cores, we analyze the energy-delay product (ED^XP) and energy-delay-area product (ED^XAP), which indicates the cost [11] to understand the interaction between energy, cost and performance characteristics of the studied applications. The objective of this analysis is to select a right number of Xeon or Atom cores that minimizes various costs, including cost driven by the area as well as the energy. Based on the analysis results presented in the previous sections we set HDFS block size at 512MB and operating frequency at the 1.8GHz.

Table 1 shows the results for the operational and capital cost of the Hadoop micro-benchmarks and real world applications with various number of Xeon and Atom cores/mappers. The number of the mappers is set to be equal to the number of cores (M). It should be noted that EDP is a function of both the execution time and power. Adding more cores to the architecture lowers the execution time, but increases the power consumption. From Table 1 results we observe that in most cases, increasing the number of cores, enhances the energy efficiency. However, in *Grep* and *Terasort* benchmarks, the lowest EDP on Atom is achieved with 6 cores. Moreover, the variations in the EDP with respect to the number of cores is more significant on Atom. Results show that EDP can be reduced by upto 5X (in sort benchmark) by utilizing a maximum number of Atom cores compared to using minimum two, yielding both higher performance and energy-efficiency.

As mentioned earlier, the capital cost of the architecture is another major cost function that affects the scheduling decisions. In order to take the capital cost into account, we study ED^XAP values, presented in Table 1. Hadoop micro-benchmarks show that while increasing the number of cores reduces the EDP, it increases the EDAP. Thus, based on the optimization goals, capital cost constraints may prompt the scheduler to use fewer number of cores. However, for the real world applications, we have observed a different trend where increasing the number of cores is reducing the EDAP. One reason is that real world applications have significantly higher execution time and power consumption compared to the Hadoop micro-benchmarks. Therefore, the performance improvement achieved by introducing more cores for the CPU-bound real world applications are more significant that even results in lowering EDAP as well.

While the scheduling decision to maximize performance and satisfy user expectation attempts to maximize the number of available cores, the scheduling decision attempts to reduce the number of cores for cost efficiency as well as energy-efficiency as it is preferred by the cloud provider. To find the middle

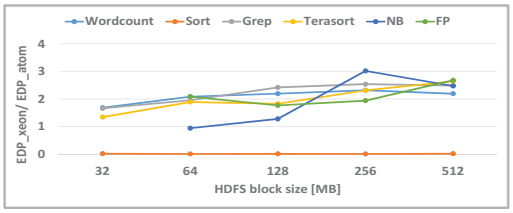

Figure 7: EDP of Hadoop benchmarks at various HDFS block size

ground between the user and cloud provider expectations, we have performed operational and cost analysis of studied benchmark normalized to the maximum number of Xeon cores (i.e. 8 cores), presented in Figure 8 (Spider graphs). Each corner of the spider graphs illustrates the operational and capital cost metrics including energy-efficiency (EDP), near real-time energy efficiency (ED^2P), cost energy efficiency (EDAP) and near real-time cost energy efficiency (ED^2AP). The spider graph is divided into two regions (labeled by 8X equal to 1), an inner region and outer region. Inner region illustrates that little core is preferable to execute the MapReduce job, however, the outer region favors the big core. The results close to the origin represent the maximum energy and cost efficiency.

For the energy efficiency (EDP) results, the comparison between Atom and Xeon shows that the Atom cores are more energy-efficient than the Xeon cores for all the studied applications with the exception of *Sort* as it has the higher performance gap between Atom and Xeon that results in lower EDP on Xeon. Additionally, we have observed that due to the high power consumption of Xeon core, even utilizing the maximum number of Atom cores (i.e. 8 cores) achieves lower EDP compared to the Xeon architecture with the 2 cores.

For the near real-time energy efficiency (ED^2P), a large number of Xeon cores (4 and more) outperform small number of Atoms cores. This is due to the fact that real-time energy efficiency gap gradually decreases with a large number of Xeon cores. While for ED^2P a large number of Xeon cores (4 and more) outperform the small number of Xeon cores, similar to the EDP, the minimum ED^2P is achieved with a large number of Atom cores (6 and 8 cores). A comparison of the cost energy efficiency (EDAP) value between Atom and Xeon shows that EDAP is lower on Atom for most applications including Hadoop micro-benchmarks and real world applications, for a certain number of cores. Additionally, we have observed that Atom with a smaller number of cores provides maximum cost energy efficiency as compared to the Xeon cores. For sort application, the EDAP on Xeon cores is lower than Atom, due to the high performance gap between Atom and Xeon.

The real-time cost energy efficiency (ED^2AP) analysis illustrates that for the *Terasort* and *Grep* applications, the Xeon architecture with 2 cores yields lower ED^2AP than the Atom architecture with 8 cores. This is an interesting observation, which allows us to use Xeon architecture with a small number of cores, rather than running the job on many Atom cores. In this case, we are able to benefit from the high performance Xeon core, yielding low ED^xAP costs. However, for the real world applications, higher computation power is required to process the applications that can be achieved with large number of Atom or Xeon cores. Considering the power consumption of Xeon, Atom cores will be a more efficient choice to execute the Hadoop applications. Additionally, through the comprehensive system and architecture analysis shown in section III-A, we have illustrated that the reliance on the utilization of the maximum number of cores for Atom architecture can be reduced by fine- tuning the system, applications and architectural parameters.

In general, our results illustrate that in cloud computing infrastructure equipped with heterogeneous architectures, the minimum operational and capital cost can be achieved for compute intensive Hadoop applications by scheduling them to large number of little cores while still satisfying user expected performance comparable to what can be achieved on big cores. The reliance on large number of little cores can be reduced significantly by fine-tuning the application, system and architecture level parameters. For I/O bound applications, Xeon core still shows to be the favorite choice for energy as well as cost efficiency.

3. 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 platform is suited to process them efficiently. The most prominent big data benchmarks, including CloudSuite, HiBench, BigDataBench, LinkBench, and CloudRank-D focus on the applications' characterization [2, 12, 13, 15]. These works analyze the application characterization of big data applications on the Hadoop platform, but they do not discuss the implication of this new set of applications on the choice of big vs little core architectures.

Many recent works have investigated the energy efficiency in the Hadoop system including energy-efficient storage for Hadoop [17-18], energy-aware scheduling of MapReduce jobs [19] and GreenHadoop [20]. Additionally, the impact of Hadoop configuration parameters has been discussed in [20]. But these works have not studied the impact of frequency scaling and its interplay on Hadoop specific parameters for optimizing the energy efficiency and the impact on the choice of big vs little core. ARIA [7] is an analytical model that utilizes the knowledge of the map and reduce task completion time as a function of the allocated resources. However, this study lacks the power and energy analysis on the low-power embedded server with various system and architecture parameters. The work in [3] 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 and energy consumption. In contrast, our work explores Hadoop configuration parameters and system parameters for the performance and energy efficiency, as well as cost efficiency of Hadoop applications in a heterogeneous architecture and the choice between big and little cores.

4. Conclusion

This paper answers the important question of whether big

Table 1: Operational and Capital Cost Analysis of Hadoop applications

		Atom				Xeon			
		M2	M4	M6	M8	M2	M4	M6	M8
EDP (J.sec)	WC	4.20E+05	3.37E+05	3.06E+05	3.05E+05	1.52E+06	6.66E+05	6.70E+05	6.50E+05
	ST	1.05E+06	6.64E+05	5.66E+05	3.40E+05	1.38E+04	8.94E+03	8.78E+03	1.31E+04
	GP	2.55E+04	1.71E+04	1.53E+04	1.85E+04	4.06E+04	3.96E+04	3.80E+04	3.94E+04
	TS	1.83E+05	1.05E+05	7.35E+04	7.71E+04	2.43E+05	2.07E+05	1.94E+05	2.04E+05
	NB	2.64E+06	9.23E+05	5.74E+05	5.51E+05	3.77E+06	9.97E+05	5.80E+05	5.26E+05
	FP	9.53E+09	3.28E+09	2.45E+09	1.77E+09	2.07E+10	5.44E+09	3.21E+09	2.74E+09
ED2P (J.sec2)	WC	1.41E+08	9.60E+07	8.59E+07	8.52E+07	4.56E+08	1.04E+08	1.03E+08	9.56E+07
	ST	5.00E+08	2.40E+08	1.79E+08	8.83E+07	3.58E+05	1.97E+05	1.76E+05	3.40E+05
	GP	2.12E+06	1.13E+06	9.77E+05	1.11E+06	2.27E+06	2.14E+06	2.05E+06	2.05E+06
	TS	3.93E+07	1.63E+07	8.96E+06	9.25E+06	2.47E+07	1.90E+07	1.73E+07	1.77E+07
	NB	2.23E+09	3.81E+08	1.49E+08	1.41E+08	1.32E+09	1.76E+08	7.60E+07	6.54E+07
	FP	4.49E+14	8.20E+13	4.57E+13	2.60E+13	5.33E+14	7.03E+13	3.13E+13	2.47E+13
EDAP (J.mm2.sec)	WC	1.34E+08	2.16E+08	2.94E+08	3.91E+08	6.56E+08	5.75E+08	8.68E+08	1.12E+09
	ST	3.38E+08	4.25E+08	5.44E+08	4.35E+08	5.95E+06	7.73E+06	1.14E+07	2.26E+07
	GP	8.16E+06	1.09E+07	1.47E+07	2.37E+07	1.75E+07	3.42E+07	4.92E+07	6.80E+07
	TS	5.84E+07	6.75E+07	7.05E+07	9.87E+07	1.05E+08	1.79E+08	2.51E+08	3.52E+08
	NB	8.46E+08	5.90E+08	5.51E+08	7.05E+08	1.63E+09	8.62E+08	7.52E+08	9.09E+08
	FP	3.05E+12	2.10E+12	2.36E+12	2.27E+12	8.96E+12	4.70E+12	4.16E+12	4.74E+12
ED2AP (J.mm2.sec2)	WC	4.53E+10	6.14E+10	8.25E+10	1.09E+11	1.97E+11	8.98E+10	1.34E+11	1.65E+11
	ST	1.60E+11	1.53E+11	1.72E+11	1.13E+11	1.55E+08	1.70E+08	2.27E+08	5.88E+08
	GP	6.77E+08	7.20E+08	9.38E+08	1.42E+09	9.82E+08	1.85E+09	2.66E+09	3.54E+09
	TS	1.26E+10	1.05E+10	8.60E+09	1.18E+10	1.07E+10	1.65E+10	2.24E+10	3.06E+10
	NB	7.14E+11	2.44E+11	1.43E+11	1.80E+11	5.72E+11	1.52E+11	9.85E+10	1.13E+11
	FP	1.44E+17	5.24E+16	4.39E+16	3.33E+16	2.30E+17	6.07E+16	4.06E+16	4.27E+16

core or little core is more energy and cost efficient to process Hadoop applications. To answer this question it is important to take into consideration tuning parameters at application, system and architecture levels as they influence performance, energy efficiency and cost efficiency of Hadoop applications. Based on real system experimental results, we have observed that for I/O intensive Hadoop applications, Xeon has a clear performance advantage, however, the gap between Xeon and Atom reduces significantly for compute intensive Hadoop applications. Also, Atom has shown to be significantly more sensitive to tuning parameters such as frequency and HDFS block size. 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. Furthermore, for the map phase, compute intensive benchmarks clearly favor the Atom for energy-efficiency, while I/O intensive favors Xeon. For the reduce phase, Atom is the favorite choice across all studied applications.

In addition, we also analyzed the operational and capital cost estimation, which helps guiding scheduling decisions in cloud environment equipped with heterogeneous architectures to find out which of big or little core is the more cost-efficient. For compute intensive applications, we found that the minimum operational and capital cost can be achieved by scheduling to a large number of Atom cores while still satisfying user expected performance comparable to the performance that can be achieved on few Xeon cores. The reliance on a large number of Atom cores can be reduced significantly by fine-tuning the application, system and architecture level parameters. For I/O intensive applications, Xeon still shows to be the favorite choice for energy as well as cost efficiency.

5. Acknowledgments

This work was supported in parts by the National Science Foundation under CSR-1526913 grant.

References

[1] YU.P., et al. “Scalable custom instruction identification for instruction set extensible processor,: in proc of CASES 2014”
[2] Ferdman, Michael, et al. "Clearing the clouds: a study of emerging scale-out workloads on modern hardware." ACM SIGPLAN 2012.
[3] Anwar, Ayesha, K. R. Krish, and Ali R. Butt. "On the use of microservers in supporting hadoop applications." In CLUSTER, 2014
[4] Kontorinis, V., et al., “Managing distributed UPS energy for effective power capping in data centers,” In 39th ISCA 2012.
[5] Reddi, et al, “Web search using mobile cores: qualifying and mitigating the price of efficiency”, in SIGARCH CAN 2010.
[6] K. Neshatpour, et al. "Energy-efficient acceleration of big data analytics applications using fpgas." In Big Data 2015
[7] Verma, et. al., “ARIA: Automatic Resource Inference and Allocation for MapReduce Environments,” in ICAC 2011.
[8] 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.
[9] Hardavellas, Nikos, et al. "Toward dark silicon in servers." IEEE Micro 31.EPFL-ARTICLE-168285 (2011): 6-15.
[10] Kontorinis, V., et al., “Enabling Dynamic Heterogeneity Through Core-on-Core Stacking,” in proc. of DAC 2014
[11] Li, Sheng, et al. "McPAT: an integrated power, area, and timing modeling framework for multicore and many core architectures." In. MICRO, 2009
[12] Huang, S., et al. "The HiBench benchmark suite: Characterization of the MapReduce-based data analysis," In the proc. of 26th ICDEW, 2010
[13] Gao,W. et al.,“BigDataBench: a Big Data Benchmark Suite from Web Search Engines,”.ASBD 2013 in conjunction with ISCA 2013
[14] M.K. Tavana, et al.,”ElasticCore: enabling dynamic heterogeneity with joint core and voltage/frequency scaling“ in DAC 2015
[15] Li, Ang, et al. "CloudCmp: comparing public cloud providers." Proc. of the 10th ACM SIGCOMM conf. on Internet measurement. ACM, 2010.
[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] Homayoun, H., et al., “Dynamically heterogeneous cores through 3D resource pooling,” In HPCA 2012.
[18] Krish, K. R., Ali Anwar, and Ali R. Butt. "[phi] Sched: A Heterogeneity-Aware Hadoop Workflow Scheduler." In MASCOTS, 2014.
[19] Yigitbasi, Nezih, et al. "Energy efficient scheduling of mapreduce workloads on heterogeneous clusters." In GCM. ACM, 2011.
[20] Goiri, Íñigo, et al. "GreenHadoop: leveraging green energy in data-processing frameworks." Proc. of the 7th ACM EuroSys, 2012.
[21] M. Malik, et al., "Characterizing Hadoop applications on microservers for performance and energy efficiency optimizations," in (ISPASS) 2016.

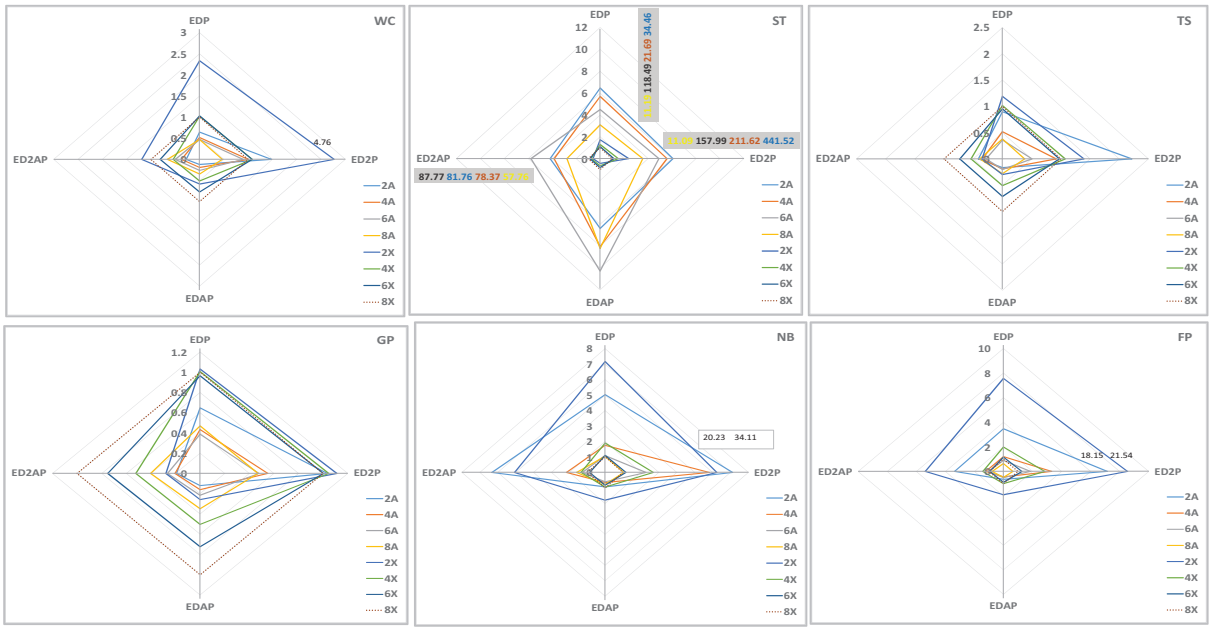

Figure 8: Results presented are for energy-efficiency (EDP), real time energy efficiency (ED^{2}P), cost energy efficiency (EDAP) and real time cost energy efficiency (ED^{2}AP) of Hadoop applications normalized to the 8 Xeon core