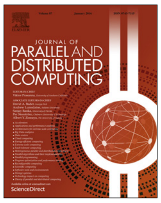
Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc



Big vs little core for energy-efficient Hadoop computing

Maria Malik [a,*], Katayoun Neshatpour [a], Setareh Rafatirad [b], Rajiv V. Joshi [e], Tinoosh Mohsenin [c], Hassan Ghasemzadeh [d], Houman Homayoun [a]

[a] *Department of Electrical and Computer Engineering, George Mason University, United States*
[b] *Department of Information Sciences and Technology, George Mason University, United States*
[c] *Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, United States*
[d] *Department of Computer Science, Washington State University, United States*
[e] *IBM T.J. Watson Research Center, Yorktown Heights, NY, United States*

HIGHLIGHTS

- Xeon has a clear performance advantage for the I/O intensive Hadoop applications.
- For map phase, energy efficient core is closely decided by the application type.
- For the reduce phase, Atom is the favorite choice across all studied applications.
- The reliance on a large number of Atom cores can be reduced significantly by fine-tuning the configuration parameters.
- Minimum operational and capital cost can be achieved by scheduling a large number of Atom cores.

ARTICLE INFO

Article history:
Received 9 May 2017
Received in revised form 11 February 2018
Accepted 20 February 2018
Available online xxxx

Keywords:
Heterogeneous architectures
Hadoop
MapReduce
Energy and cost efficiency
Big and little cores
Scheduling

ABSTRACT

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. 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, through a comprehensive system level analysis, we first characterize Hadoop-based MapReduce applications 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. Second, we study how the choice between big and little core changes across various phases of MapReduce tasks. Furthermore, we show how the choice of most efficient core for a particular MapReduce phase changes in the presence of accelerators. The characterization analysis helps guiding scheduling decisions in future cloud-computing environment equipped with heterogeneous multicore architectures and accelerators. We have also evaluated the operational and the capital cost to understand how performance, power and area constraints for big data analytics affect the choice of big vs little core server as a more cost and energy efficient architecture.

0. 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

* Correspondence to: George Mason University, 4400 University Dr., Fairfax, VA 22030, United States.

E-mail addresses: mmalik9@gmu.edu (M. Malik), kneshatp@gmu.edu (K. Neshatpour), srafatir@gmu.edu (S. Rafatirad), rvjoshi@us.ibm.com (R.V. Joshi), tinoosh@umbc.edu (T. Mohsenin), hassan@eecs.wsu.edu (H. Ghasemzadeh), hhomayou@gmu.edu (H. Homayoun).

https://doi.org/10.1016/j.jpdc.2018.02.017
0743-7315/© 2018 Elsevier Inc. All rights reserved.

work [13,29,30,37]. This new set of characteristics necessitates a change in the direction of server-class microarchitecture to improve their computational efficiency [13,37]. Moreover, physical design constraints (power and area density) have become the dominant limiting factor for scaling out data centers [2,13,33,35]. 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 [27,33]. 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 [7,10,34]. 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 [32].

To explore the choice of server architecture for Hadoop applications, in a recent paper [28], 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 [14]. 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 (application type and data size), system (HDFS block size) and architecture (operating voltage and frequency of core) parameters and the interplay among them on performance and energy-efficiency and the choice of big vs little cores.

As chips are hitting power limits, computing systems are moving away from general-purpose designs and toward greater specialization. Hardware acceleration through specialization has received renewed interest in recent years, mainly due to the dark silicon challenge. To find out the right architecture for Hadoop applications processing, it is important to understand how deploying an accelerator, such as FPGA, would necessitate adapting the choice of big vs. little cores, referred as the post acceleration code characteristics. For this purpose, we analyze the choice of big vs little core-based servers for the code that remains on the CPU after assuming the hotspot phases, e.g. map phase, are offloaded to an accelerator. Overall, our characterization and analysis across Hadoop-based applications demonstrate how the choice of big vs little core-based servers for energy-efficiency is significantly influenced by the type of application, size of data, performance constraints, and presence of accelerator and the breakdown of the execution time across phases of MapReduce.

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^{X}P) analysis — to evaluate the trade-off between power and performance- and ED^{X}AP — a recently introduced figure of merit for heterogeneous architectures to include the chip area as an indication of cost [21] — 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^{X}AP 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:

- Analyze 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.
- Evaluate how offloading the hotspot map tasks to an accelerator such as FPGA affects the choice of big vs little core-based servers to process the code that remains on the CPU for energy-efficient processing.
- Analyze the impact of input data size for performance and energy-efficiency of MapReduce application on big and little cores.
- Demonstrate how the characterization results help scheduling decision to improve operational and capital cost in heterogeneous big+little core architectures.

1. Experimental setup

1.1. 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. Table 1 summaries the architectural parameters of Atom and Xeon servers for the applications under test. To have a fair comparison between the two architectures, we used same DRAM system of 8 GB for the applications under test in both architectures. All experiments are performed on a 3-nodes Xeon and a 3-nodes Atom server. We have used Watts up PRO power meter [3] to measure the power consumption of the servers. Wattsup power meter produces the power consumption profile every one second. 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 dynamic power dissipation of the entire system. The same methodology is used in [17], for power analyses.

1.2. Operational cost and capital cost metric

We have performed the Energy-Delay^X Product (ED^{X}P) analysis — to evaluate the trade-off between power and performance- and ED^{X}AP — a recently introduced figure of merit for heterogeneous architectures to include the chip area as an indication of cost [21] — 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. We have consulted Intel data sheets to get the on-chip area reading for the studied big and little core i.e Atom 160 mm^2 and Xeon 216 mm^2.

Table 1
Architectural parameters.

Processor	Intel Atom C2758	Intel Xeon E5-2420
Operating frequency	1.8 GHz	1.8 GHz
Micro-architecture	Silvermont	Sandy Bridge
L1i Cache	32 KB	32 KB
L1d Cache	24 KB	32 KB
L2 Cache	4*1024 KB	256 KB
L3 Cache	–	15 MB
System memory	8 GB	8 GB
Memory type	DDR3 1600 MHz	DDR3 1600 MHz

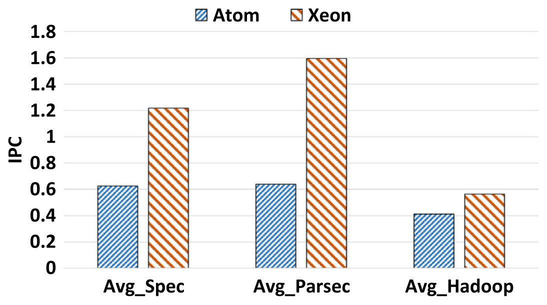


Fig. 1. IPC of SPEC, PARSEC and Hadoop applications on little and big core.

1.3. Application diversity

A Hadoop MapReduce cluster can host a variety of big data applications running concurrently. 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 and real world applications (Naïve Bayes — NB and FP-Growth — FP). Table 2 shows the selected Traditional applications, Hadoop micro-benchmarks and real world applications along their particular domain and data type [11,38].

1.3.1. Hadoop workload

- WordCount reads text files and determines how often the words appear in a set of files. Wordcount is a CPU intensive application [29,38].
- **Sort** uses the map/reduce framework to sort the input directory in the output directory. The actual sorting occurs in the internal shuffle and sort phase of MapReduce. The data is transferred to reducer that is an identity function. Sort is an I/O intensive application [38].
- **Grep** extracts matching strings provided by user from text files and sorts matching strings by their frequency. Grep is a CPU intensive application [38].
- **TeraSort** performs a scalable MapReduce-based sort of input data. It first samples the input and computes the input distribution by calculating the quantiles equal to the number of reduces that uses a sorted list of N-1 sampled keys to define the key range for each reduce. TeraGen command generates the large random data for TeraSort [38].
- Association Rule Mining is a well-known approach for exploring association between various parameters in large databases. We have analyzed **FP (Frequent Pattern)-Growth**; a resource intensive program that aims to determine item sets in a group and identifies which items typically appear together.
- Classification technique learns from the existing categorizations and groups the unclassified items to the best corresponding category. We have analyzed **Naïve Bayes (NB)**; a popular classification algorithm for data mining [38].

1.3.2. Traditional CPU benchmarks

- SPEC CPU2006 workloads are industry standard reallife applications designed to stress the CPU, memory subsystem and compiler.
- PARSEC 2.1 is an open-source parallel benchmark suite for evaluating multi-core and multiprocessor systems.

2. Traditional applications vs Hadoop applications on big and little cores

While Xeon server has been optimized for traditional CPU and parallel applications it is not clear whether it has also been optimized for big data applications and in particular Hadoop MapReduce based applications. Similarly, while Atom based server has not been designed for traditional compute intensive traditional CPU applications, it is shown to be an energy-efficient processing platform for big data applications [6]. It is therefore important to find out whether the choice of Xeon vs Atom changes for Hadoop applications compared to traditional applications. In this section, we analyze the performance and energy-efficiency measurements of Hadoop applications and compare it with the traditional CPU and PARSEC applications on big and little core-based servers.

2.1. Performance analysis

In this section, we analyze the performance measurements of Hadoop applications in terms of IPC and compare it with the traditional benchmarks. Fig. 1 presents that the average IPC of Hadoop applications is 2.16 times lower than traditional CPU benchmarks on big core and 1.55 times lower on little core. Therefore, noticeably more performance drop (39.3%, on average) is observed for Hadoop applications compared to traditional CPU applications when running on big core server compared to little core server. In general, we observe lower IPC in Hadoop applications compared with the traditional benchmarks. Furthermore, little core-based server is experiencing 1.43 times lower IPC in comparison to big core server as Xeon can process up to 4 instructions simultaneously while Atom is limited to 2 instructions per cycle.

Table 2
Studied Hadoop-based big data applications.

Type of benchmark	Application domain	Workloads	Data source	Software stacks
Hadoop Micro-benchmarks	I/O — CPU testing micro program	WordCount (WC) Sort (ST) Grep (GP) TeraSort (TS)	Text Table Text Table	Hadoop 2.6.1
Real world applications	Association Rule Mining Classification	FP-Growth (FP) Naïve Bayes (NB)	Text	Hadoop 2.6.1, Mahout 0.9
Traditional applications	CPU applications Parallel applications	Spec2006 Parsec	Reference Input Native Input	Spec 2006 Parsec 2.1

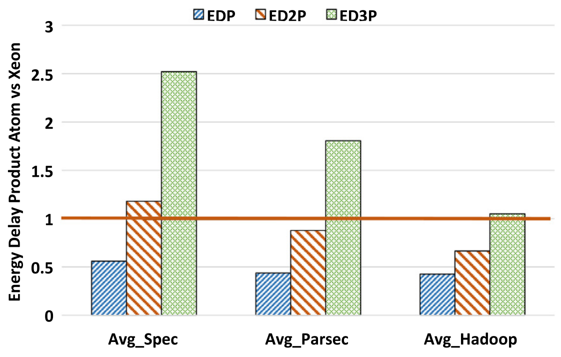


Fig. 2. EDP, ED^{2}P and ED^{3}P Analysis of SPEC, PARSEC and Hadoop applications.

2.2. *Energy-efficiency analysis*

In order to characterize the energy efficiency, we evaluate Energy Delay Product (EDP) metric to investigate trade-off between power and performance. EDP (Power x Execution time x Execution time) is a fair metric to compare various architectures, or even the impact of changing optimization knobs in an architecture. 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. In addition to EDP, we have explored the ED^{2}P and ED^{3}P to understand the impact of near real-time performance constraints on Hadoop applications and how more performance constraints affects the choice of most efficient server architecture. Fig. 2 shows the EDP, ED^{2}P and ED^{3}P ratio for little vs big core for traditional CPU, parallel benchmarks as well as Hadoop applications. Although the execution time of Spec, PARSEC and Hadoop applications is lower on Xeon compared to Atom, the high power consumption of Xeon results in a higher EDP on Xeon as compared to Atom. The ED^{X}P ($X = 1, 2, 3$) results show that compared to little core server, big core server is noticeably more efficient for traditional CPU applications in comparison with Hadoop applications. While for traditional applications there is a noticeable ED^{X}P gap between the two architectures, the ED^{X}P gap for Hadoop applications reduces significantly. With increased in real-time performance constraints big core becomes more energy-efficient compared to little core mainly due to its complex and deeper memory subsystem along with higher processing capacity (2X more than little core).

Considering 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. In addition, given that the choice of big vs little core can be impacted by various tuning knobs that exist in MapReduce computing, 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.

3. 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 (32 MB, 64 MB, 128 MB, 256 MB, and 512 MB) at different operating frequencies (1.2 GHz, 1.4 GHz, 1.6 GHz and 1.8 GHz) for Hadoop micro-benchmarks and real world applications at 1 GB and 10 GB of data per node, respectively.

3.1. *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.

3.1.1. *Application execution time*

Fig. 3 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. We have observed on average 1.74X, 15.4X, 1.39X and 1.57X reduction in execution time from Xeon to Atom for WordCount, Sort, Grep and Terasort, respectively in Fig. 3. Moreover, increasing the frequency and HDFS block size enhances the performance on both architectures. Fig. 3 illustrates 34.4%–66.60% and 47.18–74.87% reduction in execution time by tuning the frequency and HDFS block size on Xeon and Atom, respectively. 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 up to 256 MB. 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. By increasing the HDFS blocks size from 32 MB to 512 MB, we have observed up to 18.9% variation on Xeon and up to 26.18% variation 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. For instance, by changing the frequency from 1.2 GHz to 1.8 GHz, the improvement in execution time is up to 31.52% on Xeon and 44.60% on Atom, observed in Fig. 3. 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 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. Atom has a performance bottleneck that exists in its compute capacity and memory subsystem. Xeon has a high processing capacity as it can process 4 instructions per cycle (issue width of 4) compared to Atom that has issue width of 2. Additionally, Xeon has large in size three-level cache hierarchy (mentioned in Table 2) compared to the Atom that has a two-level cache hierarchy. Therefore, Xeon can hide the memory subsystem misses more effectively than Atom. 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

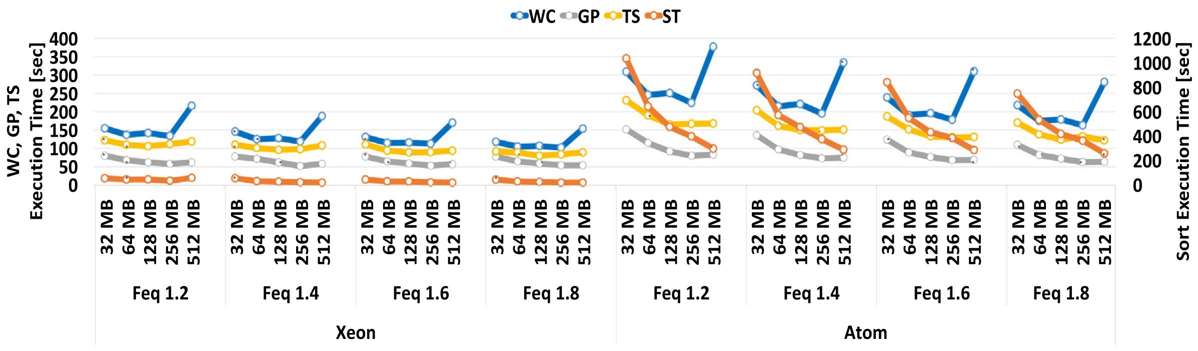


Fig. 3. Execution time of Hadoop micro-benchmarks with respect to various HDFS block size and frequency scaling.

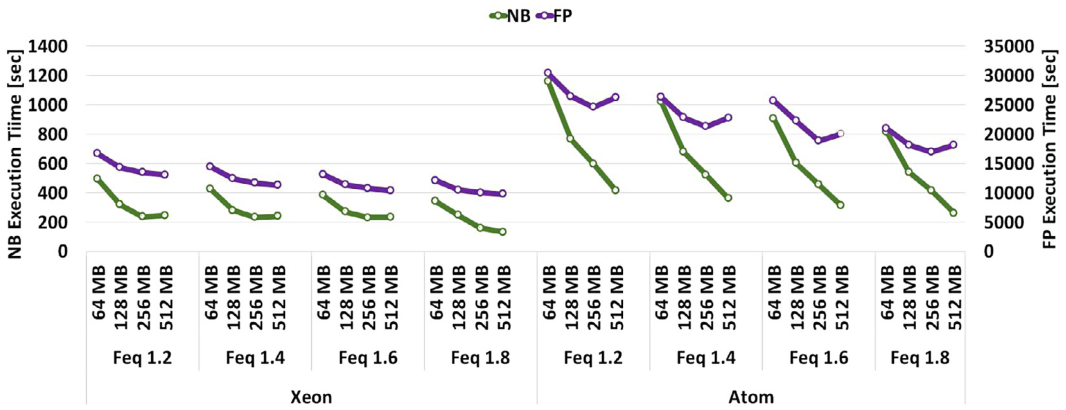


Fig. 4. Execution time [sec] of real world applications with respect to various HDFS block size and frequency scaling.

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.

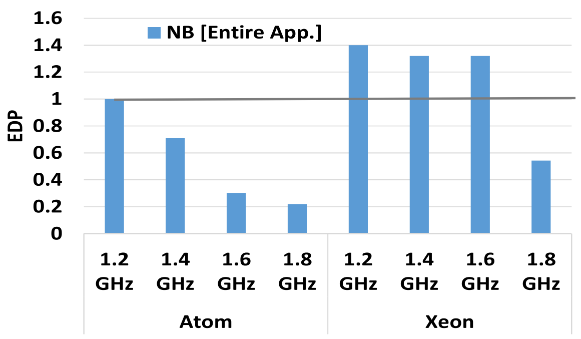
For compute-bound application — Wordcount, the trend is slightly different from I/O intensive application-Sort. The Wordcount execution time decreases with the increasing frequency on both machines, as is the case with Sort. However, while increasing the HDFS block size to 256 MB decreases the execution time, further increase in HDFS block size (e.g. 512 MB) 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 256 MB, however, considering I/O applications are benefiting from higher CPU processing capacity, we have observed a better performance at the higher block size (512 MB) for these applications and in particular on Xeon.

Fig. 4 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 64 MB 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 (64 MB) is not optimal to achieve the maximum performance improvement. The HDFS block sizes of up to 256 MB 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, 256 MB is the optimal choice to achieve the maximum performance.

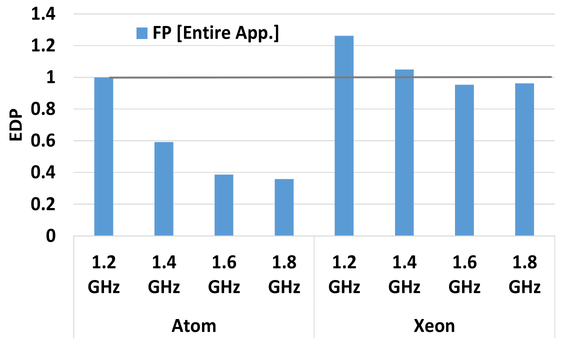
Although, the optimal HDFS block size for the peak performance is closely decided by the application type, extensive experimental search to determine the best HDFS block size can be avoided by considering 256 MB block size for the compute bound and 512 MB for other applications as an optimal choice for performance.

3.1.2. *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.

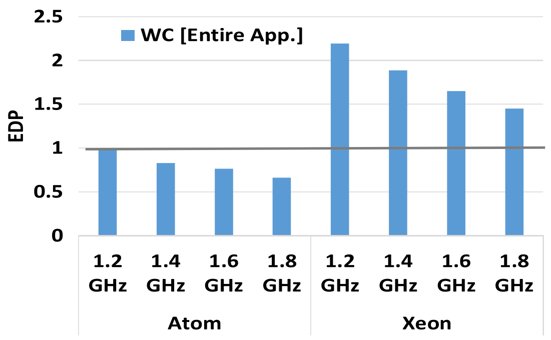


(a) NB.

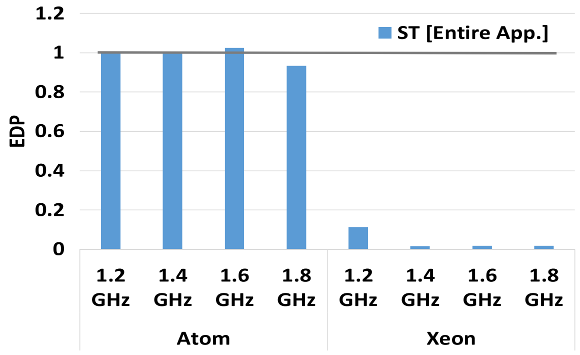


(b) FP.

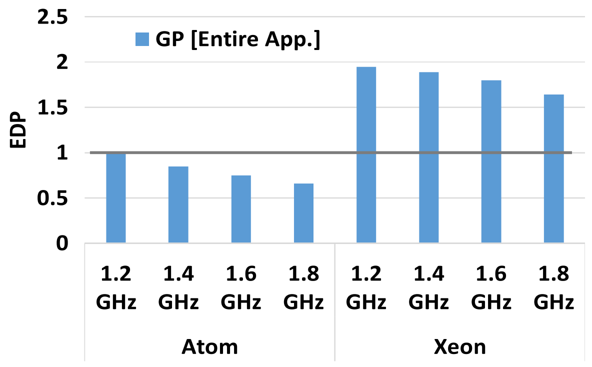
Fig. 5. EDP analysis of entire real world application on big and little core with frequency scaling.



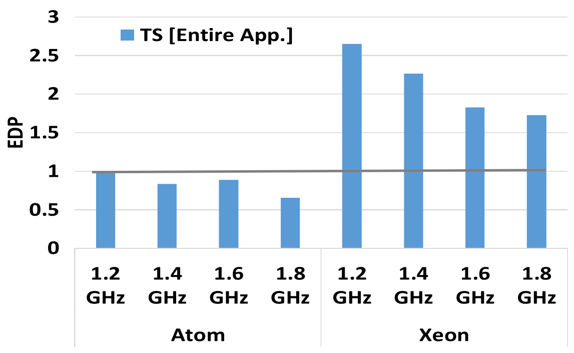
(a) WordCount.



(b) Sort.



(c) Grep.



(d) TeraSort.

Fig. 6. EDP analysis of entire Hadoop micro-benchmark on big and little core with frequency scaling.

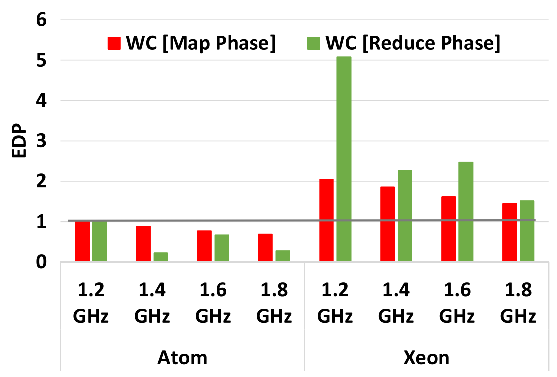
Therefore, the performance gap between Atom and Xeon architecture can be reduced significantly through fine-tuning of the system and architectural parameters on Atom, allowing maximum energy efficiency, as will be discussed later. Also, the results suggest that the optimal HDFS block size for the maximum performance is closely decided by the application type and fine tuning this parameter reduces the dependence on the highest operating frequency.

3.2. Energy-efficiency analysis

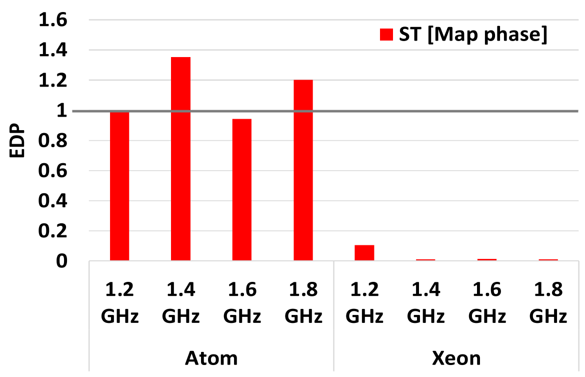
In this section, we analyze the energy-delay-product (EDP) of the studied applications while changing the frequency. Figs. 5 and 6 show the EDP results on Atom and Xeon for the entire application. Figs. 7 and 8 present the map and reduce phases of all the studied applications. In order to make fair comparisons, for each workload, the EDP values are normalized to the EDP on Atom at the lowest frequency of 1.2 GHz and with 512 MB HDFS block size.

3.2.1. EDP of the entire application

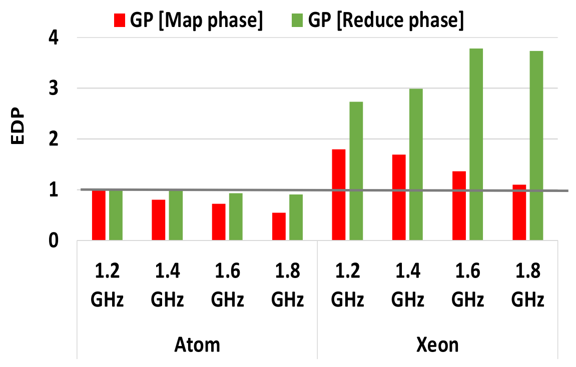
The major observation for the EDP is that for most applications, the low power characteristics of the Atom results in a lower EDP on Atom compared to Xeon, with the exception of the Sort benchmark. By scaling the frequency from 1.2 GHz to 1.8 GHz in Fig. 6, we have observed 2.27X, 2.48X and 2.64X lower EDP on Atom compared to Xeon for WordCount, Grep and Sort, respectively. 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.



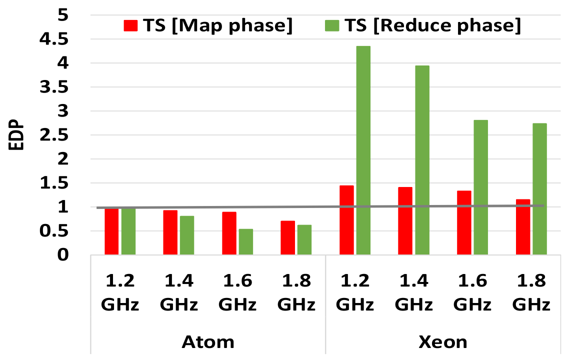
(a) WordCount.



(b) Sort.

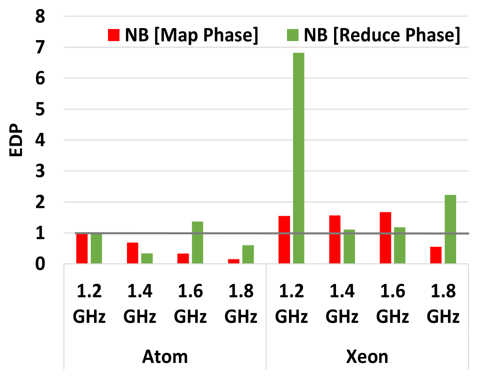


(c) Grep.

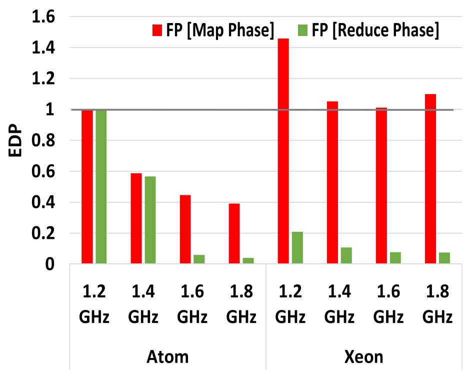


(d) TeraSort.

Fig. 7. EDP analysis of Map and Reduce phase on big and little core with frequency scaling.



(a) NB.



(b) FP.

Fig. 8. EDP analysis of Map and Reduce phase of real world applications on big and little core with frequency scaling.

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, Figs. 5 and 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.

3.2.2. *Map reduce phase analysis*

The results in Figs. 7 and 8 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 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.

3.2.3. *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. Fig. 9 presents the EDP change with respect to the HDFS block

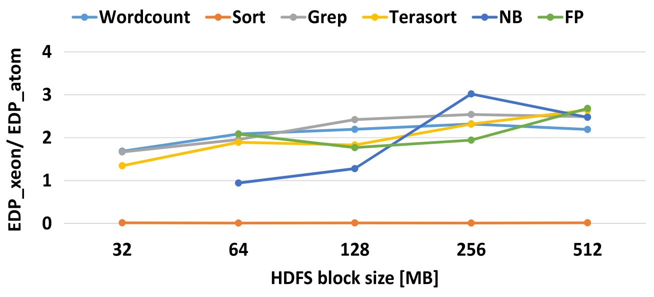


Fig. 9. EDP of Hadoop benchmarks at various HDFS block size.

size for a frequency of 1.8 GHz. The results show that increasing HDFS block size increases the EDP gap between Atom and Xeon. As we have observed in Fig. 9, with HDFS block size of 512 MB, the EDP gap between Atom and Xeon increases more than 2X. 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.

3.3. Input data size sensitivity analysis

In this section, we study the impact of input data size on performance and energy-efficiency of Atom vs Xeon. We have conducted the data size sensitivity analysis of Hadoop micro-benchmarks and real world applications with the datasets of 1 GB, 10 GB, and 20 GB per node. The HDFS block size considered for this experiment is 512 MB with 1.8 GHz operating frequency.

Figs. 10 and 11 present the normalized execution time breakdown of Hadoop applications (Map phase, Reduce phase and Others) along with the total execution time of the entire workload on big and little cores (presented as a line graph in the figure). As shown, the execution time is proportional to the input data size. Comparing the two architectures, we can observe that the execution time increases significantly more on Atom as a function of data size (10.15X, 7.75X, 27.15X, 8.59X, and 7.97X) compared to Xeon (3.45X, 7.75X, 26.07X, 7.22X, and 5.96X) for GP, WC, TS, NB and FP, respectively, as shown in Figs. 10 and 11. This is mainly due to the performance bottleneck of Atom core that exist in its compute capacity as well as memory subsystem, which is exposed more as the size of data increases.

To evaluate the trade-off of power and performance between Xeon and Atom, we have investigated the EDP metric (Figs. 12 and 13). We have observed a clear trend of rise in EDP with the increase in input data size across both architectures. Across almost all the studied benchmarks little Atom core is clearly a favorite choice for energy-efficiency, with the exception of the sort application. The increase in the data size progressively makes the big core more efficient compared to the little core across all the applications with the exception of Sort that illustrate the opposite trend. For the smaller data size, the higher processing capability of big core hides the I/O communication cost effectively. However, the rise in data size aggravates the I/O cost to an extent that it diminishes the benefit of the high processing capability of the big core for I/O intensive benchmarks. Note that big core has no advantage of big DRAM size as we have set the studied applications to utilize the same DRAM size in both architectures. Additionally, we have studied the EDP results of Atom vs. Xeon for the map and reduce phases. This experiment will help guiding scheduling decision such as the choice of the core to run map or reduce phase to improve energy efficiency. The results are presented in Fig. 13. For the map phase, we observed very similar trend to the entire application, showing clear benefit for little core, except for I/O bound applications and also when the size of data per node increases.

3.4. Performance hotspot and post-acceleration CPU code characterization

In recent years, the interest in hardware acceleration has revived mainly due to the dark silicon challenge. In addition to big, medium, and small cores, the integration of domain-specific accelerators, such as and FPGAs in cloud computing platforms has become extensive. To find out the right CPU core for Hadoop MapReduce processing, it is important to understand how deploying an accelerator, such as FPGA or GPUs, would necessitate adapting the choice of CPU. While the GPU-based platforms have achieved significant speedup across a wide range of benchmarks, their high power demands preclude them for energy-efficient computing [12,26]. FPGAs have shown to be more energy efficient [24] and they allow the exploitation of fine-grained parallelism in the algorithms. Moreover, advances in the FPGA technology advances, suggest that new generation of FPGAs will outperform GPUs. In [31] the authors compare two generations of Intel FPGAs (Arria 10, Stratix10) against the latest highest performance Titan X Pascal GPU. For a ResNet case study, their results show that for Ternary ResNet [20], the Stratix 10 FPGA can deliver 60% better performance over Titan X Pascal GPU, while being 2.3× better in performance/watt showing that FPGAs may become the platform of choice for accelerating the applications. Therefore, we have considered FPGA as a domain-specific accelerator.

The post acceleration code characteristics are important to find the right architecture for efficient processing of Hadoop applications. In this section, we analyze the choice of big vs. little core-based server for the code that remains for the CPU after acceleration, compared with the choice of big vs. little before acceleration. A key research challenge for heterogeneous architecture

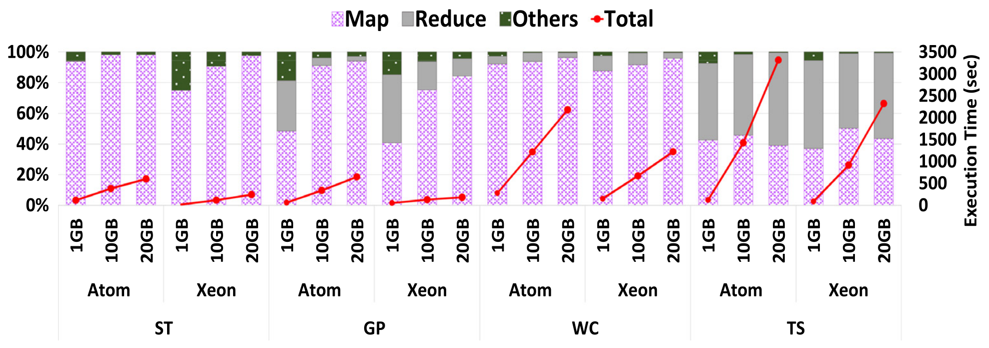


Fig. 10. Execution time and breakdown of Hadoop micro-benchmarks at various input data size.

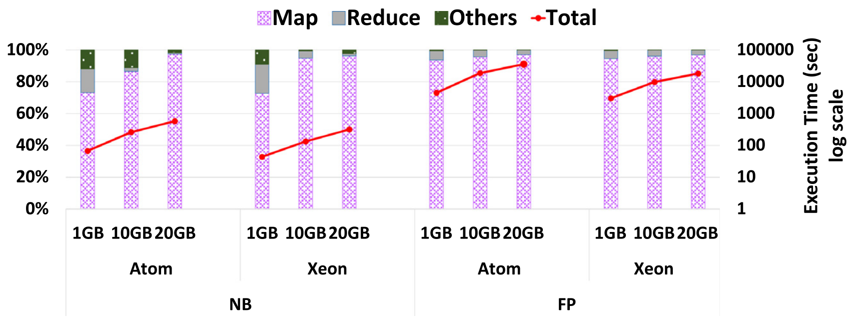


Fig. 11. Execution time and breakdown of Hadoop real applications at various input data size.

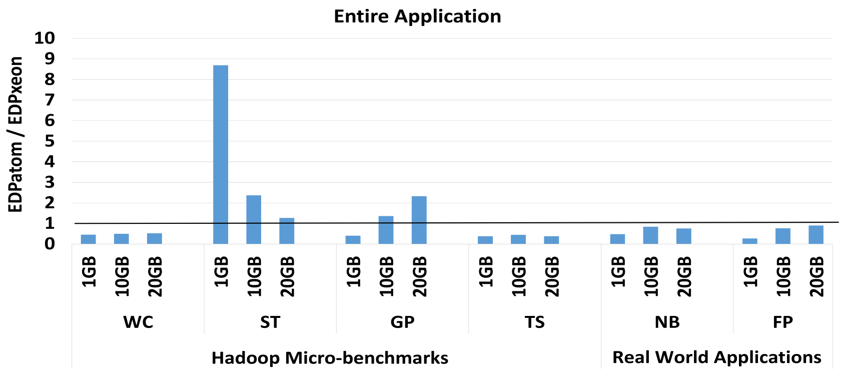


Fig. 12. EDP analysis of entire application with various input data size.

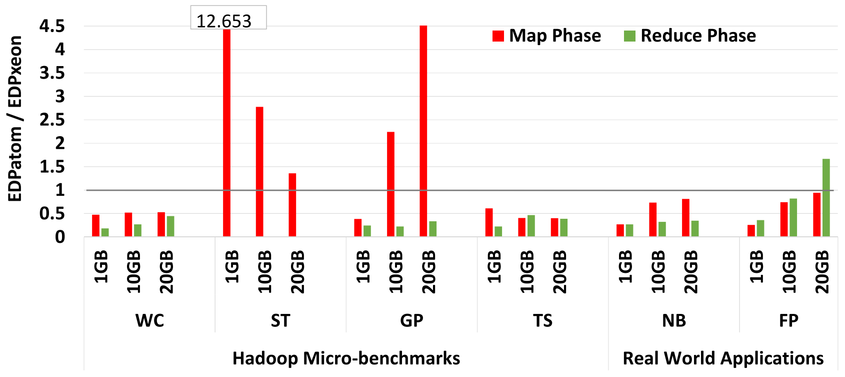


Fig. 13. EDP analysis of Map Phase and Reduce phase with respect to the input data size.

that integrates CPU and accelerator such as FPGA is workload partitioning and mapping of a given application (which is alternatively referred to as scheduling) to CPU and FPGA for power, performance, and QoS. This is commonly referred as hardware and software partitioning. A common method for HW/SW partitioning is to profile the application to find the performance hotspot region. These regions are candidates for FPGA acceleration, as long as the overhead of communication with CPU is not significant [1]. There are several tasks involved in an end-to-end big data Hadoop MapReduce environment.

To perform hotspot analysis of the applications, we identify and analyze these phases based on their execution time. Here we have considered the 512 MB HDFS block size and 1.8 GHz operating frequency. Note that sort benchmark has 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. Since the computation intensive part of the applications lie on the map and reduce phase, it is critical to understand how sensitive they are for the post-accelerated code analysis. In most of the studied applications, the map function accounts for more than half of the execution time. Additionally, given that sort has no reduce phase and in most studied applications, map execution phase is the hotspot, we assume map tasks are offloaded to an accelerator such as FPGA. The execution time of the map phase after acceleration consists of three major parts, i.e. time_cpu — the software part of the map phase that remains on the CPU-, time_fpga — the hardware part of the map function that is offloaded to the FPGA- and time_trans — the data transfer time between the FPGA and the CPU core. Time_trans is calculated based on the speed of the connections link and the amount of data that is transferred between the accelerator and the CPU. While related work [1,9,16]

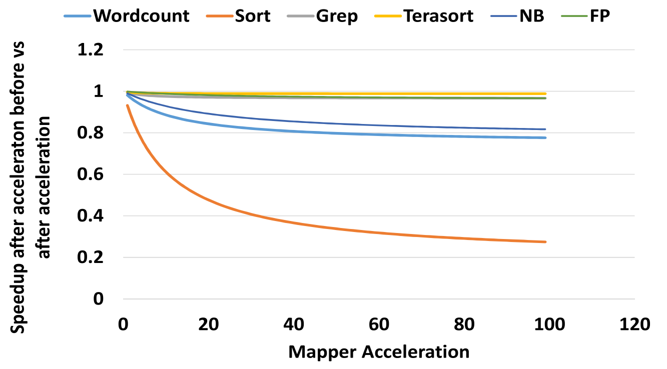


Fig. 14. Speed up of Atom vs Xeon before and after acceleration of Hadoop micro-benchmarks and real world applications.

have focused on modeling how adding various accelerators, with various interconnects improves the designs, our main goal is to characterize how adding FPGAs change tuning of various system and architecture-level parameters in the design. To this end, without diving into how each application can be accelerated, we have studied a wide range of acceleration rates of the map-phase, which ranges from no speedup (1X) to 100X (speedup is achieved as time_allCPU/[time_cpu+time_fpga+time_trans]), as in Fig. 14. After calculating the new execution time, we can study the remaining modules that are left for the big or little core-based server to run.

Fig. 14 shows the impact of post-accelerated code by investigating the speed up — migrating from Atom to Xeon before and after acceleration. We report the little vs big core speed up in terms of

$$\text{Speed up} = \frac{(\text{Exectime}_{\text{Atom}}/\text{Exectime}_{\text{Xeon}})_{\text{remaining code after acceleration}}}{(\text{Exectime}_{\text{Atom}}/\text{Exectime}_{\text{Xeon}})_{\text{entire application}}} \quad (1)$$

$(\text{Exectime}_{\text{Atom}}/\text{Exectime}_{\text{Xeon}})_{\text{remaining code after acceleration}}$ represents the speed up obtained by migrating the post-accelerated code from Atom to Xeon.

$(\text{Exectime}_{\text{Atom}}/\text{Exectime}_{\text{Xeon}})_{\text{entire applications}}$ represents the speed up obtained by migrating the application from Atom to Xeon before acceleration.

Using Eq. (1), we can evaluate the impact of Atom over Xeon speedup gain after acceleration compared to speed up before acceleration. All the benchmarks in Fig. 14 have speed up less than 1 which indicates that speedup of migrating from Atom to Xeon after acceleration reduces compared to speed up before acceleration. We have observed a negligible impact on Terasort and Grep. The contribution of Map phase execution time to the entire application execution time for these applications is lower compared to the other applications that explains why there is no significant difference between before and after acceleration for the choice of Atom vs Xeon. Overall, Xeon provides a lower execution time, however, if speedup after acceleration is very small then considering the power consumption of Xeon, Atom-based server will be a more efficient choice to execute the post-accelerated code.

3.4.1. *Frequency and HDFS block size sensitivity analysis before and after acceleration*

Additionally, we have performed the post-accelerated code analysis with respect to the frequency scaling (Fig. 15). With the exception of grep and FP at the lower frequencies, all other benchmarks have shown that the speed up of migrating from Atom to Xeon after acceleration reduces compared to the before acceleration.

Fig. 16 illustrates the post-accelerated code analysis with various HDFS block size. We have observed that for the grep and TeraSort, with the increase in the data size the speedup after acceleration increases compared to the speedup before acceleration. This behavior is consistent with the fact that reduce phase of these applications has a significant contribution to the total execution time as compared to the other applications.

However, sort shows the opposite trend as it has only the map phase that is being offloaded to the accelerator. On the other hand, FP, a compute-bound application, shows that the speedup of migrating from Atom to Xeon achieves after deploying an accelerator outperforms the speedup before acceleration as we have observe a significant performance gap between Atom and Xeon for this application.

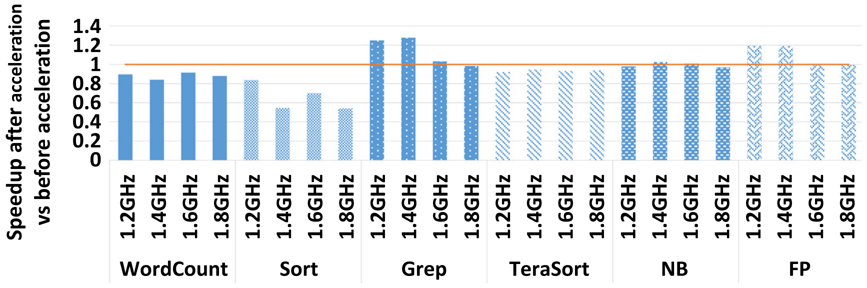


Fig. 15. Speedup of Atom vs Xeon before and after acceleration with various frequencies.

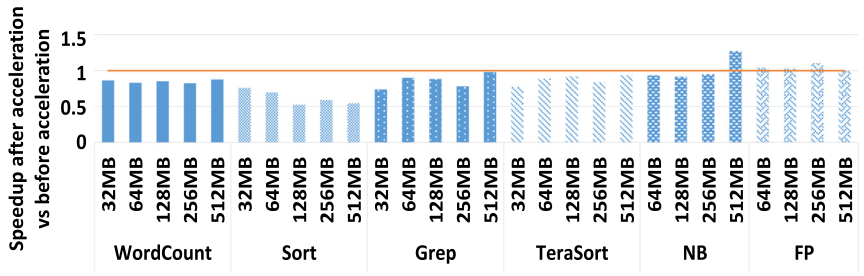


Fig. 16. Speedup of Atom vs Xeon before and after acceleration with various HDFS block size.

Table 3
Operational and capital cost analysis of Hadoop applications.

		Atom				Xeon			
		M2	M4	M6	M8	M2	M4	M6	M8
EDP (J s)	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 s^2)	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 mm^2 s)	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 mm^2 s^2)	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

3.5. Scheduling

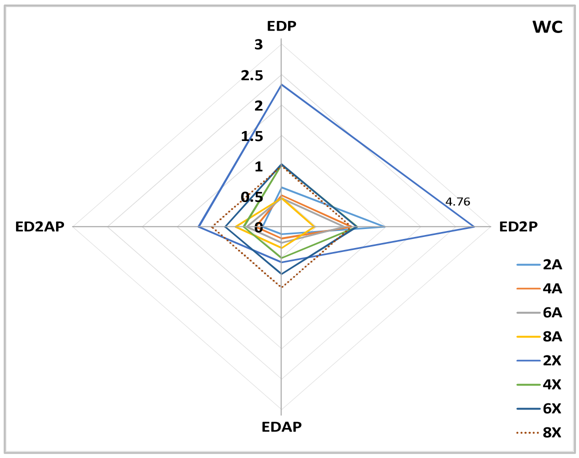
In the previous sections, we analyzed the execution time and the energy-efficiency of MapReduce benchmarks across a wide range of application, system and architecture levels parameters on Xeon and Atom cores. These analyses will help guide the scheduling optimization decisions in a heterogeneous architecture, 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 [21] 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 512 MB and operating frequency at the 1.8 GHz.

Table 3 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 3 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 up to 5X (in sort benchmark) by utilizing a maximum number of Atom cores compared to using minimum two, yielding both higher performance and energy-efficiency.

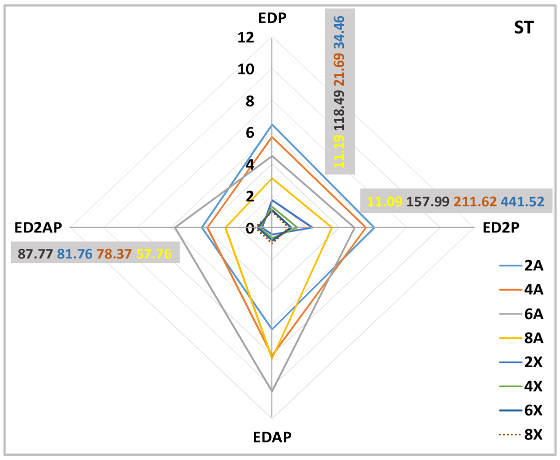
As mentioned earlier, the capital cost of the architecture is another major cost function that affects the scheduling decisions. In order to take the capital cost into account, we study ED^XAP values, presented in Table 3. Hadoop micro-benchmarks show that while increasing the number of cores reduces 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 compute-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 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 Fig. 17 (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.

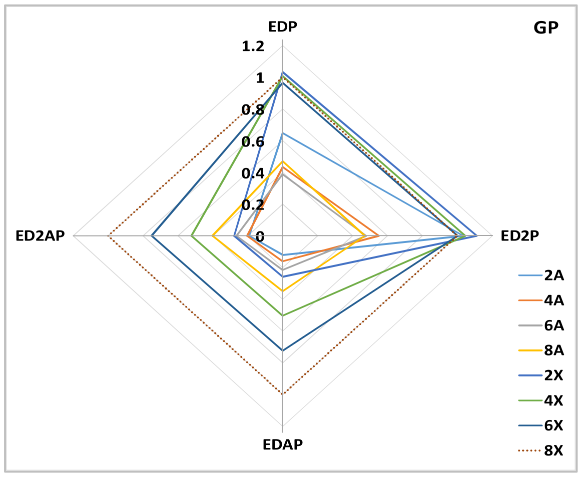
For the energy efficiency (EDP) results, the comparison between Atom and Xeon shows that the Atom cores are more energy-efficient than the Xeon cores for all the studied applications with the exception of *Sort* as it has the higher performance gap between Atom and Xeon that results in lower EDP on Xeon. Additionally, we



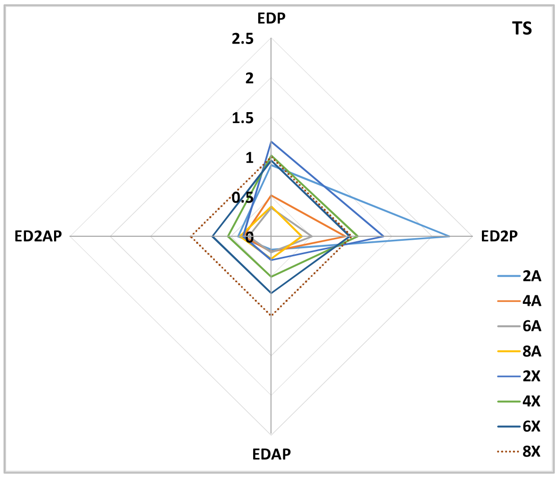
(a) Wordcount.



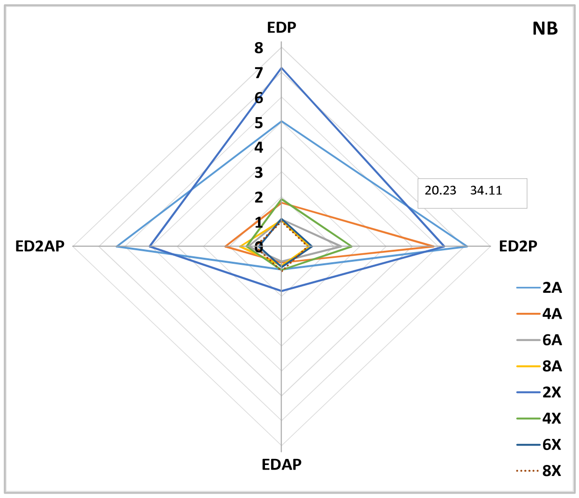
(b) Sort.



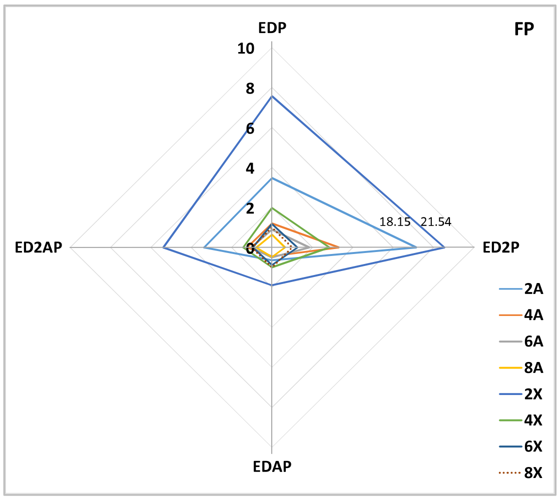
(c) Grep.



(d) TeraSort.



(e) NB.



(f) FP.

Fig. 17. Results presented are for energy-efficiency (EDP in J s), real time energy efficiency (ED2P in J s^2), cost energy efficiency (EDAP-in J mm^2 s) and real time cost energy efficiency (ED2AP in J mm^2 s^2) of Hadoop applications normalized to the 8 Xeon core.

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 architecture with 8 cores. This is an interesting observation, which allows us to use Xeon *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 level analysis in Section 3.2, we illustrate that the reliance of the maximum number of cores for Atom architecture can be reduced by fine-tuning the system, application and architectural parameters.

The pseudo code to schedule the workloads in heterogeneous server architecture to minimize the operational and capital cost is provided as follows

Schedule Big Data Application on Big or Little cores in heterogeneous architecture

```
Goal is to minimize operational cost and capital cost = {ED^XP, ED^XAP}
Applications are classified as compute bound (C), I/O bound (I) and Hybrid (H) = {C, I, H}
Number of Big/Xeon cores X = {2, 4, 6, 8}
Number of Little/Atom cores A = {2, 4, 6, 8}
Applications are referred as App

Procedure schedule_workloads (goal, App)
  If App = C
    For the min operational cost and capital cost
      Assign large number of Atom cores to run the application (A=8)
      Fine-tune configuration parameter to reduce the number of cores
  If App = I
    For the min operational cost and capital cost
      Assign small number of Xeon cores to run the application (X=4)
  If App = H
    For the min ED^2AP
      Assign small number of Xeon cores to run the application (X=2)
    Otherwise
      Assign large number of Atom cores to run the application (A=8)
      Fine-tune configuration parameter to reduce the number of cores
Return (X, A)
```

In general, our results in cloud computing infrastructure equipped with heterogeneous server architectures illustrate that by fine-tuning the system and 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.

4. Related work

Recently, there have been a number of efforts to understand the behavior of big data and cloud-scale applications [8,10], 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 [9,13,16,37,38]. 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 [15,23], energy-aware scheduling of MapReduce jobs [25] and GreenHadoop [4]. Additionally, the impact of Hadoop configuration parameters has been discussed in [4]. 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 [19] 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 [2] 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.

There have been also a number of research into application specific [32,39] and domain-specific accelerators [7,18,36]. Using tightly integrated FPGA [40] with CPU, and GPU with CPU [5], to accelerate big data processing have been proposed in recent work. While deploying programmable accelerator is a new and hot research topic, little attention has been paid on how CPU designs should be adapted to this change. To the best of our knowledge, the only work on this topic is by Arora [22], which studied the role of the CPU for a CPU+GPU architecture. They concluded that, in a CPU+GPU architecture, the CPU is running a code that is significantly different from a CPU-only code. In this paper, we demonstrated how deploying accelerator such as FPGA for big data affects the choice of big vs. little core for efficient processing.

5. Conclusion

This paper answers the important question of whether big 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.

Comparing the two architectures with respect to the input data size, we can observe that the execution time increases significantly more on Atom as a function of data size compared to Xeon for big data applications. In addition, we have observed a clear trend of rise in EDP with the increase in input data size across both architectures. The increase in the data size progressively makes the big core more efficient compared to the little core across all the applications with the exception of Sort that illustrate the opposite trend.

As future cloud-scale architecture will be equipped with on-chip accelerator it is important to understand the choice of Atom vs Xeon cores after acceleration compared to before acceleration. We observed that the speedup gain of migrating from Atom to Xeon reduces significantly after acceleration compared to before acceleration. The presence of hardware accelerator changes the most efficient architecture. Overall, Xeon provides a lower execution time, however, if speedup after acceleration is very small then considering the power consumption of Xeon, Atom is a more energy-efficient choice to execute the post-accelerated code.

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.

Acknowledgment

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

References

[1] Accelerating Hadoop* applications using Intel® QuickAssist Tech, [Online]. Available: http://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/accelerating-hadoop-applications-brief.pdf.

[2] A. Anwar, K.R. Krish, A.R. Butt, On the use of microservers in supporting hadoop applications, in: IEEE International Conference on Cluster Computing, CLUSTER 2014, pp. 66–74.

[3] Apache Mahout: scalable machine-learning and data-mining library, [Online]. Available: http://mahout.apache.org/.

[4] T.G. Armstrong, V. Ponnekanti, D. Borthakur, M. Callaghan, LinkBench: A database benchmark based on the Facebook social graph, in: ACM SIGMOD International Conference on Management of Data, ACM, 2013, pp. 1185–1196.

[5] M. Arnold, H. Corporaal, Designing domain-specific processors, in: Proceedings of the Ninth International Symposium on Hardware/Software Codesign, ACM, pp. 61–66.

[6] M. Arora, S. Nath, S. Mazumdar, S.B. Baden, D.M. Tullsen, Redefining the role of the CPU in the era of CPU–GPU integration, IEEE Micro 32 (6) (2012) 4–16.

[7] N. Arora, K. Chandramohan, N. Pothineni, A. Kumar, Instruction selection in asip synthesis using functional matching, in: VLSI Design, 2010 VLSID'10 23rd International Conference on, IEEE, 2010, pp. 146–151.

[8] C. Baru, M. Bhandarkar, R. Nambiar, M. Poess, T. Rabl, Setting the direction for big data benchmark standards, in: Technology Conference on Performance Evaluation and Benchmarking, Springer, Berlin Heidelberg, 2012, pp. 197–208.

[9] Y.M. Choi, H.K.H. So, Map-reduce processing of k-means algorithm with FPGA-accelerated computer cluster, in: IEEE 25th International Conference on Application-specific Systems, Architectures and Processors, ASAP, 2014.

[10] Eric S. Chung, John D. Davis, Jaewon Lee, Linqits: big data on little clients, ACM SIGARCH Comput. Archit. News 41 (3) (2013).

[11] Blem Emily, Jaikrishnan Menon, Karthikeyan Sankaralingam, Power struggles: Revisiting the RISC vs. CISC debate on contemporary ARM and x86 architectures, in: IEEE 19th International Symposium High Performance Computer Architecture, HPCA2013, 2013.

[12] J. Fowers, G. Brown, P. Cooke, G. Stitt, A performance and energy comparison of FPGAs, GPUs, and multicores for sliding-window applications, in: Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, ser. FPGA '12, New York, NY, USA, 2012, pp. 47–56.

[13] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A.D. Popescu, A. Ailamaki, B. Falsafi, Clearing the clouds: A study of emerging scale-out workloads on modern hardware, ACM SIGPLAN Not. 47 (4) (2012) 37–48 ACM.

[14] N. Hardavellas, M. Ferdman, B. Falsafi, A. Ailamaki, Toward dark silicon in servers, IEEE Micro 31 (4) (2011) 6–15.

[15] H. Homayoun, V. Kontorinis, A. Shayan, T.W. Lin, D.M. Tullsen, Dynamically heterogeneous cores through 3D resource pooling, in: IEEE 18th International Symposium on High Performance Computer Architecture, HPCA, 2012.

[16] T. Honjo, K. Oikawa, Hardware acceleration of hadoop mapreduce, in: Big Data, 2013 IEEE International Conference on, IEEE, 2013, pp. 118–124.

[17] S. Huang, J. Huang, J. Dai, T. Xie, B. Huang, The hibench benchmark suite: Characterization of the mapreduce-based data analysis, in: Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on, IEEE, 2010.

[18] Í Goiri, K. Le, T.D. Nguyen, J. Guitart, J. Torres, R.. Bianchini, GreenHadoop: Leveraging green energy in data-processing frameworks, in: 7th ACM European Conference on Computer Systems 2012 Apr 10, ACM, 2012, pp. 57–70.

[19] K.R. Krish, A. Anwar, A.R. Butt, [phi] Sched: A heterogeneity-aware Hadoop Workflow Scheduler, in: IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems, 2014.

[20] A. Kundu, K. Banerjee, N. Mellempudi, D. Mudigere, D. Das, B. Kaul, P. Dubey, Ternary residual networks, arXiv preprint arXiv:1707.04679, 2017.

[21] S. Li, J.H. Ahn, R.D. Strong, J.B. Brockman, D.M. Tullsen, N.P. Jouppi, McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures, in: Microarchitecture, 2009, MICRO-42.

[22] T. Li, Z. Sun, W. Jigang, X. Lu, n.d. Fast enumeration of maximal valid subgraphs for custom-instruction identification, in: Proceedings of the 2009 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems.

[23] Z. Lin, P. Chow, n.d. Zcluster: A zynq-based hadoop cluster, in: Field -Programmable Technology, FPT, 2013 International Conference on, pp. 450–453, IEEE.

[24] L. Stolz, H. Endt, M. Vaaraniemi, D. Zehe, W. Stechele, Energy consumption of graphic processing units with respect to automotive use-cases, in: Energy Aware Computing, ICEAC, 2010 International Conference on, Dec 2010, pp. 1–4.

[25] C. Luo, J. Zhan, Z. Jia, L. Wang, G. Lu, L. Zhang, C.Z. Xu, N. Sun, Cloudrank-d: benchmarking and ranking cloud computing systems for data processing applications, Front. Comput. Sci. 6 (4) (2012) 347–362.

[26] X. Luo, W. Najjar, V. Hristidis, n.d. Efficient near-duplicate document detection using FPGAs, in: Big Data, IEEE International Conference on, IEEE, 2013, pp. 54–61.

[27] M. Malik, H. Homayoun, Big data on low power cores: Are low power embedded processors a good fit for the big data workloads? in: 33rd IEEE International Conference on Computer Design, ICCD, 2015 pp. 379–382.

[28] M. Malik, K. Neshatpour, T. Mohsenin, A. Sasan, H. Homayoun, Big vs little core for energy-efficient hadoop computing, in: 2017 Design, Automation & Test in Europe Conference & Exhibition, DATE, IEEE, 2017, pp. 1480–1485.

[29] M. Malik, S. Rafatirah, A. Sasan, H. Homayoun, System and architecture level characterization of big data applications on big and little core server architectures, in: Big Data (Big Data), 2015 IEEE International Conference on, Santa Clara, CA, 2015.

[30] M. Malik, A. Sasan, R. Joshi, S. Rafatirah, H. Homayoun, Characterizing Hadoop applications on microservers for performance and energy efficiency optimizations , in: IEEE International Symposium on Performance Analysis of Systems and Software ISPASS, 2016.

[31] E. Nurvitadhi, G. Venkatesh, J. Sim, D. Marr, R. Huang, J.O.G. Hock, Y.T. Liew, K. Srivatsan, D. Moss, S. Subhaschandra, et al., Can FPGAs beat GPUs in accelerating next-generation deep neural networks? in: FPGA, 2017, pp. 5–14.

[32] Yu. Pan, Tulika Mitra, Scalable custom instruction identification for instruction set extensible processor, in: International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, ACM, 2004, pp. 69–78.

[33] Janapa Reddi, Vijay, Benjamin C. Lee, Trishul Chilimbi, Kushagra Vaid, n.d. Web search using mobile cores: Quantifying and mitigating the price of efficiency, ACM SIGARCH Comput. Archit. News 38 (3) (2010) 314–325.

[34] M.K. Tavana, M.H. Hajkazemi, D. Pathak, I. Savidis, H. n.d. Homayoun, ElasticCore: Enabling dynamic heterogeneity with joint core and voltage/frequency scaling, in: 52nd Annual Design Automation Conference, ACM, pp. 151.

[35] Kontorinis Vasileios, Liuyi Eric Zhang, Baris Aksanli, Jack Sampson, Houman Homayoun, Eddie Pettis, Dean M. Tullsen, Tajana Simunic Rosing, Managing distributed ups energy for effective power capping in data centers, in: Computer Architecture (ISCA), 2012 39th Annual International Symposium on, IEEE, 2012, pp. 488–499.

[36] A. Verma, L. Cherkasova, R.H. Campbell, ARIA: Automatic resource inference and allocation for mapreduce environments, in: 8th ACM International Conference on Autonomic Computing 2011 Jun 14, ACM, 2011, pp. 235–244.

[37] Lei Wang, Jianfeng Zhan, Chunjie Luo, Yuqing Zhu, Qiang Yang, Yongqiang He, Wanling Gao, Zhen Jia, Yingjie Shi, Shujie Zhang, 2014. Bigdatabench: A big data benchmark suite from internet services, in: International Symposium on High Performance and Computer Architecture, HPCA.

[38] Wattsuppro power meter, [Online]. Available: https://www.wattsupmeters.com/secure/index.php.

[39] N. Yigitbasi, K. Datta, N. Jain, T. Willke, Energy efficient scheduling of mapreduce workloads on heterogeneous clusters, in: Green Computing Middleware on Proceedings of the 2nd International Workshop 2011 Dec 12, ACM, 2011, p. 1.

[40] P. Yu, T. Mitra, Disjoint pattern enumeration for custom instructions identification, in: Field Programmable Logic and Applications, 2007 FPL 2007 International Conference on, IEEE, 2007, pp. 273–278.

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, scheduling MapReduce application on microserver, accelerating machine learning kernels, parallel programming languages and parallel computing.

Katayoun Neshatpour is a Ph.D. 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 Ph.D. 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 Ph.D. 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 M.S. degree from the Department of Information and Computer Science at the UC Irvine in 2010.

Rajiv V. Joshi is a research staff member at T. J. Watson research center, IBM. He received his B.Tech I.I.T (Bombay, India), M.S (M.I.T) and Dr. Eng. Sc. (Columbia University). He holds 58 invention plateaus and has over 225 US patents and over 350 including international patents. He has authored and co-authored over 185 papers. He received the Best Editor Award from IEEE TVLSI journal. He is recipient of 2015 BMM award. He is inducted into New Jersey Inventor Hall of Fame in Aug 2014 along with pioneer Nikola Tesla. He is a recipient of 2013 IEEE CAS Industrial Pioneer award and 2013 Mehboob Khan Award from Semiconductor Research Corporation. He is a member of IBM Academy of technology. He served as a Distinguished Lecturer for IEEE CAS and EDS society. He is IEEE, ISQED and World Technology Network fellow and distinguished alumnus of IIT Bombay. He is in the Board of Governors for IEEE CAS. He serves as an Associate Editor of TVLSI. He served on committees of ISLPED (Int. Symposium Low Power Electronic Design), IEEE VLSI design, IEEE CICC, IEEE Int. SOI conference, ISQED and Advanced Metallization Program committees.

Tinoosh Mohsenin received the B.S. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, and the M.S. degree in electrical and computer engineering from Rice University, Houston, TX. She is currently working toward the Ph.D. degree in electrical and computer engineering at the University of California, Davis. She is the Designer of the Split-Row, multisplit, and Split-Row Threshold decoding algorithms and architectures for low-density parity-check (LDPC) codes. She was a Key Designer of the 167-processor Asynchronous Array of simple Processors chip. Her research interests include algorithms, architectures and VLSI design for high-performance and energy-efficient computation in the areas of networking and communications, digital signal processing, and error-correction applications.

Hassan Ghasemzadeh received the B.Sc. degree from Sharif University of Technology, Tehran, Iran, the M.Sc. from University of Tehran, Tehran, Iran, and the Ph.D. from the University of Texas at Dallas, Richardson, TX, in 1998, 2001, and 2010 respectively, all in Computer Engineering. He was on the faculty of Azad University from 2003 to 2006 where he served as Founding Chair of Computer Science and Engineering Department at Damavand branch, Tehran, Iran. He spent the academic year 2010–2011 as a Postdoctoral Fellow at the West Wireless Health Institute, La Jolla, CA. He was a Research Manager at the UCLA Wireless Health Institute 2011–2013. Currently, he is Assistant Professor of Computer Science in the School of Electrical Engineering and Computer Science at Washington State University, Pullman, WA. The focus of his research is on algorithm design and system level optimization of embedded and pervasive systems with applications in healthcare and wellness.

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 Ph.D. degree from the Department of Computer Science at the UC Irvine in 2010, an M.S. degree in computer engineering in 2005 from University of Victoria, and his B.S. degree in electrical engineering in 2003 from Sharif University of technology.