(Invited Paper)

Heterogeneous Chip Multiprocessor Architectures for Big Data Applications

Houman Homayoun
George Mason University
hhomayou@gmu.edu

Abstract

Emerging big data analytics applications require a significant amount of server computational power. The costs of building and running a computing server to process big data and the capacity to which we can scale it are driven in large part by those computational resources. However, big data applications share many characteristics that are fundamentally different from traditional desktop, parallel, and scale-out applications. Big data analytics applications rely heavily on specific deep machine learning and data mining algorithms, and are running a complex and deep software stack with various components (e.g. Hadoop, Spark, MPI, Hbase, Impala, MySQL, Hive, Shark, Apache, and MangoDB) that are bound together with a runtime software system and interact significantly with I/O and OS, exhibiting high computational intensity, memory intensity, I/O intensity and control intensity. Current server designs, based on commodity homogeneous processors, will not be the most efficient in terms of performance/watt for this emerging class of applications. In other domains, 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. A heterogeneous architecture integrates cores with various micro-architectures and accelerators to provide more opportunity for efficient workload mapping. In this work, through methodical investigation of power and performance measurements, and comprehensive system level characterization, we demonstrate that a heterogeneous architecture combining high performance big and low power little cores is required for efficient big data analytics applications processing, and in particular in the presence of accelerators and near real-time performance constraints.

CCS Concepts

Computer systems organization → Heterogeneous (hybrid) systems; Hardware → Hardware accelerators

Keywords

Heterogeneous Architectures; Performance; Power; Application Characterization; Big Data; Accelerator

1. INTRODUCTION

Advances in various branches of technology – data sensing, data communication, data computation, and data storage – are driving an era of unprecedented innovation for information retrieval. The world of Big Data is constantly changing and producing huge amounts of data that creates challenges to process the applications using existing solutions. Big data applications require computing resources and storage subsystems that can scale to manage massive amounts of diverse data. Individuals, businesses, governments, and society as a whole now have access to enormous collections of big data, empowering them to build their own analytics. Data centers are therefore required to introduce more nodes to their infrastructure or replace their existing hardware with more powerful systems to respond to this growing demand. This trend increases the infrastructure cost and power consumption. 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 and power efficiency.

Emerging big data analytics applications require a significant amount of server computational power. The costs of building and running a data center to process big data applications and the capacity to which we can scale it are driven in large part by such computational resources. However, big data applications, in particular from web service domain, share many inherent characteristics that are fundamentally different from traditional desktop, parallel, and scale-out applications [1, 2]. Big data analytics applications in these domains heavily rely on big-data-specific deep machine learning and data mining algorithms, and are running complex database software stack with significant interaction with I/O and OS, and exhibit high computational intensity, memory intensity, I/O intensity and control intensity. In addition, unlike conventional CPU applications, big data applications combine a high data rate requirement with high computational power requirement, in particular for real-time and near-time performance constraints.

This new set of characteristics is necessitating a change in the direction of server-class microarchitecture to improve their computational efficiency. However, while demand for data center computational resources continues to grow as the size of data grows, the semiconductor industry has reached its physical scaling limits and is no longer able to reduce power consumption in new chips. Physical design constraints, such as power and density, have therefore become the dominant limiting factor for scaling out data centers [3, 4, 5]. Current server designs, based on commodity homogeneous processors, will therefore not be the most efficient in terms of performance/watt to process big data applications [5, 6].

Big data applications require computing resources that can scale to manage massive amounts of diverse data. As more clients, consumer devices, and industrial equipment get connected through the Internet of Things, the tsunami of data that is generated and the opportunities to make decisions faster will make today's predominant server architecture, including their homogeneous processing cores, obsolete.

In other domains, 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 [10]. A heterogeneous chip architecture integrates cores with various microarchitectures (in-order, out-of-order, varying cache and window sizes, etc.) or even instruction set architectures (e.g., Thumb and x86) with on-chip GPU or FPGA accelerators to provide more opportunity for efficient workload mapping so that the application can find a better match among various components to improve power efficiency [9,

CF '16, May 16-19, 2016, Como, Italy

http://dx.doi.org/10.1145/2903150.2908078

10, 11, 12]. In this research we investigate whether heterogeneous architectures are a good fit for energy-efficient processing of big data applications. In exploring the choice of server architecture for big data, in this paper, we present a comprehensive analysis of the measurement of power and performance of big data applications on two very distinct microarchitectures; a high performance big Xeon core and another a low power embedded-like little Atom core. These two types of servers represent two schools of thought on server architecture design: using big core like Xeon, which is a conventional approach to designing a high-performance server, and the Atom, which is a new trajectory in server design that advocates the use of a low-power core to address the dark silicon challenge facing servers [6, 7, 8]. In addition to power and performance study, we have also performed the Energy-Delay^X Product (ED^{X}P) analysis to evaluate the trade-off between power and performance to understand how near real-time performance constraints for big data analytics affects the choice of big vs. little core server as a more efficient architecture. This will provide us with insight whether a heterogeneous architecture combining big and little cores is a better fit for big data compared to traditional homogeneous architectures.

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 big data processing, it is important to understand how deploying an accelerator, such as FPGA, would necessitate adapting the choice of big vs. little cores. The post acceleration code characteristics are important to find the right architecture for efficient processing of big data applications. For this purpose, we analyze the choice of big vs. little core-based servers for the code that remains for the CPU after assuming the hotspots are offloaded to an accelerator, compared with the choice of big vs. little before acceleration.

Overall, our measurement results demonstrates that a heterogeneous architecture combining big and little cores is required for efficient big data processing, and in particular in the presence of hardware accelerators and near real-time performance constraints. Therefore, to address the computing requirements of big data, we envision a data-driven heterogeneous architecture for next generation big data server platforms that leverage the power of heterogeneity.

The rest of the paper is organized as follows. Section 2 provides background for big data. Section 3 describes the studied big data, scale-out and Traditional serial and parallel CPU benchmarks. Our methodology and experimental setup details are presented in section 4. Section 5 presents the experimental results and provides system level analysis along the micro-architectural characterization of big data applications. Section 6 provides the related work. Lastly, section 7 concludes the paper.

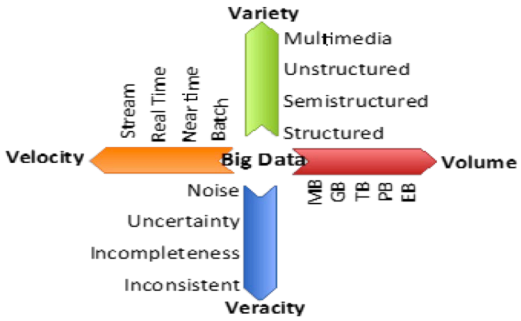


Figure 1. Illustration of Four "Vs" of Big Data

2. BACKGROUND ON BIG DATA APPLICATIONS

The "cloud" is a new platform that has been used to cost effectively deploy an increasingly wide variety of applications. Vast amount of data is now stored in a few places rather than distributed across a billion isolated computers, therefore creates opportunities to learn from the aggregated data. The rise of cloud computing and cloud data storage, therefore, has facilitated the emergence of big data applications. Big data applications are characterized by four critical features, referred as the four "Vs", shown in Figure 1 [35]: volume, velocity, variety, and veracity. Big data is inherently large in volume. Velocity refers to how fast the data is coming in and to how fast it needs to be analyzed. In other words, velocity addresses the challenges related to processing data in real-time. Variety refers to the number and diversity of sources of data and databases, such as sensor data, social media, multimedia, text, and much more. Veracity refers to the level of trust, consistency, and completeness of data. Traditionally, cloud servers mainly use high performance CPU cores such as Xeon. However, low-power embedded cores such as Atom are gradually entering the server market. Therefore, it is important to characterize emerging big data applications on these two different platforms to understand their computational needs.

3. DOMINANT BIG DATA WORKLOADS

The studied big data workloads in this paper are representative applications from 15 different domains such as graph mining, data mining, data analysis platform and pattern searching applications, which are frequently used in the real world. We provide these selected applications, along with their particular domain and data type in Table 1. In addition we also studied scale-out, PARSEC and SPEC CPU benchmark to understand how server architectures behave differently for emerging big data applications compared to more traditional benchmark suites.

4. MEASUREMENT TOOLS AND

Table 1. Studied Big Data Applications

Type of Benchmark	Application Domain	Application Type	Workloads	Data Types	Data source	Software Stacks
Micro benchmarks	I/O testing micro programs	Offline Analytics	WordCount	Unstructured	Text	Hadoop 1.2.1
			Sort		Table	
			Grep		Text	
			TeraSort		Table	
			TestDFSIO-Write			
			TestDFSIO-Read			
Application-Level Benchmark	Search Engine	Offline Analytics	PageRank	Unstructured	Graph	GraphBuilder 0.0.1, Hadoop 1.2.1
			LDA			
	Social Network	Offline Analytics	Collaborative Filtering	Semi-structured	Text	Hadoop 1.2.1, Mahout 0.9/0.4/0.6
			Clustering		Graph	
	E-commerce	Offline Analytics	Frequent Pattern Growth	Structured		
			Frequent Item Mining	Structured	Table	SPMF 0.96
			Sequential Rule Mining			
			Sequential Pattern Mining			
			Classification	Semi-structured	Text	Hadoop 1.2.1, Mahout 0.4, CloudSuite 2.0
	Client Server Application	Online Services	Memcached	Unstructured	Table	Memcached server/client 1.4.15, CloudSuite 2.0

METHODOLOGY

Figure 2 presents a methodology of our approach. 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 includes six aggressive processor cores per node with three-level of the cache hierarchy. Intel Atom C2758 has 8 processor cores per node and a two-level cache hierarchy. The operating system used is Ubuntu 13.10 with Linux kernel 3.11.

We analyze the architectural behavior using Intel VTune [13], a performance-profiling tool that provides an interface to the processor performance counters. We have used Watts up PRO power meter to measure the power consumption of the servers. Watts up power meter produces the power consumption profile every one-second of an application under test. 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 calculate the dynamic power dissipation.

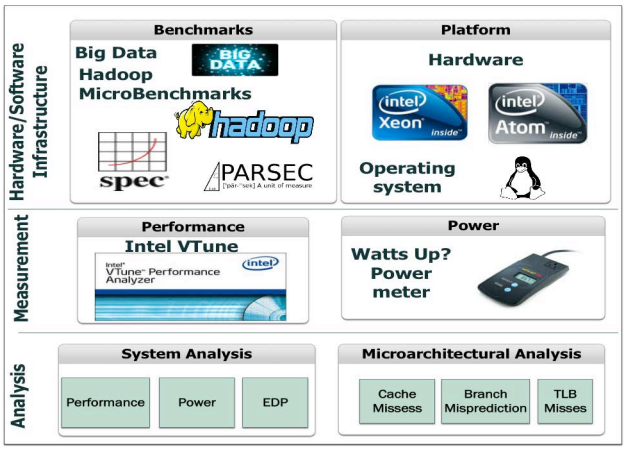


Figure 2. Methodology

5. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we discuss the system-level and micro-architecture-level analysis of big and little cores, when running traditional CPU benchmarks, parallel benchmarks, scale-out, and big data applications. Due to space constraints, we are only reporting the average results for SPEC, PARSEC and scale-out applications. Moreover, we have conducted the data size sensitivity analysis of Hadoop micro-benchmarks with the dataset of 10MB, 100MB, 1GB and 10GB per node to understand the impact of the size of data per server node on system-level parameters.

5.1 Performance Analysis

In this section, we analyze the performance measurements of big data applications in term of IPC and compare it with the traditional benchmarks. Figure 3.1 shows that the average IPC of big data is 1.65 times lower than the traditional CPU benchmarks on big core and 1.21 times on little core. Therefore, noticeably more performance drop (37%, on average) is observed for big data applications compared to traditional CPU applications when running on big core server compared to little core server. In general, we observe lower IPC in big data 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. Figure 3.2 shows the IPC of Hadoop micro-benchmarks for different data sizes. The results are consistent with the results in Figure 3.1 showing lower IPC on little core compared to big core across all data sizes. We also observe that on little core, increasing the data size reduces the IPC since the cache misses increases (mainly Icache miss). Little core, due to its low processing capacity (issue width of 2), cannot hide cache miss penalty as effective as big core. However, on big core while for most cases, increase in data size per node reduces the IPC, there are few exceptions where increasing the data size from 100MB to 1000MB per node increases the IPC. This is mainly due to higher cache locality as a result of larger and more complex cache subsystem in big core, which results in reduction in cache miss rates.

5.2 Power Characterization

Figure 4.1 shows the average dynamic power consumption of the studied applications on big and little core servers. The idle power of the servers is subtracted from the measured (run-time) power. Note that the power results reported are for the entire system, including core, cache, DRAM and on-chip communication buses. Big core consumes on average 35 Watts of dynamic power with the peak of 44 Watts in cluster application. Little core consumes much lower dynamic power as expected, ranging from 0.9 to 6 Watts with an average of 4.8 Watts. Figure 4.2 shows that the power consumption increases as the size of data per node increases in most cases across both big and little architectures. This is more noticeable in little core. While increasing in data size in little core reduces the IPC and therefore core power, it increases cache and off-chip traffic in DRAM and bus subsystem (see LLC MPKI reported in Figure 10). Therefore, for low-end little core where cache, DRAM and off-chip components are dominant power consumer (unlike high performance Xeon core), a clear rise in power consumption is observed as the size of data increases.

5.3 Energy-Efficiency Analysis

Based on the results of power consumption for both platforms, we have evaluated the trade-off between power and performance by investigating the EDP metric. Furthermore, we have explored the ED^2P and ED^3P to understand the impact of near real-time performance constraints on big data applications and how more constraints on performance affects the choice of most efficient server architecture. Figure 5.1 illustrates EDP, ED^2P and ED^3P ratio for big vs little cores. The ED^XP (X=1,2,3) results show that big core-based server is noticeably more efficient for traditional CPU applications compared with big data. Also, interestingly for scale out benchmarks, little core is always more efficient than big core for EDP, ED^2P and ED^3P metrics. For real-world big data applications EDP results show that the little core-based server is almost always a more efficient platform for CPU intensive applications. However, for heavy I/O intensive applications such as sort and terasort, for large data sizes (10000MB), the big core becomes more efficient than the little core in terms of EDP. Complex memory subsystem in big core along with higher processing capacity (2X more than little core) allow big core to be more effective in hiding the cost of high I/O communication in these applications and can explain why big core-based server is more efficient. However, with more near real-time performance constraints, i.e. for ED^2P and ED^3P metrics, big core becomes more efficient compared to little cores across most applications. Figure 5.2 presents the data sensitivity analysis of Hadoop micro-benchmarks. The increase in the data size, progressively makes the big core more efficient compared with little core, however the point where big core becomes more efficient than little core varies across applications and depends on the data size and the performance constraint.

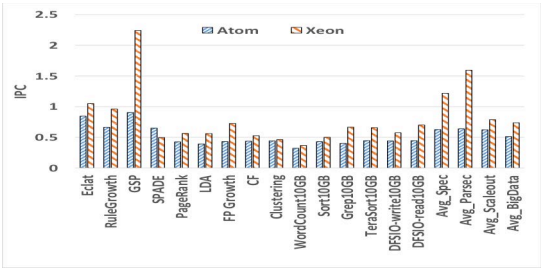
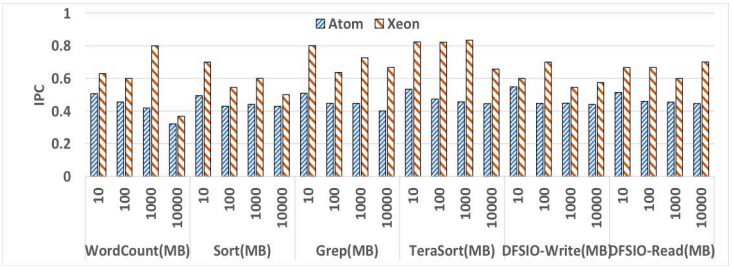


Figure 3. IPC (3.1) Big Data workloads



(3.2) Different configurations of Hadoop micro-benchmarks

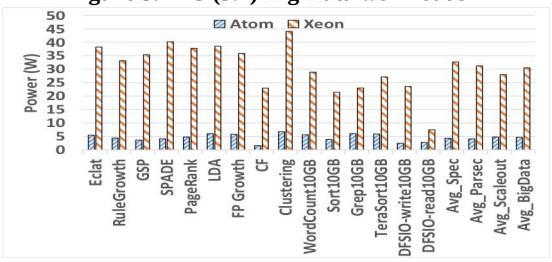
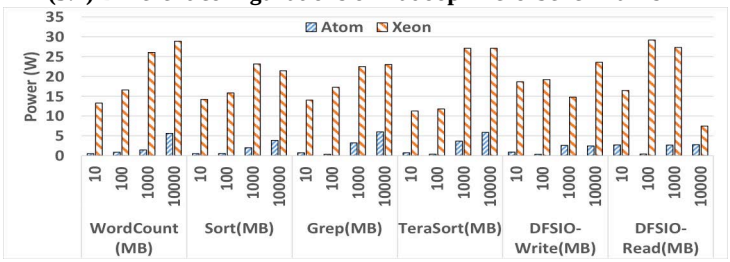
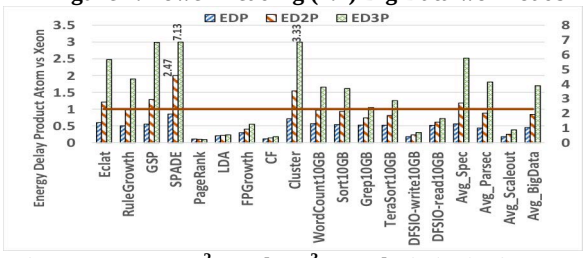


Figure 4. Power Reading (4.1) Big Data workloads



(4.2) Different configurations of Hadoop micro-benchmarks



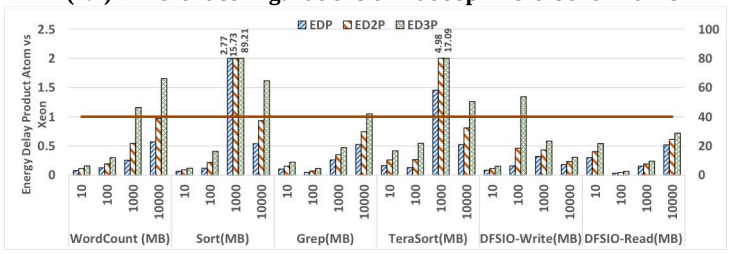


Figure 5. EDP, ED^{2}P and ED^{3}P Analysis (5.1) Big Data Workloads (5.2) Different configurations of Hadoop micro-benchmarks

The results illustrate that little core server is more efficient in terms of EDP for big data applications with the smaller data sizes. However, as the size of data increases and with more performance constraints big core server becomes more efficient.

5.4 Performance Hotspot and Post-Acceleration CPU code Characterization

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. In addition to big, medium, and small cores, the integration of domain-specific accelerators, such as GPUs and FPGAs has become extensive.

To find out the right server architecture for big data processing, it is important to understand how deploying an accelerator, such as FPGA, would necessitate adapting the choice of CPU. The post acceleration code characteristics are important to find the right architecture for efficient processing of big data 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 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 [15]. To perform hotspot analysis on big data applications, we use Intel Vtune to select the common hotspot modules of the applications running on big and little cores. First, we identify and analyze hotspot modules based on their execution time. Figure 6.1 shows the common hotspot modules of big data applications and Figure 6.2 presents Hadoop micro-benchmark hotspots with a data size of 1GB on Big and Little cores, respectively. Map and Reduce tasks represent the computation part to perform the task, such as grep, sort and etc. Libz is performing the compression and decompression task (library) for the Hadoop workload. Module dynamic contains hotspot functions such as java-finalize to perform the completion tasks of an object. Compiled java code includes the java.lang.string class to represent character strings along with the system.array, copy, math, abstractSringBuilder and object classes. Libpthread contained functions like mutex lock/unlock for the thread creation and protection and cond_wait functions to block on a condition variable.

We have also collected the IPC for each of these hotspot functions. Due to space limitation, we do not show the details of the IPC results. The results show that the performance gap between big and little cores for Map and Reduce task is 2X. This large gap shows that big core has a clear advantage over little core to run these hotspot functions.

Since the hotspot functions and their corresponding libraries are taking up most of execution time, they are candidate for acceleration, for instance with offloading mapper and reducer tasks to FPGAs [16, 17]. Several recent works have demonstrated the potential of offloading map and reduce tasks to FPGA platforms [18, 19, 20]. To analyze post-accelerated code and the choice of server architecture, we assume map and reduce tasks are offloaded to an accelerator [18, 19, 20]. We do not make any assumption

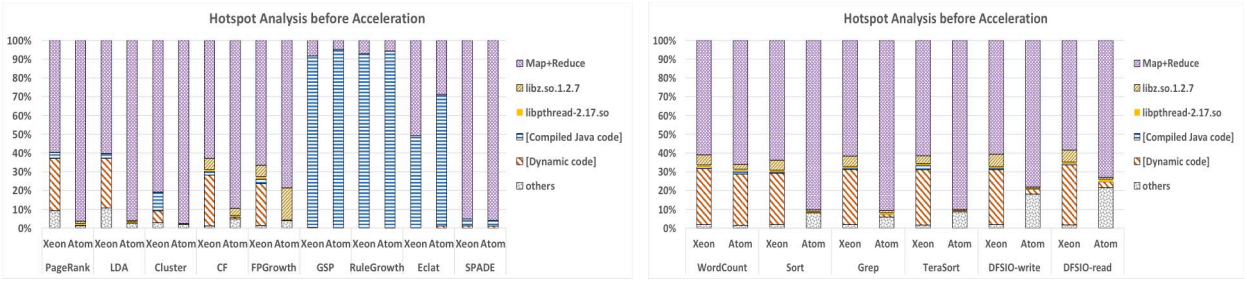


Figure 6. Hotspot Analysis before acceleration (6.1) Big Data workloads (6.2) Hadoop micro-benchmarks

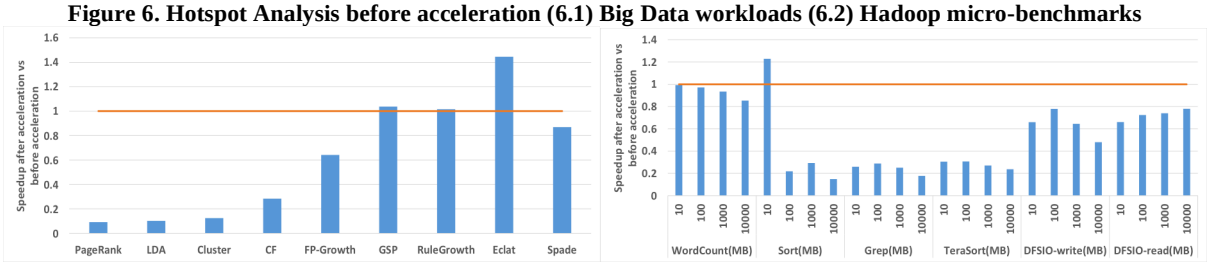


Figure 7: Speed up of Atom vs Xeon before and after acceleration (7.1) Big Data workloads (7.2) Hadoop micro-benchmarks

about the speedup gain on accelerator nor the cost of offloading to the accelerator. By taking out the time it takes to run hotspot function, we can study the remaining modules that are left for the big or little core-based server to run.

Figure 7 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

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

$(Exectime_{Atom} / Exectime_{Xeon})_{remaining\ code\ after\ acceleration}$ represents the speed up obtained by migrating the post-accelerated code from Atom to Xeon. $(Exectime_{Atom} / Exectime_{Xeon})_{entire\ applications}$ represents the speed up obtained by migrating the application from Atom to Xeon before acceleration. Using this equation, we can evaluate the impact of Atom over Xeon speedup gain after acceleration compared to speed up before acceleration. Most of the micro-benchmarks in Figure 7 have speed up less than 1 which indicates that speedup of Atom over Xeon after acceleration reduces compared to speed up before acceleration. We have also observed that most micro-benchmarks, with the increase in data size the speedup after acceleration reduces compared to the speedup before acceleration. However, there are several exceptions to this trend, including GSP, RuleGrowth, Ecalt, *WordCount, and Spade* where post acceleration code achieves higher speed up on Xeon over Atom compared with pre-acceleration code. Overall, Xeon provides a lower execution time, however, if speedup after acceleration is very small then considering the power consumption of Xeon, Atom-based will be a more efficient server to execute the post-accelerated code. Results show that the choice of big vs. little before and after accelerations is different. While most benchmarks clearly favor little core post acceleration, in several applications post accelerated code show higher speed up on big core-base server over little core-base server compared to pre-acceleration.

6. RELATED WORK

Recently, there have been a number of efforts to benchmark and characterize big data and cloud-scale applications, mainly on state-of-the-art high performance server platform. In general, there are two major approaches for benchmarking big data: A system benchmarking and a component benchmarking. A system benchmark is an end-to-end benchmarking, which includes the entire database and application software stack, including data preparation, data aggregation and data analytics [22]. A component benchmark encloses only a portion of the entire end-to-end system [22].

The most prominent big data benchmarks, include HiBench, Scale-Out, BigDataBench, CloudCmp, and LinkBench. HiBench [24] is a benchmark suite for Hadoop MapReduce. CloudCmp [25] use a systematic approach to benchmark various components of the cloud to compare cloud providers. LinkBench is a real-world database benchmark for social network applications [26]. The Transaction Procession Performance Council (TPC) has released a number of benchmark suites in recent years, including TPC-C, TPC-E, and TPC-DS for online transaction processing. BigDataBench [2] was released very recently and includes online service and offline analytics for web service applications. BigBench [27] is a new big data benchmark that adopts TPC-DS as its basis and expands it for offline analytics on Xeon high performance server. The CloudSuite [3, 4] benchmark was developed for Scale-Out cloud workloads and mainly includes small data sets, e.g., 4.5 GB for Naïve Bayes.

Several prior researches have characterized traditional CPU and parallel applications such as SPEC2006, PARSEC, and NAS on high performance server-class processors [21, 36]. It is important to also compare the characteristics of big data application with these traditional benchmark suites. We have included the SPEC CINT2006, SPEC CFP2006 and PARSEC 2.1 benchmarks for the comparison with BigData Workloads.

This work is different from all above benchmarking and characterization work as it perform a comprehensive system-level (power, performance, ED^XP) analysis of various big data applications and big data micro-benchmarks on two substantially different platforms one with high performance big core and another with low power little core to understand which of these two architectures is the choice for efficient big data processing.

There has been also a number of research into application-specific [28, 29] and domain-specific accelerators [30, 31, 32, 33]. Using tightly integrated FPGA [27, 34] with CPU, and GPU with CPU [1, 22], to accelerate big data processing have been proposed in recent work. While deploying programmable accelerator is a new and hot research topic, there has been little attention paid to how

CPU designs should be adapted to this change. To the best of our knowledge, the only work on this topic is by Arora [23], 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. They found that the post-GPU code has a lower ILP, higher branch miss prediction rate, and larger number of load and stores, and benefits less from multiple cores, as there is less TLP after GPU offloading. 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.

7. CONCLUSIONS

Heterogeneous chip architectures have emerged as an effective solution to address the power efficiency challenges the computer industry is facing. This work studied whether this is true for emerging big data applications. In this paper, we present a comprehensive system level analysis of big data applications on two distinct server platforms; the conventional server design approach, using a high performance big Xeon core; and the new trajectory in server design, using little Atom core, which advocates the use of a low-power core to address the power challenge. The characterization results show significantly larger performance drop (37%, on average) for big data applications compared to traditional CPU applications when running on big core server compared to little core server. Big core-based server provides a high performance, compared to little core, however, it is not as power efficient. Little core-based server is more efficient in terms of EDP for big data processing with small data sizes. However, as the size of data increases and with performance constraints, big core becomes an efficient choice. Overall the performance gap between these two server architectures while it is large for traditional CPU applications, it is becoming smaller for emerging big data applications. In addition, we performed the post-acceleration CPU code analysis to find out the most efficient server architecture to process the remaining code of big data applications after acceleration. The results show that there is a difference between the choice of big vs. little core-based server before and after accelerations. While most benchmarks clearly favor little core post acceleration, several applications show higher speed up on big core over little core post acceleration compared to pre-acceleration.

Overall, the results suggest that a heterogeneous architecture combining big and little cores is required for efficient processing of emerging big data analytics applications.

8. REFERENCES

[1] Wu, Ren, Bin Zhang, and Meichun Hsu. "GPU-accelerated large scale analytics." IACM UCHPC (2009).

[2] Gao,W. “BigDataBench: a Big Data Benchmark Suite from Web Search Engines”.ASBD 2013 in conjunction with ISCA 2013

[3] Ferdman, M., et al. "Clearing the clouds: a study of emerging scale-out workloads on modern hardware." ACM SIGARCH Computer Architecture News40.1 (2012): 37-48.

[4] Ghazal, A. “Bigbench: Towards an industry standard benchmark for big data analytics”. In: ACM SIGMOD Conference (2013)

[5] Gutierrez, A. et al. "Integrated 3D-stacked server designs for increasing physical density of key-value stores." ASPLOS. 2014.

[6] Reddi, V. J., et al. "Web search using mobile cores: quantifying and mitigating the price of efficiency." ACM SIGARCH Computer Architecture News38.3 (2010): 314-325.

[7] Andersen, D. G. et al. “FAWN: A Fast Array of Wimpy Nodes”. In the Proceedings of ACM SIGOPS 22nd SOSP, pages 1–14, 2009.

[8] Hardavellas, Nikos, et al. "Toward dark silicon in servers." IEEE Micro 31.EPFL-ARTICLE-168285 (2011): 6-15.

[9] Neshatpour, Katayoun, et al. "Energy-efficient acceleration of big data analytics applications using FPGAs." Big Data (Big Data), 2015 IEEE International Conference on. IEEE, 2015.

[10] Kumar, Rakesh, et al. "Heterogeneous chip multiprocessors." Computer 11 (2005): 32-38.

[11] Kontorinis, Vasileios, et al. "Enabling dynamic heterogeneity through core-on-core stacking." Proceedings of the 51st Annual Design Automation Conference. ACM, 2014.

[12] Homayoun, Houman, et al. "Dynamically heterogeneous cores through 3D resource pooling." High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on.

[13] Intel VTune Amplifier XE Performance Profiler. http://software.intel.com/en-us/articles/intel-vtune-amplifier-xe/

[14] Homayoun, Houman, et al. "Reducing execution unit leakage power in embedded processors." Embedded Computer Systems: Architectures, Modeling, and Simulation. Springer, 2006. 299-308

[15] Nilakantan, S., et al. "Platform-independent analysis of function-level communication in workloads." IISWC, IEEE, 2013.

[16] Neshatpour, Katayoun, Maria Malik, and Houman Homayoun. "Accelerating machine learning kernel in hadoop using fpgas." Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on. IEEE, 2015.

[17] James T Kukunas, et al. “High Performance ZLIB Compression on Intel®Architecture Processors”, White paper, April 2014.

[18] Shan, Y., et al. “FPMR: Mapreduce framework on FPGA,” in Proc ACM/SIGDA Int Symp Field Programmable Gate Arrays, 2010.

[19] Neshatpour, Katayoun, et al. "Accelerating big data analytics using fpgas." Field-Programmable Custom Computing Machines (FCCM), 2015 IEEE 23rd Annual International Symposium on. IEEE, 2015.

[20] Z. Lin and P. Chow, “Zcluster: A zynq-based hadoop cluster,” in Int. Conf. FPT, Dec 2013, pp. 450–453.

[21] T. K. Prakash et al. Performance Characterization of SPEC CPU2006 Benchmarks on Intel Core 2 Duo Processor. In Transactions on Computers and Software Engineering, No. 1, Vol 2, pp. 36-41, 2008.

[22] Baru, C., et al. “Setting the Direction for Big Data Benchmark Standards”,Lecture Notes in Computer Science

[23] Arora, Manish, et al. "Redefining the Role of the CPU in the Era of CPU-GPU Integration." Micro, IEEE 32.6 (2012): 4-16.

[24] Huang, S., et al. "The HiBench benchmark suite: Characterization of the MapReduce-based data analysis." In the proc. of ICDEW, 2010

[25] Li,A.,et al. “CloudCmp: comparing public cloud providers.” ACM,’10

[26] Armstrong, et al. "Linkbench: a database benchmark based on the facebook social graph." Proceedings of the ACM SIGMOD, 2013.

[27] Xi Luo, Walid A. Najjar, Vagelis “Hristidis: Efficient near-duplicate document detection using FPGAs”. BigData 2013

[28] YU, P., et al. “Scalable custom instructions identification for instruction-set extensible processors”. In Proc. of the CASES’04. ACM, New York.

[29] YU, P. et al. “Disjoint pattern enumeration for custom instructions identification”. In Proceedings of the FPL’07, 273–278.

[30] ARNOLD, M. et al. “Designing domain-specific processors.” In Proceedings of the 9th CODES. ACM 2001.

[31] Clark, N. T., et al. "Automated custom instruction generation for domain-specific processor acceleration." Computers, IEEE Transactions on 54.10 (2005): 1258-1270.

[32] Homayoun, Houman, and et. al. "ZZ-HVS: Zig-zag horizontal and vertical sleep transistor sharing to reduce leakage power in on-chip SRAM peripheral circuits.", 2008, ICCD, IEEE International Conference on Computer Design.

[33] Arora, N, et al. "Instruction selection in asip synthesis using functional matching." VLSI Design, 2010.

[34] Chung, E. S., et al. "Linqits: Big data on little clients." ACM SIGARCH Computer Architecture News. Vol. 41. No. 3, 2013

[35] http://www.chipestimate.com/tech-talks/2013/07/16/Cadence-5-Emerging-DRAM-Interfaces-You-Should-Know-for-Your-Next-Design-

[36] S. Bird, et al. Performance Characterization of SPEC CPU Benchmarks on Intel's Core Microarchitecture based processor, in Proceedings of 2007 SPEC Benchmark Workshop, Jan 2007.