Maria Malik and Houman Homayoun, Under Review

Big or Little Cores for Big Data?

Performance and Power Characterization of BigData on Intel™ Xeon and Intel™ Atom

Abstract— Emerging Big Data applications require a significant amount of server computational power. Big data applications require computing resources that can efficiently scale to manage massive amounts of diverse data. However, the rapid growth in the data yields challenges to process data efficiently using current servers. Furthermore, physical design constraint- power and density, have become the dominant limiting factor for scaling out data centers. In this study, we analyze the performance and power characterization of the BigData applications on two state-of-the-art servers, Intel Xeon (high performance server) and Intel Atom (low power server). The results presented in this study provide essential architectural insight and will help architects to address the new challenges presented in BigData workloads.

***Index Terms*—Performance, Characterization, BigData, Intel VTune, High-Performance server, Low-Powered server.**

I. 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 BigData is changing constantly and momentously from the amount of data that is generated to the way in which it is being analyzed and used for decision-making. BigData applications with their volume and high processing demand requires specialized design and further investigation in terms of computing hardware, data storage, information extraction algorithms. However, big data applications, in particular from the web service domain, share many inherent characteristics that are fundamentally different from traditional desktop, parallel, and scale-out applications [3, 6]. This new set of characteristics is necessitating a change in the direction of server-class microarchitecture to improve their computational and memory efficiency.

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 are now having access to enormous collections of big data, empowering them to build their own analytics. Different processors have their own processing specialties, which presents distinct behaviors when dealing with various categories of applications. In this work, our goal is to identify the right computing platform for BigData workloads and find hardware resources that are the bottleneck for the big data applications.

A first step to understand the computational requirements of big data applications is to characterize them effectively. The characterization information allows us to find the bottleneck in current servers when running big data, thus allowing us to modify their underlying micro-architecture for performance and power-efficiency improvement. Characterizing BigData applications will help computer architects to optimize applications that process huge data and to customize the hardware design. In general, BigData benchmarking is challenging, mainly due to the diversity of applications, the size and complexity of applications, the security and privacy issues related to real-life data sets. Although there has been a number of recent work on benchmarking big data, still it is an open question whether a single set of benchmarks can be representative to all of these fields, or whether different groups will require different benchmarks [1]. To accomplish this goal we have investigated the performance characterization of BigData workloads to analyze their behavior using two state-of-the-art servers; Xeon and Atom. These two types of servers represent two schools of thought on server architecture design: using Xeon, which is conventional approach to design high-performance server, and the Atom, which is a new direction in server design that advocates the use of a low-power core to address the dark silicon challenge facing servers. This particular behavioral information will allow architecture designers to modify the underlying micro-architecture for performance and power innovations required in microprocessors design as compared to today's modern processors.

Recently, there have been a number of efforts to benchmark BigData and cloud suite applications [3, 9]. While all these benchmark suites are important to evaluate computer system optimization for BigData. They do not cover all aspects of BigData- namely diversity in workload and data type. In particular, these benchmarks mainly focus on web service applications or internet services. Additionally, BigData applications share many inherent characteristics that are fundamentally different from traditional desktop and parallel applications. Therefore, we compare the behavioral characteristics of BigData workload to scale-out applications and the existing well known traditional benchmark suites like SPECINT2006, SPECFP2006, and PARSEC. This comparison provides the comprehensive information of similarities and differences of big data workload among the other benchmarks.

In addition, we present the power characterization of BigData applications on both servers; Xeon and Atom. Power dissipation becomes important concern of system configuration in modern architecture design. We believe this is the right time to exploit the heterogeneous multicore systems. These systems contains multiple cores configured with distinct abilities. Assign the applications on the efficient core that can satisfy its performance requirements, as computational intensive applications will run on Xeon (big core; high performance core) and other applications will run on Atom (little core; power efficient core). Assigning the computational intensive applications to big core and other executions on the little core will improve performance and consequently energy on architectures. This study assists in determining whether heterogeneous multicore systems is the solution to handle big data workloads performance and energy consumption efficiency.

The main contributions of the paper are as follows:

- We exploit the performance characterization of BigData applications using hardware performance

counter on Atom and Xeon.

- We present the power characterization using performance/watts metric and power traces on Atom and Xeon.

Rest of the paper is organized as follows. Section II provides the related work. Section III describes the brief description of the BigData workload, selected scale-out workloads and Traditional benchmark, which we have run on Intel Xeon and Intel Atom server. Our methodology and experimental setup details are presented in section IV. Section V presents the results. Lastly, we have presented the brief conclusion acquire from our analysis.

II. Related Work

There are two major approaches for benchmarking big data: system benchmarking and component benchmarking. A system benchmark is end-to-end benchmarking, which includes the entire database and application software stack, including data preparation, data aggregation and data analytics [1]. Big data applications are still evolving quickly and continuously; therefore, creating a single end-to-end benchmark suite to represent the evolving applications is difficult. A component benchmark encloses only a portion of the entire end-to-end system [1]. This feature of component benchmarks makes it relatively easier to deploy and test, as they do not need to interact with other system components. On the other end component benchmark do not provide the same level ofinformation as an end-to-end benchmark.

Recently, there have been a number of efforts to benchmark and characterize big data and cloud-scale applications. HiBench [5] is a benchmark suite for Hadoop MapReduce, and Hive. It targets the Hadoop mapreduce framework for performance characteristics of Hadoop clusters using HiBench [5]. However, our study provides the micro-architectural rather than system-side characterization. CloudCmp [7] 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 [8]. The Transaction Procession Performance Council (TPC) have released a number of benchmark suites in recent years, including TPC-C, TPC-E, and TPC-DS for online transaction processing. The closest one to our work is BigDataBench [6], which was released very recently and includes online service and offline analytics for web service applications. Although this is an important step toward benchmarking big data, it does not encompass key emerging areas of big data analytics or diverse data types, such as image, audio, and video. Also BigDataBench has been characterized on high performance architectures to better understand the limitations of today's servers. BigBench [9] is a new big data benchmark that adopts TPC-DS as its basis and expands it for offline analytics. Currently, BigBench is not open-source; therefore, its usage, evaluation, and further adaptation by the community is unclear. The CloudSuite [3, 9] benchmark was developed for Scale-Out cloud workloads and mainly includes small data sets, e.g., 4.5 GB for Naïve Bayes. In addition, the Scale-Out workload mainly covers the classification workloads for big data mining and it misses other key data science cases, such as collaborative filtering, clustering, and frequent item-set mining. Moreover, CloudSuite does not cover a diverse range of real-world data sets and workloads, such as online big data analytics. We have also included these applications in our studies. In addition, we have also included more performance parameters like branch misprediction, ITLB and DTLB miss rate to provide more elaborated insight.

Several prior researches have characterized traditional CPU and parallel applications such as SPEC2006, PARSEC, and NAS on high performance server-class processors [10,11,12,13,14,15]. 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.

III. Dominant BigData Workloads

The studied BigData workloads in this paper are representative algorithms from the 15 different domains like graph mining, data mining, data analysis platform and pattern

Table 1: Studied BigData 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 0.6
			Clustering		Graph	
	E-commerce	Offline Analytics	Frequent Pattern Mining	Structured		
			Frequent Item Mining	Structured	Table	SPMF 0.96
			Sequential Rule Mining			
			Sequential Pattern Mining			
			Classification	Semi-structured	Text	Hadoop 1.2.1, Mahout 0.6, CloudSuite 2.0
	Client Server Application	Online Services	Memcached	Unstructured	Table	Memcached server/client 1.4.15, CloudSuite 2.0

searching applications, which are frequently occurred in real world. We provide these selected applications, along with their particular domain and data type in Table 1.

A. BigData Workload

Hadoop Microbenchmark. Apache Hadoop is an open-source Java-based framework of MapReduce implementation. It assists the processing of large datasets in a distributed computing environment and stores data in highly fault-tolerant distributed file system, HDFS. Hadoop have the numerous micro-benchmarks; from which we have included five in our study to stress test a Hadoop application [28].

- *WordCount* reads text files and determines how often the words appear in a set of files. Mapper proceeds with a line as input and fragment it into words. Reducer sums the counts for each word and provides the aggregated value for each word.
- *Sort* uses the map/reduce framework to sort the input directory into the output directory. The inputs and outputs must be sequence files where the keys and values are BytesWritable (a kind of class in Hadoop).
- *Grep* firstly extracts matching strings provided by user against text files. Secondly, it sorts matching strings by their frequency and stores the output in a single output file.
- *TeraSort* sorts any amount of data as fast as possible to test the CPU/Memory power of the cluster. TeraGen command generates the data for the TeraSort. This micro-benchmark combines examining the HDFS and MapReduce layers of a Hadoop cluster.
- *TestDFSIO-write/read:* TestDFSIO benchmark is a storage throughput test that is divided into two parts, TestDFSIO-Write and TestDFSIO-Read.
  - TestDFSIO-Write writes the data to HDFS
  - TestDFSIO-Read reads the earlier written data from HDFS

  TestDFSIO-read does not generate its own input files. For this reason, we have performed the write test and then follow up with a read test.

GraphMining. Graph construction can be very challenging because of complex iterative and data-dependent nature of graph. Hadoop is well-suited for this task but required expertise to handle graph complexities. GraphBuilder address this gap by providing a scalable graph construction software library for Hadoop. This benchmark performs parallel graph construction including graph formation, tabulation, compression, transformation, partitioning, output formatting and serialization, transformation and verifications that are convenient for graph mining. GraphBuilder constructs graphs for PageRank and LDA algorithms implemented on PowerGraph.

- *PageRank graph* is a directed graph with vertices refering documents and edges representing hyper-links. This application constructs a list of all Wikipedia documents based on to their hyper-link ranking within the Wikipedia site [17].
- *LDA (Latent Dirichlet Allocation) graph* is a bipartite graph connecting documents to words where edge weights equal to the frequency of each word in each corresponding document [17].

Collaborative Filtering (CF) Recommendation is a technique used by many recommender systems to predict the preference of user based on their previous rating history. This application finds similarities between different items in the dataset using mathematical formulations like *similarity_euclidean_distance* to building a database of preference for items [21, 24, 25].

Clustering is one of the fundamental tasks in Data Mining. Cluster assembles data items into group based on their similar features. We have analyzed meanshift clustering as it is a nonparmetric clustering technique that does not require prior knowledge of the number of clusters [22].

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 [23]. FP-Growth endures high communication cost and restricts the percentage of computation that can be parallelized.

SPMF (Sequential Pattern Mining Framework is an open-source data mining library written in Java. It offers numerous data mining algorithms for sequential pattern, rule mining and frequent item mining [16].

- *Frequent Item Mining.*
  Equivalence Class Transformation (Eclat) observes frequent item-sets in a transaction database. It uses depth-first search for discovering frequent item-sets instead of breath-first search.
- *Sequential Rule Mining.*
  RuleGrowth explores sequential rules that appear in sequence databases. It is fast and memory efficient algorithm.
- *Sequential Pattern Mining*
  i) *Generalized Sequential Pattern (GSP)* discovers frequent sequential patterns in sequence databases using Apriori-like approach.

  ii) *Sequential Pattern Discovery using Equivalence classes (SPADE)* determines all frequent sequential patterns occurring in a sequence databases. It reduces I/O costs by minimizing database scans and decrease the computational costs by using efficient search schemes.

B. Scale-out Workloads

Classification technique learns from the existing categorizations and groups the unclassified items to the best corresponding category [24, 25, 27].

Graph-analysis is performed by implementing the TrunkRank on GraphLab [3, 27]. This application studies the impact of a Twitter user for graph analysis.

DataCaching. Memcached is a high-performance, general-purpose distributed memory caching system. It uses in-memory key-value storage mechanism for small chunks of arbitrary data API calls or results of database calls [26, 27]. Memcached data caching server, imitates the action of a Twitter caching server using the twitter dataset.

C. Traditional Benchmarks

SPEC CPU2006. SPEC CPU Benchmark suite is one of the most frequently used suites for computer architecture research. These workloads are designed to stress the hardware of the machines by having computer intensive workload based on the real applications. SPEC CPU 2006 is classified into two groups: SPEC CINT2006; representing integer computation and SPEC CFP2006; corresponding floating point computation.

PARSEC 2.1. Princeton Application Repository for Shared-Memory Computers (Parsec) is open-source parallel benchmark suite for evaluating multi-core and multiprocessor systems. This particular benchmark suite assembles diverse and large workloads of scientific acquisition. The applications exhibit various memory behaviors, different workload partitions and different data sharing patterns.

IV. 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 includes six aggressive processor cores per node with three-level cache hierarchy: L1 and L2 cache are private to each core while LLC (L3) is shared among all cores. Secondly, Intel Atom C2758 has 8 processor cores per node and two level cache hierarchy. Table 2 summarizes the key architectural parameters of the system.

TABLE 2. ARCHITECTURAL PARAMETERS

Processor	22nm Intel Atom C2758 @ 2.40GHz	32nm Intel Xeon E5-2420 @ at 1.9GHz
Micro-architecture	Silvermont	Sandy Bridge
L1d Cache	24 KB	32 KB
L2 Cache	1024 KB	256KB
LLC (L3) Cache	-	15MB
System Memory	8GB	32GB

We analyze architectural behavior using Intel VTune [Itel 2013], a graphical performance-profiling tool that provides an interface to the processor performance counters [2]. Moreover, we have used Watts Up PRO power meter to measure the power consumption [18].

A. BigData Workload Experimental setup.

a) *Hadoop micro-benchmarks.* We benchmark Hadoop 1.2.1 and generate random data on HDFS. Data set of size 10MB, 100MB, 1GB and 10GB per node are studies to observe the data sensitivity analysis of Hadoop micro-benchmarks.

b) Graphbuilder. We benchmark graphbuilder 0.0.1 on Hadoop 1.2.1 to construct graphs for PageRank and LDA.

c) Collaborative Filtering. We benchmark CF recommendation algorithm from the mahout 0.9 library on Hadoop 1.2.1. The data set is obtained from MovieLens website that consists of 100,000 ratings (1-5) from 943 users on 1682 movies.

d) *Clustering*. We benchmark meanshift clustering algorithm from the mahout 0.4 library on Hadoop 1.2.1. The data set is collected from UCI Machine Learning Repository.

e*) Association Rule Mining.* We benchmark FP-Growth algorithm from the mahout 0.6 library on Hadoop 1.2.1. The dataset is obtained from [19], which contains traffic accident information.

f) SPMF. We benchmark the spmf v0.96 with the dataset accidents.txt and Kosarak_converted.txt obtained from the Frequent Itemset Mining Dataset Repository [19].

B. Scale-out Workloads experimental setup.

a) Classification. We benchmark Bayesian classification algorithm from the mahout 0.4 library on Hadoop 1.2.1, running. This algorithm accepts the Wikipedia pages and guess the country tag of each article to Wikipedia pages.

b) Graph-Analysis. We benchmark the trunkrank implemented on GraphLab 2.1 [27]. This algorithm works with the twitter data-set of 25GB with 41M vertices.

c) Data-Caching. We benchmark the memcached v-1.4.15 with the prerequisite libraray Libevent v2.0.21. This application is configured for single server with the twitter dataset of 1.2GB size and 90% number of requests per second.

C. Traditional Benchmark Experimental Setup

a) SPEC2006: We benchmark the SPEC2006 with the first reference input.

b) Parsec 2.1: We benchmark Psarsec 2.1 with the native input.

V. Experimental Results and Analysis

In this section, we evaluate micro-architectural behavior of BigData workloads and compare it with the traditional benchmark workloads and Scale-out (CloudSuite) applications on Xeon and Atom server. Due to the space constraints, we are just showing the results of BigData applications along with the average values of Spec, Parsec and Scale-out applications. Moreover, we have conducted the data size sensitivity analysis

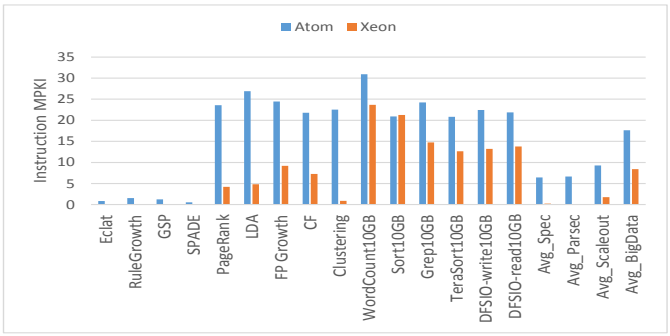


Figure 1.1: Instruction MPKI of BigData workloads

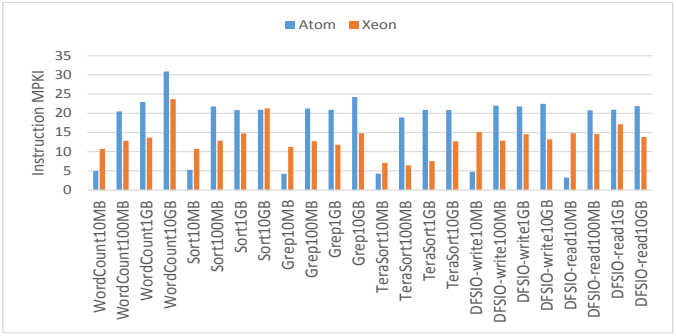


Figure 1.2: Instruction MPKI of different configurations of Hadoop micro-benchmarks

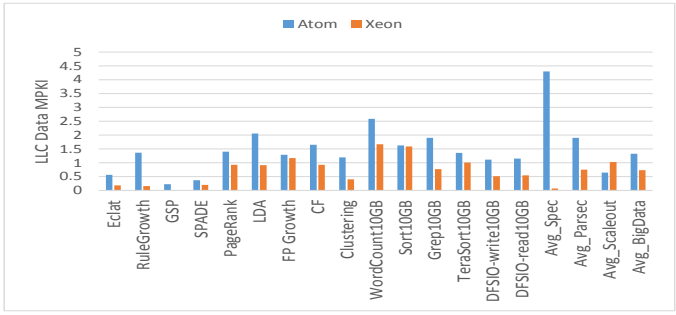


Figure 1.1: LLC Data MPKI of BigData workloads

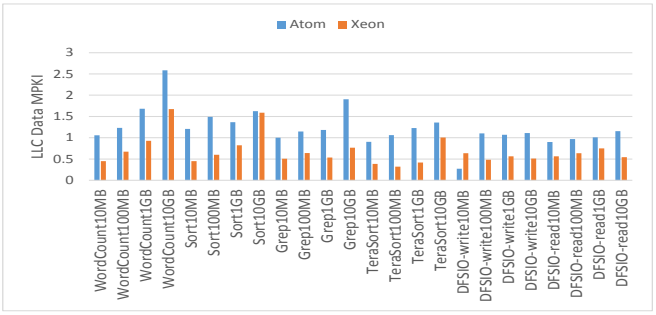


Figure 2.1: LLC Data MPKI of different configurations of Hadoop micro-benchmarks

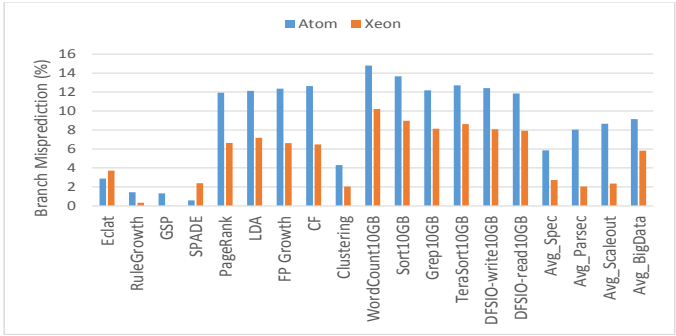


Figure 3.1: Branch Misprediction of BigData workloads

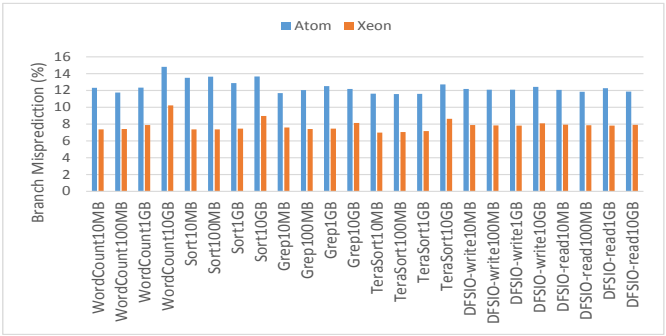


Figure 3.2: Branch Misprediction of different configurations of Hadoop micro-benchmarks

of Hadoop micro-benchmarks with the dataset of 10MB, 100MB, 1GB and 10GB. The details of the observed parameters are explained below:

A. Instruction cache misses

Frequent cache misses decreases the fetch rate and introduces the stalls in the frontend of the pipeline. Fetch rate reduction constraints the number of instruction that frontend delivers to the backend for ILP extraction and therefore increases the chance of the whole pipeline to be stalled.

Figure 1.1 presents the Instruction cache misses per kilo instructions (MPKI) on Xeon processor and Atom processor. We leave out the L1-I cache misses of the Xeon as it has 2 level instruction cache and L2-I misses experience high miss penalty. The average L-I cache MPKI of BigData applications is at least 2.6 times higher than the traditional benchmarks and 1.86 times than the scale-out applications on Atom, while on Xeon it is 38 times higher than traditional benchmarks and 4.76 times higher than scaleout applications. In comparison to Atom and Xeon, Atom shows two times higher average L-I cache MPKI of BigData applications than Xeon. Figure 1.2 shows data sensitivity analysis of Hadoop micro-benchmarks. We have observed that except 10MB data size all other data sizes have high instruction MPKI on Atom. Data sensitivity analysis results also shows increase in instruction MPKI as the data size increases on both Xeon and Atom.

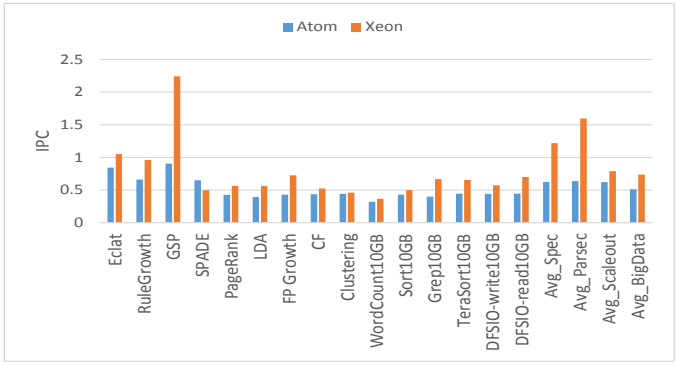


Figure 4.1: IPC of BigData workloads

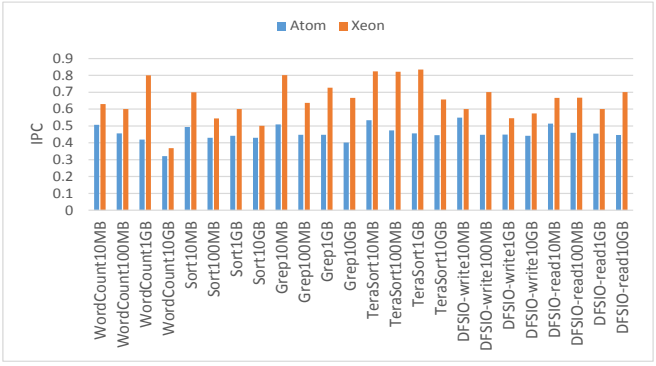


Figure 4.2: IPC of different configurations of Hadoop micro-benchmarks

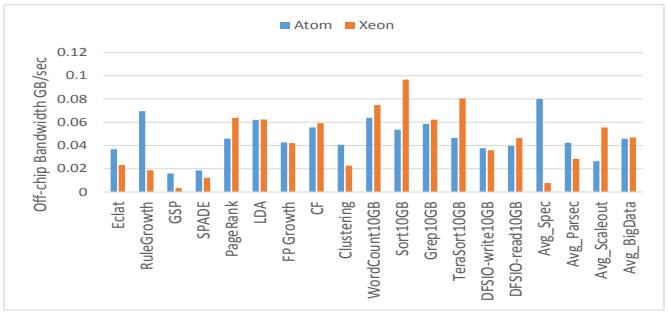


Figure 5.1: Off-bandwidth utilization of BigData workloads

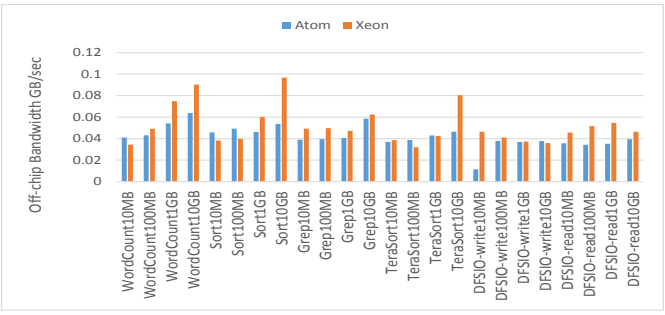


Figure 5.2: Off-bandwidth utilization different configurations of Hadoop micro-benchmarks

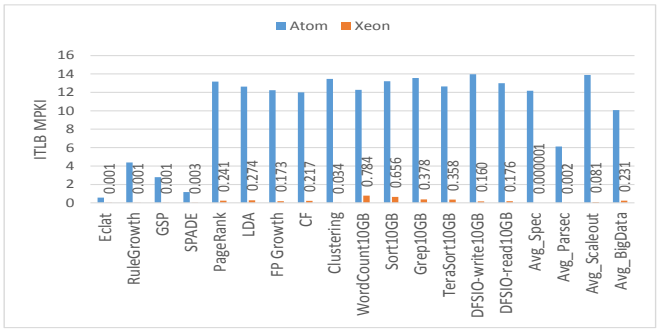


Figure 6.1: ITLB MPKI of BigData workloads

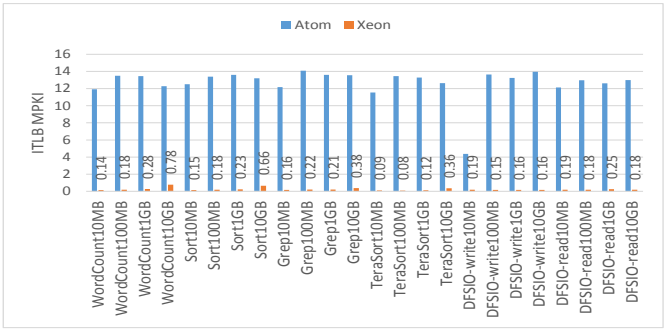


Figure 6.2: ITLB MPKI of different configurations of Hadoop micro-benchmarks

Observation. The general observation is that the instruction working sets of most BigData workloads exceed the L1 capacity of both Xeon and Atom processor. Fast cache access demand restrains the architect designer to increase the instruction cache size. However, Xeon and Atom comparison shows that two level instruction cache hierarchy mitigate instruction cache MPKI. Increase in data size results increasing the number of dynamic instructions (as the number of iteration in many loops in the code increases). These reasons lead to higher L1 instruction misses as the size of input data increases.

Current prefetchers rely on the repetitiveness and sequential behavior of the instructions to predict the same sequence in future. The possible main factors leading to the high LI misses are the huge code size and deep software stack of the big data workloads. This behavior implies that better LI cache performance is demanded for the BigData workloads. Moreover, modern processors need to include the instruction prefetchers that can predict complex patterns with the advance replacement policy to eliminate the wasted cycles caused by front-end stalls.

B. Data Cache misses

Intel Xeon have three-level cache hierarchy and Intel Atom has two-level hierarchy to reduce the gap between processor and memory speed. In this study, we have analyzed the LLC of both processors; L3-D cache level of Xeon and L2-D cache level of Atom for the fair comparison. Figure 2.1 shows the LLC MPKI on Atom and Xeon of BigData applications along the average data MPKI of Spec, Parsec and Scale-out. Firstly, the average LLC data cache MPKI of BigData is at least 2.5 times higher than the traditional benchmarks and 3.8 times higher than Scale-out applications on Xeon. In contrast, Atom shows the opposite behavior; SPEC experiences 3.25 times higher data MPKI than big data on Atom. Secondly, in comparison to Atom and Xeon, average LLC data cache MPKI

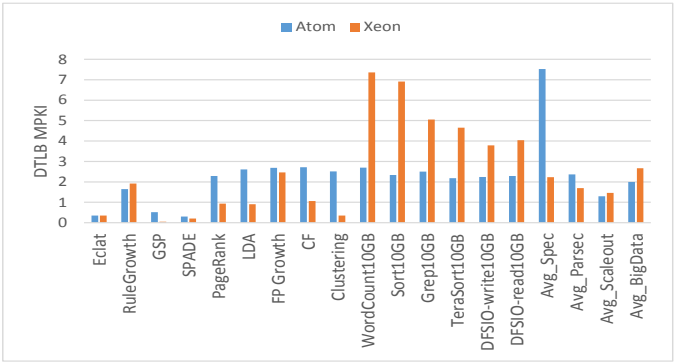


Figure 7.1: DTLB MPKI of BigData workloads

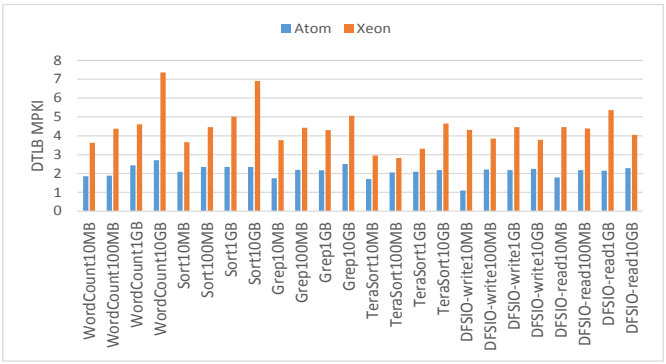


Figure 7.2: DTLB MPKI of different configurations of Hadoop micro-benchmarks

of BigData on Atom is 1.8 times higher than Xeon.

Figure 2.2 shows the data sensitivity analysis of Hadoop micro-benchmarks on Xeon and Atom. Data MPKI on Atom increases with the increase in data size however Xeon shows no clear trend. In comparison to Atom and Xeon; the results for the data size except 10MB are consistent with the Figure 2.1 by having high instruction MPKI on Atom.

Observation. We conclude that BigData applications might have good data locality or there are fewer computation operations to memory access compared with traditional benchmarks corroborating the observation in [3]. Furthermore, this observation shows that a small 1MB LLC cache in Atom is sufficient for the BigData workloads. Three-level cache hierarchy with the large size in Xeon processor certainly has the advantage on Atom processor. But for the power sensitive and area constraint applications, atom illustrates small data MPKI difference against Xeon while having small, two-level cache hierarchy. In addition, we observed that while for most cases increasing data size increases LLC MPKI, there are exceptions that require careful consideration when choosing the size of data to be allocated to each node. Modern processor can generally only handle very regular access pattern. Architects need to consider the prefetchers that can anticipate the complex pattern to reduce data cache stalls.

C. Branch Misprediction.

Figure 3.1 presents the branch misprediction of BigData applications along the average branch misprediction of Spec, Parsec and Scale-out applications. Figure 3.2 shows the branch misprediction of the Hadoop micro-benchmarks with different data sizes. The average misprediction of BigData is 9.14, while that of Spec, Parsec and scale-out are 5.84, 8.01 and 8.66 respectively on Atom processor. In addition, on Xeon, average misprediction of BigData is 5.82, while that of Spec, Parsec and scale-out are 2.72, 2.04 and 2.34 respectively. This shows that misprediction of BigData is at least 1.2 times higher on Atom and 2.13 times higher on Xeon than the traditional benchmarks. In comparison to Atom and Xeon, atom experiences 1.5 times higher branch misprediction than Xeon processor. Figure 3.2 results are consistent by having the higher branch misprediction rate on Atom as compared to the Xeon.

Observation. Traditional CPU and parallel benchmarks are mainly having instruction working sets with tight loop (as in matricx algebra), making them easy for the branch predictor to predict correctly and reduce the mispredictions. In BigData workloads however loops are seldom, and instead they have abundant fetch record (LD), Match Key (CMP), and non-loop branches (Branch to handler or BC) operations. This results in higher branch mispredicitons rate affects the application performance by creating the stall in the pipeline.

D. Instructions per cycle (IPC).

Figure 4.1 and 4.2 presents the processor's performance for big data applications along the average value of Spec, Parsec and scale-out applications on both processors Xeon and Atom in terms of IPC. The average IPC of BigData is at least 1.65 times lower than the traditional benchmarks on Xeon and 1.21 times on Atom. In general, we have observed lower IPC in BigData applications compared with the traditional benchmarks. Furthermore, Atom is experiencing 1.43 times lower IPC in comparison to Xeon. Xeon can fetch upto 4 instructions simultaneously but Atom is limited to fetch 2 instructions per cycle. Figure 4.2 results are consistent with the last statement by having the low IPC on Atom verses Xeon. In comparison to traditional benchmarks, BigData workloads do not yield significant performance benefit, as the average IPC is low and there is not much variation across this class of applications.

Observation. This behavior is consistent with what we have observed in the Figure 1.1 and Figure 3.1. Performance of BigData applications is impacted by high instruction cache misses and branch misprediction. This might also indicates that parallelism of these workloads is not fully exploited. Figure 4.2 shows the IPC values of Hadoop micro-benchmarks with the increasing data size. We have observed that increase in data size have reduces the IPC as it increases cache misses.

E. Bandwidth:

In Figure 5.1, we have plotted the off-chip bandwidth utilization of BigData workloads, Spec, Parsec and scaleout applications. On average the BigData workloads are utilizing higher DRAM bandwidth in comparison to traditional benchmarks on Xeon architecture, however, we have observed the opposite trend on Atom. We calculate the off-chip

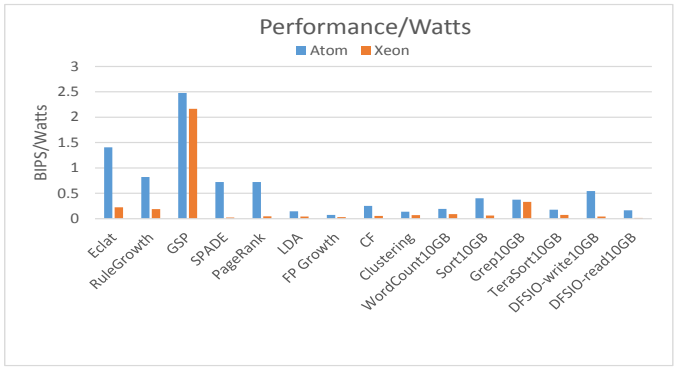


Figure 8.1: Performance/Watts of BigData Workloads

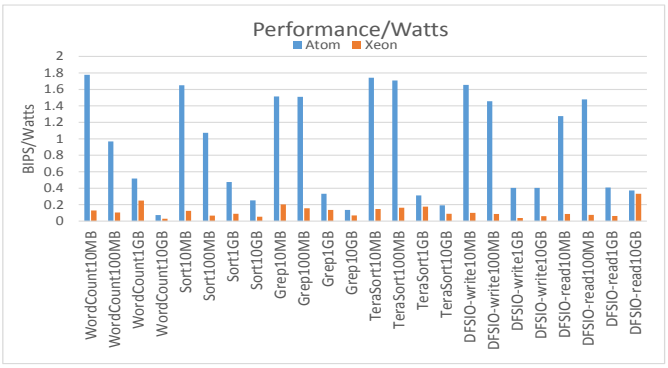
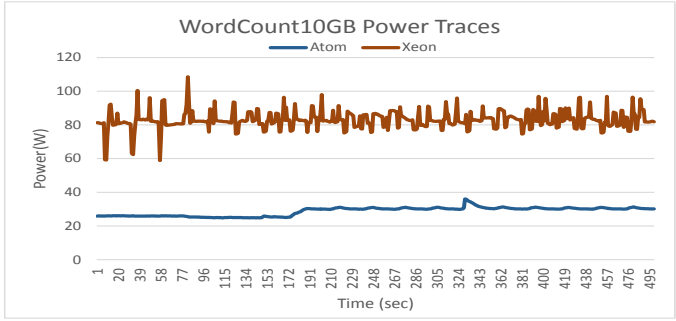


Figure 8.2: Performance/Watts of different configurations of Hadoop micro-benchmarks



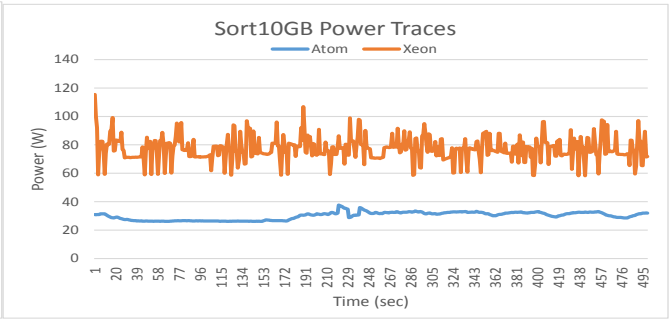


Figure 9: Power Traces of WordCount10GB and Sort10GB on Atom and Xeon

bandwidth utilization by multiplying the number of LLC misses per cycle with the number of bytes fetched and the frequency of the processor in GHz [3]. This formula indicates that bandwidth is dependent on LLC misses of running application. This behavior is expected on Atom as BigData applications suffer from less LLC cache misses (Figure 3.1).

Figure 5.2 shows the data sensitivity analysis of Hadoop micro-benchmarks on Xeon and Atom. Most of the micro-benchmarks on Xeon are utilizing more bandwidth than the Atom processor. In addition, *wordcount, sort* and *grep* are presenting the clear trend of increase in the bandwidth utilization with the increase in data size on Xeon. And Atom shows the increasing trend of bandwidth utilization with the increasing data size in all micro-benchmarks.

Observation. The characterization results in [3, 6] for bandwidth utilization are consistent with Figure 5.1 showing that big data on Xeon stress DRAM more significantly than traditional benchmark. In contrast, we observed an opposite trend on Atom where big data on little core stresses DRAM less compared with traditional applications. If we focus on the BigData workloads on Atom and Xeon then the difference for bandwidth utilization among them is minimal. Xeon has complex memory controller that consumes a lot of power and chip in contrast to Atom. This observations refers that architects can reduce memory controller complexity and work with the little core without effecting the application performance.

F. TLB misses:

TLB (Translation look-aside buffer) misses are costly, taking up to several hundred of cycles that are also expensive in terms of energy dissipation. We have measured the Instruction TLB (ITLB) and Data TLB (DTLB) MPKI Figure 6.1 and Figure 7.1 for BigData applications along the average reading for Spec, Parsec and scaleout applications. Our results reveal that average ITLB MPKI of Parsec is the lowest on Atom while BigData has the highest ITLB MPKI on Xeon with the average ITLB MPKI of 0.23 while that of Spec, Parsec and Scale-out is 7.44E-07, 0.0019, and 0.081 respectively. The average DTLB MPKI of BigData is highest as compared to traditional benchmarks and scale-out on Xeon but on Atom Spec have the highest average DTLB MPKI reading. In general, Xeon shows the clear trend of highest ITLB and DTLB MPKI than the traditional benchmarks. In contrast to this behavior Spec is experiencing the high ITLB and DTLB MPKI on Atom. Furthermore, Figure 6.2 and Figure 7.2 are the data sensitivity analysis of micro-benchmarks with respect to ITLB MPKI and DTLB MPKI. Atom incur huge ITLB MPKI as compared to Xeon, but lower DTLB MPKI in comparison to Xeon.

Observation. Overall there is a large variation in TLB misses in traditional CPU and parallel benchmarks. However, for the big data applications there is not much variation. Figure 6.2 does not show any clear trend however Figure 7.2 depicts that *WordCount, Sort* and *Grep* are clearing have increase in DTLB MPKI as the data size increases.

G. Performance /watts and Power Traces analysis.

Emerging big data applications require a significant amount of server computational power. 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, 20]. The results indicate that high performance servers Intel Xeon, which rely on aggressive superscalar technology, exhibit significantly lower energy efficiency in terms of energy-delay metric when running big data applications, compared to traditional CPU applications. Based on this observation, we have evaluated performance/watts comparison for BigData application along the Hadoop micro-benchmarks on both the Xeon processor and Atom processor in Figure 8.1 and Figure 8.2. The results present the clear trend of high performance/watts reading on Atom in comparison to Xeon processor. One reason might be fact at Spec and Parsec developers have spent generous amount of time on the optimization of these algorithms however BigData applications required optimization at many levels. This observation got us intrigued to observe the power traces. We have selected two big data applications *wordcount10GB* and *sort10GB* to observe power traces every second on both processors; Atom and Xeon. Figure 9 shows the high variation in the power traces on Xeon. Power ranges between 32 to 92 Watts. In contrast to Xeon, Atom does not show that much variation in the power. It consumes less power than Xeon as predicted and power value ranges from 0.9 to 12Watts.

Observation. This behavior illustrates that although Xeon has better performance capability, Atom is the better suit to deliver high performance/watts. And if applications have huge working data set and they are not computational intensive than Atom is better platform than the Xeon server

VI. CONCLUSIONS

Emerging BigData applications required a significant amount of server computational power. Since the increase in the information data is introducing challenges to analyze and evaluate this much data, it is very critical to assess the modern processors for the BigData applications.

To gain the detail insight of BigData applications, in this paper, we have concertized the performance characterizations of the various BigData workloads. Firstly, we have reported the BigData workloads characterization in comparison with the traditional benchmark and scale-out applications on high performance server, Intel Xeon. Secondly, we have performed the same study on the low-power server, Intel Atom. Thirdly, we have investigated the data sensitivity analysis of Hadoop micro-benchmarks on both Xeon and Atom servers. Lastly, we have done comparison among the Xeon and Atom results. The analysis shows that deep software stack of big data applications, along with the excessive non-loop branches affects L1 cache hit rate and branch predictor accuracy, respectively. In addition, our analysis indicates that data size has a non-trivial impact on several micro-architecture parameters. Moreover, BigData applications require efficient instruction prefetcher that can predict complex patterns and sophistication branch predictor that can handle the unknown control flow and predicting the instruction path when unstable data-dependent branches are encountered. Furthermore, off-chip bandwidth utilization results shows that we can reduce memory controller complexity without effecting the application performance. Specially, our performance/watts analysis shows that simple and multiple core are more suitable for the future processor designs and they will provide architectural designers more area and power range to play for advance prefetcher and branch predictor techniques. The analysis on Atom shows that except branch predictor and L1 cache, other micro-architectural parameters are satisfying the requirements of big data.

In sum, we believe this is the right time to explore and build the foundation of an efficient computing server platform customized for big data applications as big data analytics is just evolving and becoming a legitimate market. Realizing that highly over-provisioned out-of-order cores are not efficient for several types of server applications, and little core shows the low IPC, high branch predictor and high L1 cache misses, it is the high time to exploit the heterogeneous multicore systems. These systems comprise multiple cores with different capabilities. This new trajectory in server designing will advocates the effective utilization of Big Core for computational intensive applications and direct the power critical applications on little core.

References

[1] Baru, Chaitanya, et al. Setting the Direction for Big Data Benchmark Standards,Lecture Notes in Computer Science

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

[3] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, , and B. Falsafi. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. In Proceeding of the Int'l Conference on Architectural Support for Programming Languages and Operating Systems, Mar. 2012.

[4] N. Hardabellas, I. Pandis, R. Johnson, N. G. Marcheril, A. Ailamaki and B. Falsafi. Database Servers on Chip Multiprocessors: Limitations and Opportunities. In The 3rd Biennial Conference on Innovative Data Systems Research, January 2007.

[5] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang. The HiBench benchmark suite: characterization of the MapReduce-based data analysis. In 26th International Conference on Data Engineering Workshops, March 2010.

[6] W. Gao, "BigDataBench: a Big Data Benchmark Suite from Web Search Engines".(ASBD 2013) in conjunction with ISCA 2013

[7] S. Huang,The HiBench benchmark suite: characterization of the MapReduce-based data analysis, 26th ICDE Workshops, March 2010.

[8] Li,Ang,et al. CloudCmp: comparing public cloud providers. ACM,'10

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

[10] SPEC CPU2006, Standard Performance Evaluation Corporation

[11] A. Kejariwal, A. V. Veidenbaum, A. Nicolau, X. Tian, M.Girkar, H. Saito, U. Banerjee. Comparative Architectural Characterization of SPEC CPU2000 and CPU2006 Benchmarks on the Intel R CoreTM 2 Duo Processor. In International

[12] S. Bird, A. Phansalkar, L. K. John, A. Mericas and R. Indukuru. Performance Characterization of SPEC CPU Benchmarks on Intel's Core Microarchitecture based processor, in Proceedings of 2007 SPEC Benchmark Workshop, Jan 2007.

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

[14] T. Li, L.K. John, V. Narayanan, A. Sivasubramaniam, J. Sabarinathan and A. Murthy. Using Complete System Simulation to Characterize SPECjvm98 Benchmarks. ICS 2000 Santa Fe New Mexico USA.

[15] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. The PARSEC benchmark suite: characterization and architectural implications. In Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques, October 2008.

[16] SPMF; A sequential Pattern Mining Framework, http://www.philippe-fournier-viger.com/spmf/index.php?link=download.php

[17] T. L. Willke and N. Jain. GraphBuilder – A Scalable Graph Construction Library for Apache™ Hadoop™, in Big Learning WS at NIPS. 2012. Las Vegas.

[18] WattsUpPro power meter https://www.wattsupmeters.com/secure/index.php

[19] Frequent Itemset Mining Dataset Repository; http://fimi.ua.ac.be/data/

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

[21] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, Item-Based Collaborative Filtering Recommendation Algorithms, in Proceeding 10th Int'l WWW Conf., 2001

[22] L. Rokah and O. Maimon, Clustering Methods Chapter of Data Mining and Knowledge Discovery Handbook

[23] H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Y. Chang. Pfp: parallel fp-growth for query recommendation. In Proc. of the 2008 ACM RecSys conference, pages 107-114, 2008

[24] Apache Mahout; http://hortonworks.com/hadoop/mahout/

[25] Apache Mahout: scalable machine-learning and data-mining library. http://mahout.apache.org/.

[26] Memcached home page. http://memcached.org/

[27] Cloudsuite 2.0 home page. http://parsa.epfl.ch/cloudsuite/memcached.html

[28] .Apache™ Hadoop http://hadoop.apache.org/

[29] A. Jaleel and B. Jacob. In-Line Interrupt Handling for Software-Managed TLBs, in Proceeding of ICCD-19, 2001.

[30] J. Huck and J. Hays. Architectural Support for Translation Table Management in Large Address Space Machines. In proceedings of the 19th International Symposium on Computer Architecture, pages 39-51, May 1992.

[31] Intel® 64 and IA-32 Architectures Optimization Reference Manual. Order Number: 248966-029 , March 2014

[32] CloudRank home page. http://prof.ict.ac.cn/BigDataBench/

[33] A. Ghazal, T. Rabl, M. HU, F. Raab, M. Poess, A. Crolotte and H. Jacobsen: Bigbench: Towards an industry standard benchmark for big data analytics. In: ACM SIGMOD Conference (2013)

[34] DCBench home page. Dchttp://prof.ict.ac.cn/DCBench/

[35] S. Singh and N. Singh. Big Data Analytics, 2012 International Conference on Communication, Information & Computing Technology (ICCICT), Oct. 19-20, Mumbai, India

[36] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron. Rodinia: A benchmark suite for heterogeneous computing. In Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC '09), pages 44–54, Austin, TX, USA, 2009. IEEE.

[37] Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, pages 132–141