Application-driven Energy-efficient Architecture Explorations for Big Data

Xiaoyan Gu
Institute of Computing Technology
Chinese Academy of Sciences
Beijing 100190, China
Graduate University of Chinese Academy of Sciences
Beijing 100049, China
guxiaoyan@ncic.ac.cn

Rui Hou
Institute of Computing Technology
Chinese Academy of Sciences
Beijing 100190, China
hourui@ict.ac.cn

Ke Zhang
Institute of Computing Technology
Chinese Academy of Sciences
Beijing 100190, China
zhangke@ict.ac.cn

Lixin Zhang
Institute of Computing Technology
Chinese Academy of Sciences
Beijing 100190, China
zhanglixin@ict.ac.cn

Weiping Wang
Institute of Computing Technology
Chinese Academy of Sciences
Beijing 100190, China
wpwang@ncic.ac.cn

ABSTRACT

Building energy-efficient systems is critical for big data applications. This paper investigates and compares the energy consumption and the execution time of a typical Hadoop-based big data application running on a traditional Xeon-based cluster and an Atom-based (Micro-server) cluster. Our experimental results show that the micro-server platform is more energy-efficient than the Xeon-based platform. Our experimental results also reveal that data compression and decompression accounts for a considerable percentage of the total execution time. More precisely, data compression/ decompression occupies 7-11% of the execution time of the map tasks and 37.9-41.2% of the execution time of the reduce tasks. Based on our findings, we demonstrate the necessity of using a heterogeneous architecture for energy-efficient big data processing. The desired architecture takes the advantages of both micro-server processors and hardware compression/ decompression accelerators. In addition, we propose a mechanism that enables the accelerators to perform more efficient data compression/decompression.

Categories and Subject Descriptors

C.0 [**Computer Systems Organization**]: General-Systems architectures

General Terms

Measurement, Experimentation, Performance

ASBD 11, October 10, 2011, Galveston Island

Keywords

Energy-efficient, Big Data, Performance

1. INTRODUCTION

Big data becomes ubiquitous in today's information-based society where massive data sets are being produced at unprecedented rates, especially with the rapid growth of the number of devices connected to the Internet. IDC predicted that the amount of digital information created and replicated in a year are growing by a factor of 44, which means there will be about 35 Zetta Bytes generated in 2020 [5]. The deluge of data poses makes it extremely difficult to handle big data efficiently. Three of the most significant issues for processing big data are energy consumption, operation (analysis, query etc.) latency and throughput, and storing capacity. Various efforts have been made in attempts to overcome these obstacles [6, 12].

Recently, the concept of Micro-Server was proposed as one of the solutions to address the energy-saving requirement of big data centers [17, 21, 20]. Some researchers have suggested the use of low-power, low-frequency, simple, even "wimpy" processors as the computing nodes instead of high-performance and "giant" CPUs, like Intel Xeon [7], in data centers. In addition, several industrial vendors like Intel and ARM have announced their low-power processors for energy-efficient server systems. However, much remains to be done to explore the practicality of using "wimpy" CPUs, like Intel's Atom, in data center servers.

This paper presents the initial findings in our study of the conventional servers with high-performance CPUs and the micro-servers with low-performance CPUs. Our studies uses MapReduce and Hadoop as the testing benchmarks. MapReduce [13] is one of the most popular programming models for processing large data sets and Hadoop [2] is Apache's free and open source implementation of MapReduce. Our tests reveals that Hadoop can be both time-consuming and storage-consuming. The storage requirement

of Hadoop is extraordinarily high because it can generate a large amount of intermediate data. To reduce the requirment on the storage capacity, Hadoop often compresses data before storing it. In addition, while Hadoop and other conventional databases tend to store a large volume of data in the row-order, some big data platforms (such as Monet [4], RCFile [23], CStore/Vertica [22] and Mastiff [15]) tend to store data in the column order, for faster partial information retrieval. In addition, storing data in the column-order also allows better data compression rates, because data items with the same property are stored near each other.

The use of data compression raises a number of questions for the server designers: whether the computing nodes can meet the computing requirement of the data compressing and decompressing operations; how important is the performance of the compression/decompression operations to the overall execution time; and whether it makes sense to integrate compression/decompression accelerators onto the systems to release the main processors from doing data compression and decompression, especially when the main processors are "wimpy" CPUs and do not have strong computing capability.

In this paper, we intend to answer these questions by investigating a few server systems with different processors running real applications. In particular, we run a typical big data scenario, called Mastiff (a locally-developed extension of Hadoop with column-store mechanism, more details in next chapter), on two comparable clusters. One cluster is based on high-performance Xeon processors, while the other one is based on low-power Atom processors. Through measuring the energy consumption and breakdown of the execution time of Mastiff on the two clusters, we come up with the following main findings: (1) Atom cluster is more energy-efficient; (2) data compression and decompression takes a significant percentage of the total execution time; (3) the characteristics of data compression and decompression operations have a common streaming pattern with multiple interleave. Motivated by these findings, we propose a heterogeneous architecture for big data applications.

The paper is structured as follows. Section II presents background and Section III shows the evaluation methodology. Section IV gives our experimental results and the experiences and findings from running Mastiff on the two clusters. A new architecture design motivated by the findings is proposed in Section V. Finally, we compare our work with others and conclude this paper.

2. TARGETED APPLICATION SCENARIO : MASTIFF

Mastiff is chosen as our targeted application, because it is a typical representative of big data processing engine and can run with real production workloads with the column-major store policy [15].

Table 1: Compression ratio comparisons between Mastiff and native Hadoop HDFS. (The lower the better)

	Compression ratio on 3GB data	Compression ratio on 100GB data	Compression ratio on 500GB data
Mastiff	0.54	0.53	0.518
Hadoop HDFS	0.72	0.71	0.7

As a typical management system for big data, Mastiff implements an efficient column-major data storage layer based on Hadoop framework which provides the scalability of data processing and a scalable fault-tolerance Distributed File System (HDFS). With the column-major storage policy, data items with the same property are aggregated consecutively. Therefore, data compression has a better opportunity to get a much better compression ratio, which consequently could lead to more efficient I/O transactions. We compare the compession efficiency between Mastiff and the native Hadoop HDFS that uses the row-major policy, with the same Lempel-Ziv-Oberhumer (LZO) compression algorithm. Table 1 shows that the compression ratio of Mastiff is better than that of the native Hadoop HDFS in all three test cases.

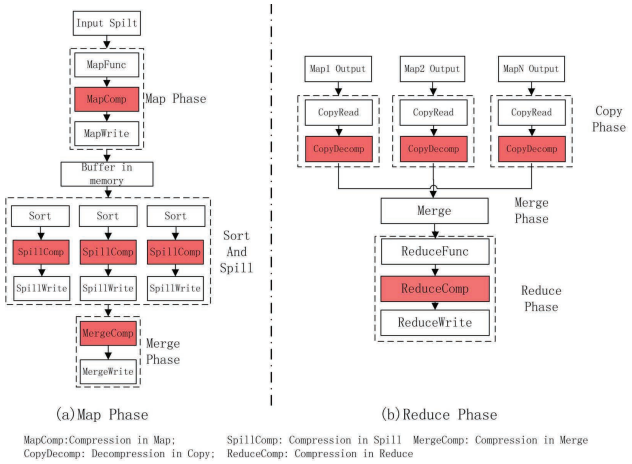

Figure 1: Mastiff working flow

To better understand Mastiff, Figure 1 describes its workflow. Mastiff composes of a Map phase and a Reduce phase. The Map function processes the data in the unit of $\langle key, value \rangle$ pair, and generates a set of intermediate $\langle key, value \rangle$ pairs. Such intermediate data are then shuffled over the network to the Reduce nodes, whose Reduce function emits a final set of $\langle key, value \rangle$ pairs.

It can be observed that compression and decompression are widely used in the major workflow of Mastiff(marked as red color in Figure 1). As shown in Figure 1(a), the Map phase of Mastiff adopts multi-level compression mechanism. The application level compression (MapComp) is firstly invoked, once the size of the raw output data reaches a prespecified threshold (depended on the compression algorithm, the default value is often slightly larger than a "page". In our studies, a page in Mastiff is 128K bytes). Compressed data is stored in the internal memory buffer. When the content in the internal buffer reaches specified size (such as 80% of the buffer capacity), the second-round compression is triggered (SpillComp). In addition, a third round compression (MergeComp) may be performed before the final output of the Map phase is written to the disk. Figure 1(b) shows that the Reduce task includes three major sub-phases: copy, merge, and reduce. Decompression is immediately needed after the data of the same partition is loaded from multiple remote nodes in the copy sub-phase. When all the map outputs have been copied and decompressed, the reduce task

moves into the merge phase. After the merge, the output is compressed again and written directly to the underlying file system.

3. METHODOLOGY

Table 2: Hardware and software configurations.

		Atom cluster	Xeon cluster
Number of Nodes		31 nodes	31 nodes
Chassis	Model	SuperCloud SC-R6280	Dawning I610r-H
	Size	2U with 8 nodes	1U with 1 node
Node	CPU	Intel Atom D525 (2 Cores/4 Threads/ 1.8 GHz/1M L2 cache/13W Max TDP)	Intel Xeon E5310 (4 Cores/4 Threads/ 1.6 GHz/8M L2 cache/80W Max TDP)
	Chipset	Intel ICH9R Chipset	Intel 5100+ICH9R Chipset
	Memory	2*2GB DDR3 Non-ECC 800MHz SO-DIMM	2*2GB FBD DDR2 667MHz ECC
	Disk	500GB SATA 5400RPM	1TB SATA 7200RPM
	Network	2*Intel 82574L Gigabit Ethernet	2*Intel 82573E Gigabit Ethernet
	Power module	720W (1+1) backup	520W
OS		CentOS release 5.5 Final (Linux Kernel: 2.6.18-194.el5 x86-64)	
Software		JDK1.6.0-16, Hadoop 0.20.2, Mastiff-0.1.2, Hive-0.5.0	

Our testing infrastructure includes two clusters, a HPC cluster with 32 Xeon-based nodes, and a Micro-Server cluster with 32 Atom-based nodes. The former represents high performance systems common deployed in the data centers and the latter represents low power systems being suggested today for more energy efficient data centers. Table 2 summarizes the major setting of these two clusters. One node in each cluster plays the role of the master nodes, which is in charge of recording metadata and managing tasks, including dispatching tasks to other nodes. The other nodes, named as slave nodes, are used to run dispatched tasks. The nodes within each cluster are connected through a 1Gbps Ethernet switch with 32 ports.

Mastiff is evaluated via using TPC-H benchmark [1]. A synthetic dataset with 1 Terabytes size is generated for evaluation. Every job is executed three times to compute the average result. The two scenarios we choose are data load and data query(TPC-H Query 6), which are the major operations in data flow management system.

A Fluke NORMA 4000 power meter equipment is used for our power and energy measurement. The power meter is attached to the wall socket of the slave nodes in our clusters. Due to the fact we do not have enough power meters to monitor all nodes in each cluster and our current study does not measure the power consumed by the network routers, we have to make two methodological simplifications in our initial study. First, power of network router is evenly counted towards to each node. Second, the energy consumption of each node in a cluster is similar. The results presented in this paper is based on the measurement of one slave node. A test is run multiple times. The median value of the resulting multiple measurements is reported in this paper. The initial results indicate the variance among different runs and different nodes are small. Nevertheless, we are installing more power meters into the systems to monitor each node and each network router.

4. POWER AND PERFORMANCE EVALUATION

In this section, we present the power and performance numbers of Mastiff on both the Xeon-based cluster and the Atom-based cluster and other interesting behaviors we have observed.

4.1 Power and performance measurement

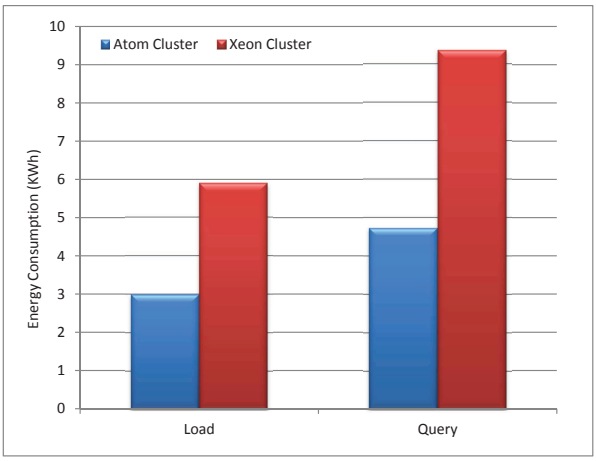

Figure 2: Total energy consumption comparison between a 30-node Atom cluster and a 30-node Xeon cluster

Table 3: Execution time comparison for different clusters with different scale.

	Time on Atom Cluster (30 nodes)	Time on Xeon Cluster (30 nodes)	Time on Xeon Cluster (15 nodes)
Data Load	3.435 hours	1.543 hours	3.242 hours
Data Query	5.877 hours	2.724 hours	5.564 hours

First of all, we compare the power consumption and performance of these two kinds of cluster with the same number of nodes (31 nodes), the same workload (Data load and query in Mastiff) and the same input data (1 Terabytes data). Table 3 shows the performance results and Figure 2 illustrates the energy consumption. The figure shows that, for both data load and data query operations, the energy consumption of Atom-based cluster is only around 50% of that of Xeon-based cluster. However, more detailed analysis reveals that the execution time of Atom-based cluster is twice of that of the Xeon-based cluster. It is a classical case of trading execution time for energy.

To make a comparison with the same number of cores, we scale down the size of the Xeon-based cluster to 16 nodes (15 slaves and 1 master) but keeps the Atom-based cluster the same. From table 3, we can observe that the execution time of the two clusters is similar. It should be pointed that the energy consumption of Atom-based cluster

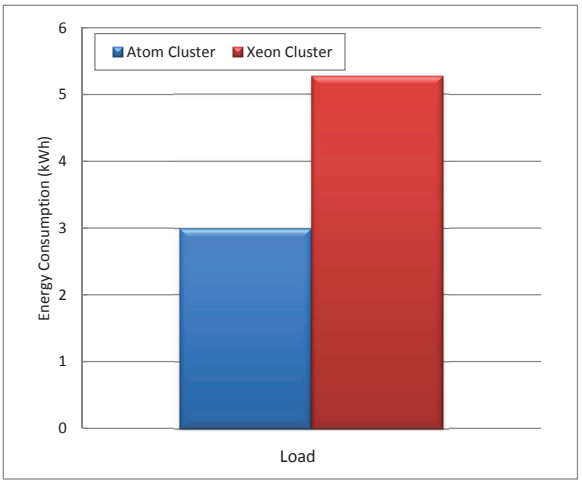

Figure 3: Total energy consumption comparison between a 30-node Atom cluster and a 15-node Xeon cluster

is still around 50% of that of Xeon-based cluster (Figure 3). Which means, in order to complete one job, although the size of Atom-based cluster is 2X than Xeon-based cluster, the energy consumption is still 0.5X of that of Xeon-based cluster. As a result, we think Atom-based cluster is more energy efficient than Xeon-based cluster.

4.2 Execution time breakdown

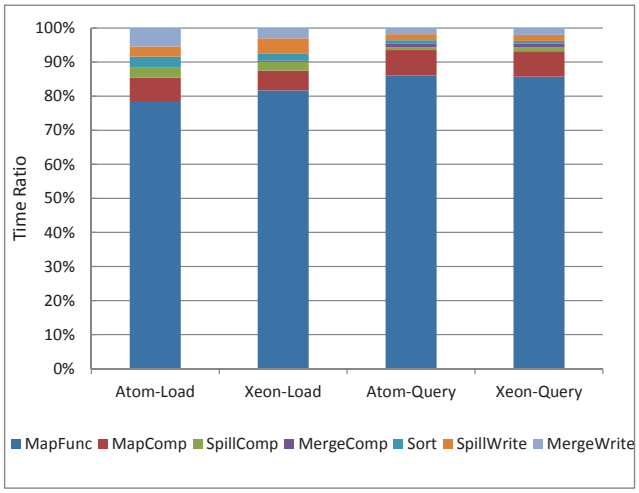

Figure 4: Time breakdown in Map phase

In order to figure out the bottleneck of Mastiff, we break down the execution of two major processing phases (map and reduce phase). Figure 4 shows the execution breakdown of the Map phase. Most of the time is spent on the application level task (map task itself) instead of the Hadoop framework. Data compression takes a significant amount of time. In particular, data compression is included in both the map task and the Hadoop framework (MapComp in Mapper task, SpillComp in the sort and spill sub-phase). Overall, data compression operation takes 11% of the execution time of the Map phase on Atom-based cluster and 7% on Xeon-based cluster.

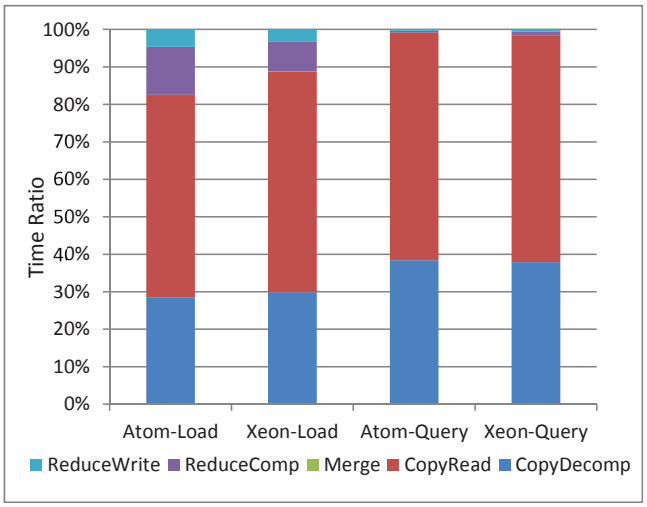

Figure 5: Time breakdown in Reduce phase

Figure 5 depicts the execution breakdown of the Reduce phase. In this case, most of the time is spent in the Hadoop framework instead of the application-level reduce task. As in the Map Phase, compression/decompression operations occupy significant percentages of the execution time for all test cases. For example, for the data load job on the Atom-based cluster, compression occupies 12.8% and decompression occupies 28.4% of the reduce phase. For data query, decompression takes 38.4%. For Xeon cluster, the numbers are 8.1%, 29.8% and 37.9% respectively.

4.3 Stream working mode for compression and decompression operation

An in-depth code analysis of both Mastiff and Hadoop implementation reveals that there is one common streaming pattern for the compression and decompression operations. In particular, these compression and decompression operate at the granularity of a single or a group of $\langle key, value \rangle$ pairs, instead of all $\langle key, value \rangle$ pairs. The major reason behind this mechanism is due to the widely-used software pipeline implementation in Mastiff and Hadoop. Using the reduce phase of Hadoop as an example, Hadoop immediately starts decompression just after compressed data is loaded from remote nodes. If decompression had started until all the data have become available, it would have had the concurrency. As an optimization, decompression is triggered once a single or a group of $\langle key, value \rangle$ pairs have been loaded.

4.4 Summary

This section summarizes the insights we obtained from above analysis.

1. The Atom platform is more power efficient than the Xeon platform;

2. Data compression and decompression occupies a significant percentage of the total execution. More precisely,

it takes 7%-11% of the execution time to decompress the data in the map phase, and 37.9%-41.2% of the execution time to compress then decompress the data in the reduce phase;

3. Data compression and decompression are typically done in the software pipeline fashion;
4. It is common for Hadoop to run multiple map and reduce tasks on the same processor, each of which usually contains compression or/and decompression operations.

5. ENERGY EFFICIENT ARCHITECTURE FOR BIG DATA

Motivated by the outcome from the analysis of Hadoop applications, this section proposes a new energy efficient architecture for big data applications.

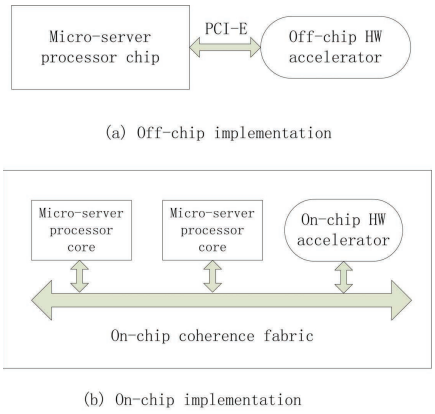

Figure 6: Off-chip and on-chip accelerator implementations

5.1 Heterogeneous architecture: Low-power cores with hardware accelerators

Our experiments show that Atom platform is more power efficient than Xeon platform. Although such observation may not be generalized for every kind of applications, Hadoop is one of the most frequently used applications today and is a good representative of big data processing. Consequently, our energy-efficient architeture exploration starts with a low power processor.

Considering the fact that data compression/decompression accounts for a considerable fraction of the total execution time, we propose a heterogeneous architecture which can offload data compression and decompression operations from the general-purpose processors to the special hardware accelerators. By implementing certain special operations in hardware, an hardware accelerator can perform the special operations much faster and more efficiently than general-purposes architectures. Typically, an accelerator can improve performance and energy efficiency by orders of magnitude over a general-purpose core. As a matter of fact, various heterogeneous architectures have been adopted in commercial products. For example, compression and decompression accelerator can be found in IBM Z10 processor [24], IBM PowerEN processor [14] and RMI XLP processors [18]. The maturity of compression/decompression accelerators makes it easy to incorporate them in servers optimized for big data.

As shown in Figure 6, there are at least two common options to implement such heterogeneous designs. One option is through the PCIe based acceleration card, and the other is through on-chip accelerators. The first option is easily doable with the existing commercial systems. The second one requires redesigning processor chips, a much higher investment and a more fundamental change.

5.2 Desired features of compression and decompression accelerator

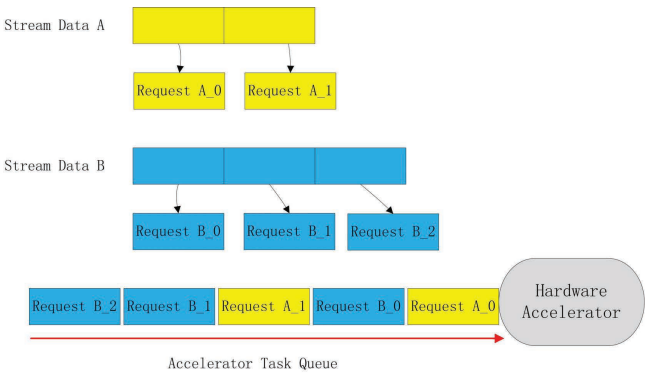

Figure 7: Multiple interleaved compression or decompression tasks

Traditional compression or decompression accelerator works on one block of data at a time and processing of different data blocks do not share any state. This kind of processing is referred as stateless. Due to the limited size of a data block, the efficiency of the stateless compression/decompression operation is correspondingly limited. On the other hand, the upper-level application tasks works in much large data sets within the context of Hadoop and Mastiff, albeit in a software pipeline mode. The efficiency of compression/decompression could be greatly improved if the accelerator can work in longer data streams. Thus, we hereby propose to enable the accelerator to support so-called stateful processing.

As shown in Figure 7, one input data stream to be compressed or decompressed is usually composed of many sub-blocks. Compared to stateless accelerators, stateful accelerators do not need to wait for all the sub-blocks of the whole data stream to be ready. The software can issue accelerator request for each sub-block once it is available. The accelerator then starts processing data once it receives a request of a sub-block. Such stateful working mode can lead to higher performance and better power efficiency. To support stateful operation, the accelerator has to maintain a context for each sub-block request associated with the same data stream. Yu et al. proposed a similar design for a stateful decompression accelerator with the networking processing [25].

We further observed that Hadoop usually allocates multiple map and reduce tasks on the same node at the same

time, and the number of software tasks are larger than the number of hardware threads. It implies that multiple software tasks may share the same stateful hardware accelerator, while each task issues multiple accelerator requests for processing one big data stream. The interleaving of accelerator requests from different tasks will lead to frequent context switches within an accelerator, causing considerable performance loss. Therefore, an accelerator with stateful operation mode should either support efficient context switching or reduce the number of context switches when processing interleaved requests. Such optimizations have been proposed in other contexts and can be adopted into architectures discussed here. For example, X.T. Chang et al. have proposed and evaluated dynamic reordering of the requests of different streams in a hybrid on-chip/memory based request queue in order to reduce the associated overhead [8]. Such feature is important for an accelerator optimized for Hadoop-like applications.

Another critical factor affecting the acceleration speed is the efficiency of data communication. In the case of PCI-E based accelerator card implementation, advanced DMA mechanism would be valuable for improving the performance of accelerators. And for on-chip accelerators, direct cache access and advanced cache coherence protocols can be adopted to avoid unnecessary off-chip data communication and improve data access latency. For example, R. Hou et al. have proposed handshake-sharing cache coherence protocol to support efficient data transfer of on-chip accelerators [16].

6. RELATED WORK

Building an energy-efficient architecture or system for big data has received growing attention from both industry and academia. It would be impossible to comprehensively review the existing literature in this section, so we only focus on research projects which are most relevant to our work.

Some other researchers have noticed that low-power processors are potential candidates for the computing elements of an energy-efficient big data system. In particular, Andersen et al. propose a cluster architecture that consists of a large number of slow, low-power embedded processors (AMD Geode LX or Intel Atom) coupled with CompactFlash or SATA-based Flash storage [3]. The resulting system is more energy-efficient in terms of queries per Joule than conventional disk-based systems for several I/O intensive workloads. Gordon [7] demonstrate similar findings.

Chun et al. propose a way to lower energy consumption in data centers through exploring the potential of hybrid designs that combine low-power platforms with high-performance platforms [11]. Reddi et al. evaluate Microsoft Bing web search engine on both Xeon and Atom processors [19].

Chen et al. instrument a Hadoop cluster with a power meter to explore configuration effects on power and energy consumption [10]. Based on their findings, they propose a new decision algorithm that helps MapReduce users identify when and where to use compression. The decision is based on the analysis of I/O tradeoffs for energy efficiency [9].

7. CONCLUSION AND FUTURE WORK

In this paper, we investigate and compare the energy consumption and the execution time of a Hadoop-based big data processing application running on a Xeon-based cluster and an Atom-based cluster. Our experimental results show that the Atom-based server is more energy-efficient than the Xeon-based platform for the application under test. In addition, we have identified that data compression and decompression can be major bottlenecks in such applications.

Inspired by the result of our investigation, we demonstrate the necessity of using energy efficient heterogeneous architectures for big data applications. An ideal big data server should include both low-power processors and hardware compression/decompression accelerators. In this paper, we briefly discuss the difference between off-chip accelerators and on-chip accelerators. Furthermore, we qualitatively discuss the potential benefit of supporting stateful compression/decompression within the accelerators.

Much more work remains to be done to fully understand the advantages and disadvantages of conventional high-performance servers and emerging energy-efficient micro servers. Some of our future work includes exploring a large set of applications, performing more fine-grained power measurements, investigating other types of platforms such as ARM-based clusters, as well as proposing a complete system design for big data applications.

8. ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China (Youth) No. 61100010, and National Science and Technology Major Project of China, No. 2011ZX01028-001-002.

9. REFERENCES

[1] `http://www.tpc.org/tpch/spec/tpch2.9.0.pdf`.

[2] B. A., M. Cafarella, D. Cutting, and O'lley. Hadoop: a framework for running applications on large clusters built of commodity hardware. `http://lucene.apache.org/hadoop`.

[3] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan. Fawn: A fast array of wimpy nodes. In *Communications of the ACM, July 2011*, July 2011.

[4] P. A. Boncz and M. L. Kersten. Monet: An impressionist sketch of an advanced database system. In *Proc. Basque International Workshop on Information Technology*, 1994.

[5] M. Bowman, S. K. Debray, and L. L. Peterson. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.*, 15(5):795–825, November 1993.

[6] J. Braams. Babel, a multilingual style-option system for use with latex's standard document styles. *TUGboat*, 12(2):291–301, June 1991.

[7] A. M. Caulfield, L. M. Grupp, and S. Swanson. Gordon: Using flash memory to build fast, power-efficient clusters for data-intensive applications. In *14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS09)*, Mar 2009.

[8] X. Chang, Y. Ma, H. Franke, K. Wang, R. Hou, H. Yu, and T. Nelms. Optimization of stateful hardware acceleration in hybrid architectures. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–4, Mar 2011.

[9] Y. Chen, A. Ganapathi, and R. H. Katz. To compress or not to compress - compute vs. io tradeoffs for

mapreduce energy efficiency. In *First ACM SIGCOMM Workshop on Green Networking, co-located with SIGCOMM 2010*, pages 23–28, 2010.

[10] Y. Chen, L. Keys, and R. H. Katz. Towards energy efficient mapreduce. Technical Report UCB/EECS-2009-109, EECS Department, University of California, Berkeley, Aug 2009.

[11] B.-G. Chun, G. Iannaccone, G. Iannaccone, R. Katz, G. Lee, and L. Niccolini. An energy case for hybrid datacenters. In *Proceedings of the 2nd Workshop on Hot Topics in Power-Aware Computing and Systems (HotPower 2009)*, pages 76–80, Oct 2009.

[12] M. Clark. Post congress tristesse. In *TeX90 Conference Proceedings*, pages 84–89. TeX Users Group, March 1991.

[13] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Sixth Symposium on Operating System Design and Implementation, OSDI2004*, pages 107–113, Dec 2004.

[14] H. Franke, J. Xenidis, C. Basso, B. M. Bass, S. S. Woodward, J. D. Brown, and C. L. Johnson. Introduction to the wire-speed processor and architecture. *IBM Journal of Research and Development*, 54(1):3:1–3:11, January 2010.

[15] S. Guo. Massive stream data's storage and query technology research based on hadoop (in chinese). Master's thesis, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, P.R.China, 2010.

[16] R. Hou, L. Zhang, M. C. Huang, K. Wang, H. Franke, Y. Ge, and X. Chang. Efficient data streaming with on-chip accelerators: opportunities and challenges. In *17th International Conference on High-Performance Computer Architecture (HPCA-17)*, pages 312–320, Feb 2011.

[17] I. L. power Micro-server. `http://newsroom.intel.com/servlet/JiveServlet/download/38-4248/Intel_Micro_Server_factsheet.pdf`.

[18] R. X. S. processor.

[19] J. Reddi, Vijay, Lee, B. C., C. Trishul, and V. Kushagra. Web search using mobile cores: quantifying and mitigating the price of efficiency. In *Proceedings of the 37th annual international symposium on Computer architecture*, pages 314–325, 2010.

[20] M. A. server chip. `http://www.eetimes.com/electronics-news/4199239/Marvell-ARM-Servers`.

[21] A. server solution: Caldexa. `http://www.calxeda.com`.

[22] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O'Neil, P. O'Neil, A. Rasin, N. Tran, and S. Zdonik. C-store: a column-oriented dbms. In *Proc. of the 31st international conference on Very Large Data Bases (VLDB'05)*, pages 553–564, Dec 2005.

[23] A. I. H.-. M. H. support column based storage. `http://issues.apache.org/jira/browse/HIVE-352`.

[24] C. Webb. Ibm z10: The next-generation mainframe microprocessor. *Proceedings of IEEE Micro*, 28(2):19–29, March 2008.

[25] H. Yu, H. Franke, G. Biran, A. Golander, T. Nelms, and B. Bass. Stateful hardware decompression in networking environment. In *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pages 141–150. ACM, 2008.