A Study of Big Data Processing Constraints on a Low-Power Hadoop Cluster

Chanwit Kaewkasi
School of Computer Engineering
Suranaree University of Technology
Nakhon Ratchasima, 30000, Thailand
Email: chanwit@sut.ac.th

Wichai Srisuruk
School of Computer Engineering
Suranaree University of Technology
Nakhon Ratchasima, 30000, Thailand
Email: wichai@sut.ac.th

***Abstract*—Big Data processing with Hadoop has been emerging recently, both on the computing cloud and enterprise deployment. However, wide-spread security exploits may hurt the reputation of public clouds. If Hadoop on the cloud is not an option, an organization has to build its own Hadoop clusters. But having a data center is not worth for a small organization both in terms of building and operating costs. Another viable solution is to build a cluster with low-cost ARM system-on-chip boards.**

This paper presents a study of a Hadoop cluster for processing Big Data built atop 22 ARM boards. The Hadoop's MapReduce was replaced by Spark and experiments on three different hardware configurations were conducted to understand limitations and constraints of the cluster.

From the experimental results, it can be concluded that processing Big Data on an ARM cluster is highly feasible. The cluster could process a 34 GB Wikipedia article file in acceptable time, while generally consumed the power 0.061-0.322 kWh for all benchmarks. It has been found that I/O of the hardware is fast enough, but the power of CPUs is inadequate because they are largely spent for the Hadoop's I/O.

***Keywords*—Big Data, Hadoop, cluster.**

I. Introduction

Big Data processing with Apache Hadoop [1] has been emerging since the past five years. It is motivated by, at least, the two research works of Google, the Google File System [2] and the Map-Reduce programming paradigm [3]. Recently, Hadoop has been the de facto standard software for Big Data. It consists of two major components, the Hadoop Distributed File System (HDFS) [4] and the Hadoop MapReduce engine. Hadoop plays an important role in several Big Data software companies, such as Cloudera, MapR and Hortonworks. Focusing on clouds and enterprise deployment, these companies have been invested more than $1 billion already by many venture capitals, for example [5].

Unfortunately, wide-spread security exploits may hurt the reputation of cloud computing. For example, an effect of the Heartbleed exploit [6] is severe because this bug is hidden in one of the most popular SSL implementations. This bug allows direct memory access of the server, so an invader can retrieve a large amount of data to look for recently changed passwords, numerical components of private keys, or even active logging sessions to bypass security layers [7]. Despite inexpensiveness, transfering the organization's sensitive data to the cloud just for processing them may be risky todate.

A natural alternative of public cloud computing is to build the organization's own data center. However, a data center is costly in both terms of building and operating. The Hamilton's cost model [8] showed that the building cost is 4% of the overall data center, and it is approximately the cost of 4,000 server machines. In addition, the operating cost, which includes the electrical power and the power distribution & cooling, is a high ratio around 31%. According to this model, it can imply that building a data center is not worth for an organization that requires less than a thousand of servers.

Another viable choice is to develop a Hadoop cluster with low-power processors, which can be done in two different approaches, a specially designed ARM-based server versus a commodity cluster with low-cost ARM boards. Ignited by Raspberry Pi [9], there are many efforts studied [10], [11] and built [12] computing clusters with ARM system-on-chip (SoC) boards. Recently, a number of Hadoop clusters on ARM SoC boards have been studied and reported to be slow and suggested to not be ready for production [13], [14]. Naturally, ARM processors are slower than desktop or server classes by design because of their low power consumption, but it is believed that utilizing a cluster of ARM boards for Big Data is still possible. This is based on the work by Zaharia et al. showing that Hadoop is slow because it performs disk-based operations and can be improved by replacing the Hadoop's MapReduce by their in-memory computational engine [15].

The research questions of this work are that if the software stack is changed, could an ARM cluster be able process Big Data at the acceptable rate? What are limitations of the cluster with the new software stack? Can the cluster still preserve its low-power characteristic? To answer these questions, a cluster with 22 ARM-based Cortex-A8 processors is developed and the experiments with Spark over HDFS are conducted on it.

The contributions of this paper are as follows. Firstly, a set of data processing rate measurements with different configurations of the cluster is presented. Secondly, limitations of the cluster in the context of Big Data from different perspectives, focusing on I/O constraints, power consumption and CPU usage, are presented. It is found that the capability of in-memory computation can spare the CPUs to process data rather than busy computing data integrity for the HDFS.

978-1-4799-4963-2/14/$31.00

Finally, it is found that Big Data software stack plays an important role on a constrainted hardware like an ARM board.

The remainings of this paper is organized as follows. Section 2 presents three different hardware configuration of the cluster. Section 3 discusses benchmarks and the data size selection for the experiments. Section 4 presents the experimental results and discusses the cluster's characteristics in terms of *bottlenecks*, *power consumption*, *performance* and *memory constraints*. Section 5 discusses related work, and the paper ends with conclusion and future works in Section 6.

II. Cluster Configurations

Testing by gradually increasing a number of nodes reflects the scaling property of the cluster, but it is out of the scope of interests. Instead, three differrent hardware configurations of the cluster were prepared. The first one is the network-based cluster with 2 solid state drives (SSDs), denoted `2-sdd`. The second one, denoted `20-hdd`, and the third one, denoted `20-ssd`, consist of the same physical cluster, with 20 hard drives and 20 SSDs, respectively.

The physical cluster is a 22-node Cubieboard [24]. Each node equipped with an ARM Cortex-A8 processor at 1 GHz with 1 GB RAM. Each network connector is 100 Mbps Ethernet. Although our application is out of the area of High Performance Computing (HPC), but it is good to note about the hardware's performance. The raw performance of each node is around 2,000 MIPS (Dhrystone) and 200 MFLOPS (Whetstone), according to the study from [16]. Thus, the whole cluster's raw performance would be approximately 40,000 MIPS and 4 GFLOPS.

The cluster's operating system is Linaro Ubuntu 13.06 with Linux kernel 3.4.47. For Big Data software, Apache Hadoop 0.20.205 [1] and Apache Spark 0.9.0 [25] were used in conjunction with Java Development Kit 1.7.0_51 for the experiments. The 0.20 version of Hadoop was chosen rather than newer ones because the space limitation of the board's built-in NAND flash device. The file system `ext4` was chosen because it is the most stable file system for this version of Linux kernel.

There are 2 logical cluster layers atop the physical cluster, the HDFS cluster and the Spark cluster. The HDFS cluster consists of a single *Name Node* and a number of *Data Nodes*, while the Spark cluster consists of a *Master*, a *Driver* and 20 *Workers*.

For the first configuration, there are 2 SSDs attached to 2 physical nodes as shown in Fig. 1. This means that the HDFS cluster consists of 2 Data Nodes sharing physical nodes with 2 Spark's Workers. The HDFS's Name Node also shares the same physical node with the Spark's Master. This configuration is the early design of our cluster. It is used for observing data processing over network and the behaviour of the HDFS's data locality, when a number of Data Nodes is small. From the economy perspective, this is the cheapest cluster. Data replication for HDFS is set to be 2, as only 2 SSDs are available for this configuration.

Shown in Fig. 2, the second configuration consists of 20 Data Nodes for HDFS. The 2 SSDs in the first configuration were replaced by 20 hard drives. This means that each Spark's

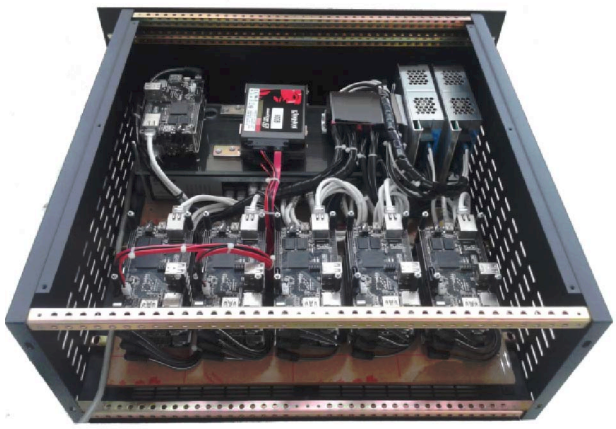

Fig. 1. The first cluster configuration, the network-based cluster with 2 SSDs

Fig. 2. The second cluster configuration with 20 mechanical hard drives

Worker node has its own HDFS Data Node for retrieving data. It does not require transfering data over network, so this cluster configuration has the full data locality (FDL) property. This configuration is for observing the FDL property over a set of mechanical drives. This configuration has the largest storage. The data replication level is set to 3, so it is similar to a production cluster.

For the third configuration, The 20 hard drives of the second configuration were replaced by 20 SSDs. Similar to the second one, this configuration also maintains the FDL property. It is for observing the FDL property over a set of SSDs. The data replication is also set to 3 for this configuration. Fig. 3 shows the third hardware configuration.

III. Benchmarks

Unlike Hadoop, there is no standard set of benchmarks for Apache Spark yet. But it has been found that *word-count*, *sort* and *grep* micro-benchmarks are listed as a part of BigDataBench [17]. These three micro-benchmarks were combined into a program for finding the most frequent words (MFW). MFW loads a text file from HDFS, then split each line as a list of words. The programs then maps and reduces to count all duplicated words. Later, each entry is mapped

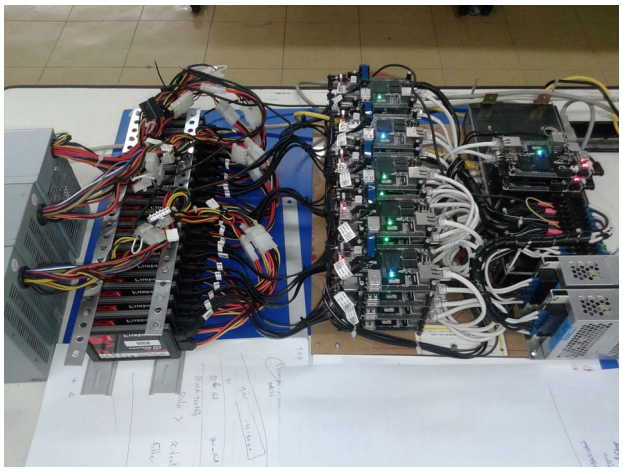

Fig. 3. The third cluster configuration with 20 SSDs

```
var tf = sc.textFile("hdfs://...")
tf.flatMap(_.split(" ")).
    map(_, 1).
    reduceByKey(_ + _).
    map(_.swap).
    sortByKey(false).
    take(10)
```

Fig. 4. The non-filtered version of the MFW benchmark

again by swapping the key and the value. After that, all entries are sorted. Spark will be separating the running program in several transformation stages while executing, where the cluster characteristics can be observed.

Benchmarks used in our experiments are two versions of MFW. The non-filtered version, suffix denoted `-nf`, process the text file directly while the filtered version, suffix denoted `-fl`, uses a regular expression, `\w+`, to filter only words for further processing, hence six combinations of configuration. For example, `2-hdd-nf` is the second cluster configuration running the non-filtered version of the benchmark program. Fig. 4 and 5 show both versions of the program written in Scala [18].

The size of data used in all experiments is 34 GB. It is the English Wikipedia article file of the year 2012. This data size was chosen because it is about 2 times larger than the median Big Data job size reported by Microsoft [19], which is around 14 GB. There is also a report that the average Yahoo! job size

```
var tf = sc.textFile("hdfs://...")
tf.flatMap(_.split(" ").
    filter(_.matches("\\w+"))).
    map(_, 1).
    reduceByKey(_ + _).
    map(_.swap).
    sortByKey(false).
    take(10)
```

Fig. 5. The filtered version of the MFW benchmark

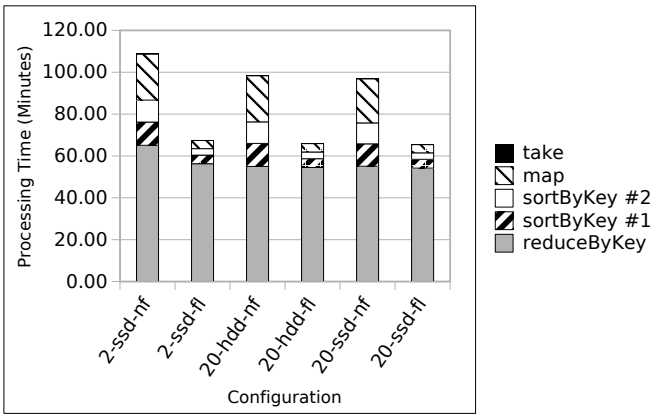

Fig. 6. Processing time for MFW benchmarks broken down as Spark's stages

is approximately at 12.5 GB [20]. This means that 34 GB size of file is reasonable to be a representative of the data size in general. The file is stored in the HDFS, separated into 538 blocks and 64 MB each. So the maximum parallelism level to process this file is 538. Of course, the cluster could process 20 blocks at a time.

IV. Cluster Characteristics

A. Bottlenecks

Using both versions of benchmark, the cluster's bottlenecks can be more clearly identified that they are from the I/O, when reading data via network and disk. The total execution time, separated as Spark's transformation stages, is shown in Fig. 6. After the first stage (`reduceByKey`), all configuration spent time insignificantly different. Unsurprisingly, the 100 Mbps network I/O is the biggest burden of the cluster, indicated as the gray portion of `2-ssd-nf`. Also, the cluster performed slowest with the `2-ssd-nf` configuration.

The second bottleneck is concluded to be the power of processors. At first, the disk I/O was suspected as both hard drive and SSD configurations performed the same. So the disk I/O was measured using the `dd` command and resulting in 125 MB/s for disk reading. This value seems to be the bus limitation of the board, but it is far faster than the processing rate. Then the CPU usage at the first Spark's stage was examined, as shown in Fig. 7. It was found that more than 95% of the CPU time are spent by the first Spark's tasks (`reduceByKey` in Fig. 6), and another 2% (figures excluded from this paper) are spent by a Hadoop's Data Node. From the presentation by Capper [21], the author mentioned that Hadoop uses a large portion of CPU for computing and verifying CRC32 to provide data integrity. Thus, it is more understandable that the cluster's CPUs are spent for reading data from HDFS. After the confirmation that the second bottleneck is at the CPU's processing power and also relates to the software layer, it is clearly that there are still rooms for improvement to make an ARM cluster performs better by optimizing the software stack.

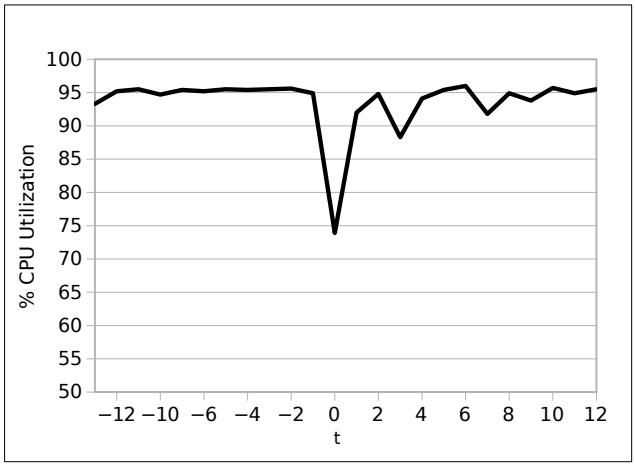

Fig. 7. CPU utilization sampling from a Spark node in percent, centering (t=0) at the time a processing job of an 64 MB data block is finished

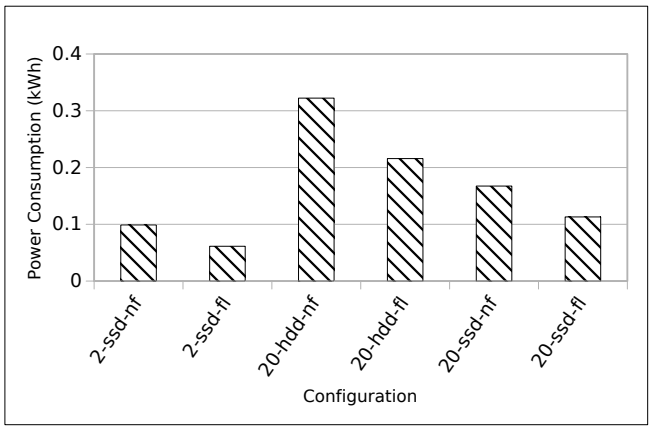

Fig. 8. Power consumption for completing each benchmarks in kWh

B. Power Consumption

An ARM processor is normally operating at low Watts. This is a strong advantage of this family of processors over desktop or server classes, e.g Intel or AMD. All cluster configuration consumes quite low power, as shown in Fig. 8. The most power consuming is the `20-hdd-nf` configuration as the cluster attached 20 mechanical hard drives and the benchmark program is non-filtered. To finish the benchmarks, the cluster consumed 0.322 and 0.215 kWh respectively. In contrast, the first configuration with 2 SSDs attached, consumed power as low as 0.099 and 0.061 kWh for the non-filtered and the filtered benchmarks respectively. However, the low-power configuration has the performance trade-off as the first configuration has only 2 Data Nodes and the data must be transferred mostly via the network.

The third configuration consumed the power under 0.2 kWh for the both benchmarks, 0.167 and 0.113 kWh respectively. It is the fastest, slightly faster than the second configuration, in terms of performance. Although the third configuration consumed lower than 0.2 kWh, a peak power consumption had been identified, slightly more than 0.250 kW, for a small period of time at the cluster's startup. This power consumption rate may be the largest possible power required by all 20 SSDs. With this spiking, it is found that the third configuration is less stable than the second configuration.

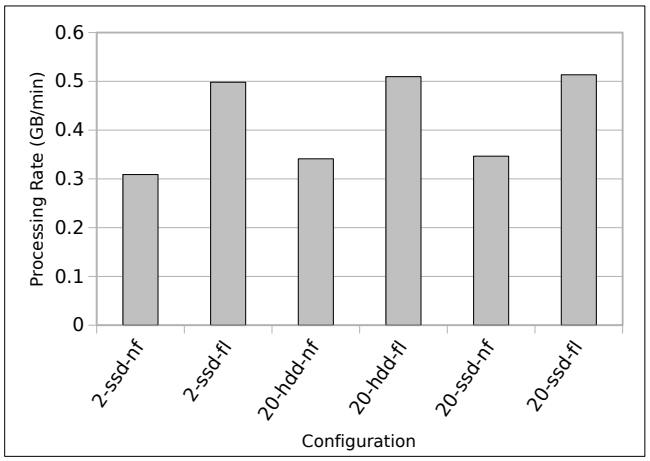

Fig. 9. Data processing rate for each cluster configuration in GB/min

C. Performance

Measurement of data processing is normally done in a unit of GB per minute (GB/min) [22]. Fig. 9 shows data processing rate of all configurations. The fastest configuration is `20-ssd-fl`, which is the third cluster running the filtered benchmark. The third cluster configuration is slightly faster than the second one for both benchmarks at 0.347 and 0.513 GB/min, while the first cluster configuration is slowest at 0.309 and 0.498 GB/min.

D. Memory Constraints

Each node has 1 GB of RAM. It requires around 150 MB for the OS. Each HDFS's Data Node required approximately 120 MB for the tested data. The 512 MB of RAM was allocated for each Spark's Worker. It had been tried to allocate 640 MB each, but the Workers failed before completing tasks, so the setting was rolled back to 512 MB per node.

All configuration used the default 60% of the Spark storage and and 30% of shuffle memory. It had been tried changing these settings to 10% and 80% respectively in hope of speeding the cluster up, but it had no effect.

It is concluded that the memory of 1 GB is enough for a single core board to form a cluster. The memory of 4 GB is not necessary as a 32-bit JVM can utilize only 1.5 GB per instance. But if there were 2 CPU cores, the memory at 2 GB would be more appropriate so 2 Spark Workers can be on the same physical node.

V. Related Works

A. Low-power Clusters

In 2008, Adams presented Microwulf [23], a Beowulf cluster with a price/performance ratio is below $100/GFLOPS.

TABLE I. SUMMARY OF THE EXPERIMENTAL RESULTS. EMPHASIZED FIGURES ARE THE BEST CONFIGURATIONS.

Configuration	Processing time broken down as Spark's stages (min.)					Processing Time (min.)	Data Processing Rate (GB/min)	Power Consumption (kWh)
	reduceByKey	sortByKey #1	sortByKey #2	map	take			
2-ssd-nf	65.1	11.1	10.5	21.9	15.5/60	108.9	0.309	0.099
2-ssd-fl	56.3	4.1	3.1	3.9	5.4/60	67.5	0.498	**0.061**
20-hdd-nf	55.0	11.0	10.3	22.0	15.4/60	98.6	0.341	0.322
20-hdd-fl	54.5	4.2	3.2	4.0	5.2/60	66.0	0.509	0.215
20-ssd-nf	55.1	10.7	10.0	21.0	13.2/60	97.0	0.347	0.167
20-ssd-fl	54.2	4.2	3.1	3.9	5.5/60	**65.5**	**0.513**	0.113

The cluster is able to run at room temperature and plug into a standard power socket. A Beowulf cluster is a computing cluster mainly designed for MPI programs. It is different from a Hadoop cluster, presented in this paper, which is focusing on processing Big Data and related to the cluster's I/O rather than computing-intensive tasks.

Cox et al. developed Iridis-Pi [12], a 64-node cluster of Raspberry Pi Model B, a node equipped with a 700 MHz processor and 512 MB RAM. Iridis-Pi focuses on computational task using MPI, while the software configuration of our cluster has been tuned for Big Data processing. We do not choose Rasberry Pi because our software stack is based on Hadoop, which requires a Java Virtual Machine (JVM). This work aims at exploring feasibility of using ARM cluster for processing Big Data in the future, so it considers JVM as a part of the system's constraints.

Whitehorn created a 5-node Raspberry Pi cluster that be able to run Hadoop. The heap size of each node could be limitedly allocated for the JVM at 272 MB. It is reported that the cluster cannot be used in production [1]. Cubietech, the maker of Cubieboard, formed an 8-node cluster with their boards. Their cluster does not have any storage device attached and is demonstrated to start and run Hadoop [24]. Both works inspired us to find real constraints behind the incapability of processing Big Data with an ARM cluster.

B. Performance on Big Data

Performance of Hadoop MapReduce has been known that it is slow because its nature of disk-based operations for preventing data loses during computation. In contrast, Apache Spark uses a fault-tolerant working set, called Resilient Distributed Dataset (RDD) [15], to prevent data losses as an RDD can track and recompute the missing tasks when a computing node is failed. Spark uses RDD as its main in-memory computational technique, so it achieved at least 10 times faster than Hadoop's MapReduce in several benchmarks [25].

Similarly, Zhang et al. proposed iMapReduce [26], a framework for iterative computation also to solve the performance problem of the Hadoop's MapReduce implementation. The authors reported five times performance gain compared to the original MapReduce. We chose Spark to experiment over HDFS because it clearly performs faster than other in-memory engine, including iMapReduce.

Uthayopas and Benjamas studied the impact of I/O and execution scheduling strategies on a large scale multi-core system via a simulation [27]. Their work discussed the importance of choosing the right I/O and execution scheduling algorithm to improve the cluster's performance. Also algorithms at the application level could benefit from their studies. This work is similar to ours in terms of identifying bottlenecks of the system. However, we focus on a specific hardware and software stack, in this case, Hadoop and an ARM-based cluster.

Balakrishnan studied High Performance Computing by measuring an ARM cluster built with the Pandaboard ES with a variety of benchmarks. The author found that the modern ARM architectures, such as ARMv7, do not only provide good raw performances, but also use less power than the older architectures [28].

VI. CONCLUSION AND FUTURE WORKS

This paper described a study of a 22-node ARM SoC cluster designed for Big Data. A cluster for Big Data is different from an MPI-based cluster in terms of the area of applications and the software stack. MPI-based clusters focuses on CPU-bound computational tasks, while a Big Data cluster performs data processing, which is I/O-bound.

Many works reported that an ARM cluster is roughly 2-9 times slower than an Intel-based cluster, but better in terms of power consumption for several benchmarks [10], [11], [29]. But, Big Data processing on a cluster made with ARM SoC boards has not been widely studied. and other work concluded that this kind of cluster is not ready for Big Data with Hadoop [14]. Despite that, we have successfully achieved a good result by demonstrating our cluster with Apache Spark over HDFS.

According to Table I, the cluster best configuration can process 34 GB size of data in approximately 65 minutes. The cluster's best processing throughput is 0.51 GB/min, while the Gray Sort winner's throughput per node, 12 cores, is 0.69 GB/min (total throughput is 1.42 TB/min for 2,100 Hadoop nodes) [22]. Although, they cannot be compared directly as it is different benchmark and the data's size are also at the different scale, 34 GB v.s. 100 TB. The cluster's lowest power consumption is the network-based configuration with 2 SSDs attached. It consumed only 0.099 and 0.061 kWh for both versions of benchmarks, and it is unsurprisingly slowest configuration. The fastest configuration is the cluster with 20 SSDs attached as it provides full data locality and consumed power only 0.167 and 0.113 kWh. Although it is just slightly faster than the configuration with 20 mechanical hard drives attached, but it consumed only 50% of the power for the same task, thanks to the nature of SSDs.

Although Big Data processing is an I/O-bound, it is found that the cluster's CPUs are spent for the I/O rather than the computing tasks. To perform better, the cluster still requires more powerful processors nonetheless.

Not only the CPU's limitation, but also the software stack prevents the use of low-cost ARM boards for processing Big Data. Fortunately, many newer version of this family of boards will equip faster and also multi-core processors. Hardware

upgrade of the cluster will be obviously done by replacing with higher CPU boards and the faster networking, as it is found that the disk I/O is still enough. From the observation, a cluster made with ARM SoC boards has an important property, commodity, beyond a specially designed ARM server.

We also observe that there are some ARM boards with a large number of GPU processors, e.g. 192 cores. Unlike computing-intensive tasks, that kind of co-processors does not fit the area of Big Data directly, but it may help improving general computation of the common software stack, like HDFS. At the moment, we have been focusing on improving software stacks around HDFS to gain better processing rate for ARM clusters. We will also observe and study the performance, as well as the garbage collection, of JVMs in hope of the optimization will improve the overall cluster throughput.

ACKNOWLEDGMENT

This work is supported by Suranaree University of Technology. The authors would like to thank anonymous reviewers for several useful feedbacks.

REFERENCES

[1] T. White, *Hadoop: The Definitive Guide*, 1st ed. O'Reilly Media, Inc., 2009.

[2] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," in *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, ser. SOSP '03. New York, NY, USA: ACM, 2003, pp. 29-43.

[3] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107-113, Jan. 2008.

[4] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," in *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, May 2010, pp. 1–10.

[5] N. Randewich. (2014, Mar.) Intel invested $740 million to buy 18 percent of Cloudera. [Online]. Available: http://www.reuters.com/article/2014/03/31/us-intel-cloudera-idUSBREA2U0ME20140331

[6] MITRE Corporation. (2014) CVE-2014-0160. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160

[7] R. Lawler. (2014, Apr.) Cloudflare challenge proves worst case scenario for Heartbleed is actually possible. [Online]. Available: http://www.engadget.com/2014/04/11/heartbleed-openssl-cloudflare-challenge/

[8] J. Hamilton, "Cooperative expendable micro-slice servers (CEMS): low cost, low power servers for internet-scale services," in *Conference on Innovative Data Systems Research (CIDR)*, 2009.

[9] E. Upton and G. Halfacree, *Raspberry Pi User Guide*. John Wiley & Sons, 2013.

[10] Z. Ou, B. Pang, Y. Deng, J. Nurminen, A. Yla-Jaaski, and P. Hui, "Energy- and cost-efficiency analysis of ARM-Based clusters," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, May 2012, pp. 115–123.

[11] K. L. Keville, R. Garg, D. J. Yates, K. Arya, and G. Cooperman, "Towards fault-tolerant energy-efficient high performance computing in the cloud," in *2013 IEEE International Conference on Cluster Computing (CLUSTER)*. Los Alamitos, CA, USA: IEEE Computer Society, 2012, pp. 622–626.

[12] S. J. Cox, J. T. Cox, R. P. Boardman, S. J. Johnston, M. Scott, and N. S. OBrien, "Iridis-pi: a low-cost, compact demonstration cluster," *Cluster Computing*, pp. 1–10. [Online]. Available: http://link.springer.com/article/10.1007/s10586-013-0282-7

[13] J. Whitehorn. (2013, Nov.) Raspberry flavoured Hadoop. [Online]. Available: http://www.iDataSci.com/1/post/2013/11/stratu-eu-ignite-slides-raspberry-flavoured-hadoop.html

[14] Y. Wang. (2013, Dec.) Hadoop MapReduce performance study on ARM cluster. [Online]. Available: http://www.slideshare.net/airbots/hadoop-mapreduce-performance-study-on-arm-cluster

[15] M. Zaharia, et al., "Spark: Cluster computing with working sets," in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'10. Berkeley, CA, USA: USENIX Association, 2010, p. 10-10.

[16] F. Sonnenwald. (2013, Jul.) Pi-Alikes: Let's take a look at some of the other low cost computers. [Online]. Available: http://pi.gate.ac.uk/pages/hardware.html

[17] L. Wang, et al., "BigDataBench: A big data benchmark suite from internet services," *arXiv preprint arXiv:1401.1406*, 2014. [Online]. Available: http://arxiv.org/abs/1401.1406

[18] M. Odersky, et al., "An overview of the Scala programming language," in *Technical Report IC/2004/64*. EPFL, Lausanne, Switzerland, 2004.

[19] A. Rowstron, et al., "Nobody ever got fired for using Hadoop on a cluster," in *Proceedings of the 1st International Workshop on Hot Topics in Cloud Data Processing*, ser. HotCDP '12. New York, NY, USA: ACM, 2012, pp. 2:1-2:5.

[20] K. Elmeleegy, "Piranha: Optimizing short jobs in Hadoop," *Proc. VLDB Endow.*, vol. 6, no. 11, p. 985996, Aug. 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2536222.2536225

[21] S. Capper. (2013, Mar.) Hadoop DFS performance. [Online]. Available: http://www.slideshare.net/linaroorg/lca13-hadoop-1

[22] C. Nyberg, M. Shah, and N. Govindaraju. (2013) Sort benchmark home page. [Online]. Available: http://sortbenchmark.org/

[23] J. C. Adams and T. H. Brom, "Microwulf: A Beowulf cluster for every desk," in *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '08. New York, NY, USA: ACM, 2008, pp. 121-125.

[24] Cubietech Limited. (2013) Hadoop on Cubieboard. [Online]. Available: http://cubieboard.org/2013/08/01/hadoophigh-availability-distributed-object-oriented-platform-on-cubieboard/

[25] M. Zaharia, et al., "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012, p. 2-2.

[26] Y. Zhang, Q. Gao, L. Gao, and C. Wang, "iMapReduce: a distributed computing framework for iterative computation," *Journal of Grid Computing*, vol. 10, no. 1, pp. 47–68, Mar. 2012. [Online]. Available: http://link.springer.com/article/10.1007/s10723-012-9204-9

[27] P. Uthayopas and N. Benjamas, "Impact of I/O and execution scheduling strategies on large scale parallel data mining," in *6th International Conference on New Trends in Information Science and Service Science and Data Mining (ISSDM)*, Oct. 2012, pp. 654–660.

[28] N. Balakrishnan, "Building and benchmarking a low power ARM cluster," Master's thesis, The University of Edinburgh, 2012.

[29] M. Jarus, S. Varrette, A. Oleksiak, and P. Bouvry, "Performance evaluation and energy efficiency of high-density HPC platforms based on intel, AMD and ARM processors," in *Energy Efficiency in Large Scale Distributed Systems*, ser. Lecture Notes in Computer Science, J.-M. Pierson, G. D. Costa, and L. Dittmann, Eds. Springer Berlin Heidelberg, Jan. 2013, pp. 182–200.