EVALUATION OF A LOW-POWER HADOOP CLUSTER BASED ON THE ZYNQ ARM-FPGA SOC

Ovidiu PLUGARIU[1], Alexandru CALIN[2], and Lucian PETRICA[3]

Distributed computing is important to many web and scientific applications. The quest for lower power dissipation and energy consumption in distributed applications has led to the implementation of ARM processor based distributed systems. The advent of integrated accelerators such as Field Programmable Gate Arrays (FPGA) promises to provide further reductions in power, but it is yet unclear to the scientific community how FPGAs should be utilized to maximize their benefit to the distributed system. To answer this research question, in this paper we evaluate a ARM-FPGA distributed system which utilizes the Xilinx Zynq SoC as processor and Apache Hadoop as a programming environment and task scheduler. We evaluate the system on industry-standard benchmarks: MRBench, TestDFSIO, Wordcount. Compared to a similar system with x86 processors, the ARM-based system dissipates 5 times less power. Additionally, we observe that all applications are IO bound on the ARM system, which indicates that the most efficient use for FPGAs in such a system is for offloading IO compression algorithms.

Keywords: Hadoop, ARM, distributed computing

1. Introduction

All modern web-based applications, ranging from web analytics, to search, and social network infrastructure, must process thousands of requests per second. These applications therefore require vast amounts of computing power, i.e. hundreds to thousands of co-located, networked servers which work simultaneously and collaboratively to process information and generate results. This type of computing infrastructure is called a cluster, and the process of collaborative task solving is called distributed computing. Modern distributed computing occurs in data centers, facilities which house interconnected servers and associated components, such as networking and storage systems [1]. As such, data-center efficiency dictates the profitability of most on-line businesses. The majority of data-center operational expenditure is the cost for electrical

[1]PhD Student, DCAE Department, University "Politehnica" of Bucharest, Romania

[2]PhD Student, DCAE Department, University "Politehnica" of Bucharest, Romania

[3]Lector, DCAE Department, University "Politehnica" of Bucharest, Romania, e-mail: lucian.petrica@upb.ro

energy for servers and cooling equipment, which may be reduced by improving the energy-efficiency of the server processors utilized.

Low-power computing is an area of great interest for industry and academia because it enables reductions in the cost of distributed computing in data-centers. Low-power servers based on ARM processors have the potential of increased efficiency - more operations per Watt and per Joule, compared to servers which utilize processors of the x86 and PowerPC architectures, which are faster at the cost of significantly greater power dissipation [2]. With the advent of mass-produced ARM-based computing hardware such as the Raspberry Pi, it has even become possible for enthusiasts to construct cheap, energy-efficient clusters [3, 4, 5]. By utilizing SDCard storage instead of expensive disk drives, and by utilizing exclusively passive cooling, this new class of cluster opens up distributed computing to virtually anyone.

Despite the potential of these low-power clusters to replace x86 clusters in some applications, to our knowledge there is no comparison in the scientific literature between the two types of cluster on industry-standard benchmarks. Without such a comparison it is difficult to evaluate the computational capability and efficiency of low-power clusters. How much more efficient is a low-power cluster compared to a x86 cluster? Are applications on these clusters IO-bound because of the slow speed of the SDCard storage? On the answers to these questions depends the feasibility of new lines of research. For example, it is possible to utilize IO compression to reduce the amount of data transferred to and from storage, but the overhead of compression and decompression typically reduces overall application performance even further [6]. A potential solution lies with combined use of ARM processors and Field Programmable Gate Arrays (FPGAs), a type of accelerator capable of emulating digital circuits at high speed [7]. FPGAs may be utilized to offload computationally difficult tasks from the ARM CPU, including the IO compression and decompression [8, 9].

In this paper we present an evaluation of a low-power, passively cooled computing cluster similar to the previously mentioned Raspberry Pi clusters, but based on Zynq-7000 ARM-FPGA Systems-on-Chip (SoC). We evaluate the cluster utilizing industry-standard benchmarks for distributed computing. In previous work on ARM-FPGA clusters in [10], the authors offload signal processing applications, i.e. a FIR filter, to the Zynq FPGA. Instead of taking this ad-hoc route, our benchmarks aim to identify the principal computational and IO weaknesses of our cluster, when compared to a x86 cluster, in order to help the research community to formulate a coherent strategy to utilize the FPGA to mask or minimize these weaknesses. We determine that application execution is almost universally IO-bound on our ARM-FPGA cluster, which indicates that IO compression on FPGA is a potentially useful and unique scenario for ARM-FPGA SoCs compared to ARM SoCs.

2. Low-Power Distributed Computing with Hadoop

Apache Hadoop [11] is an open-source software framework written in Java for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware, i.e., any computer with a Linux operating system. The core of Apache Hadoop consists of a storage module, the Hadoop Distributed File System (HDFS) [12], a programming model, MapReduce [13], and a task scheduler, YARN (Yet Another Research Negotiator) [14]. HDFS splits files into large blocks and distributes them amongst the nodes in the cluster for storage. To process the data, YARN transfers packaged MapReduce code for nodes to execute in parallel on the data stored locally. This approach takes advantage of data locality nodes manipulating the data that they have on hand to allow the data to be processed without moving it between nodes, and therefore faster and more efficiently.

A Hadoop cluster consists of a single master node and multiple worker nodes. The master node hosts several management services: a Job Tracker, Task Tracker, NameNode, and DataNode. A worker node acts as both a DataNode and TaskTracker, though it is possible to have data-only worker nodes and compute-only worker nodes. In a larger cluster, the HDFS is managed through a dedicated NameNode server to host the file system index, and a secondary NameNode that can generate snapshots of the namenode's memory structures, thus preventing file-system corruption and reducing loss of data.

The MapReduce Engine consists of one JobTracker, to which client applications submit MapReduce jobs. The JobTracker pushes work out to available TaskTracker nodes in the cluster, striving to keep the work as close to the data as possible. If a TaskTracker fails or times out, that part of the job is rescheduled.

MapReduce is a programming model for data processing. Hadoop can run MapReduce programs written in various languages (Java, Ruby, Python, C++ etc.). Most importantly, MapReduce programs are inherently parallel, thus putting very large-scale data analysis into the hands of anyone with enough machines at their disposal. MapReduce works by breaking the processing into two phases: the map phase and the reduce phase. Each phase has key-value pairs as input and output, the types of which may be chosen by the programmer. The programmer also specifies two functions: the map function and the reduce function.

A typical Hadoop cluster (Figure 1) has one NameNode who coordinates a list of DataNodes. The NameNode splits the tasks to the DataNodes, who performs the actual computing. The speed of the computation performed by the cluster is the result of the node-network aggregate, which is given by the nodes hardware architecture and the speed of the network. The Secondary NameNode is optional and is used to assure redundancy for the NameNode.

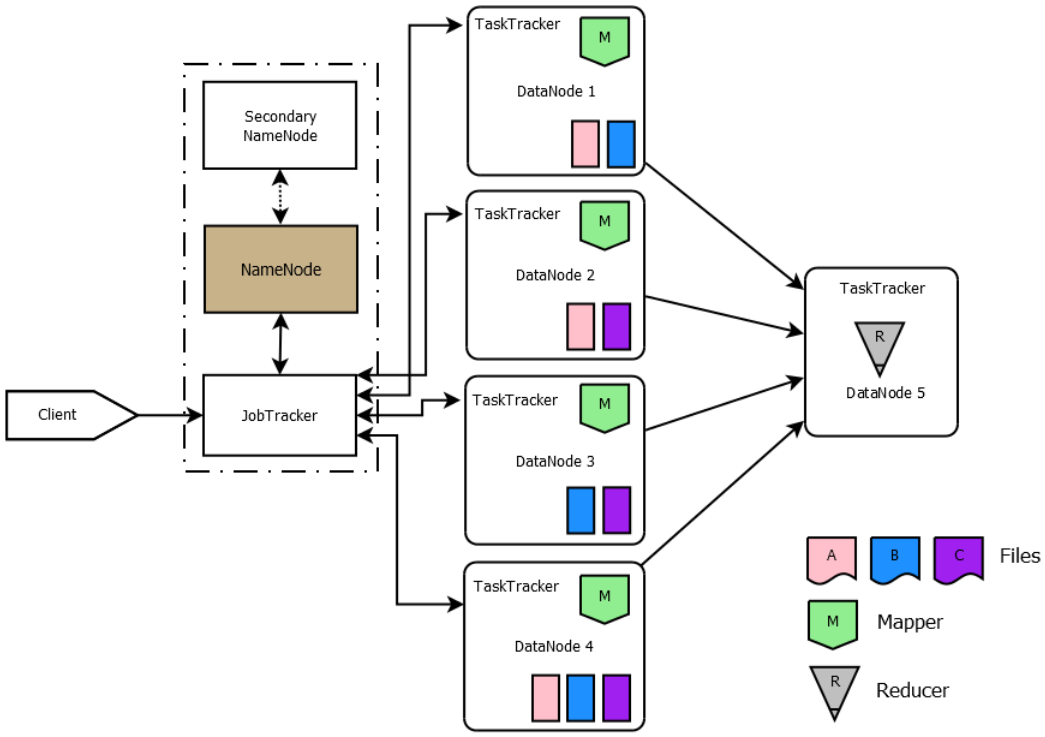

Fig. 1. Hadoop cluster architecture [15]

3. The ARM and x86 Clusters

3.1. The ARM Processing Cluster

In Figure 2 we present the block diagram of the ARM processor cluster utilized in our experiments. The cluster does not consist entirely of ARM processors, as the master node is a x64 CPU HP ProLiant server, with sufficient RAM for executing the NameNode of HDFS which maintains the file system and metadata for all processes. It also ensures load balancing and the replication of data. These cluster maintenance tasks generate many processes, thereby a powerful x86 CPU and a larger memory is preferred to an ARM processor. The NameNode has a dual-core 64bit Intel Core CPU operating at 3GHz, 200GB hard-drive and 2GB RAM memory. The operating system is Ubuntu 14.04.2 LTS.

The JobTracker schedules all data processing tasks to ARM nodes. These are implemented with ZedBoard development boards, each equipped with a Xilinx XC7Z020 Zynq SoC, which integrates a dual-core ARM Cortex-A9 at 667MHz and an FPGA. The board is also equipped with 512MB of DDR memory and a 8GB SD card for storage. The SD card is rated for approximately 14MB/s write speed and a cached read speed of 364.36MB/s. Each ZedBoard system executes a Xillinux operating system, which is a Linux distribution

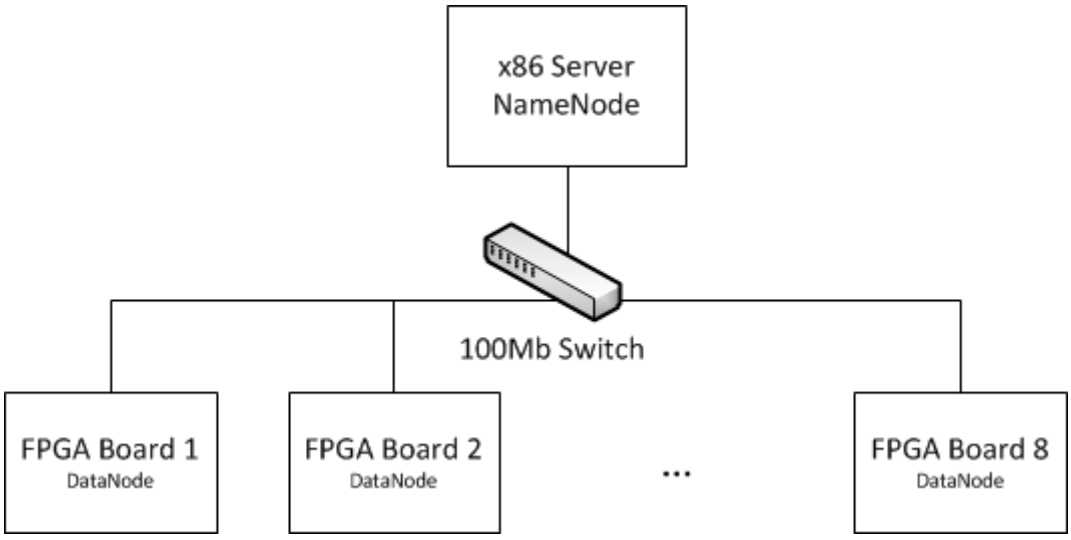

Fig. 2. Architecture of ARM Zedboard cluster

based upon Ubuntu LTS 14.04 for ARM. The ZedBoards are capable of Gigabit Ethernet networking, however as we do not propose to benchmark network performance, the boards are connected to each-other and the ProLiant server through a 100Mb switch, as illustrated in Figure 2.

3.2. The x86 Processing Cluster

The x86 Hadoop cluster consists of commodity computers co-located inside a laboratory, with hostnames Catalyst01 to Catalyst15. For reasons such as ease of deployment and dependency isolation, we utilize Linux containers (LXC) as a way of hosting the Hadoop environment. We selected Linux Containers (LXC) because it is a lightweight virtualization technology for computers running GNU/Linux, facilitating cluster deployment while not hindering performance significantly. Inside of an LXC container on each physical computer executes Ubuntu 14.04 Trusty (32 bit) that hosts the Hadoop environment. Containers communicate through a software network bridge on the Host OS of the physical node. A schema of how the Hadoop components can be found throughout the cluster can be seen in Figure 3. A generic relationship between two host computers and their corresponding isolated Hadoop environment is depicted in Figure 4.

Two types of hardware nodes have been utilized, which have the following specifications:

- HDD: 70GB,RAM : 2GB, CPU: Intel Pentium 4 CPU @ 3.00GHz, Host OS : LinuxMint LMDE Cinnamon Edition, LXC OS: Ubuntu 14.04 (32 bit)

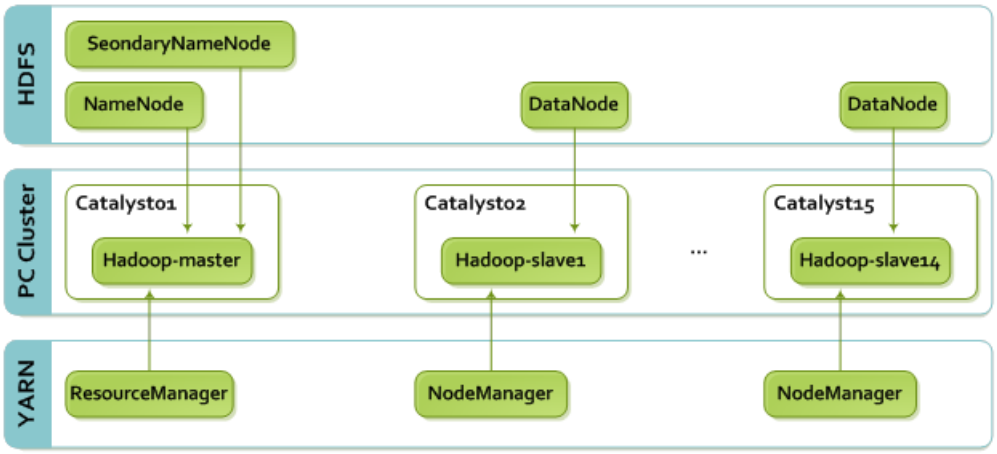

Fig. 3. Hadoop components in the x86 cluster

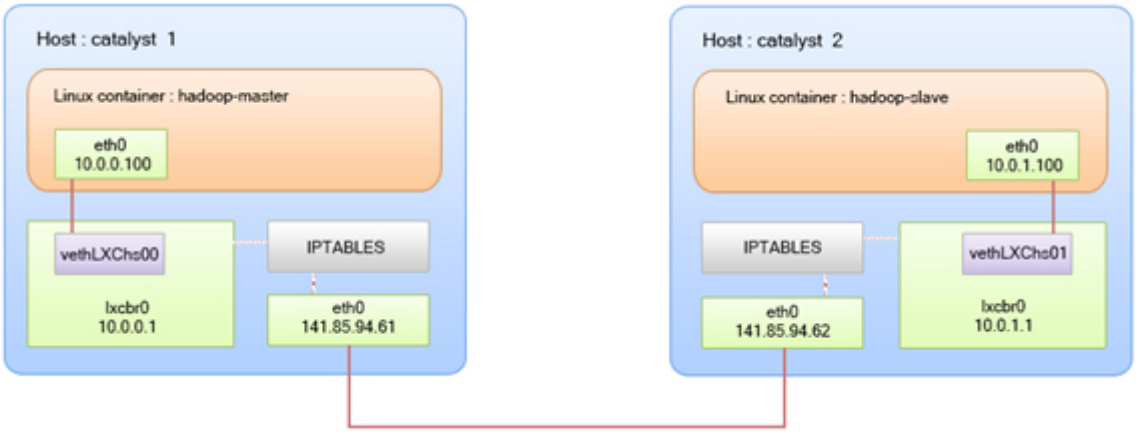

Fig. 4. Architecture of x86 cluster

- HDD: 224GB, RAM: 3GB, CPU: Intel Celeron CPU E3400 @ 2.60GHz, Host OS : LinuxMint LMDE Cinnamon Edition, LXC OS: Ubuntu 14.04 (32 bit)

The complete Hadoop cluster totals 15 computers with 1 master NameNode and 14 slave nodes, for the purpose of this paper, only 8 slave nodes have been utilized, 4 of each hardware configuration. We selected only 8 nodes in order to provide a meaningful comparison to the 8-node ARM cluster with regard to processing performance. As is the case with the ARM cluster, only slave nodes process data.

4. Evaluation

We perform a performance evaluation of both clusters utilizing standard Hadoop benchmarks designed to evaluate all aspects of Hadoop performance.

4.1. Wordcount

The Wordcount benchmarking application counts all the occurences of a word from a given set of text files in HDFS, using MapReduce. The input for this benchmarking application is the Wikipedia Corpus [16], totaling 9.6 GB of text after it is processed into a corpus database. Wordcount is an evaluation of compound Hadoop performance (HDFS and MapReduce), but because the computation is exceedingly simple, i.e. repeated string comparison, Wordcount is more IO intensive than computationally intensive.

We loaded the Wikipedia text corpus into HDFS and performed Wordcount utilizing all 8 nodes of each cluster type. The ARM cluster processed the corpus in 19386 seconds (5.23 hours) while the x86 cluster completed Wordcount in 1982 seconds (0.55 hours). The x86 cluster performed almost 10 times faster compared with the ARM cluster. Because the data transfer speed of the SD card, which is the principal storage device on the Zedboards, is an order of magnitude slower than a mechanical hard drive, HDFS performance is a bottleneck for the ARM cluster.

4.2. Terasort

Terasort is the most well-known Hadoop benchmark. The goal of Terasort is to sort 1TB of data (or any other size) as fast as possible in a distributed fashion. This benchmark exercises both the HDFS and the MapReduce layers of the cluster. Since the computed algorithm is more complex than that of Wordcount, Terasort is focused more on computing performance and less on HDFS performance. We generated unsorted data sets ranging in size from 0.5 GBytes up to 3 GBytes in 0.5 GByte increments, utilizing the Teragen software bundled with Terasort. We loaded the generated data-sets into HDFS.

Figure 5 illustrates the performance of the two clusters while executing Terasort. For both clusters, the time required to completely sort the data-set is proportional to the size of the data-set. The x86 cluster is on average two times faster, for all dataset sizes. The advantage over the ARM cluster is less than with Wordcount because Terasort is less IO-bound, and more compute-intensive.

4.3. MRBench

MRBench executes a small Hadoop job in a loop for a large number of times, evaluating the performance of the job management mechanisms of the cluster. The number of jobs launched is the key parameter of MRBench. As such it is a very complimentary benchmark to the large-scale TeraSort benchmark suite because MRBench checks whether small job runs are responsive and running efficiently on the cluster. It focuses on the MapReduce layer as its impact on the HDFS layer is very limited. However, MRBench is not computationally intensive for slave nodes, but instead stresses the JobTracker.

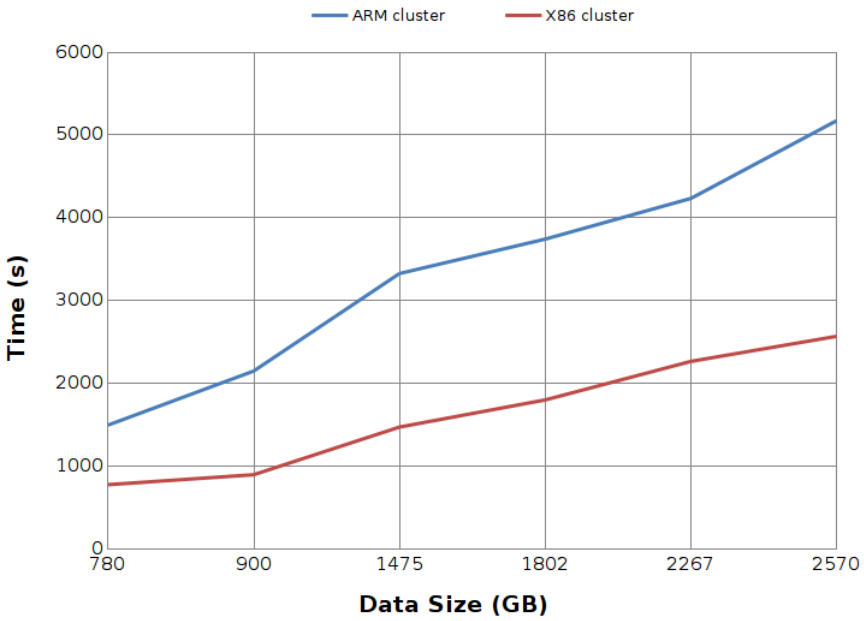

Fig. 5. Terasort results

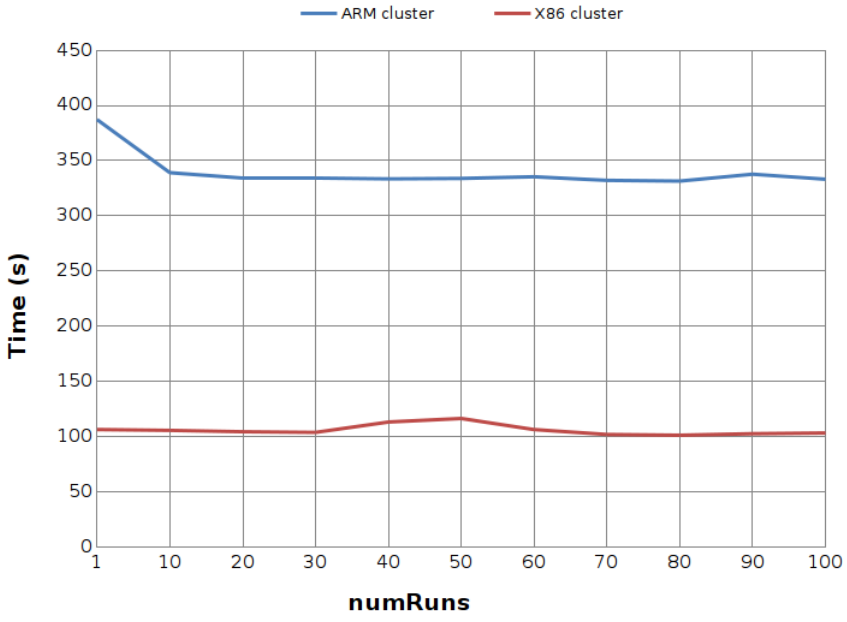

Fig. 6. MRBench results

We executed MRBench with a number of jobs ranging from 1 to 100. Figure 6 presents the performance comparison of the two clusters on MRBench. It is immediately evident that jobs are started and managed more slowly on

the ARM cluster. Despite the fact that the JobTracker of the ARM cluster is a traditional x86 server, the under-powered ARM processors of the slave nodes take longer to respond to requests and overall MRBench performs 3 times better on x86 than on the ARM cluster.

4.4. TestDFSIO

Test DFSIO is a benchmark for read/write operations from and to HDFS. It is useful for discovering IO performance bottlenecks. We perform several evaluations for a data volume of 2GB split equally into an increasingly larger number of files. The purpose of this evaluation is to identify how the data granularity is affecting the IO speed of the Hadoop cluster.

The most notable metrics for TestDFSIO are Throughput and Average IO rate, expressed in MBytes/second. Both metrics are computed from the size of the file written (or read) by the individual map tasks, and the time elapsed for reading or writing. Equation 1 calculates the throughput of a TestDFSIO job which utilizes N map tasks. The average IO rate is defined in Equation 2. The index $i \in [1, N]$ denotes the individual map tasks, while S_i denotes the file size read or written by the map task, and T_i denotes the time required to perform the read or write of the file.

$$Throughput(N) = \frac{\sum_{n=1}^{N} S_i}{\sum_{n=1}^{N} T_i} \tag{1}$$

$$AverageIOrate(N) = \frac{1}{N} \sum_{n=1}^{N} \frac{S_i}{T_i} \tag{2}$$

Table 1

Average TestDFSIO Performance

Operation	Average I/O (MB/s)
x86 Cluster HDFS Read	23.47
ARM Cluster HDFS Read	9.7
x86 Cluster HDFS Write	5.95
ARM Cluster HDFS Write	2.1

Average performance for both reads and writes is summarized in Table 1. Figure 7 illustrates the average read IO throughput of the two clusters under evaluation. Except when handling large files, the read throughput is 2-3 times greater on the x86 cluster, which utilizes mechanical hard-disks.

Write performance is illustrated in Figure 8. Similarly to reads, the x86 cluster outperforms the ARM cluster. For both clusters, write performance degrades when files become smaller, as the HDFS file management overhead becomes more time-consuming than the actual writes to the storage devices of the cluster nodes.

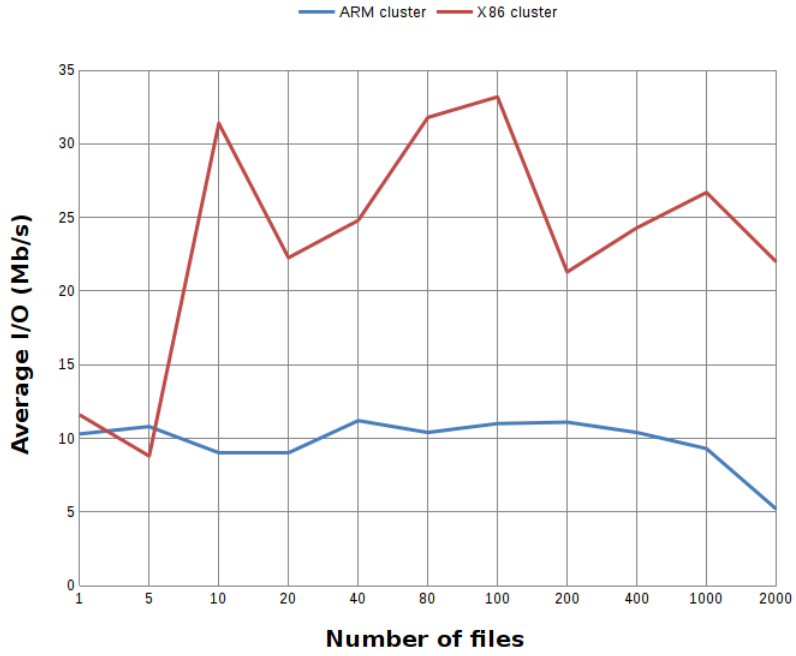

Fig. 7. TestDFSIO Read Average I/O

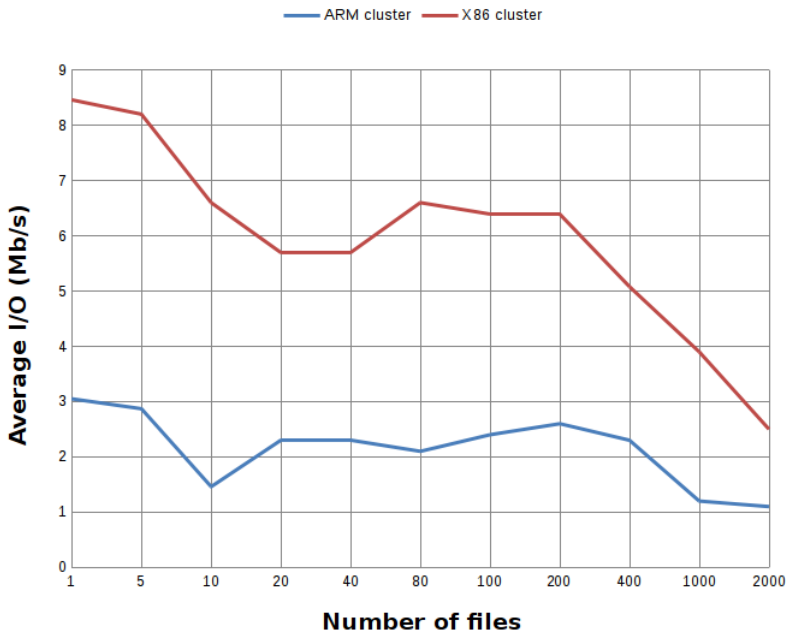

Fig. 8. TestDFSIO Write Average I/O

4.5. Power Dissipation

We evaluate the power dissipation of the ARM cluster in order to provide a rough comparison of the two clusters for this important data-center metric. We utilize a Ubiquity mPower Pro smart power outlet, with power measurement capability. The ARM and x86 cluster average power dissipation is recorded during the execution of the distributed computing benchmarks and presented in Table 2. We note that the power dissipated in the ARM cluster mostly by the name-node server, at 190 Watts. The worker ARM nodes dissipate a maximum of 20 Watts together. By taking into account the execution times for the Wordcount, Terasort, and MRBench benchmarks, we have also calculated the energy consumed by each cluster to compute the benchmarks. The ARM cluster is more energy-efficient on all benchmarks except Wordcount, which is IO bound on the ARM cluster and where the performance difference between the two systems is the most pronounced.

Table 2

Power Dissipation

Computing System	ARM Cluster	x86 Cluster
Power Dissipation (W)	210	1200
Wordcount Energy (MJ)	4.07	2.38
Terasort Energy (MJ)	0.73	1.92
MRBench Energy (MJ)	0.07	0.12

5. Conclusions

We have presented a comparative evaluation of two Hadoop clusters, equipped with x86 and ARM processors respectively. The ARM cluster was constructed from low-cost, commercially available ZedBoard development boards, which are frequently utilized in the research community for FPGA offloading applications. We utilized tha standard set of Hadoop benchmarks to determine the relative performance of the clusters.

From our evaluations it has become evident that IO throughput, in both reads and writes, is the principal drawback of the ARM cluster as presented in our work. Despite this fact, the ARM cluster dissipates five times less power than the x86 cluster and is more energy-efficient on most benchmarks. Our results guide future work towards solving the IO bottleneck through e.g. FPGA offload of Hadoop compression, in order to further increase possible power and energy benefits of the ARM cluster.

REFERENCES

[1] "EPA datacenter report." http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf. Accessed: 2016-04-07.

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

[3] P. Abrahamsson, S. Helmer, N. Phaphoom, L. Nicolodi, N. Preda, L. Miori, M. Angriman, J. Rikkila, X. Wang, K. Hamily, *et al.*, "Affordable and energy-efficient cloud computing clusters: The Bolzano Raspberry Pi cloud cluster experiment," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 2, pp. 170–175, IEEE, 2013.

[4] F. P. Tso, D. R. White, S. Jouet, J. Singer, and D. P. Pezaros, "The Glasgow Raspberry Pi cloud: a scale model for cloud computing infrastructures," in *Distributed Computing Systems Workshops (ICDCSW), 2013 IEEE 33rd International Conference on*, pp. 108–112, IEEE, 2013.

[5] 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*, vol. 17, no. 2, pp. 349–358, 2014.

[6] Y. Chen, A. Ganapathi, and R. H. Katz, "To compress or not to compress-compute vs. IO tradeoffs for MapReduce energy efficiency," in *Proceedings of the first ACM SIGCOMM workshop on Green networking*, pp. 23–28, ACM, 2010.

[7] C. Bira, R. Hobincu, L. Petrica, V. Codreanu, and S. Cotofana, "Energy-efficient computation of L1 and L2 norms on a FPGA SIMD accelerator, with applications to visual search," pp. 32–37, 2014.

[8] S. Rigler, W. Bishop, and A. Kennings, "FPGA-based lossless data compression using Huffman and LZ77 algorithms," in *Electrical and Computer Engineering, 2007. CCECE 2007. Canadian Conference on*, pp. 1235–1238, IEEE, 2007.

[9] J. Ouyang, H. Luo, Z. Wang, J. Tian, C. Liu, and K. Sheng, "FPGA implementation of GZip compression and decompression for IDC services," in *Field-Programmable Technology (FPT), 2010 International Conference on*, pp. 265–268, IEEE, 2010.

[10] Z. Lin and P. Chow, "Zcluster: A Zynq-based Hadoop cluster," in *Field-Programmable Technology (FPT), 2013 International Conference on*, pp. 450–453, IEEE, 2013.

[11] T. White, *Hadoop: The definitive guide.* " O'Reilly Media, Inc.", 2012.

[12] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pp. 1–10, IEEE, 2010.

[13] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[14] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, *et al.*, "Apache hadoop yarn: Yet another resource negotiator," in *Proceedings of the 4th annual Symposium on Cloud Computing*, p. 5, ACM, 2013.

[15] "Introducing Hadoop, part II." http://www.rohitmenon.com/index.php/introducing-hadoop-part-ii/. Accessed: 2016-04-07.

[16] "Creating a text corpus from wikipedia." http://trulymadlywordly.blogspot.ro/2011/03/creating-text-corpus-from-wikipedia.html. Accessed: 2016-04-07.