An ARM-Based Hadoop Performance Evaluation Platform: Design and Implementation

Xiaohu Fan[1(✉)], Si Chen[1], Shipeng Qi[1], Xincheng Luo[1], Jing Zeng[1], Hao Huang[2], and Changsheng Xie[3]

[1] School of Computer Science and Technology, HUST, Wuhan, China
{fanxiaohu, M201272616, qishipeng, luoxc613, zengjing}@hust.edu.cn
[2] School of Software Engineering, HUST, Wuhan, China
thao@hust.edu.cn
[3] Wuhan National Laboratory for Optoelecgtronics, 1037 Luoyu Road, Wuhan, China
cs-xie@hust.edu.cn

Abstract. As the growth of cluster scale, huge power consumption will be a major bottleneck for future large-scale high performance cluster. However, most existing cloud-clusters are based on power-hungry X86-64 which merely aims to common enterprise applications. In this paper, we improve the cluster performance by leveraging ARM SoCs which feature energy-efficient. In our prototype, cluster with five Cubieboard4, we run HPL and achieve 9.025 GFLOPS which exhibits a great computational potential. Moreover, we build our measurement model and conduct extensive evaluation by comparing the performance of the cluster with WordCount, k-Means (etc.) running in Map-Reduce mode and Spark mode respectively. The experiment results demonstrate that our cluster can guarantee higher computational efficiency on compute-intensive utilities with the RDD feature of Spark. Finally, we propose a more suitable theoretical hybrid architecture of future cloud clusters with a stronger master and customized ARMv8 based TaskTrackers for data-intensive computing.

Keywords: HPC · ARM cluster · Cost-effective · Data-intensive

1 Introduction

As scale and dimension increase with the combination of Cloud Computing and Internet of Things in the Big Data era, massive data burden more pressure on compute and storage performance. Fortunately, Hadoop framework enables an alternative collaboration cluster implementation of High Performance Computing (HPC) and Data Center, which holds a strong market share. However, CTOs hesitated by the cost and security problems of an enterprise deployment. Besides, data-intensive applications require tremendous power. Cases [1, 2] show that total energy cost over a few years of operation exceeds the cost of the hardware. Along with the expansion of cluster scale, future HPC systems will be limited by power consumption.

S. Guo et al. (Eds.): CollaborateCom 2015, LNICST 163, pp. 82–94, 2016.
DOI: 10.1007/978-3-319-28910-6_8

Consequently, the weight of supercomputers ranking criterion gradually shifted from Top500 [3] to Green500 [4]. Most of the Top500 and Green500 leaders are x86 processor based system combined with GPU and other accelerate technologies. Analysis of current HPC shows that 40–60 % of the energy consumption is due to the compute nodes, 10 % to the interconnect and storage, and major remainder to infrastructure itself especially, cooling [1, 5]. Taking the performance and cost of power into account, the born nature of ARM chips is historically related to lower power consumption and cost efficient, which makes ARM a promising candidate in terms of future Data Centers. Nowadays, the 8 cores Cortex-A15 [6] processors provide satisfied performance in cost guaranteed A15 giant share of mobile field. The next generation ARMv8 instruction set, namely the A50 series [7] features a 64-bit address space, which makes ARM chips real alternate player for HPC and cloud computing.

Following this trend, we attempt to evaluate the performance of ARM based Hadoop cluster, considering network bandwidth, workload, performance and Total Own Cost of future enterprise Hadoop cluster. In this paper, we use A15 based Cubieboard4 to build a cluster with Hadoop and Spark. In order to verify whether ARM clusters is suitable for common data-intensive applications or not, we try to evaluate the performance and seek for constrains. With hardware environment limited and following common practice, we tried 3 different configuration settings to verify the optimized cluster performance with same workload on same application in weak scaling approach. Main contribution lies in: (1) Evaluate the feasibility of a Hadoop cluster based on the current leader chip in the mobile domain with common applications. (2) Present a simple method to estimate efficiency of cluster by abstracting the CPU execution time of a small cluster as prototype and learn experience. (3) Propose a customized hybrid architecture with stronger master node and ARMv8 TaskTrackers cluster. The experiments show that this architecture can achieve great performance under higher cluster bandwidth.

This paper is organized as follows. The Sect. 2 gives some background information of low power HPC. In Sect. 3, we propose our theory, measure method and model, bind the micro CPU time and macro execution time to evaluate the performance and the bottleneck of ARM based cluster. Experimental results, analysis and a case study of improvement is shown in Sect. 4 with a theoretical future architecture. Finally, Sect. 5 presents the conclusions.

2 Related Works

As great methods that Green500 leaders have taken. Google published its GFS [8], BigTable [9] and Map-Reduce [10] scheme during last decade, confirmed the distributed cluster is feasible in large-scale data applications. Supported by the Apache foundation, Hadoop collaborates the cluster performs satisfied and the rapid development is in continuous [11–13]. Besides, the emerging spark [14] using RDD to implement in-memory compute enhanced the performance of Hadoop clusters greatly.

However, vast majority Hadoop clusters are x86 based, barely ARM based trial except Raspberry Pi [15, 16] clusters of Cortex-A8 based with a 700 MHz processor and 512 MB RAM. Comparison of x86 and ARM architectures of data center has been

discussed by [17]. In comparison of big data workload on wimpy nodes and traditional big nodes, the Cortex-A8 based AppleTV2 cluster [18, 19] running HPL in a cluster environment attempt to assess the low power solution, achieved 460.4 MFLOPS. For common utility, [16] establishing pi spark cluster, however, is constrained by A8's performance, 100 Mb network and I/O bandwidth. Barcelona Supercomputing Center used 96 cores of Cortex-A9 based Barcelona cluster [20] in 2013 and update the cores to Cortex-A15 as Tibidabo [2] to achieve a theoretical 1046MFLOPS/W, but it is a super-compute-oriented special customized cluster. The NUS researchers reach new levels of performance and cost-efficiency [21], and proved that ARM based cluster is not fit for I/O intensive workloads but good at databases query processing. Large data processing in the recent fiery geographic information [22–25] required larger amount of power-hungry clusters.

Generally, most existing work about improving the energy efficiency and performance of cluster focused on x86 based Hadoop clusters. As for wimpy node clusters, either attempt the simple application scenario or build super customized cluster. In this paper, we use 5 standard Cortex-A15 based Cubeborad4 to construct a small Hadoop cluster without other accelerate technique to establish analysis models and run several common algorithms to evaluate the performance and cluster efficiency. It carried out a comparative study of applications with same workload under different configuration and we tried to find out constraints and improvement via analysis. The results of comparison verify the feasibility and advantage of our scheme.

3 Measurements-Driven Modeling

Ideally, all nodes keep peak capability determines the performance upper bound of the cluster. Affected by network bandwidth, communication overhead, I/O and other latency, it is widely accepted that the total cluster performance gradually decrease with the nodes expansion. As for single node, performance of ARM CPU can be measured by the rate of CPU running time in the total program execution time to get the efficiency of single node. For the cluster, we focus on the parallel processing efficiency, and use throughput or parallel speedup ratio to indicate the efficiency. The whole roadmap of our experiments is described in Fig. 1.

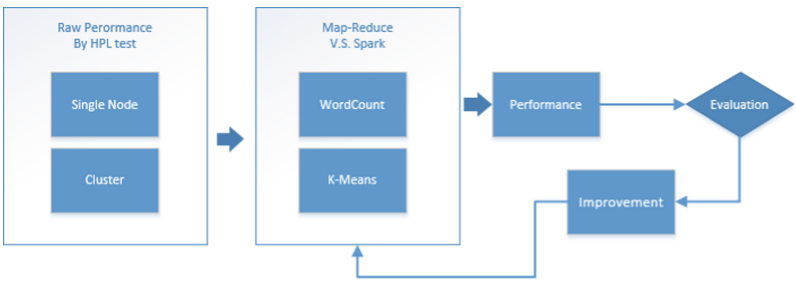

Fig. 1. Roadmap of experiments. Firstly, get peak performance. Then, analysis the performance according to the result, find the bottleneck to improve the efficiency of cluster and compare the result under different hardware settings

Firstly, we test the I/O and CPU performance of raw Cubieboard4 using simple command line and HPL with Ubuntu 14.04 on both single node and cluster to get actual results. Then, we evaluate the performance by running some applications in Hadoop and Spark. Finally, we improve the throughput according to our evaluation with bottleneck of Cubieboard4 cluster found.

We characterize data intensive execution on 1–5 small nodes by evaluating the performance of some well-known applications such as HPL, HDFS read/write, WordCount, k-Means, comparisons between Map-Reduce mode and Spark RDD. The analysis is based on the measurement of execution time, each actual throughput, CPU efficiency and power consumption.

Considering some negative effects: Cubieboard4, Linux OS, and Hadoop overhead themselves, communication offset of cores, attenuation of each node exist, we use the computation intensive HPL as a benchmark to determine whether the actual computing performance is suitable. Then, we use the performance of single node as the baseline of the cluster to evaluate the extension. According to [26] research, due to the overhead of CPU communication and offset of I/O the efficiency of large-scale clusters actually accounts for only a fraction of the running time. Thus, a Hadoop operation process is abstracted as Fig. 2 shown below, T is the total running time of the process, and T_{exe} is the CPU running time abstracted from the grey in the upper part of Fig. 2, $T_{cpu(i)}$ is the actual execution time of each CPU in the blue part. The red part in the T_{exe} is the communication synchronous overhead. Deviation does exist in this method, but simpler to measure and estimate with a viable accuracy.

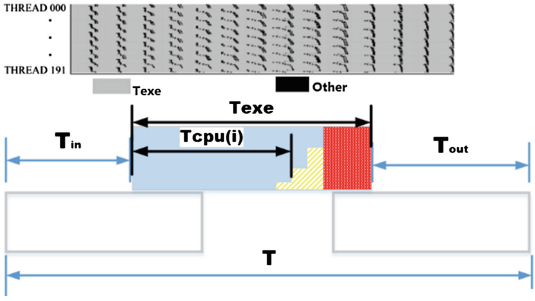

Fig. 2. Abstract CPU time in process of application on Hadoop (Color figure online)

As can be seen in Fig. 2, $T_{cpu(i)}$ is a micro measured value, T is a macro measured time. In order to facilitate the measurement we did not take a fixed large amount of the total task. For fairly comparison, each node is assigned the same sized task load, similar to a weak-scaling approach. According to Gustafson's law [27], communication overhead is a linear positive correlation with number of nodes in the cluster [28]. Thus, we are able to estimate the delay and overhead of cluster, and modeling the performance of cluster with deviation. We propose our estimate model as follow, n is the number of nodes, C is a constant for overhead, $C \times n$ is linear correlation with n.

$$Performance_{cluster} = \frac{Total\ Task}{Total\ Time} = \frac{n \times Task_{single-node}}{Time_{single-node} + C \times n} \quad (1)$$

In this method, once given the Task data and application, the workload is confirmed. With the results of the single node as a basic reference value, in addition with the results of the rest nodes, C value under the fixed workload could be calculated by fitting approach. The smaller constant C value is the greater performance of cluster achieves. Otherwise, similar type of application might not fit for ARM based cluster extension. In this model, execution time and total run time of the progress, CPU records are required for both single compute node and the whole cluster.

The cluster bandwidth depends on the minimal of network, board I/O interface, read/write of storage. Eff_{cpu} is a micro numerical calculation, while the Compute Ratio is a macro time accounting measurement, we can use results above to estimate the efficiency of the cluster, defined as $Eff_{cluster}$.

$$Eff_{cluster} = Eff_{cpu} \times Compute\ Ratio = \frac{\frac{\sum_{i=0}^{n} T_{cpu(i)}}{n}}{T_{exe}} \times \frac{T_{exe}}{T} = \frac{\sum_{i=0}^{n} T_{cpu(i)}}{n \times T} \quad (2)$$

4 Results and Analysis

We use 5 CubieBoard4 development boards to establish our cluster for evaluation via different hardware configuration running same application under same workload, in order to verify that whether the performance of cluster can be improved when hardware

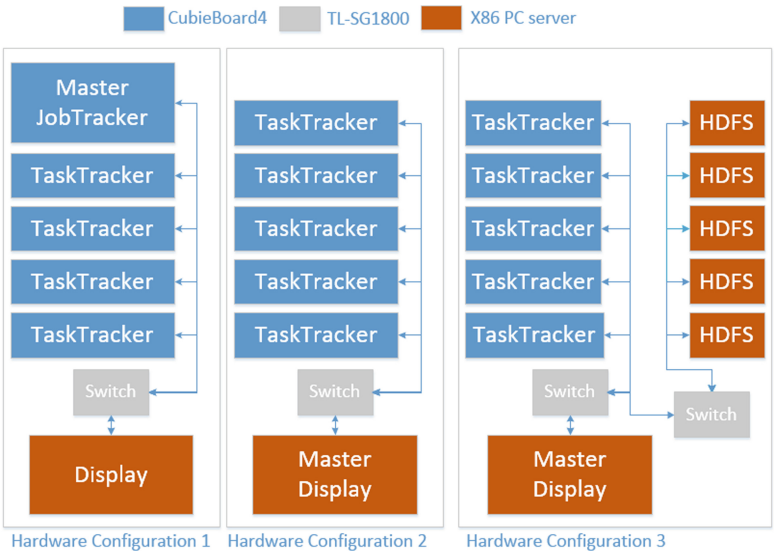

Fig. 3. Abstract architecture of 3 hardware configurations

limitation along with the factors changed. Following common practice, the bottleneck and constrains of the development board can be infer from the actual performance measurement. The architecture of hardware configurations is shown in Fig. 3.

Initially, we used 5 Cubieboard4 development boards to construct a pure ARM based cluster, as Configuration 1 shows. One x86 workstation is only used for console control and display by operators. The hardware parameters of experimental environment is shown in Table 1. Eight ports TP-LINK TL-SG1008 gigabit switch connected the cluster as a LAN,every ARM node and the x86 workstation are connected with CAT6 network cable.

Table 1. Hardware environment

Item	Intel	ARM	
CPU	Core i7	Cortex A15	Cortex A7
ISA	X86-64	ARMv71	
Cores	4	4	
Frequency	3.50 GHz	0.60–2.00 GHz	300–1.00 GHz
L1 Data cache	4x32 KB	32 KB	
L2 Cache	4x256 KB	2 MB	
Memory type	DDR3 2x8 GB	DDR3 2 GB	
USB	USB3.0	1xUSB3.0 OTG 4xUSB2.0	

4.1 Single Node Performance

We use Coremark, which is a platform-independent testing program developed by EEMBC, to evaluate the single node I/O performance of raw development board and x86 workstation before the Map-Reduce approach on the cluster. The testing results on single node comparison between ARM and x86 is shown in Table 2.

Table 2. Raw performance of single node

Performance		Intel	ARM	
		Core i7	A15 (big)	A7 (little)
CPU	CoreMark (iterations/MHz)	5.3	3.52	5.0
	Dhrystone (MIPS/MHz)	5.8	3.1	3.7
	Power Consumption (w/h)	50.6	15	11
Storage system	Write (MB/s)	165.0	24.5	21.7
	Read (MB/s)	173.0	31.6	26.5
Network	TCP bandwidth (Mbps)	944	421	411
	UDP bandwidth (Mbps)	810	405	399
	Ping delay (ms)	0.20	0.59	0.59

4.2 Performance of ARM Cluster

For the computational performance, we use HPL to test the computing ability of ARM cluster. Based on the single node performance, we record the results of $1 \sim 5$ nodes and calculate the Eff_{cpu} and Speed-Up Ratio and actual performance of FLOPs and power, shown in Table 3.

Table 3. HPL performance results of raw cluster.

Nodes	Eff_{cpu}	Speed-up ratio	GFLOPS	PPW (MFLOPS/w)
1	100 %	Base	2.177	151.2
2	94.3 %	1.709	3.720	123.6
3	88.3 %	2.559	5.572	145.5
4	85.1 %	3.287	7.155	143.3
5	82.9 %	4.146	9.025	138.8

In order to verify our Hypothesis under fair environment, we perform a weak scalability test. We keep same feasible size per node as weak-scaling way to evaluate the speed-up ratio and performance. The actual measured data and model prediction comparison as follow.

Performance of WordCount. WordCount is a classic application which features on frequency read and write on HDFS in Map-Reduce way, while decreasing the read/write rate with RDD on Spark indicate better performance. Thus, we try to verify our idea with the actual results of both Map-Reduce and Spark on the cluster, and the results shown in conjunction with Fig. 5.

From the results of experiments we found that the performance of our original hardware configurations as a pure ARM based cluster is disappointing through the analysis of dashboard of master node. Constrained by CPU and memory capacity, the compile and delivery of task occupied too much time during the whole process, both in the Map-Reduce and Spark way. Making full use of x86 based workstation, larger memory, L2 cache and board bandwidth guarantee the better performance of master node for the scheduling and monitoring works. So we propose the hardware configuration 2 scheme: let x86 workstation be the master alone, and the rest ARM boards as the computational nodes. As presented in Fig. 4, the performance is greatly improved by the way of having a strong master node of configuration 2 than 1.

As can be seen from Fig. 4, a stronger master node indicated better performance of the cluster. However, WordCount is not a typically compute-intensive application, comparative low computational work with the size of task shows poor scalability. Besides, a decline shows in the speed up of the Map-Reduce mode at 5 nodes. The cluster efficiency is decreasing dramatically with node extension, the low throughput of HDFS I/O determines that this kind of application features frequently read or write on HDFS, is not fit for the ARM based cluster. The performance shrink ascribed to the mechanism of HDFS, which illustrated in Sect. 4.3.

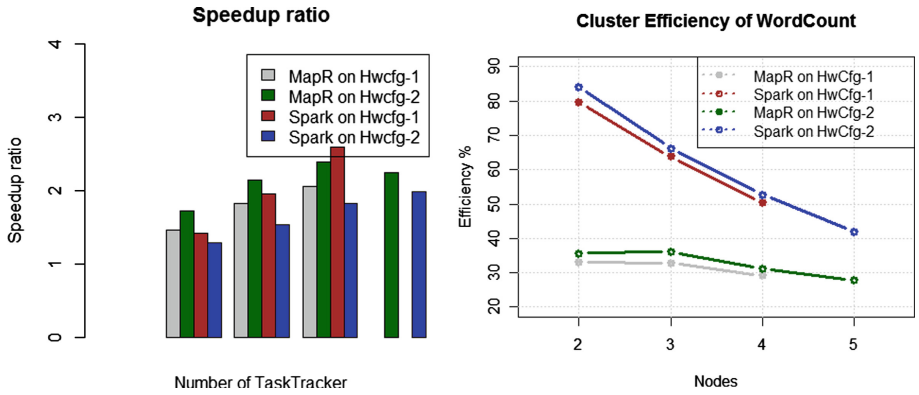

Fig. 4. Speed-up ratio and efficiency of cluster on WordCount.

Performance of k-Means. The nature of k-Means is the operation process of an iterative calculation. During this process, a lot of intermediate results and small data update will be put onto the HDFS in the Map-Reduce way, which indicates a great impact on Hadoop. Owing to the RDD of Spark approach, none write operation would be called until the result output, theoretically means great improvement in the performance of Spark. Thus, given 200 MB of workload, k = 8 and iteration for 6 times we calculate the results as shown in Fig. 5. 5, k-Means on Spark features CPU intensive with a relatively small amount of read/write tasks shows that I/O still is the main factor which restrict the performance of clusters.

Considering the number of nodes, running time, throughput and power, we can ascribe the inefficiency drawbacks on Map-Reduce model: mandated shuffle phrase and cumbersome reduce joins waste a large amount of computation on scheduling and synchronous, and communication overhead in HDFS. Map-Reduce model seems not fit for ARM based clusters, but RDD model on Spark with compute intensive application is feasible to ARM features.

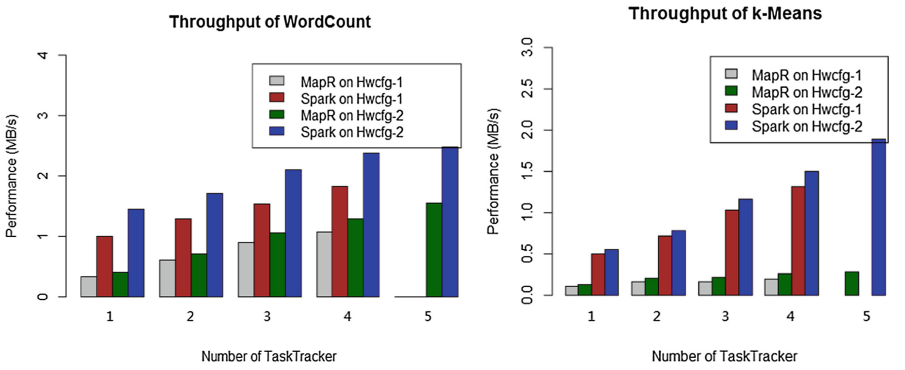

Fig. 5. Comparison of WordCount and K-Means by different configurations.

4.3 Case Study of Improvement on Different Setting

The results of CubieBoard4 is not satisfied enough due to the non-optimized development boards, as shown in Table 1, the inefficiencies due to the bottleneck of the I/O interface chips of USB. If the hardware performance of HDFS is improved, better performance would be achieved. So we try the hardware configuration 3 scheme: One x86 workstation as the master node, additional 5 × 86 workstation with sata3 HDD as HDFS, connect ARM cluster and x86 HDFS with gigabit networks. In this way, performance of HDD based HDFS is better than the 400 Mbps CubieBoard4 network capacity. Thus, the bottleneck is transferred from the USB to the network interface. So we use the TestDFSIO, which is the benchmark testing software to evaluate the throughput of HDFS on different hardware configuration, the results shown in Fig. 6.

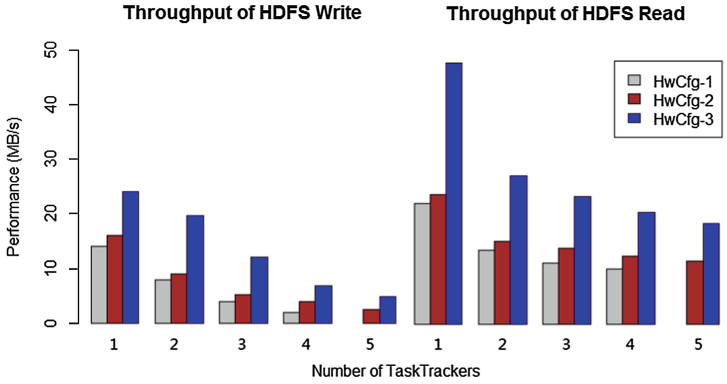

Fig. 6. Throughput of HDFS read and write operation.

As presented in Fig. 6, the throughput of HDFS decreased as nodes increased. Partly due to the hardware capability, as Configuration 3 performs better because higher I/O on HDD. Other factors of decrease lies in the scheduling and the 3-copies-redundance mechanism of HDFS itself. In hardware configuration 1, the performance of master node prolonged the total execution time. We replaced with a higher performance master node as configuration 2, the USB I/O interface inhibit the total performance. As we transfer the bottleneck to network capacity in configuration 3, the performance almost reach the hard limitation of the non-optimized development boards.

Then we recur our experiments of k-Means and WordCount on Spark under same workload by different hardware configurations to see the performance, plus a Map-Reduce results as basic reference value, shown in Fig. 7. Spark performs better than Map-Reduce model as the grey line compared with black line, and improved with higher hardware configurations. Compute-intensive application shows better improvement of performance. The cluster efficiency of each application is described in Table 4. Affected by HDFS, the efficiency of cluster decayed as the redundancy backup tripled the I/O loads.

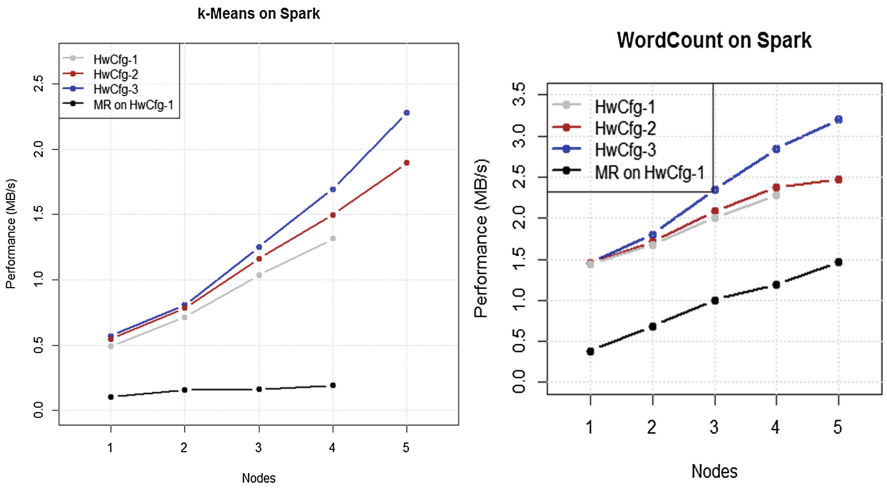

Fig. 7. Performance of K-Means and WordCount on 3 hardware configurations

Table 4. Cluster efficiency of spark

Application	k-Means			WordCount		
Hardware configuration	1	2	3	1	2	3
Nodes	Cluster efficiency					
2	85.88 %	88.38 %	90.8 %	79.7 %	81.4 %	85.8 %
3	71.56 %	75.98 %	82.78 %	63.79 %	66.23 %	74.5 %
4	58.41 %	64.7 %	74.97 %	50.43 %	52.68 %	62.99 %
5	N/A	51.5 %	68.89 %	N/A	41.96 %	54.47 %

The RDD mechanism reduced the HDFS read and write frequency, solved part of the I/O intensive operations inefficiency of ARM cluster. But compared with HPL result, the current performance of cluster is far from its limitation, which means abundant potential space for ARM cluster to improve. Besides, 2 GB memory constrains the performance of Spark, CPU utility is not high enough when the memory is full occupied, and insufficient to handle larger dataset. In order to get a fair compare, we use standard example programs without any optimized methods. Analysis under the current CubieBoard4 hardware configuration, to construct an ARM cluster only fit for compute intensive applications on Spark within a limited scale constrained by memory size.

ARMv8 with 64bit address bandwidth will solve the memory boundary, and customized hardware configuration that improve the I/O, network and board bandwidth to match the CPU performance are required in enterprise usage of ARM based cluster. Besides, there are lots of optimized methods at system and software level. Thus, one strong master node, which required by the Spark RDD mechanism, with customized ARM cluster, in addition with optimize methods is adequate to build an enterprise cluster under home-level temperature environment, which means a lower cost to build a

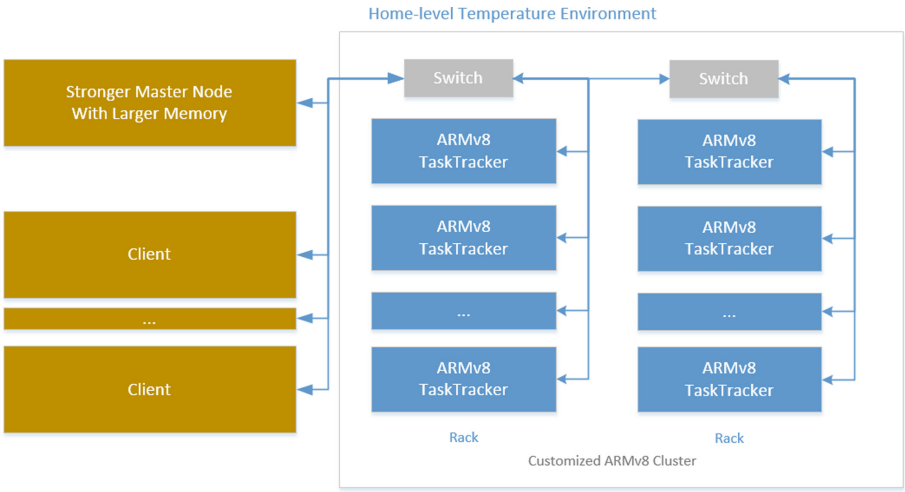

Fig. 8. Theoretical architecture of future ARMv8 cluster for enterprise application on spark

private Spark cluster. The theoretical architecture of the future ARM based clusters is shown in Fig. 8.

5 Conclusions

In this paper we present an ARM based Hadoop cluster with 5 CubieBoard4 development boards, which achieves 9.025 GFLOPS on HPL. We put forward our unique measurement modeling method to evaluate the cluster efficiency. Under 3 testing settings we systematically run the common applications to measure and evaluate the performance, efficiency of the cluster to verify the improvement. Through the experiments we assess the bottleneck of development boards and try to improve the performance, demonstrate that the RDD performance of spark on the ARM cluster is significantly better than the non-optimized Map-Reduce mode. The limitation of I/O interface chips of the pure development board constrained the performance since cluster bandwidth can not match the computational capacity. However, a customized setting is required for enterprise utility. Thus we present a heterogeneous architecture of a strong master node with customized I/O interfaces which may benefit the performance of the cluster.

One high performance JobTracker/Namenode with larger memory takes charge of scheduling, compiling and delivery the jobs is required by the Spark mechanism itself, TaskTrackers as the distributed compute nodes use a customized ARM cluster in Tibidabo way is competent to compute-intensive enterprise applications, such as regression, clustering, classification, collaborative Filtering recommendation and so on, in a lower cost way under home-level temperature environment. ARMv8's features may encouraging industrial attempt to accomplish the optimization of hardware and software, and ARM clusters are the alternate candidates in the near future.

References

1. GöDdeke, D., Komatitsch, D., Geveler, M., et al.: Energy efficiency vs. performance of the numerical solution of PDEs: an application study on a low-power ARM-based cluster. J. Comput. Phys. **237**, 132–150 (2013)
2. Rajovic, N., Rico, A., Puzovic, N., et al.: Tibidabo: making the case for an ARM-based HPC system. Future Gener. Comput. Syst. **36**, 322–334 (2014)
3. www.top500.rog
4. www.green500.org
5. Ebrahimi, K., Jones, G.F., Fleischer, A.S.: A review of data center cooling technology, operating conditions and the corresponding low-grade waste heat recovery opportunities. Renew. Sustain. Energy Rev. **31**, 622–638 (2014)
6. Turley, J.: Cortex-A15 "Eagle" flies the coop. Microprocess. Rep. **24**(11), 1–11 (2010)
7. ARM Ltd.: Cortex-A50 series. http://www.arm.com
8. Ghemawat, S., Gobioff, H., Leung, S.T.: The Google file system. ACM SIGOPS Oper. Syst. Rev. **37**(5), 29–43 (2003). ACM
9. Chang, F., Dean, J., Ghemawat, S., et al.: Bigtable: a distributed storage system for structured data. ACM Trans. Comput. Syst. (TOCS) **26**(2), 4 (2008)
10. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
11. Leverich, J., Kozyrakis, C.: On the energy (in) efficiency of hadoop clusters. ACM SIGOPS Oper. Syst. Rev. **44**(1), 61–65 (2010)
12. Shvachko, K., Kuang, H., Radia, S., et al.: The hadoop distributed file system. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), pp. 1-10. IEEE (2010)
13. Zaharia, M., Konwinski, A., Joseph, A.D., et al.: Improving MapReduce performance in heterogeneous environments. In: OSDI, 8(4), p. 7 (2008)
14. Zaharia, M., Chowdhury, M., Das, T., 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, p. 2. USENIX Association (2012)
15. Fox, K., Mongan, W., Popyack, J.: Raspberry HadooPI: a low-cost, hands-on laboratory in big data and analytics. In: Proceedings of the 46th ACM Technical Symposium on Computer Science Education, p.687. ACM (2015)
16. Kaewkasi, C., Srisuruk, W.: A study of big data processing constraints on a low-power Hadoop cluster. In: 2014 International Computer Science and Engineering Conference (ICSEC), pp. 267–272. IEEE (2014)
17. Aroca, R.V., Gonçalves, L.M.G.: Towards green data centers: a comparison of x86 and ARM architectures power efficiency. J. Parallel Distrib. Comput. **72**(12), 1770–1780 (2012)
18. Klausecker, C., Kranzlmüller, D., Fürlinger, K.: Towards energy efficient parallel computing on consumer electronic devices. In: Kranzlmüller, D., Toja, A.M. (eds.) ICT-GLOW 2011. LNCS, vol. 6868, pp. 1–9. Springer, Heidelberg (2011)
19. Fürlinger, K., Klausecker, C., Kranzlmüller, D.: The AppleTV-cluster: towards energy efficient parallel computing on consumer electronic devices. Whitepaper, Ludwig-Maximilians-Universitat (2011)
20. Rajovic, N., Vilanova, L., Villavieja, C., et al.: The low power architecture approach towards exascale computing. J. Comput. Sci. **4**(6), 439–443 (2013)
21. Dumitrel Loghin, B.M.T., Zhang, H., Ooi, B.C., et al.: A performance study of big data on small nodes. Proc. VLDB Endow. **8**(7), 762–773 (2015)

22. Gu, L., Zeng, D., Guo, S., Yong, X., Hu, J.: A general communication cost optimization framework for big data stream processing in geo-distributed data center. IEEE Trans. Comput. (ToC) (2015)
23. Lin, G., Zeng, D., Li, P., Guo, S.: Cost minimization for big data processing in geo-distributed data centers. IEEE Trans. Emerg. Topics Comput. **2**(3), 314–323 (2014)
24. Hu, C., Zhao, J., Yan, X., Zeng, D., Guo, S.: A MapReduce based parallel niche genetic algorithm for contaminant source identification in water distribution network. Ad Hoc Netw. **35**, 116–126 (2015)
25. Gu, L., Zeng, D., Guo, S., Barnawi, A., Stojmenovic, I.: Optimal task placement with QoS constraints in geo-distributed data centers using DVFS. IEEE Trans. Comput. (ToC) **64**(7), 2049–2059 (2014)
26. Plugaru, V., Varrette, S., Pinel, F., et al.: Evaluating the HPC performance and energy-efficiency of Intel and ARM-based systems with synthetic and bioinformatics workloads. In: CSC (2014)
27. McCool, M., Reinders, J., Robison, A.: Structured Parallel Programming: Patterns For Efficient Computation. Elsevier, Waltham (2012)
28. Chou, C.-Y., Chang, Hsi-Ya., Wang, S.-T., Tcheng, S.-C.: Modeling message-passing overhead on NCHC formosa PC cluster. In: Chung, Y.-C., Moreira, J.E. (eds.) GPC 2006. LNCS, vol. 3947, pp. 299–307. Springer, Heidelberg (2006)