A Low-Cost Energy-Efficient Raspberry Pi Cluster for Data Mining Algorithms

João C Saffran, Gabriel C Garcia, Matheus Souza, Pedro Henrique Penna, Marcio Castro, Luís F. Góes, Henrique C Freitas

A low-cost energy-efficient Raspberry Pi cluster for data mining algorithms

João Saffran[1], Gabriel Garcia[1], Matheus A. Souza[1], Pedro H. Penna[2], Márcio Castro[2], Luís F. W. Góes[1], and Henrique C. Freitas[1]

[1] Computer Architecture and Parallel Processor Team (CArT)
Pontifícia Universidade Católica de Minas Gerais (PUC Minas), Brazil

[2] Laboratório de Pesquisa em Sistemas Distribuídos (LAPeSD)
Universidade Federal de Santa Catarina (UFSC), Brazil

{joao.saffran, gabriel.garcia, matheus.alcantara}@sga.pucminas.br,
pedro.penna@posgrad.ufsc.br, marcio.castro@ufsc.br
{lfwgoes,cota}@pucminas.br

Abstract. Data mining algorithms are essential tools to extract information from the increasing number of large datasets, also called Big Data. However, these algorithms demand huge amounts of computing power to achieve reliable results. Although conventional High Performance Computing (HPC) platforms can deliver such performance, they are commonly expensive and power-hungry. This paper presents a study of an unconventional low-cost energy-efficient HPC cluster composed of Raspberry Pi nodes. The performance, power and energy efficiency obtained from this unconventional platform is compared with a well-known coprocessor used in HPC (Intel Xeon Phi) for two data mining algorithms: Apriori and K-Means. The experimental results showed that the Raspberry Pi cluster can consume up to 88.35% and 85.17% less power than Intel Xeon Phi when running Apriori and K-Means, respectively, and up to 45.51% less energy when running Apriori.

Keywords: Raspberry Pi cluster, Intel Xeon Phi, Apriori, K-Means

1 Introduction

Petaflop computing relied on the advent of massively parallel architectures, such as vector processing units and manycore processors. For instance, Graphics Processing Units (GPUs) and the Intel Xeon Phi have been used extensively in current High Performance Computing (HPC) platforms, since they have proven to significantly increase the overall processing power of these platforms. However, several challenges still have to be overcome to reach the exascale computing era [11], [15]. First, current cutting-edge parallel machines are power-hungry. This characteristic has motivated the community to seek for strategies to reduce energy consumption while delivering high performance [3], [16]. Second, HPC platforms are expensive to obtain, which may be unaffordable for small- and

medium-size institutes. In this context, the use of such architectures might be unworkable, but the demand for performance must not be disregarded.

One of the research domains that are gaining attention in the HPC community is Big Data, which is a term applied to data sets whose size or type is beyond the ability of relational databases to capture, manage, and process the data with low-latency. Big Data comes from sensors, devices, video/audio, networks, log files, transactional applications, web, and social media (much of it generated in real time and in a very large scale). To extract insights from unstructured data in a feasible time, Big Data applications usually exploit HPC platforms.

The already mentioned massively parallel architectures (*e.g.*, GPUs and the Intel Xeon Phi) have been used to process Big Data applications with high performance. However, their expensive financial cost and recent issues related to energy consumption leads the scientific community to consider the use of low-cost and low-power architectures to build scalable machines. This way, high performance may be achieved with lower financial cost [4], [13].

In this paper, we study the use of an unconventional low-cost energy-efficient HPC cluster composed of eight Raspberry Pi boards, interconnected by a network switch, to verify whether it can be used as an alternative for HPC. Although these boards are not optimized for high performance, they can be considered as a good candidate to scalable systems with very low energy consumption.

Our main goal is to evaluate the performance, power and energy consumption of the Raspberry Pi cluster for two well-known data mining algorithms commonly used in Big Data applications: Apriori and K-Means. The results obtained with the Raspberry Pi cluster are compared with a well-known coprocessor used extensively in HPC (Intel Xeon Phi). Our results show that the Raspberry Pi cluster can achieve better energy efficiency than Intel Xeon Phi, consuming up to 45.51% less energy than Intel Xeon Phi when running Apriori and K-Means kernels.

The remainder of this paper is organized as follows. Section 2 presents an overview of the related work. In Section 3 we briefly present the architectures used in this work. In Section 4 we describe the applications we have chosen and explain their implementations. Section 5 presents our experimental methodology and the evaluation of the observed results. Finally, in Section 6, we present our conclusions and suggestions for future research.

2 Motivation and Related Work

As aforementioned, the energy efficiency of parallel computers is an obstacle to be overcome on the way to the exascale era. Future HPC computers will be limited to about 20 MW of power consumption in the coming decade [3], [18]. Thus, energy efficiency is a relevant aspect to be addressed.

Recently, many high performance architectures have been designed driven by low-power consumption constraints. Kruger *et al.* [9] proposed a cluster of low-cost Parallella boards, each one featuring a 16-core Epiphany RISC System-on-Chip (SoC) and a dual-core ARM Cortex-A9. They concluded that the cluster

achieved better performance than an Intel i5-3570. However, Parallella lacks hardware for complex arithmetic operations, which can degrade the performance of few specific applications. Although they mention the importance of energy efficiency, the work did not present a rigorous power consumption analysis.

Similarly to our work, d'Amore *et al.* [5] proposed the use of Raspberry Pi boards. They built a cluster of six Raspberry Pi boards to evaluate the performance of Big Data applications. The authors concluded that the cluster is an affordable solution to their problem, but the work only focused on how the data can be retrieved and used. However, the authors did not conduct any power/energy or quantitative performance evaluation.

In order to analyze the power efficiency of different processors, Aroca *et al.* [2] compared Intel, AMD and ARM processors. They used five different processors in their analysis, running web servers, database servers and serial applications from the Linpack benchmark. They concluded that devices with low-power characteristics (*e.g.*, the Intel Atom and ARM ones) are good candidates to compose the infrastructure of data centers. On the other hand, aspects of HPC systems were not considered, such as parallel applications that are mainly CPU-bound.

Other initiatives are also worried about these issues related to energy consumption. ARM processors attempt to be a suitable alternative for low-power consumption in HPC. This is the proposal of the *Mont-Blanc and Mont-Blanc2* projects [14], which intend to design a new computer architecture to establish HPC standards based on energy-efficient platforms. Similarly, the *Glasgow Raspberry Pi Cloud (PiCloud)* [17] is a cluster of Raspberry Pi devices designed to be a scale model of a cloud computing platform, in order to address cloud simulations and applications.

Both projects proved to be energy efficient in their proposals, which reinforces the applicability of our work, that evaluates the behavior of Big Data application kernels in parallel architectures, as well as in [5]. We compare the use of a cluster of low-cost and low-power platforms with a coprocessor designed specifically for HPC (Intel Xeon Phi). Furthermore, we measure the power and energy consumption of both architectures when running two data mining algorithms used in the Big Data research domain.

3 Experimental Platforms

In this section we detail the platforms that we considered in this work, highlighting their main features.

3.1 Intel Xeon Phi

The Intel Xeon Phi is a 64-bit coprocessor that primarily targets HPC workloads. This coprocessor features Intel's Many Integrated Core (MIC) architecture and provides full compatibility with the x86-64 Instruction Set Architecture (ISA)[3].

[3] Intel markets this coprocessor with several different specifications. In this work we refer as Intel Xeon Phi to the one codenamed Knight's Corner.

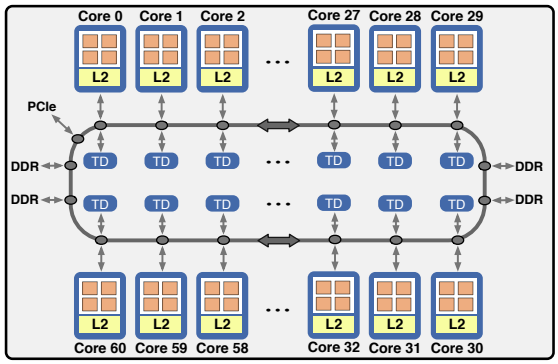

Fig. 1. Overview of the Intel Xeon Phi coprocessor.

An overview of the Intel Xeon Phi is presented in Figure 1. It features 61 dual-issue, out-of-order processing cores with 4-way simultaneous multithreading, enabling up to 244 working threads. Cores are interconnected by a bidirectional ring topology. Each core has 32 kB instruction and 32 kB data L1 caches and 256 kB of L2 cache, which is the last level of cache in the architecture. All caches are private and coherent. With regard to power consumption, this coprocessor has a Thermal Design Power (TDP) of 300 W. The Intel Xeon Phi coprocessor is usually connected to a host machine through a Peripheral Component Interconnect Express (PCIe) bus. A single Intel Xeon Phi board may have up to 16 GB of memory.

3.2 Raspberry Pi Cluster

The Raspberry Pi is a low-cost general purpose SoC. It has a quad-core 64-bit processor based on the ARMv7 architecture. In our work, we used a Raspberry Pi 2, which features a quad-core Cortex-A7 CPU running at 900 MHz. Each of the four cores has 64 kB instruction and 64 kB data L1 caches and 512 kB of L2 cache shared among all cores. The L2 is the last level cache in Raspberry Pi 2. The total amount of memory is 1 GB.

This type of SoC is not usually used in HPC due to its low performance. However, some characteristics make it much more energy efficient, *e.g.*, the absence of peripheral controllers and hardware support for complex arithmetic operations. Hence, given the possibility of scaling this type of architectures, the overall energy efficiency of a larger system could be improved while achieving decent performance. The Raspberry has all components that conventional computers have, thus, it can be connected to a local area network using an Ethernet interface. Many Raspberry units can be clustered, conceiving a low-power and low-cost system with high potential to achieve high performance [4], [13].

A schematic diagram of our Raspberry Pi cluster is depicted in Figure 2. In our work, we opted to group eight Raspberry Pi 2 devices in order to conceive

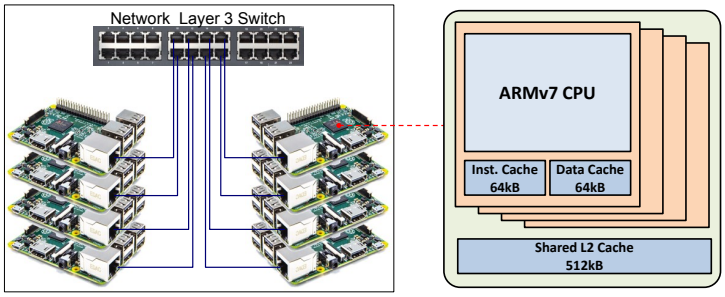

Fig. 2. The Raspberry Pi cluster (left) and a overview of the ARM Cortex-A7 (right).

the cluster. The devices were interconnected by the means of a layer 3 switch, constituting a local network.

4 Data Mining Algorithms

To conduct our work, we chose two data mining algorithms used in the Big Data domain. These algorithms play an important role in different fields, including pattern recognition, image analysis and bioinformatics [19]. In this section, we detail these algorithms and their parallel versions.

4.1 Association Rule Learning

Association rule learning is a very common method used to discover relations between variables in large databases. Apriori is a state-of-the-art association rule machine-learning technique used for frequent itemset mining [1]. Given a list of itemsets, it identifies association rules between those items based on their frequency. These rules reveal subsets of items that frequently occur together in the same itemsets. The algorithm is driven by the following rule: all non-empty frequent itemsets must be also frequent. This rule allows the algorithm to eliminate all itemsets that are not composed of frequent item subsets, reducing significantly the search space. The Apriori algorithm works as follows. For each association rule $A \rightarrow B$, where A and B are subsets of items of a frequent itemset, the Apriori algorithm calculates its *confidence*, presented in Equation 1. High confidence levels mean that most of the time an itemset A is present in a frequent itemset, the itemset B is also there.

$$\texttt{Conf}(A, B) = \frac{support(A \wedge B)}{support(A)} \tag{1}$$

In this paper, the multi-threaded version of the algorithm follows a Map-Reduce parallel pattern. After identifying the itemsets of size $K = 1$, items from S are distributed among the threads (Map stage). Each thread then counts the occurrences of its subset of items $e_i \in S$. With all frequencies calculated, the

subsets are regrouped (Reduce stage). Thus, the itemsets that do not meet the minimum support (i.e. a threshold used to eliminate infrequent itemsets) are removed, and then the confidence is calculated to form the association rules. This steps are repeated incrementing K until the subsets of size K are empty.

In the distributed version, the itemsets are assigned first to the nodes instead of threads. Each node then runs a multi-threaded version of the algorithm, exactly as described before.

4.2 K-Means Clustering

The K-Means clustering is a clustering approach widely used and studied [12]. Formally, the K-Means clustering problem can be defined as follows. Given a set of n points in a real d-dimensional space, the problem is to partition these n points into k partitions, so as to minimize the mean squared distance from each point to the center of the partition it belongs to. Several heuristics have been proposed to address the K-Means clustering problem [6], [8]. We opted to use the Lloyd's algorithm [7], which is based on an iterative strategy that finds a locally minimum solution for the problem. The minimum Euclidean distance between partitions and centroids is used to cluster the data points. The algorithm takes as input the set of *data points*, the number of partitions k, and the minimum accepted distance between each point and the centroids.

The K-Means algorithm works as follows. First, data points are evenly and randomly distributed among the k partitions, and the initial centroids are computed. Then, data points are re-clustered into partitions taking into account the minimum Euclidean distance between them and the centroids. The centroid of each partition is recalculated taking the mean of all points in the partition. The whole procedure is repeated until no centroid is changed and every point is farther than the minimum accepted distance.

The multi-threaded version of this algorithm takes an additional parameter t, that is the total number of threads. An unique range of points and partitions is assigned to each thread, with two processing phases, A and B. In phase A, each thread re-clusters its own range of points into the k partitions. In phase B, each thread works in its own range of partitions, in order to recalculate centroids.

The distributed K-Means algorithm takes the number of available nodes that, by themselves, spawn working threads. The strategy employed in this algorithm is to first distribute the data points and replicate the data centroids among the available nodes, and then to loop over a two-phase iteration. In the first phase, partitions are populated, as in the multi-threaded algorithm, and in the second phase, data centroids are recalculated. For this recalculation, first each node uses its local data points to compute partial centroids, *i.e.*, a partial sum of data points and population within a partition. Next, nodes exchange partial centroids so that each peer ends up with the partial centroids of the same partitions. Finally, nodes compute their local centroids and broadcast them.

5 Experimental Results

In this section, we first describe the experimental setup and discuss important implementation details for each platform. Then, we present the results obtained with the data mining algorithms on both platforms.

5.1 Setup and Implementation Details

We used the multi-threaded implementations of both algorithms to carry out the experiments on the Intel Xeon Phi, since the memory is shared among all cores in this processor. More precisely, we parallelized the algorithms using the OpenMP programming model and we compiled them with Intel C/C++ Compiler (`icc`) version 16.0.1.

The Intel's MIC System Management and Configuration (MICSMC) tool was used to monitor the processor's power consumption. Power measurements obtained from MICSMC are very accurate as shown in [10]. Intel Xeon Phi can be either used in *offload* or *native* modes. In the former mode, the main application code is executed on the host and performance-critical sections of the application are offloaded to the coprocessor. In the latter mode, the entire application is executed on the coprocessor. In this paper, we chose the latter since we intend to compare the performance and energy consumption of the coprocessor only.

For the Raspberry Pi cluster, on the other hand, we adopted a hybrid programming model: we used the OpenMPI library to implement the distributed version of the applications (*i.e.*, to make use of all nodes available in the Raspberry Pi cluster) and OpenMP inside each node to exploit four cores available in it. We compiled both applications with `mpicc` version 1.4.5 with GNU C Compiler (GCC) version 4.6.3. To measure the energy consumption of this cluster, we used a watt-meter instrument connected before the power supply of the devices, except the network switch, taking the total consumption in kilowatts-hour (kWh) of the whole system. The Equation 2 was used to convert our results to Joules (J).

$$E_{(J)} = 1000 \times 3600 \times E_{(kWh)} \tag{2}$$

The power consumption was calculated for the Raspberry Pi cluster, as well as the energy consumption, for the Intel Xeon Phi. To calculate these values, we used Equation 3.

$$E_{(J)} = P_{(W)} \times t_{(s)} \tag{3}$$

We ran each application varying the number of available nodes or threads, depending on the target platform. For the Raspberry Pi cluster, we used four threads per cluster and varied the number of nodes in 4 (12.5% of the total number of cores), 8 (25% of the total number of cores), 16 (50% of the total number of cores) and 32 (all available cores). Similarly, we varied the number of threads proportionally to the number of cores available in the Intel Xeon Phi,

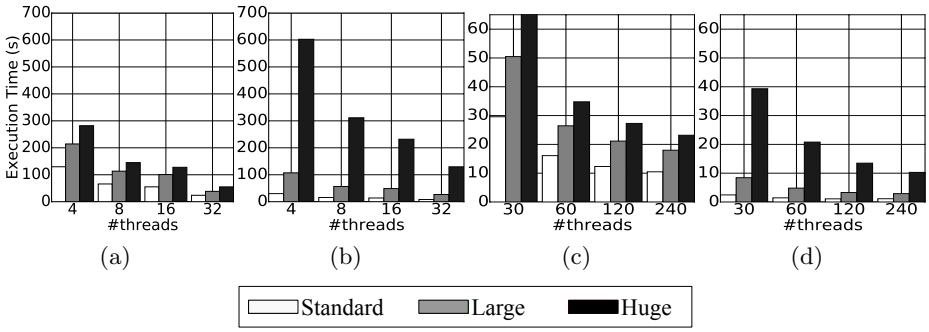

Fig. 3. Execution time: (a) Raspberry Pi cluster - Apriori, (b) Raspberry Pi cluster - K-Means, (c) Intel Xeon Phi - Apriori (d), Intel Xeon Phi - K-Means.

i.e., 30, 60, 120 and 240 threads. This allows us to compare both platforms when the same percentages of the overall resources available in each platform are used.

Three workload sizes were defined for each algorithm, to evaluate the behavior of both systems when the workload increases. For the Apriori algorithm, the increase on the workload size can be achieved by reducing the minimum support. We used the following minimum support values: 70 for the standard workload, 60 for large and 50 for huge. For K-Means, we increased the number of data points to be clustered. We used the following number of data points: 2^{14} for the standard workload, 2^{15} for large and 2^{16} for the huge one.

Finally, we performed 10 runs for each configuration (number of threads/nodes and workload sizes) and computed the average execution time and power. The maximum standard deviation observed in Raspberry Pi cluster executions was 11.04%, while the Intel Xeon Phi presented at most 7.07%.

5.2 Evaluation

We used three metrics to compare the platforms: execution time, power consumption, and energy consumption. Figure 3 presents the execution time of the algorithms when executed on both architectures. As it can be observed, Apriori proved to be more scalable than K-Means. It was possible to reduce the execution time by 82.05% with the Apriori when changing from 1 node to 8 nodes in the Raspberry Pi cluster, while in K-Means with same configurations the reduction was about 74.97%. This is due to the fact that the Apriori algorithm has more independent work units than the K-Means, which leads to less synchronization when parallel work finishes.

Comparing the Raspberry Pi cluster and Intel Xeon Phi execution time, the Intel Xeon Phi presented better results. It is an architecture with more processing power, featuring a larger thread support (up to 240), thus it was an expected behavior. However, the Intel Xeon Phi presented poorer scalability than the Raspberry Pi cluster for both algorithms. For instance, the maximum

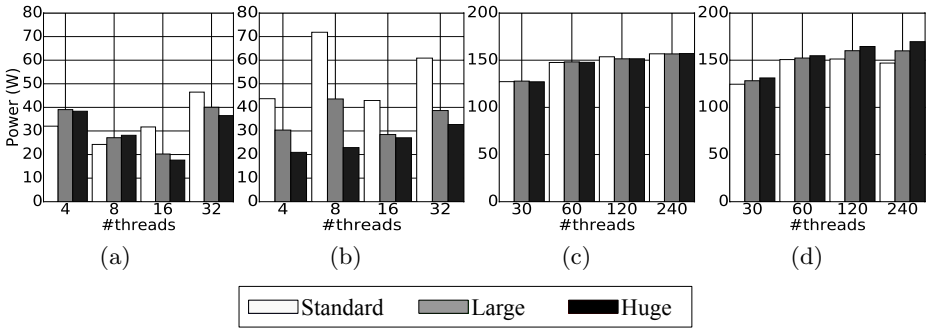

Fig. 4. Power consumption: (a) Raspberry Pi cluster - Apriori, (b) Raspberry Pi cluster - K-Means, (c) Intel Xeon Phi - Apriori (d), Intel Xeon Phi - K-Means.

execution time reduction, starting from 30 threads, was 73.85% when running K-Means with the full architecture (240 threads). The communication in the Raspberry Pi cluster, although done by the means of a local area network, is not surpassed by the synchronization time spent by the 240 threads on the Intel Xeon Phi, in the case of Apriori. Considering K-Means, it presents a similar behavior when increasing the resources of the architectures. On the other hand, with this application, the increase in the workload has run better in the Intel Xeon Phi. It is worth noting that the multi-threaded version of the codes are the same for both architectures, without specific optimizations. Thus, despite the Intel Xeon Phi code could be improved, the Raspberry Pi cluster presents better efficiency with a lower number of threads (32 at full use).

Figure 4 presents the observed power consumption in our experiment. The applications are naturally unbalanced, thus, the power consumption varies depending on the execution time spent by each work unit, which are irregular (the cores and nodes present different times to solve their own work unit). With the Apriori algorithm, changing the workload size results in less variation in power consumption than when running the K-Means algorithm. Usually, with Apriori, the power consumption reduces when increasing the workload size. With respect to K-Means running in the Raspberry Pi cluster, the power consumption reduces when the workload size is increased, but with the Intel Xeon Phi the opposite occurs. In the Raspberry Pi cluster, each node has its own slave process and, when it finishes its computation, the power consumption of the entire node is drastically reduced while other nodes are still active.

We noticed a much more significant variation in power consumption on the Raspberry Pi cluster than on the Intel Xeon Phi. This is due to the fact that when less cores are used on the Intel Xeon Phi, the coprocessor idle cores keep consuming a portion of the energy, since they are in the same device. On the Raspberry Pi cluster, there is less impact from this fact, since the devices are completely independent. Overall, the Raspberry Pi cluster is superior when compared to Intel Xeon Phi. It was possible to obtain up to 88.35% of reduction in

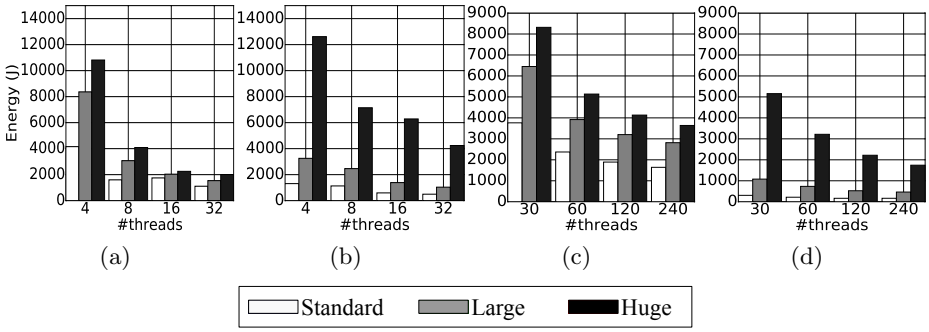

Fig. 5. Energy consumption: (a) Raspberry Pi cluster - Apriori, (b) Raspberry Pi cluster - K-Means, (c) Intel Xeon Phi - Apriori (d), Intel Xeon Phi - K-Means

power consumption with Apriori when using the Raspberry Pi cluster over the Intel Xeon Phi. In the same way, for K-Means, the biggest reduction was 85.17%.

Figure 5 presents the energy consumption results, which were obtained by multiplying the average power by the execution time. As it can be observed, the energy consumption increases as we increase the workload size. This is due the increase in the execution time that the algorithms spent to reach a solution.

Another observation concerns the energy consumed when varying the number of threads/nodes. We observed a significant reduction in energy consumption when more threads are used. This can be explained by the fact that if more resources are used, the power consumption increases, but the time to solution tends to decrease, due to the increase in the computational power. Thus, since the power consumption is less determinant than the execution time in our experiment, the energy consumption decreases when more resources are used.

In summary, the Raspberry Pi cluster proved to be more energy efficient than the Intel Xeon Phi for Apriori, although the opposite occurs with K-Means. This is due to the higher execution time difference for K-Means, since as more time is spent running an application more energy is consumed during this time. The Apriori algorithm was less energy-efficient in the Raspberry Pi cluster when using a single node in comparison with 30 threads from the Intel Xeon Phi, however, when more resources are employed proportionally, the Raspberry Pi cluster starts to be more energy-efficient, consuming up to 45.51% less energy than the Intel Xeon Phi. With respect to the financial costs, as mentioned before, the Raspberry Pi cluster is about ten times cheaper than the Intel Xeon Phi, thus presenting better price-performance ratio (*i.e.*, cost-benefit).

6 Concluding Remarks

In this paper we evaluated the performance, power and energy consumption of an unconventional low-cost energy-efficient HPC cluster composed of Raspberry Pi nodes when running two well-known data mining algorithms used in Big Data

(Apriori and K-Means). The results obtained on this cluster were compared to a coprocessor widely adopted in the HPC domain (Intel Xeon Phi). Our results showed that the Raspberry Pi cluster achieved a better tradeoff between execution time and power consumption for the Apriori kernel. On the other hand, the Intel Xeon Phi presented better performance on K-Means.

As future work, we propose to apply load balancing strategies on both applications to improve their performances. Moreover, we intend to implement parallel versions of these applications for Graphical Processor Units (GPUs), and use more HPC devices, for instance, a cluster of Xeon Phi boards. This would allow us to compare this architecture with the ones used in our work. Finally, we also intend to study the impacts on the energy efficiency and performance of the Raspberry Pi cluster when running application kernels from other domains, such as image processing and computational fluid dynamics.

7 Acknowledgement

This work was partially supported by FAPEMIG, FAPESC, CAPES, CNPq and STIC-AmSud and was developed in the context of EnergySFE and ExaSE cooperation projects, also supported by FAPERGS and INRIA.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th International Conference on Very Large Data Bases. pp. 487–499. VLDB '94, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1994)
2. Aroca, R.V., Gonçalves, L.M.G.: Towards green data centers: A comparison of x86 and arm architectures power efficiency. Journal of Parallel and Distributed Computing 72(12), 1770–1780 (2012)
3. Ashby, S., Beckman, P., Chen, J., Colella, P., et al.: The opportunities and challenges of exascale computing. Tech. rep., Summary report of the advanced scientific computing advisory committee (ASCAC) subcommittee - Office of Science, U.S. Department of Energy Fall (2010)
4. Cox, S.J., Cox, J.T., Boardman, R.P., et al.: Iridis-pi: a low-cost, compact demonstration cluster. Cluster Computing 17(2), 349–358 (2013)
5. d'Amore, M., Baggio, R., Valdani, E.: A practical approach to big data in tourism: A low cost raspberry pi cluster. In: Information and Communication Technologies in Tourism 2015, pp. 169–181. Springer (2015)
6. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1988)
7. Kanungo, T., Mount, D., Netanyahu, N., et al.: An efficient k-means clustering algorithm: analysis and implementation. Pattern Analysis and Machine Intelligence, IEEE Transactions on 24(7), 881–892 (Jul 2002)
8. Kaufman, L., Rousseeuw, P.J.: Finding groups in data: an introduction to cluster analysis. John Wiley and Sons, New York (1990)
9. Kruger, M.J.: Building a Parallella board cluster. Bachelor of science honours thesis, Rhodes University, Grahamstown, South Africa (2015)

10. Lawson, G., Sosonkina, M., Shen, Y.: Energy evaluation for applications with different thread affinities on the Intel Xeon Phi. In: Workshop on Applications for Multi-Core Architectures (WAMCA). pp. 54–59. IEEE Computer Society (2014)
11. Lim, D.J., Anderson, T.R., Shott, T.: Technological forecasting of supercomputer development: The march to exascale computing. Omega 51, 128 – 135 (2015)
12. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics. pp. 281–297. University of California Press, Berkeley, Calif. (1967)
13. Pfalzgraf, A.M., Driscoll, J.A.: A low-cost computer cluster for high-performance computing education. In: IEEE International Conference on Electro/Information Technology. pp. 362–366 (June 2014)
14. Rajovic, N., Carpenter, P.M., Gelado, I., Puzovic, N., Ramirez, A., Valero, M.: Supercomputing with commodity cpus: Are mobile socs ready for hpc? In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. pp. 40:1–40:12. SC '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2503210.2503281
15. Simon, H.: Barriers to exascale computing. In: High Performance Computing for Computational Science (VECPAR), pp. 1–3. Springer, Kope, Japan (2012)
16. Trefethen, A.E., Thiyagalingam, J.: Energy-aware software: Challenges, opportunities and strategies. Journal of Computational Science 4(6), 444 – 449 (2013), scalable Algorithms for Large-Scale Systems Workshop (ScalA2011), Supercomputing 2011
17. Tso, F.P., White, D.R., Jouet, S., Singer, J., Pezaros, D.P.: The glasgow raspberry pi cloud: A scale model for cloud computing infrastructures. In: 33rd International Conference on Distributed Computing Systems Workshops. pp. 108–112. IEEE (July 2013)
18. Villa, O., Johnson, D.R., O'Connor, M., et al.: Scaling the power wall: a path to exascale. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 830–841. IEEE Press (2014)
19. Xu, R., Wunsch, D., I.: Survey of clustering algorithms. Neural Networks, IEEE Transactions on 16(3), 645–678 (May 2005)