On the Use of Microservers in Supporting Hadoop Applications

Ali Anwar, Krish K.R., Ali R. Butt
Virginia Tech
Email: {ali, kris, butta}@cs.vt.edu

***Abstract*—**

The use of economical, low-power microservers comprising of embedded CPUs is on the rise in supporting a myriad of applications. State of the art microservers can already match the performance of low-end traditional servers, and have been advocated as an energy-efficient alternative computing substrate for data centers as well. In this paper, we explore whether cluster comprising microservers can support the popular Hadoop platform. We conduct a quantitative study of six representative Hadoop applications on five hardware configurations. To compare the different clusters, we also define a comprehensive metric, *PerfEC*, which unifies the performance, energy consumption, and the acquisition and operating costs of the applications, and helps identify appropriate clusters for Hadoop applications.

Experiments on our test clusters suggest that for applications such as *TeraSort*, *RandomWriter* and *Grep* microservers offer up to two orders of magnitude better efficiency in terms of *PerfEC* than traditional clusters. Similarly, a 3000-node cluster simulation driven by a real-world trace from Facebook shows that on average the studied microservers can match the performance of standard servers, while providing up to 31% energy savings at only 60% of the acquisition cost. We also compare *PerfEC* to the extant Total Cost of Ownership (TCO) metric, and find that our approach is better able to capture the trade-offs involved.

I. Introduction

MapReduce/Hadoop [5] has emerged as an efficient distributed computing framework that supports many large-scale applications [8], [14], [33]. While the framework was originally designed to run on uniform resources, Hadoop is being quickly adapted to non-uniform settings [15], [19], [27], [41] comprising a range of hardware from massive-core machines to low-power ARM-based devices [12], [44].

A major obstacle to sustaining Hadoop at scale is that the energy footprint of the large clusters and data centers that support Hadoop is increasing sharply [28] and imposes significant financial burden [1]. To mitigate this, microserver-based clusters have been proposed as an alternative [3], [4], [11]. Microservers are networked embedded devices designed specifically for low powered environments. They employ the Server-on-Chip (SoC) approach to have a CPU, networking, and I/O fully integrated onto a single server chip [3]. Microservers can be synthesized around numerous embedded architectures such as PowerPc [35], ColdFire [42], Mips [20] and ARM [24]. These devices are cheaper compared to traditional servers, energy efficient, and are becoming readily available. Therefore, microservers can potentially be used to support Hadoop. However, the above benefits come at the cost of reduced performance, raising the question whether microservers can realistically support Hadoop applications.

A promising development in this direction is that the ARM architecture, driven by its widespread use in smart phones, tablets, and miniPC devices [39], is beginning to offer increasingly high performance that is even competing with general purpose CPUs [37]. Thus, ARM has become a key enabler for microservers [21] and ultra-low power clusters [3]. Industry players such as AMD are forecasting that ARM is expected to cover 25% of the server market by 2019 [2]. The caveat here is that such predictions are for ARM-based traditional well-endowed servers, and the use of microservers in the data center, e.g., to support Hadoop, has not been explored.

In this paper, we present a quantitative study to explore the efficacy of microservers in supporting Hadoop applications, which aims to consider all factors of interest, i.e., performance, energy consumption, reliability, operating costs, ease of use, etc. Although several works [12], [44] have demonstrated the potential of microservers for specific applications, there is little understanding of how a general platform like Hadoop will behave on a substrate of microservers. To this end, we use six representative Hadoop applications, and study their execution characteristics on five different hardware configurations ranging from workstations to microservers. The goal is to quantify the performance impact, and identify design trade-offs that can help realize Hadoop on microservers.

Furthermore, we define a comprehensive metric, *PerfEC*, which captures the impact of the three often competing constraints of performance, energy consumption, and the acquisition and operating costs on overall data center design. *PerfEC* allows for better understanding of the relationships between Hadoop applications and the underlying data center hardware, in terms of suitability to efficiently support the applications. While we focus on Hadoop in this work, *PerfEC* offers a generic metric that is equally suitable for exploring microserver-based clusters for other applications as well. We note that *PerfEC* is better than metrics such as Total Cost of Ownership (TCO) that considers only the acquisition and operating costs of the hardware, and Joulesort [38] that considers only the energy characteristics of the applications. Thus, we can use the metric to analyze how our studied hardware configurations fare in comparison for particular applications. Moreover, the metric can also be used to choose appropriate hardware substrates for applications.

Specifically, this paper makes the following contributions:

- We customize the Hadoop software stack to run on five different hardware configurations.
- We port, tune and measure the performance, energy,

and cost for six representative Hadoop applications on the studied hardware targets.

- We define a new metric, *PerfEC*, to understand the interactions between energy, cost, and performance characteristics of the studied applications and hardware targets.
- We employ *PerfEC* to identify the best hardware configuration for the studied applications.

We compared *PerfEC* with the extant TCO metric and find that our approach is better able to capture the trade-offs involved when studying microservers for supporting Hadoop applications. Our evaluation reveals that microservers can match or exceed the performance delivered by a traditional workstation setup for the studied applications at a fraction of energy consumption and cost. We also performed a simulation study driven by a 45-day trace of real world workload from a 3000-node Facebook cluster [10], and show that on average the studied microservers can match the performance of state-of-the-art standard servers, while providing up to 31% energy savings at only 60% of the acquisition cost.

II. Enabling Technologies

In this section, we present the background and enabling technologies for our study of the suitability of microservers for supporting Hadoop.

A. Hadoop

Hadoop offers an open-source implementation of the MapReduce framework that provides machine-independent programming at large scale. All data in MapReduce is represented as key-value pairs [48]. Programmers specify user defined map and reduce functions to specify the operations to be performed on the key-value pairs. A pre-specified number of *Mappers* execute the map function on the distributed resources to processes the input data and generate intermediate data that is then consumed by *Reducers* that implement the reduce function to aggregate and consolidate the data. The process of sorting and transferring the output of the mappers to the reducers is known as the shuffle phase, and is automatically handled by the Hadoop runtime. A Hadoop cluster node consists of both compute processors and directly-attached storage. A small number of nodes (typically $12-24$ [7]) are grouped together and connected with a network switch to form a rack. One or more racks form the Hadoop cluster.

Hadoop provides a *JobTracker* component that accepts jobs from the users and also manages the compute nodes that each run a *TaskTracker*. The data management is provided by the Hadoop Distributed File System (HDFS). The main functions of HDFS are to ensure that tasks are provided with the needed data, and to protect against data loss due to failures. HDFS uses a *NameNode* component to manages worker components called *DataNodes* running on each Hadoop node. HDFS divides all stored files into fixed-size blocks (chunks) and distributes them across DataNodes in the cluster. Moreover, the system typically maintains three replicas of each data block, two placed within the same rack and one on a different rack.

TABLE I. Cost and power specifications of studied hardware configurations.

Machine	Initial Cost (USD)	Power (Watts)
Raspberry Pi (+SD card)	40(+10)	2-4
MK908	65	2-4
WandBoard Quad (+HDD)	125 (+60)	3-6(+8)
2-socket Intel Xeon E5450	1200	120-180
2-socket Intel Xeon E5620	3500	180-300

B. ARM-Based Microservers

Microservers can be synthesized around numerous embedded architectures such as PowerPc [35], ColdFire [42], Mips [20] and ARM [24]. However, the ARM architecture is emerging as a promising enabler for microservers [21], mainly due to its low-power characteristics, cost, widespread use in mobile phones, tablets, and miniPC devices [39], and an increase in the offered performance.

ARM-based single to quad-core processors are already available, which can compete with general purpose CPUs [37]. Such cores are being explored also in the context of high performance computing [32] that has traditionally been the domain of massive multi-core systems. Moreover, the ARM processors continue to target low-power applications, and thus are designed with emphasis on not only reducing the idle power consumption but also the peak power consumption. Yet another important feature is the high reliability and MTTF, as ARM-based devices are often designed to support critical operations such as real-time control systems [39]. Consequently, as the performance of these microservers rises, they are increasingly becoming suitable substrates for supporting even the more rigorous of the data center demands.

Although most current ARM processors are 32-bit, AMD has already announced its first ARM Cortex A57 based system on chip (SoC), a 64-bit/8-core Opteron A1100 [2]. AMD is forecasting that ARM is expected to cover 25% of the server market by 2019 [2]. These developments are giving rise to solutions such as the recent self-contained, highly-extensible, ultra-low power cluster [3] that is based on microservers. Yet, the domain is only being initially explored and microservers have not been adopted by any key Hadoop data center operators at scale.

Table I lists the initial cost and energy consumption specifications of typical workstations to that of various ARM based microservers that we consider in this paper. We see that the microservers are order(s) of magnitude cheaper and consume much less power. The power-performance characteristics also vary, e.g., MK908 and Raspberry Pi both consume the same amount of power, i.e., 2-4 Watts, but Mk908 boasts of quad cores and can yield much higher performance than the single core of the Raspberry Pi. The devices vary in terms of disk and network I/O capabilities as well. Consequently, the studied microservers provide a rich exploration space for studying the impact of hardware configurations on Hadoop application performance.

C. Power Measurement

The power consumption of a data center can be measured at three reference states: idle, under peak-usage, or a combination of the two [38]. Idle power shows the minimum energy overhead of operating a data center, while measuring peak-power is crucial for determining the vertical and horizontal scaling potential of a data center. Numerous tools such as power meters [6] and hardware performance counters [17] can be used to measure the idle and peak power of a machine. We use the Whatsup Pro [6] power meters sampled every two seconds to create a power consumption profile of an application under test.

III. *PerfEC*: Measuring Performance, Cost and Energy

The emerging microserver architectures exhibit unique characteristics in terms of energy efficiency, performance, reliability and cost. This makes it challenging to match hardware to application needs and Service Level Agreements (SLA). To address this problem, we define a metric, *PerfEC*, to quantify how microservers would perform under Hadoop. *PerfEC* captures the energy consumption of a microserver, the cost of the microserver, and resulting performance achieved for the workload. In this work, we focus on applying *PerfEC* to Hadoop clusters, however, the metric can be easily adapted for use in general clusters as well and is not specific to Hadoop applications only.

PerfEC is unique from other metrics — such as TCO that captures the data center characteristics and joulesort [38] that captures application characteristics — as it strikes a balance between different factors by considering the cost associated with executing an application in a data center as well as the energy consumed during the execution of the application. We define the metric as follows:

$$PerfEC = K \times \frac{P}{D \times E \times IC \times NC}$$

where K = normalization constant; $P = \frac{1}{t}$, where t is the execution time of the application; E = total energy consumed by the application; D = Input data size; IC = initial cost of the hardware; and $NC = n \times c_n$, where n is the number of nodes and c_n is the cluster networking equipment cost amortized for each node. c_n further comprises the initial cost of the networking equipment such as routers and related hardware, as well as the operational cost of the networking equipment. The operational cost of the cluster networking equipment is not dependent on the type of device used, e.g., microservers or standard servers, and remains constant for the different clusters considered our study. A high value of *PerfEC* implies high overall efficiency. We take into account the time to completion for the job in P as it also captures the indirect costs [38] of running the job in a data center, e.g., physical space costs, cooling costs, etc. The reason for including the initial cost is that it is directly related to TCO, and allows us to subsume TCO into *PerfEC*.

The data size is included for the following reason. The number of mappers and reducers associated with a Hadoop job can be interpreted as expressing the hardware resources needed by the job. Moreover, the number of mappers and reducers required by a job depends on the input size. Thus, using input size in *PerfEC* helps to capture the specific Hadoop application demands. Similarly, the intermediate data size is also important. However, the amount of intermediate data created is not known beforehand and difficult to model. Fortunately, the impact of intermediate data can be captured indirectly; lower or larger amount of intermediate data will result in a shorter or longer job completion time, respectively, which is captured by the parameter P.

A limitation of *PerfEC* is that it depends on application characteristics, so it is not possible to predict the efficiency of a hardware configuration for a new application based on an already studied application. Furthermore, since we use the data size as an indication of an application's resource needs, we have to be careful not to interpret this factor as suggesting that using a small dataset size is better (given the inverse relationship to *PerfEC*).

Finally, the reliability of the system, especially at scales where failures are a norm and not an exception, is also crucial. Since microservers are typically designed for low-power embedded applications, they tend to have much higher MTTFs than standard servers. However as mentioned earlier, microservers have not been studied in the context of large clusters and their true MTTF in such scenarios entail further study. Thus, we do not incorporate this factor into *PerfEC* yet. Nevertheless, MTTF can be incorporated by adding it as a multiplicand in the above formula (with K adjusted accordingly).

IV. Experimental Configurations

In this section, we first present the hardware configurations that we used in our study. Next, we describe how we ported Hadoop to run on the selected hardware. Then, we briefly describe the representative applications that we have considered in our exploration.

A. Cluster Setup

Table II summarizes the hardware specifications, as well as the initial cost and power ratings, of five different type of nodes that we use to build test clusters in this work. Two of the configurations (E5450 and E5620) are based on traditional servers, while the remaining three comprise of different 32-bit ARM-based microservers, namely Raspberry Pi (RasPi) [41], MK908 [40], and WandBoard Quad (WBQuad). Each test cluster consists of five homogeneous nodes of the same type, except for RasPi that has ten nodes. The main difference between the MK908 and WBQuad microservers is the sequential disk access and network I/O rates as shown in Table III. WBQuad has a SATA connector along with two micro SD card slots and 1 Gbps network connection, whereas MK908 has built-in 8 GB NAND flash storage and only one slot for micro SD card with 100 Mbps network connection.

For cluster networking, we used a NETGEAR 48-Port 10/100 Smart switch that draws 12–14 Watts. Furthermore, we use Hadoop version 1, and execute both the JobTracker and NameNode on a separate high-end machine. The energy, cost and other values of this machine have been properly accounted for in the presented *PerfEC* values for each considered cluster

TABLE II. SPECIFICATIONS OF THE HARDWARE CONFIGURATIONS CONSIDERED IN OUR STUDY.

Hardware	CPU	Cores/ Threads	DRAM	Cache	Cost ($)	Power (Watts)
RasPi (+SD card)	ARM1176JZF-S @ 700 MHz	1/1	512 MB	128 KB L2	40 (+10)	2-4
MK908	ARM Cortex A-9 @ 1.6 GHz	4/4	2 GB	1 MB L2	65	2-4
WBQuad (+HDD)	ARM Cortex A-9 @ 1.0 GHz	4/4	2 GB	1 MB L2	125 (+60)	3-6 (+8)
E5450	Dual-Socket Intel Xeon E5450 @ 3.0 GHz	8/8	16 GB	12 MB L2	1200	120-180
E5620	Dual-Socket Intel Xeon E5620 @ 2.8 GHz	8/16	48 GB	12 MB L3 & 4 MB L2	3500	180-300

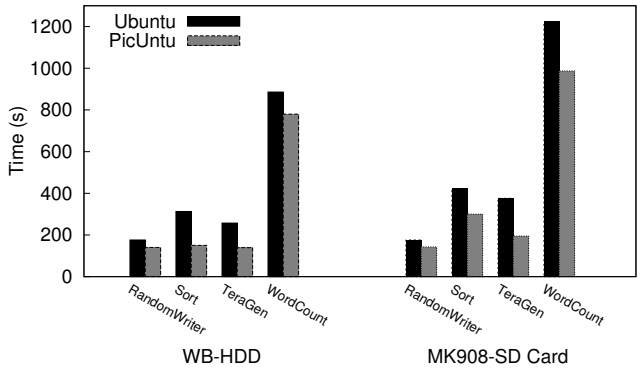


Fig. 1. Single-node Hadoop performance on Ubuntu vs. PicUntu. The dataset size is 250 MB.

configuration. Our comparison focuses on the performance of Hadoop applications, and running the Hadoop management components on a separate machine ensures that the impact of node hardware on the application performance is highlighted. Moreover, this also implies that the reported comparison for the applications will hold for newer versions of Hadoop, such as YARN [43], as well.

B. Software Porting and Tuning

The microservers have limited storage and memory, so it is not feasible [26] to run out-of-the-box Hadoop on them. In the following, we describe how we ported the software to execute Hadoop on the microservers and fine tuned its performance.

1) MK908 Microserver: MK908 is a MiniPC device and runs Android OS as default. Although there has been work done to port Hadoop on Android [34], the open source implementation of Hadoop does not support Android. Therefore, we first ported embedded Linux to MK908, which required cross-compiling the bootloader, kernel and adapting the root file system (RFS). The resulting binaries were then copied to the built-in NAND of MK908 using the "flash" tool provided by the manufacturer. We also configured the bootloader to mount the RFS either from the NAND flash or the micro SD card at runtime.

We tested two different Linux distributions, Ubuntu 13.10 [36] and PicUntu 4.4.3 [25] both with 3.13.1-armv7-x9 kernel, to determine the one that works best for the microserver. Figure 1 shows the single-node performance of four representative Hadoop applications under the two distributions. We observe that for the MK908, PicUntu performs better and selected it for our tests.

TABLE III. DISK AND NETWORK I/O RATES OBSERVED FOR THE STUDIED HARDWARE USING LARGE DISK READ/WRITE AND `iperf` BENCHMARKS, RESPECTIVELY.

Hardware	Read BW (MB/s)	Write BW (MB/s)	Network BW (MB/s)
RasPi	18	8	9
MK908	17	8.5	9.25
WBQuad	40	39	93
E5450	40	39	95.25
E5620	84	68	99.4

2) WBQuad Microserver: For WBQuad, the manufacturer provides an Android OS as well as Ubuntu distribution for the microserver. Even though WBQuad has a SATA connector, the default OS does not support mounting the RFS from a SATA-attached hard disk drive (HDD). According to the instructions provided by the manufacturer [49], RFS over SATA can be enabled using hardware rewiring (removing two resistors and placing one resistor back at the bottom side of the WBQuad at the cost of voiding the warranty). Instead, we chose to cross-compile for the WBQuad with a custom bootloader. This allowed us to change the boot parameters and mount the RFS directly from a SATA HDD. Similarly as for the MK908, we tested and chose PicUntu for WQBand as well.

3) RasPi Microserver: For RasPi, we used the default distribution provided by the manufacturer, Pidora, which is a Fedora Remix optimized for the Raspberry Pi with Linux kernel version 3.6.11. The RasPi has limited memory and fairly small computing power, so launching the TaskTracker and the DataNode processes on the same RasPi node did not succeed. To overcome this, we used a larger 10-node RasPi cluster: One node as JobTracker; one as NameNode; three as DataNodes; three with one mapper each; and two with one reducer each.

C. Studied Hadoop Applications

In the following, we describe six applications from the well-known Hadoop HiBench Benchmark Suite [22], which we have used in our study. These applications are representative of typical Hadoop workloads.

RandomWriter: is a map-only application where each map task takes as input the name of a file and writes random keys

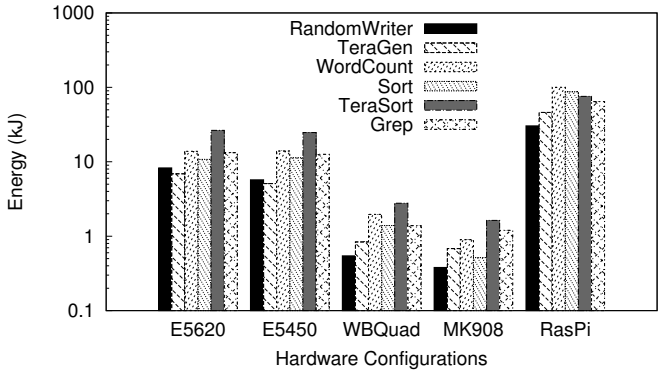


Fig. 2. Energy consumed (kilojoules, log scale) by the studied hardware configurations under Hadoop Workloads.

and values to the file. There is no intermediate output, and the reduce phase is a identity function.

TeraGen: generates a large number of random numbers, and is typically used for driving sorting benchmarks. This is a also a map-only application and does not take any input.

WordCount: counts the frequency of all the different words in the input. The map task simply emits $(word, 1)$ for each word in the input, a local-combiner computes the partial frequency of each word, and the reduce tasks performs a global combine to generate the final result.

Sort: performs a sort of the input. A mapper is an identity function and simply directs input records to the correct reducer. The actual sorting happens thanks to the internal shuffle and sort phase of MapReduce, thus the reducer is also an identity function.

TeraSort: performs a scalable MapReduce-based sort of input data. It first samples the input data and estimates the distribution of the input by determining r-quantiles of the sample (r is the number of reducers). The distribution is then used as a partition function to ensure that each reducer works on a range of data that does not overlap with other reducers. The sampling-based partitioning of data also provides for an even distribution of input across reducers. The sorting is achieved similarly as in *Sort*.

Grep: searches for all occurrences of a pattern in a collection of documents. Each mapper reads in a document, and runs a traditional "grep" function on it. The output size depends on the number of occurrences and can range from zero to the size of the input. A reducer in grep is simply an identity function, so this is also a map-only application.

V. Evaluation

In this section, we present the results of running six representative applications — chosen from well-known Hadoop/MapReduce works [10], [22], [46] (Section IV-C)— on the cluster configurations of Section IV.

A. Measurements on Real Clusters

1) Energy Consumption: Figure 2 shows the energy consumed under different cluster configurations when running the studied applications. We measured the energy using Whatsup Pro [6] power meters that were sampled every two seconds. Note that as shown in Table I, the idle power ratings of the microservers is significantly less than that of the traditional servers. So including idle power in the energy consumption would have given unfair advantage to the microservers. To address this, we only report the energy consumed by a cluster when Hadoop jobs are running on it, and factor out the idle power. This provides for a more fair comparison between the servers.

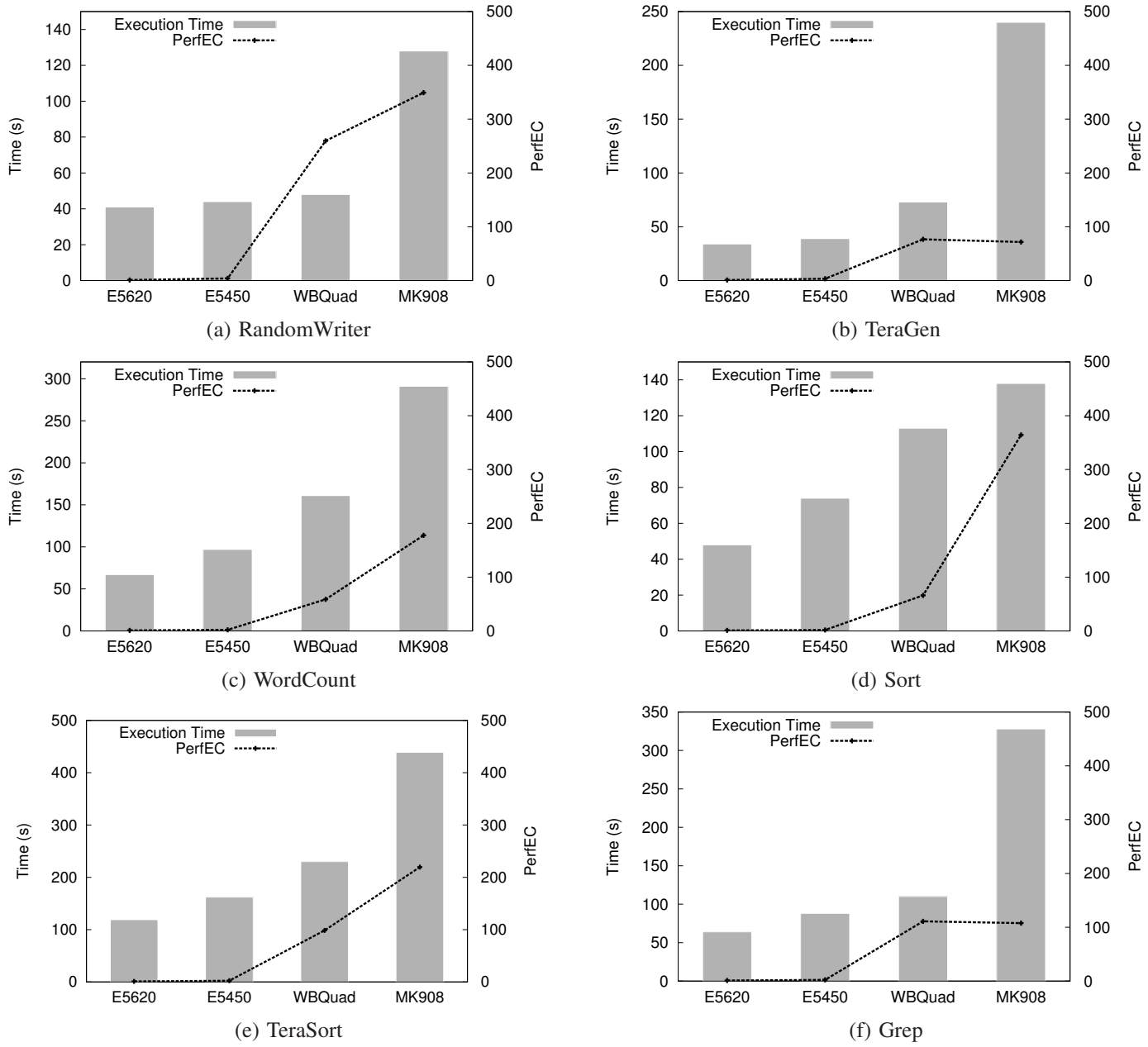
We observe that the energy consumed by WBQuad and MK908 is just a fraction of that consumed by the traditional servers. Across all applications, WBQuad and MK908 consume 88.8% and 93.3% less energy than E5620, respectively. This is promising. However, the importance of selecting an appropriate microserver is highlighted by the case of RasPi, where the total energy consumed for the studied workloads is $5\times$ that of E5620. This is because, even though RasPi consumes much less power than the traditional servers, the applications also take much longer on the RasPi setup to complete due to reduced performance, hence consuming more energy overall.

2) Performance Comparison: In our next experiment, we measure the execution time of the studied applications on our cluster configurations. Figure 3 shows the execution time and normalized *PerfEC* with respect to the case of E5620 using a 2 GB dataset. The RasPi results are shown separately in Figure 4, as this cluster had 10 much-slower nodes resulting in higher execution time. As observed in the figures, *PerfEC* was accurately able to capture the suitability of the servers. For example, RasPi has very low values for *PerfEC* indicating that it is not a suitable substrate due to its unacceptably low performance that negates its benefits of low cost and energy saving ability.

Similarly, consider the case of E5620 that has the least execution time but has high energy consumption and initial cost. Thus, it is not a efficient compute substrate for the applications. We observe that the corresponding value of *PerfEC* for E5620 is lower than the microservers, correctly suggesting that this server is not efficient overall for the studied workloads.

Next, we studied the impact of dataset size on *PerfEC*. For this experiment, MK908 and RasPi were unable to support test cases of larger than 2 GB due to limited resources, thus we do not include them in the results shown in Figure 5. We see that *PerfEC* is not constant as data size is varied for the applications, indicating the metric's ability to distinguish the hardware configuration that can achieve the best combination of performance, energy, and cost for the workload.

Further analysis revealed that Hadoop applications need larger network provisioning as both file system I/O operations and MapReduce operations are dependent on the network connectivity. We find that for I/O-intensive applications—such as *TeraGen*, *RandomWriter*, and *Grep*—the overall execution time on WBQuad is 44% and 19% slower than E5620 and E5450, respectively. On the other hand, for CPU-intensive applications, WBQuad exhibit even lower comparative performance. In spite of performance loss, WBQuad has a higher *PerfEC*, because the energy consumption and the initial cost



(a) RandomWriter (b) TeraGen

(c) WordCount (d) Sort

(e) TeraSort (f) Grep

Fig. 3. Performance and *PerfEC* of different Hadoop applications on the studied clusters. Dataset used is 2 GB.

of WBQuad is significantly lower than the other two servers. However, we can potentially mitigate the performance loss by employing a higher number of WBQuad servers, and yet can realize energy and cost savings. This is why, by considering the holistic efficiency, the *PerfEC* metric favors this microserver over the traditional server. One caveat here is that a larger number of microservers would entail more networking equipment, related costs and energy consumption, thus these additional costs should be considered while calculating *PerfEC*; a simple linear relationship should not be assumed.

B. Large Cluster Simulation Results

A challenge that we face when comparing the different clusters is how to study how other components such as networking and large data sizes will interact with the servers as the cluster size increases. So in our next set of experiments,

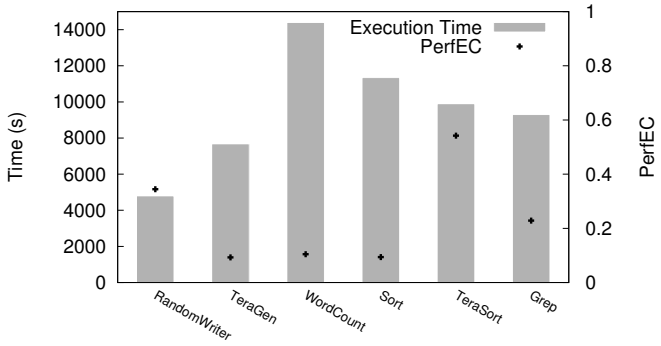


Fig. 4. Performance and *PerfEC* of different Hadoop applications running on 10-node RasPi cluster. Dataset used is 2 GB.

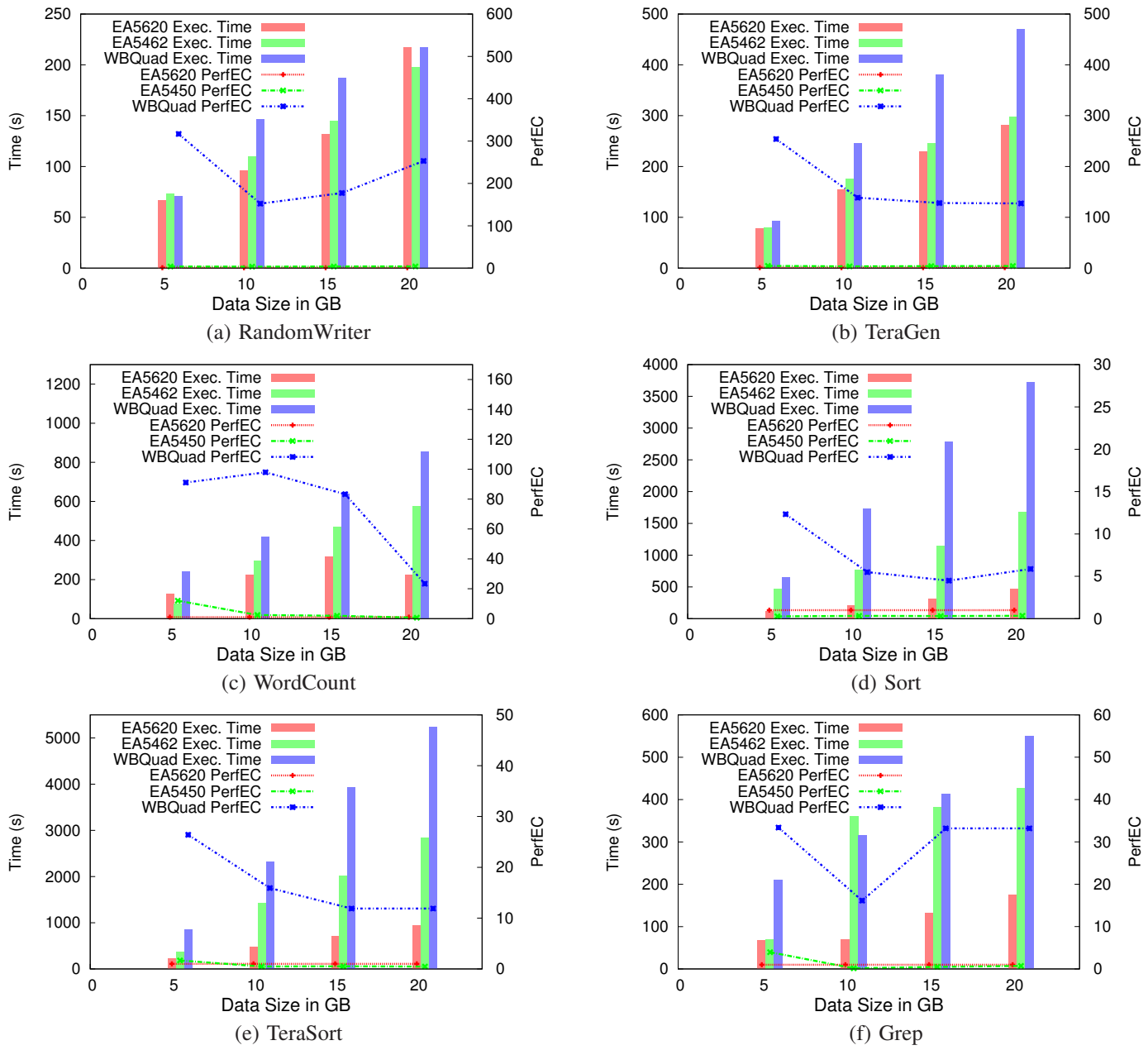


Fig. 5. Performance and *PerfEC* of different Hadoop applications running on WBQuad and traditional clusters, with increasing dataset sizes.

we simulated four large setups to explore how the studied configurations will perform at scale. To this end, we leveraged our MRPerf simulator [46], which has been extensively used to study the impact of data placement, networking, and application characteristics on overall Hadoop application performance [45]–[47]. We also validated the microserver-based simulations using our small testbed and found that the simulation results were within 5% of the actual results.

We considered clusters with: 3000 nodes of E5620, 3000 nodes of E5462, 3000 nodes of WBQuad, and 6000 nodes of WBQuad (WBQuadx2). We chose a multi-rack tiered-topology with 24 nodes per rack, and in our estimation factored in the extra networking costs related to the larger WBQuadx2 setup as well. We assume a locality-aware scheduler with minimal cross-rack traffic.

We used a 45-day (from Oct. 1 to Nov. 15, 2010) Hadoop job trace from the primary 3,000-machine Facebook production cluster [9]. The trace contains detailed information about more than 1 million jobs and the associated input, intermediate and output data size, job completion time, time taken for map and reduce phase, and the number of tasks involved in a job.

Figure 6 shows the execution time and *PerfEC* normalized to that of E5620 for the four simulated clusters. We find that *PerfEC* for WBQuad is 8× that of E5620 and E5450, but the execution time is 14× and 7× of E5620 and E5450, respectively. Similarly as in the real testbed case, *PerfEC* is higher for WBQuad due to its significantly cheap cost and low energy consumption. Therefore, we see that under WBQuadx2, where we doubled the number of WBQuad nodes, the performance degradation reduced to 2.6× that of E5620,

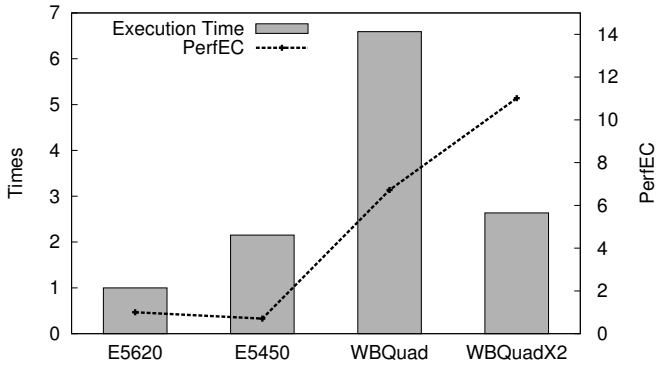


Fig. 6. Performance and *PerfEC* comparisons for the simulated large cluster.

TABLE IV. COMPARISON OF TCO AND *PerfEC*.

Machine	TCO	1/TCO	*PerfEC*
E5620	1	1	1
E5450	1.0245	0.9760	0.7056
WBQuad	0.3256	3.0711	8.5194
WBQuadx2	0.1291	7.74	13.315

but still offered 89% lower energy consumption at 90% less initial cost compared to E5620. This is promising, as it indicates that horizontal scaling of microservers can offer overall high efficiency as well as match the performance of traditional servers, provided networking equipment can scale at reasonable cost (e.g., $\leq$ linear). In fact, we repeated the experiment with 6$\times$ WBQuad nodes than E5620, and matched the performance with 31% energy consumption at 60% of the initial cost of E5620.

C. Evaluation of PerfEC

In our next experiment, we compare the efficacy of *PerfEC* by comparing it with the extant TCO [16] metric. For this test, we used our simulation setup driven by the 45-day trace as described earlier. Table IV shows the values for *PerfEC* and TCO normalized to that of E5620 for all the studied clusters. Given that a lower value of TCO is desirable unlike *PerfEC* where a larger value is desirable, the table also shows $1/TCO$ for comparison. We observe that TCO predicts the efficiency of WBQuadx2 to be more than double that of the WBQuad, which is not the case in reality. Furthermore, TCO is not able to properly distinguish between the E5620 and E5450 (TCO of E5450 is predicted to be just 2.5% higher to that of E5620), even though a clear difference is observed in performance and other measurements. In contrast, *PerfEC* was able to capture the differences between the servers realistically and in line with measured performance, energy, cost. For instance, for the given workloads the value of *PerfEC* for E4550 is only 70% of that for E5620. We also compared *PerfEC* with JouleSort [38]. However, JouleSort is not directly comparable to *PerfEC* as JouleSort only provides the SortRecords/Watt and lacks other factors such as cost of the equipment, scalability, and the ability to capture other applications besides sort. Thus, *PerfEC* is a viable metric that subsumes or exceeds the ability of existing metrics such as TCO.

VI. RELATED WORK

A number of works, such as Green Hadoop [18] and energy aware scheduling on MapReduce jobs [50], have focused on reducing the operational cost of data centers. Similarly, Nan Zhu et al. [51] aim to tame the peak power consumption in a MapReduce cluster by using adaptive power regulation. Although we share with these works the consideration of energy consumption, our study is novel and different in its focus on evaluating the suitability of microservers to support Hadoop applications.

Much recent research investigates the possibility of building clusters with ARM CPUs. Irdis-pi cluster [13] consists of 64 Raspberry Pi Model B nodes each equipped with a 700 MHz ARM processor, and is used for educational applications. FAWN [3] considers low-power embedded CPUs to develop a key-value store. Luarra Keys et al. [29] has used embedded and mobile devices to find energy efficient building blocks for data centers to support Dryad [23]. These works are complementary to ours in terms of exploring new usecases for microservers, but differ in that we focus on the Hadoop platform.

A broadband embedded computing system for MapReduce [27] is perhaps the closest to our work, as it also uses microservers for Hadoop. But the main focus of their work is to design a heterogeneous cluster to combine cloud computing with distributed embedded computing.

Finally, ours is the first work that combines the factors that affect data center design and selection of appropriate microserver architectures into a comprehensive metric that considers performance, energy consumption, and cost. Other researches [30], [31] that propose to improve the performance of Hadoop under heterogeneous environment are complementry to our findings and can co-exist with our work.

VII. CONCLUSIONS

We investigated whether microservers provide a suitable substrate for supporting Hadoop. For this purpose, we proposed a unique metric, *PerfEC*, to calculate the performance, overall cost and energy consumed when executing an application on a particular microserver. Using this metric, we studied six Hadoop applications on three different microservers and two traditional servers.

Our analysis using a real testbed revealed that for applications such as *TeraGen*, *RandomWriter* and *Grep*, microservers can yield two orders of magnitude better efficiency in terms of *PerfEC* than traditional servers. Similarly, simulation results for a large cluster also verified our findings. This initial study bode well for microservers supporting Hadoop. Finally, we compared *PerfEC* with the extant TCO metric and showed that *PerfEC* is better able to capture the trade-offs involved when studying microservers for supporting Hadoop applications.

ACKNOWLEDGMENT

This work is sponsored in part by the NSF under the CNS-1016793, CCF-0746832, CNS1422788 and CNS1405697 grants.

REFERENCES

[1] How Clean is Your Cloud?, 2012. http://www.greenpeace.org/international/en/publications /Campaign-reports/Climate-Reports/How-Clean-is-Your-Cloud/.

[2] AMD. Amd unveils its first arm cpu, the 64-bit 8-core opteron a1100, 2014. http://www.extremetech.com/computing/175583-amd-unveils-its-first-arm-based-cpu-the-64-bit-8-core-opteron-a1100.

[3] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan. Fawn: A fast array of wimpy nodes. In *ACM SIGOPS 22nd*, 2009.

[4] D. G. Andersen and S. Swanson. Rethinking flash in the data center. *IEEE micro*, 2010.

[5] Apache Software Foundation. Hadoop, 2011. http://hadoop.apache.org/core/, accessed on Nov 6, 2013.

[6] C. Barnum, D. Dayton, K. Gillis, and J. O'Connor. Making connections-teaming up to connect users, developers, and usability experts. In *IEEE IPCC*, 2005.

[7] D. Borthakur. Facebook has the world's largest hadoop cluster!, 2010. http://hadoopblog.blogspot.com/2010/05/facebook-has-worlds-largest-hadoop.html.

[8] D. e. Borthakur. Apache hadoop goes realtime at Facebook. SIGMOD '11. ACM, 2011.

[9] Y. Chen, S. Alspaugh, D. Borthakur, and R. Katz. Energy efficiency for large-scale mapreduce workloads with significant interactive analysis. In *Proceedings of the 7th ACM european conference on Computer Systems*, 2012.

[10] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz. The case for evaluating mapreduce performance using workload suites. In *IEEE MASCOTS*, 2011.

[11] B.-G. Chun, G. Iannaccone, G. Iannaccone, R. Katz, G. Lee, and L. Niccolini. An energy case for hybrid datacenters. *ACM SIGOPS*, 2010.

[12] E. S. Chung, J. D. Davis, and J. Lee. Linqits: Big data on little clients. In *ACM ISCA*, 2013.

[13] 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*, pages 1–10, 2013.

[14] F. Dong. *Extending Starfish to Support the Growing Hadoop Ecosystem*. PhD thesis, Duke University, 2012.

[15] A. Dou, V. Kalogeraki, D. Gunopulos, T. Mielikainen, and V. H. Tuulos. Misco: a mapreduce framework for mobile systems. In *ACM PETRA*, 2010.

[16] L. M. Ellram. Total cost of ownership. In *Handbuch Industrielles Beschaffungsmanagement*, pages 659–671. Springer, 2002.

[17] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. W. Cameron. Powerpack: Energy profiling and analysis of high-performance systems and applications. *IEEE TPDS*, 21(5):658–671, 2010.

[18] Í. Goiri, K. Le, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini. Greenhadoop: leveraging green energy in data-processing frameworks. In *Eurosys*, 2012.

[19] B. Gu, D. Choi, and Y. Kwak. Potentiality for executing hadoop map tasks on gpgpu via jni. In *Multimedia and Ubiquitous Engineering*. Springer, 2013.

[20] J. R. Hauser and J. Wawrzynek. Garp: A mips processor with a reconfigurable coprocessor. In *IEEE FCCM*, 1997.

[21] G. Horvat, D. Vinko, and D. Zagar. Household power outlet overload protection and monitoring using cost effective embedded solution. In *IEEE MECO*, 2013.

[22] S. Huang, J. Huang, Y. Liu, L. Yi, and J. Dai. Hibench: A representative and comprehensive hadoop benchmark suite. In *Proceedings of the ICDE Workshops*, volume 2010, 2010.

[23] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. *ACM SIGOPS*, 2007.

[24] D. Jaggar. *ARM Architecture Reference Manual*. Prentice Hall, 1996.

[25] J.-P. Jamont, L. Médini, and M. Mrissa. Les objets communicants.

[26] S. B. Joshi. Apache hadoop performance-tuning methodologies and best practices. In *IEEE SIPEW*, 2012.

[27] Y. Jung, R. Neill, and L. P. Carloni. A broadband embedded computing system for mapreduce utilizing hadoop. In *IEEE CloudCom*, 2012.

[28] R. T. Kaushik and M. Bhandarkar. Greenhdfs: towards an energy-conserving, storage-efficient, hybrid hadoop compute cluster. In *USENIX ATC*, 2010.

[29] L. Keys, S. Rivoire, and J. D. Davis. The search for energy-efficient building blocks for the data center. In *Computer Architecture*, pages 172–182. Springer, 2012.

[30] K. Krish, A. Anwar, and A. R. Butt. hats: A heterogeneity-aware tiered storage for hadoop. 2014.

[31] K. Krish, A. Anwar, and A. R. Butt. sched: A heterogeneity-aware hadoop workflow scheduler. 2014.

[32] I. Manousakis and D. S. Nikolopoulos. Epc: a power instrumentation controller for embedded applications. *SIGBED Review*, 9(2):28–32, 2012.

[33] R. Mantri, R. Ingle, and P. Patil. Scdp: Scalable, cost-effective, distributed and parallel computing model for academics. In *ICECT'2011*.

[34] E. E. Marinelli. Hyrax: cloud computing on mobile devices using mapreduce. Technical report, 2009.

[35] C. May, E. Silha, R. Simpson, H. Warren, et al. *The PowerPC Architecture: A specification for a new family of RISC processors*. Morgan Kaufmann Publishers Inc., 1994.

[36] M. G. Noll. Running hadoop on ubuntu linux (multi-node cluster). 2007. http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-nodecluster/.

[37] N. Rajovic, A. Rico, N. Puzovic, C. Adeniyi-Jones, and A. Ramirez. Tibidabo: Making the case for an arm-based hpc system. *Future Generation Computer Systems*, 2013.

[38] S. Rivoire, M. A. Shah, P. Ranganathan, and C. Kozyrakis. Joulesort: a balanced energy-efficiency benchmark. In *ACM SIGMOD*, 2007.

[39] B. Smith. Arm and intel battle over the mobile chip's future. *Computer*, 2008.

[40] Tronsmart. Mk908, 2013. www.geekbuying.com/item/Tronsmart-MK908-Google-Android-4-1-Mini-PC-TV-Box-RK3188-Quad-Core-2G-8G-BT-Black-315155.html.

[41] 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 *IEEE ICDCSW*, 2013.

[42] J. Turley. Coldfire doubles performance with v4. *Microprocessor Report*, 26, 1998.

[43] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler. Apache hadoop yarn: Yet another resource negotiator. In *Proc. SOCC*. ACM, 2013.

[44] C. Wang, X. Li, X. Zhou, Y. Chen, and R. C. Cheung. Big data genome sequencing on zynq based clusters. In *ACM SIGDA*, 2014.

[45] G. Wang, A. R. Butt, H. Monti, and K. Gupta. Towards synthesizing realistic workload traces for studying the hadoop ecosystem. In *IEEE MASCOTS*, 2011.

[46] G. Wang, A. R. Butt, P. Pandey, and K. Gupta. A simulation approach to evaluating design decisions in mapreduce setups. In *IEEE MASCOTS*, 2009.

[47] G. Wang, A. Khasymski, K. Krish, and A. R. Butt. Towards improving mapreduce task scheduling using online simulation based predictions. In *IEEE MASCOTS*, 2013.

[48] T. White. *Hadoop: The Definitive Guide: The Definitive Guide*. O'Reilly Media, 2009.

[49] Wnadboard. Sata boot, 2013. http://www.wandboard.org/index.php/50-20131017-wandboard-sata-boot.

[50] N. Yigitbasi, K. Datta, N. Jain, and T. Willke. Energy efficient scheduling of mapreduce workloads on heterogeneous clusters. In *2nd International Workshop on Green Computing Middleware (GCM)*, 2011.

[51] N. Zhu, L. Rao, X. Liu, J. Liu, and H. Guan. Taming power peaks in mapreduce clusters. In *ACM SIGCOMM Computer Communication Review*, 2011.