Towards Energy Awareness in Hadoop

Krish K. R.†, M. Safdar Iqbal†, M. Mustafa Rafique‡, Ali R. Butt†
†Department of Computer Science, Virginia Tech
‡IBM Research, Ireland
Email: {kris, safdar}@cs.vt.edu; mustafa.rafique@ie.ibm.com; butta@cs.vt.edu

***Abstract*—With the rise in the use of data centers comprised of commodity clusters for data-intensive applications, the energy efficiency of these setups is becoming a paramount concern for data center operators. Moreover, applications developed for Hadoop framework, which has now become a de-facto implementation of the MapReduce framework, now comprise complex workflows that are managed by specialized workflow schedulers, such as Oozie. These schedulers assume cluster resources to be homogeneous and often consider data locality to be the only scheduling constraint. However, this is increasingly not the case in modern data centers. The addition of low-power computing devices and regular hardware upgrades have made heterogeneity the norm, in that clusters are now comprised of several logical sub-clusters each with its own performance and energy profile. In this paper we present ϵSched, a workflow scheduler that profiles the performance and the energy characteristics of applications on each hardware sub-cluster in a heterogeneous cluster in order to improve the application-resource match while ensuring energy efficiency and performance related Service Level Agreement (SLA) goals. ϵSched borrows from our earlier work, ϕSched, a hardware-aware scheduler, that improves the resource–application match to improve application performance. We evaluate ϵSched on three clusters with different hardware configurations and energy profiles, where each sub-cluster comprises of five homogeneous nodes. Our evaluation of ϵSched shows that application performance and power characteristics vary significantly across different hardware configurations. We show that the hardware-aware scheduling can perform 12.8% faster, while saving 21% more power than hardware oblivious scheduling for the studied applications.**

I. Introduction

In recent years, the implementation of MapReduce [10] in Hadoop [3] has emerged as an efficient framework that is being extensively deployed to support a variety of big-data applications [5], [12], [23]. Modern Hadoop deployments are evolving from homogeneous clusters of commodity computers to a range of hardware from massive-core accelerators to low-power ARM-based devices [9], [29]. The data center setups are becoming heterogeneous, both from the use of advance hardware technologies and as a result of regular upgrades to the cluster hardware.

Trends indicate that data centers in the US consume upwards of 100 billion KWh per year [13]. The scale of data centers have made their power consumption an imperative issue. Power management has become of unprecedented importance not only from an economic perspective but also for environment conservation. This increases the electricity cost and aggravates carbon dioxide emissions. Most of the past work on power management in computing has been focused on energy harvesting by scheduling based on power profiling [15]. These techniques cannot be directly applied to Hadoop because the scheduling and data placement decisions in Hadoop assume that the underlying hardware is homogeneous and cannot take into account the heterogeneity of the underlying hardware architectures, which makes it impossible to effectively harvest energy benefits by using any profiling information.

Hadoop applications are becoming more intricate, and routinely comprise complex workflows with a large number of iterative jobs [17], interactive querying [17], as well as traditional batch-friendly long running tasks [6]. These applications show varying performance and energy behavior depending on the characteristics of different types of underlying hardware. Traditional Hadoop workflow schedulers are oblivious to the underlying hardware architecture and the energy consumption characteristics of the applications. Thus, in their scheduling decisions, the schedulers do not consider the variation in the application execution characteristics, such as performance and energy consumption, of Hadoop applications on heterogeneous computing substrates that are quickly becoming the norm. In these environments, it is suboptimal to schedule applications ignoring the effects of this variation. Instead, if applications are scheduled in a manner attuned to these variations, taking into account the power and the performance characteristics of cluster, substantial savings in the power or large increases in the performance can be achieved.

In this paper, we propose to improve the application-resource match by considering heterogeneous Hadoop deployments that comprise of one or more homogeneous sub-clusters. The set of tasks to be executed on the heterogeneous deployment cluster will be scheduled to the sub-clusters such that the total energy consumption is minimized while the performance goals specified in the Service Level Agreement (SLA) are met. We propose simple, application characteristic-aware task scheduling in Hadoop to reduce the power consumption or to improve the throughput.

To this end, we present ϵSched, a heterogeneity-aware and power-conserving task scheduler for Hadoop. ϵSched extends our own ϕSched system [21] – a hardware characteristic-aware scheduler that improves the resource-application match. We extend Hadoop's hardware-aware scheduler, which is optimized only for performance, to be an energy efficient scheduler. We adopt a quantitative approach where we first study detailed behavior of applications, such as performance and power characteristics, of various representative Hadoop applications running on four different hardware configura-

tions. Next, we incorporate findings of these experiments into ϕSched. To ensure that job associated data is available locally to (or nearby) a cluster in a multi-cluster deployment, ϕSched configures a single Hadoop Distributed File System (HDFS) [21] instance across all the participating clusters. As part of ϵSched, we also design and implement a region-aware data placement and retrieval policy for HDFS in order to reduce the network overhead and achieve cluster-level data locality.

Specifically, this paper makes the following contributions:

- We design a workflow management system to effectively schedule jobs to multiple heterogeneous clusters while considering the power and performance goals;
- We develop an effective mechanism to track, record, and analyze the performance and the power characteristics of Hadoop applications on clusters with different hardware configurations; and
- We validate the design and techniques therein using experiments on a real deployment using various MapReduce applications.

We evaluate our approach on a deployment with three clusters with different hardware configurations, where each cluster has five homogeneous nodes. Our evaluation of ϵSched reveals that application performance varies significantly across different hardware configurations. The results of our evaluation show that the ϵSched can speed up job by 12.8%, while operating at 21% the power of hardware- and power-oblivious scheduling for the studied applications.

II. Background

Hadoop offers an open-source implementation of the MapReduce framework that provides machine-independent programming at scale. Hadoop provides a *JobTracker* component that accepts jobs from the users and also manages the compute nodes that each run a *TaskTracker*. Each TaskTracker has one or more map and reduce slots, and applications will have tens of hundreds of map and reduce tasks running on these slots. In the case of heterogeneous clusters, the map/reduce tasks executing on the slowest node will determine the execution time of the application [2]. Although speculative execution [34] can reduce this dependency, it leads to significant resource wastage due to re-execution of tasks.

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 manage worker components called *DataNodes* running on each Hadoop node. Typically, each MapReduce cluster is configured with one instance of HDFS, and the data in one cluster is not accessible (directly) from other clusters. 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. The fact that HDFS distributes its data across all the underlying nodes restricts the opportunity to save energy by turning off the inactive nodes. As these set of nodes may have all the replicas of the files that are currently being processed and turning off the nodes may result in data unavailability.

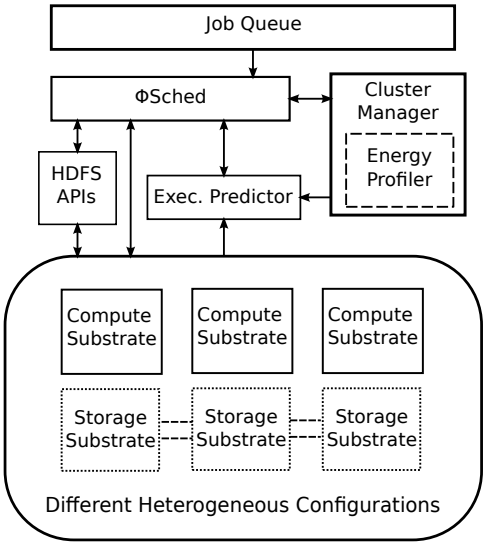

Fig. 1. ϵSched architecture overview.

Workflows have become an integral part of modern Hadoop applications, and are managed by workflow managers such as Apache Oozie [19] and Nova [26]. The workflow scheduler is responsible for co-ordinating the various events/tasks in a multi-cluster setup.

III. Design

A. Enabling Technology: ϕSched

ϕSched [21] is a hardware-heterogeneity-aware workflow scheduler for Hadoop that proposes to consider applications behavior on specific hardware configurations when scheduling Hadoop workflows. It assumes that a deployment is made of one or more *resource clusters* each with a different hardware configuration, and that the resources within a cluster are homogeneous. It focuses on variations in performance characteristics, such as CPU, memory, storage, and network usage, for a range of representative Hadoop applications on different hardware configurations. Next, based on our understanding of the applications, ϕSched: i) profiles applications execution on different clusters and performs statistical analysis to determine a suitable resource–application match; and ii) effectively utilizes the matching information to schedule future jobs on clusters that will yield the highest performance. To schedule a job, ϕSched also examines the current utilization of the clusters and the suitability of clusters to support the job based on prior profiling. Based on these factors, ϕSched suggests the best cluster to execute the job. Such profiling is feasible as recent research [6], [24] has shown the workflows to have very predictable characteristics, and the number of different kinds of jobs to be less than ten.

ϕSched treats a Hadoop deployment to consist of multiple separate clusters to handle resource heterogeneity. This leads

```
For workload W, arrange clusters C_1 to C_n in C in the
descending order of performance to power ratio ;
foreach C_i in C do
    if C_i meets SLA then
        if isResourceAvailable(C_i,W) then
            Schedule W in C_i;
            Increment cluster utilization for C_i;
            break;
        end
    end
end
```

Algorithm 1: Energy Profiler in ϵSched.

to the problem that the best cluster, C_B, in order to run an application in terms of execution characteristics, such as energy efficiency or execution time, may not have the data associated with the application, entailing data movement to C_B from the cluster, C_D, that has the data. C_D may not be able to support the application due to hardware constraints. Moreover, the data movement may be very expensive and negate the performance gain that can be realized by running the job on C_B. To this end, ϕSched proposes configuring a single HDFS instance for all the Hadoop clusters and logically arrange participating HDFS nodes by associating each node's storage within a virtual storage group referred to as a *region*. The placement and retrieval policy are region-aware meaning we can specify the set of nodes (in terms of "region") to store and retrieve the data.

B. Energy Profiler

As shown in Figure 1, ϵSched integrates an Energy Profiler component to ϕSched to track, record and analyze the power usage characteristics of the application on a particular hardware. ϵSched computes the performance to power ratio for different jobs in each cluster. As shown in Section IV, different cluster show different performance and power characteristics. There is no single cluster that is optimal for power and performance for all workloads.

As illustrated in Algorithm 1, when a job is to be scheduled in one of the homogeneous sub-clusters, $C_1, C_2, ..., C_n$ in a heterogeneous cluster, C, deployment, the Energy Profiler component accesses its profiled power characteristics. The sub-clusters are arranged in the order of performance to power ratio. Energy Profiler will traverse through the ordered list of sub-clusters to find the cluster that would meet the SLA of the workflow to be scheduled. Upon recognizing the optimal sub-cluster that will ensure the least power usage while meeting the SLA requirements, the Cluster Manager in ϕSched checks for the availability of resources in optimal sub-cluster to schedule the workflow W. If the availability of resources is determined, W is scheduled, else the process is repeated to search for the next optimal cluster.

TABLE I
HARDWARE CONFIGURATIONS CONSIDERED IN OUR EXPERIMENTS.

Name	CPUs	RAM (GB)	Storage (GB)	Network	Map Slots	Reduce Slots
Cluster-1	16	16	HDD	10 Gbps	8	4
Cluster-2	2	2	HDD	128 Mbps	2	2
Cluster-3	8	8	SSD	1 Gbps	4	2

C. Discussion

The HDFS enhancement provided by ϕSched will ensure data availability in a smaller set of nodes by modifying the placement policy. At least one replica of a file is stored in a small subset of nodes (i.e., the sub-cluster with low power-usage) called Covering Subset [1]. This will enable us to apply energy harvesting techniques such as turning a sub-cluster off or running in it low power mode for any of the under-utilized sub-clusters that are not a part of Covering Subset.

It is important to note that the power characteristics of a workload on a cluster is linearly dependent on the data size. Thus the profiler component can linearly extrapolate the power characteristics to different data sizes based on the studied workloads. Similar observations were made for performance characteristics in ϕSched [21]. Moreover the number of sub-clusters in a real deployment will be less than ten, so the overhead constituted by the energy profiler component is minimal.

IV. EVALUATION

We evaluate the energy characteristics of Hadoop applications on using a real deployment on a medium-scale cluster. We first study the characteristics of 8 representative Hadoop applications on three different cluster hardware configurations.

A. Experimental Setup

We used three clusters of five homogeneous nodes each, where each cluster has a different hardware configuration as listed in Table I. In all of the Hadoop deployments considered in our tests, the master node ran both the Hadoop JobTracker and NameNode, and was co-located with a worker node. Moreover, all worker nodes were configured with varying number of map and reduce slots depending on the configuration of the machines (Table I) along with a DataNode component.

For measuring the power usage we used Watts Up? PRO [11] power meters in the worker nodes. The power values represented in this section are usage characteristics of one worker node in every sub-cluster. All worker nodes in a single sub-cluster are homogeneous and showed similar power characteristics. We do not consider the power consumption of the master node, as we do not propose any optimization to the Hadoop master components.

B. Studied Applications

We have used 8 applications from the well-known Hadoop HiBench Benchmark Suite [18] in our study. These applications are representative of batch processing jobs and iterative

TABLE II
REPRESENTATIVE MAPREDUCE (HADOOP) APPLICATIONS USED IN OUR STUDY.

Application	Map		Mappers	Reducers
	Input	Output		
WordCount	6 GB	12 KB	120	8
DFSIOE-Read	8 GB	–	128	1
DFSIOE-Write	8 GB	–	128	1
Kmeans	1 GB	1 GB	20	1
PageRank	128 MB	12.5 MB	16	8
Bayes	128 MB	4.5 GB	16	1
Sort	6 GB	3 GB	120	8
TeraSort	15 GB	15 GB	240	8

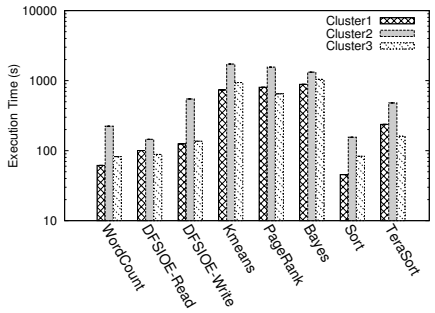

Fig. 2. Time taken for each application in studied cluster.

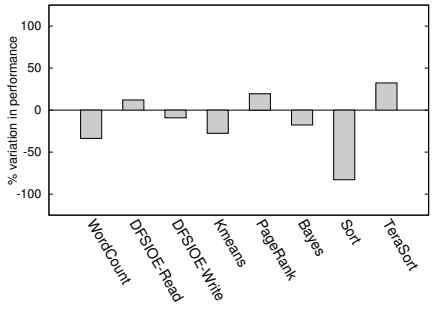

Fig. 3. Performance improvement observed on *Cluster-1* compared to *Cluster-2*.

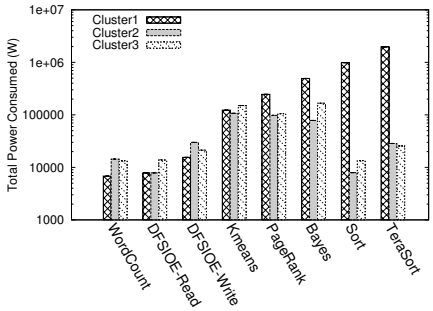

Fig. 4. Total power consumption for each application in studied cluster.

jobs. Table II lists the applications, and summarizes parameters, i.e., the input and the output data sizes, the number of mappers and reducers, for each application.

In this experiment, we measured the execution time of our applications on each of the studied clusters. As shown in Figure 2, the execution time of the applications varies across different cluster configurations. We find that across all applications, on average, *Cluster-1* performs 43% and 31% faster than *Cluster-2* and *Cluster-3*, respectively. However, we observe that the variation in performance is not similar across all applications. For instance, in the case of *DFSIOE-Read* which involves significant I/O, *Cluster-3* performs 14% faster than *Cluster-1*, whereas for the same cluster, *PageRank* performs only 22% slower.

To study this variation in detail, we compared the performance under *Cluster-1* and *Cluster-2* across all the studied applications. Figure 3 shows the results. For applications such as *WordCount*, *DFSIOE-Write*, *Kmeans*, *Bayes*, and *Sort*, *Cluster-1* performs better, while for the rest of the applications *Cluster-2* performs better. One reason for this is the varying resource requirements of the applications. For example, *Kmeans*, which is a CPU intensive application, performs 27.5% faster in *Cluster-1* that has more memory, and *TeraSort*, which involves significant network and I/O usage, performs better in *Cluster-3* that has better interconnects.

C. Power Usage Comparison

In this experiment, we measured the total power consumption of our test applications on the studied clusters. As shown in Figure 4 (Y-axis — log scale), the total power consumption of the applications varies across different cluster configurations. As expected, the power consumption of an application is not only dependent on the underlying hardware architecture but also on the execution time of the application. For applications such as *WordCount*, *DFSIOE-Write* and *DFSIOE-Write*, the optimal power consumption is observed when executed on *Cluster-1*; for *Kmeans*, *PageRank*, *Bayes* and *Sort* the optimal power consumption is observed under *Cluster-2*; finally, *TeraSort* shows the least power consumption under *Cluster-3*.

To further study this variation in detail, we compared the power consumption under *Cluster-1* and *Cluster-2* across all the studied applications. Figure 5 shows the results. For applications such as *WordCount*, *DFSIOE-Read*, and *DFSIOE-Write*, *Cluster-3* shows lower power consumption while for *Kmeans*, *PageRank*, *Bayes*, *Sort* and *TeraSort Cluster-1* shows the lower power consumption. Similarly Figure 6 shows the comparison of power consumption between *Cluster-1* and *Cluster-3*. For applications such as *WordCount*, *DFSIOE-Write*, *DFSIOE-Write* and *Kmeans*, *Cluster-3* performs better while *Cluster-1* performs better for the rest of the applications.

We observe that in spite of the high execution time, for a subset of applications such as *WordCount*, *DFSIOE-Read* and *DFSIOE-write*, the total power consumption is the least in *Cluster-2* because of its low average power consumption as shown in Figure 7. It is observed that *Cluster-2* consumes the least power and *Cluster-1* and *Cluster-3* consume $2\times$ and $3\times$ the power respectively. A similar trend is observed across all applications.

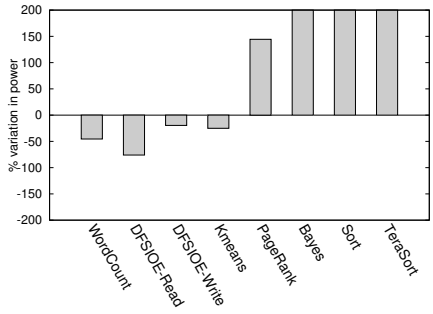

Fig. 5. Power Consumption observed on *Cluster-1* compared to *Cluster-2*.

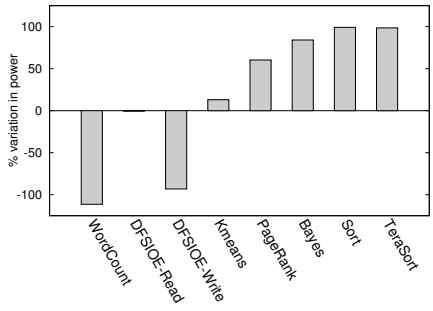

Fig. 6. Power Consumption observed on *Cluster-1* compared to *Cluster-3*.

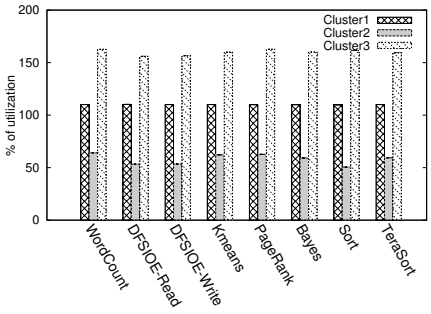

Fig. 7. Average power consumption for each application in studied cluster.

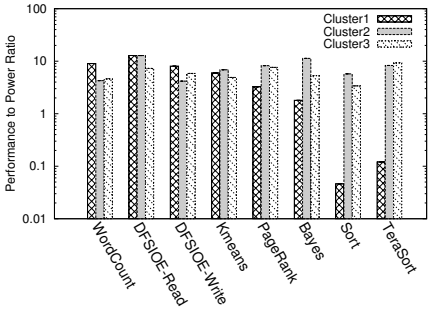

Fig. 8. Performance to power ratio for each application in studied cluster.

D. Performance to Power Ratio Comparison

Figure 8 compares the ratio (higher the better) of performance to the total power in order to find the optimum cluster in terms of both performance and power. For applications such as *WordCount*, *DFSIOE-Write* and *DFSIOE-Write*, the optimal cluster is *Cluster-1*, for *Kmeans*, *PageRank*, *Bayes* and *Sort* the optimal cluster is *Cluster-2*, and for *TeraSort* the optimal cluster is Cluster-3. In Figure 9, detailed examination reveals a variation in the application behavior, similar to the above cases.

For our next set of experiments, we develop an accurate simulator for ϵSched to observe the power consumption and the execution time of the considered clusters. Our fine-grained simulator takes into account details such as the effect of compute capacity, network and storage infrastructure, and application-hardware affinity with reference to power and execution time. We simulate a 300-node cluster, consisting of three 100-node homogeneous sub-clusters. The configuration of each node in a sub-cluster is similar to *Cluster-1*, *Cluster-2* and *Cluster-3* shown in Table I. We use the publicly available synthetic Facebook production traces [7] for driving the simulation. We replay a snippet of these traces using HiBench [17] applications (Table II). The traces run for a length of 3 hours under baseline Hadoop (hardware oblivious scheduling).

Figure 10 shows the completion time and the power consumed by the workloads under both ϵSched and hardware oblivious scheduling. To highlight the power savings achieved using ϵSched we neglect the power consumption of the system in idle state, assuming that power management schemes can be applied to nodes that are in idle state for a longer period of time. ϵSched achieves both performance benefits and power savings over baseline Hadoop. By the application-hardware match improvements in ϵSched, the completion time of the application improves by 12.8%, while consuming 21% less power than baseline Hadoop.

In summary, the above experiments validate the claim that different application-hardware interactions produce different power consumption and performance characteristics. Understanding these characteristics will enable us to achieve our goal of scheduling tasks in a heterogeneous cluster deployment such that the total power consumption is minimized while the performance goals specified in the SLA are met.

V. Related Work

Many hardware and software-based solutions have been proposed to reduce the power consumption of enterprise systems, ranging from shutting down unused components to low energy circuit designs [25], [27], [4], [16], [30]. Weiser et al. [31] first discussed the problem of scheduling tasks to reduce the energy consumption of CPUs. Yao et al. [32] propose an offline scheduling algorithm for independent tasks running with variable speed, assuming worst-case execution time. We aim to provide a similar power management solution for the Hadoop ecosystem tailored to the unique characteristics for Hadoop applications.

Much of the recent work on energy efficiency in the Hadoop ecosystem focuses on storage [20], [22], [1]. Rini et al. [20] propose energy-aware date placement, where unpopular data is placed in a subset of Hadoop cluster nodes, generating sig-

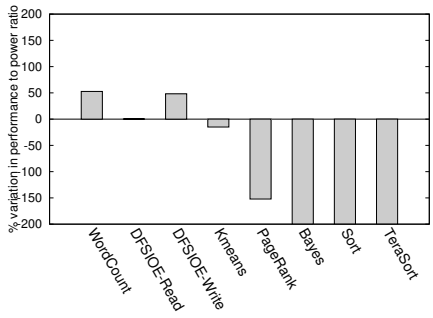

Fig. 9. Performance to Power ratio improvement observed on *Cluster-1* compared to *Cluster-2*.

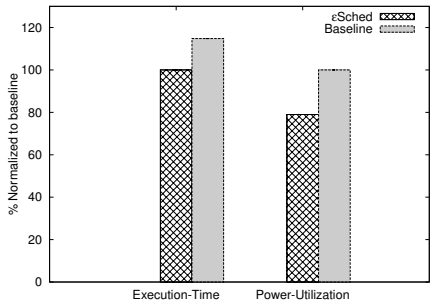

Fig. 10. Power and performance improvement of ϵSched over Hadoop.

nificant periods of idleness enabling these low-activity nodes to operate in a high-energy-saving mode without affecting nodes containing the hot data. This framework increases the skewness in popularity as hot data is concentrated on a subset of nodes, resulting in degraded throughput compared to a setup with hot data spread throughout the entire cluster. Amur et al. [1] and Leverich et al.[22] propose maintaining a primary replica of data on a subset of nodes that are guaranteed to be in active power state and using these nodes as the Hadoop compute nodes, while other replicas of the data are maintained on secondary set of nodes that are in low power modes. These approaches to achieve energy efficiency are based on an underlying assumption that the cluster is always over-provisioned in terms of number of nodes, so that the Hadoop jobs are not affected by the shrinking number of active compute nodes, which may not always be the case.

A number of works, such as GreenHadoop [14] and energy-aware scheduling on MapReduce jobs [33], have focused on reducing the operational cost of data centers. Similarly, Nan Zhu et al. [35] 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 power consumption, our study is novel and different in its focus on evaluating the resources to support Hadoop applications. JouleSort [28] is a software benchmark that considers only the energy characteristics of the applications and schedules the applications.

Chen et al. [8] proposes energy efficiency of MapReduce as a new perspective for increasing MapReduce energy efficiency in particular. They characterize the performance of the Hadoop implementation of MapReduce under different workloads. The authors propose quantitative models using traditional metrics such as job duration. However, to the best of our knowledge ϵSched is the fist workflow scheduler to propose energy efficient scheduling for Hadoop.

VI. Conclusions

In this paper, we design and implement ϵSched, a novel hardware-aware workflow scheduler for Hadoop. We observe that different workflows have different performance and power usage characteristics under varying cluster configurations, and making workflow managers aware of the underlying configuration can significantly increase overall performance and power consumption. We study the impact of ϵSched on Hadoop application performance using a range of representative applications and configuration parameters. Our evaluation shows that ϵSched managing three different clusters can achieve performance improvement of 12.8%, on average, while consuming 21% less power when compared to hardware oblivious scheduling.

Acknowledgment

This work is sponsored in part by the NSF under the CNS1422788 and CNS1405697 grants.

References

[1] H. Amur, J. Cipar, V. Gupta, G. R. Ganger, M. A. Kozuch, and K. Schwan. Robust and flexible power-proportional storage. In *ACM SOCC*, 2010.

[2] G. Ananthanarayanan, A. Ghodsi, A. Wang, D. Borthakur, S. Kandula, S. Shenker, and I. Stoica. Pacman: Coordinated memory caching for parallel jobs. In *Proc. USENIX NSDI*, 2012.

[3] Apache Software Foundation. Hadoop, 2011. http://hadoop.apache.org/core/.

[4] J. L. Berral, Í. Goiri, R. Nou, F. Julià, J. Guitart, R. Gavaldà, and J. Torres. Towards energy-aware scheduling in data centers using machine learning. In *Proceedings of the 1st International Conference on energy-Efficient Computing and Networking*, pages 215–224. ACM, 2010.

[5] D. e. Borthakur. Apache hadoop goes realtime at Facebook. Proc. ACM SIGMOD, 2011.

[6] Y. Chen, S. Alspaugh, and R. Katz. Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads. *Proc. VLDB Endowment*, 5(12):1802–1813, 2012.

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

[8] Y. Chen, L. Keys, and R. H. Katz. Towards energy efficient mapreduce. Technical Report UCB/EECS-2009-109, EECS Department, University of California, Berkeley, Aug 2009.

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

[10] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[11] E. E. Devices. Watts up pro, 2009.

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

[13] G. Fettweis and E. Zimmermann. Ict energy consumption-trends and challenges. In *Proceedings of the 11th International Symposium on Wireless Personal Multimedia Communications*, volume 2, page 6, 2008.

[14] 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.

[15] Y. Hotta, M. Sato, H. Kimura, S. Matsuoka, T. Boku, and D. Takahashi. Profile-based optimization of power performance by using dynamic voltage scaling on a pc cluster. In *IEEE IPDPS*, 2006.

[16] J. Hu and R. Marculescu. Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints. In *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, volume 1, pages 234–239. IEEE, 2004.

[17] S. Huang et al. Hibench: A representative and comprehensive hadoop benchmark suite.

[18] S. Huang, J. Huang, Y. Liu, L. Yi, and J. Dai. Hibench: A representative and comprehensive hadoop benchmark suite. In *Proc. ICDE Workshops*, 2010.

[19] M. Islam, A. K. Huang, M. Battisha, M. Chiang, S. Srinivasan, C. Peters, A. Neumann, and A. Abdelnur. Oozie: towards a scalable workflow management system for hadoop. In *Proc. ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies*, 2012.

[20] R. Kaushik, M. Bhandarkar, and K. Nahrstedt. Evaluation and analysis of greenhdfs: A self-adaptive, energy-conserving variant of the hadoop distributed file system. In *Proc. IEEE International Conference on Cloud Computing Technology and Science*, 2010.

[21] K. Krish, A. Anwar, and A. R. Butt. ϕsched: A heterogeneity-aware hadoop workflow scheduler. In *Proc. IEEE MASCOTS*, 2014.

[22] J. Leverich and C. Kozyrakis. On the energy (in)efficiency of hadoop clusters. In *ACM SIGOPS Operating Systems Review*, 2010.

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

[24] M. Mihailescu, G. Soundararajan, and C. Amza. Mixapart: Decoupled analytics for shared storage systems. In *Proc. USENIX HotStorage*, 2012.

[25] R. Mishra, N. Rastogi, D. Zhu, D. Mossé, and R. Melhem. Energy aware scheduling for distributed real-time systems. In *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, pages 9–pp. IEEE, 2003.

[26] C. Olston, G. Chiou, L. Chitnis, F. Liu, Y. Han, M. Larsson, A. Neumann, V. B. Rao, V. Sankarasubramanian, S. Seth, et al. Nova: continuous pig/hadoop workflows. In *Proc. ACM SIGMOD*, 2011.

[27] J. Peltonen. Energy-aware scheduling. In *Proceedings of Seminar on Energy-Aware Software*, page 7, 2007.

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

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

[30] L. Wang, G. Von Laszewski, J. Dayal, and F. Wang. Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with dvfs. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pages 368–377. IEEE, 2010.

[31] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced cpu energy. In *Mobile Computing*, pages 449–471. Springer, 1996.

[32] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 374–382. IEEE, 1995.

[33] N. Yigitbasi, K. Datta, N. Jain, and T. Willke. Energy efficient scheduling of mapreduce workloads on heterogeneous clusters. In *GCM*, 2011.

[34] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica. Improving mapreduce performance in heterogeneous environments. In *Proc. USENIX OSDI*, 2008.

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