Analysis of Dynamically Switching Energy-Aware Scheduling Policies for Varying Workloads

Pradyumna Kaushik, Akash Kothawale, Renan DelValle, Abhishek Jain, and Madhusudhan Govindaraju

Cloud and Big Data Laboratory, State University of New York (SUNY) at Binghamton

{pkaushi1, akothaw1, rdelval1, ajain13, mgovinda}@binghamton.edu

***Abstract*—Over the past decade, the compute resource requirements of applications has continued to grow. This leads to an increase in the required infrastructure that poses additional challenges to mitigating energy consumption and peak power draws. Apache Mesos has been a dominant player in the cluster management space, acting as a middleware between frameworks and the underlying infrastructure. Due to its high availability, fault-tolerance, and scalability, Mesos is used in the industry to run massive scale workloads. Datacenters typically handle continuous dynamically varying workloads, potentially leading to high peak power and energy consumption, in-turn leading to high cost of operation. Given a set of scheduling policies that have been proven to be beneficial in lowering peak power and/or energy consumption, we propose a mechanism of switching between these policies in order to adapt to the dynamic variation in the workloads and the changes in the state of the cluster. Our experiments show that adapting to the variation in the workload can lead to a 9.2% reduction in max peak power consumption, 6.9% in total energy consumption, and a 6.4% reduction in makespan when compared to using a single scheduling policy.**

I. Introduction

Efficient resource management has become increasingly important as the compute resource requirements and the computation complexities of applications continue to grow at a fast rate. Large-scale datacenters and clouds promote the execution of massive diverse workloads. With the growing availability of academic clouds, providers such as Jetstream [1] and Chameleon [2] have allowed researchers to gain easy access to computing resources, large-scale platforms, and deeply programmable cloud services. In datacenters and clouds, cluster management software such as YARN [3], Borg from Google [4], Torque [5], and Apache Mesos [6] have gained significant traction in the industry. Apache Mesos is currently being used to manage large clusters by industry leaders such as Bloomberg, Twitter, Uber, Ebay, PayPal, Netflix, and Apple. Apache Mesos acts as a middleware between the scheduler and the cluster by abstracting the cluster to be viewed as a single pool of resources [6]. YARN, Mesos, and other similar cluster managers have proven successful for execution of large and diverse workloads, and for co-scheduling of jobs onto worker nodes. However, peak power considerations and energy budgets are not supported by Mesos and YARN in their off-the-shelf packages [7].

Studying the effect that co-scheduled tasks have on peak power draws of compute nodes is important in the development of power-aware scheduling policies. The alignment of peak power draws of worker nodes leads to a phenomena called *Coincidence Peak* where multiple nodes draw high amounts of power in a specific interval of time. During this interval, the cost of powering the datacenter may rise to several times the base rate [8]. Maintaining a low power envelop is therefore important to 1) reduce the cost of powering a datacenter and 2) reduce the net energy demand.

It has been shown that reduction in peak power draws can be achieved by efficiently co-scheduling tasks [9]. The widely used approach is to consider the mapping of tasks, with various power requirements, to nodes with different energy and power profiles, as a multi-dimensional Bin-Packing problem [10]. The collective power usage of these co-scheduled power-intensive tasks is limited by the Thermal Design Power (TDP) of the list of nodes on which they are scheduled to execute. On the contrary, if power-intensive tasks are distributed across all the available nodes in the cluster, the collective power usage of those tasks is now limited by the aggregate of the TDPs of all the corresponding nodes. Thus, Bin-Packing tasks onto nodes can lead to lower peak power draws as compared to distributing the tasks across the cluster. However, Bin-Packing can result in excessive collocation of power-intensive tasks, thereby potentially increasing resource contention, overall makespan, energy consumption and wait times (as tasks take longer to complete and resources are not free).

In our earlier work, we introduced Electron [11], a pluggable power-aware Mesos framework capable of deploying heuristic driven scheduling policies to mitigate peak power and energy draws of heterogeneous clusters. We also showed that prior knowledge of the distribution of tasks in the workload can be used to develop variants of Bin-Packing in order to achieve energy savings [7].

Different scheduling strategies have been developed that have shown to be beneficial in lowering peak power and/or energy consumptions of clusters for certain specific kinds of workloads such as HPC workloads [12], Bag-of-Tasks [13], or for different distributions of power intensive tasks in workloads [7].

Problem Statement. Given a set of scheduling policies that are power/energy efficient for certain specific distributions of power intensive tasks in workloads, we propose a mechanism for switching between these policies in order to adapt to the dynamic variation in the workloads.

II. Mesos & Electron

A. *Mesos*

Mesos is a highly available, fault-tolerant, scalable cluster manager that pools together compute resources such as CPU and memory across all the worker nodes in the cluster and advertises them in the form of *Resource Offers* to Mesos Frameworks.

Mesos consists of masters nodes that each run a *master daemon* process and worker nodes (called agents) that each run an *agent daemon* process. The Mesos agents advertise the available resources, such as CPU, and memory to the Mesos master. The Mesos master then sends this information, in the form of coarse-grained resource offers, to registered frameworks. If the resource offers satisfy the resource requirements of a task (or several tasks), then that offer is consumed and the respective tasks are launched on the host corresponding to the consumed offer. If an offer cannot be consumed by the framework, it is declined. Any unused resources are recycled and sent back to the framework in subsequent resource offer cycles.

B. *Electron - Updated*

Electron was introduced as a pluggable power-aware Mesos framework [11] with built-in scheduling policies to help mitigate peak power and/or energy consumption. We updated the architecture of Electron framework for this work by including a Scheduling Policy Selector. Electron's interactions with Mesos are shown in Figure 1. For this work, the key components of Electron are the following:

- **Task Queue:** Set of tasks that are pending execution.
- **Scheduling Policy Selector:** Responsible for selecting the appropriate scheduling policy to schedule the next set of pending tasks. The Scheduling Policy Selector takes as input a *Switching Criteria*, the Pending Task Queue, and Mesos Resource Offers. A detailed outline of the Scheduling Policy Selector is shown in Figure 2. This module constitutes the following components:
  - **Resource Availability Tracker:** Tracks the clusterwide availability of compute resources such as CPU and memory using Mesos resource offers.
  - **Window Calculator:** Determines the scheduling window based on cluster resource availability and the set of pending tasks (described in Section IV-A1). This window constitutes the set of pending tasks that the next policy schedules.
  - **Policy Selector:** If the *Switching Criteria* is task distribution based, then the Policy Selector selects the most appropriate scheduling policy based on the distribution of low power consuming (L_{pc}) and high power consuming (H_{pc}) tasks in the scheduling window. The tasks are categorized into L_{pc} and H_{pc} using the well known *k-means* clustering algorithm where each task's MMPU values (described in Section V-C) are used as observations. Otherwise, the Policy Selector selects the next scheduling policy based on a Round-Robin ordering.

  When launching Electron, a Scheduling Policy Configuration File (SP-Config) is provided. This file includes the ratio of number of L_{pc} to H_{pc} tasks that each policy is most appropriate for, as well as the switching criteria (*Task Distribution* based or *Round-Robin*).
- **Scheduler:** Receives resource offers from Mesos and forwards them to the selected scheduling policy. Once all the tasks in the scheduling window have been scheduled, the Scheduling Policy Selector module is invoked to select the next policy and the process repeats until there are no more tasks pending execution.

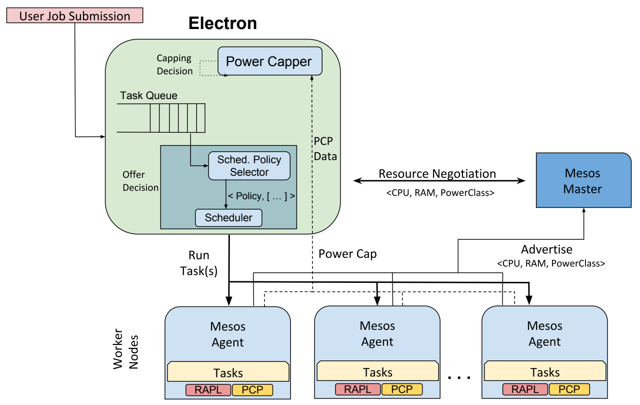


Fig. 1. Architecture of Electron with Apache Mesos. Resource offers are consumed by the dynamically selected scheduling policy.

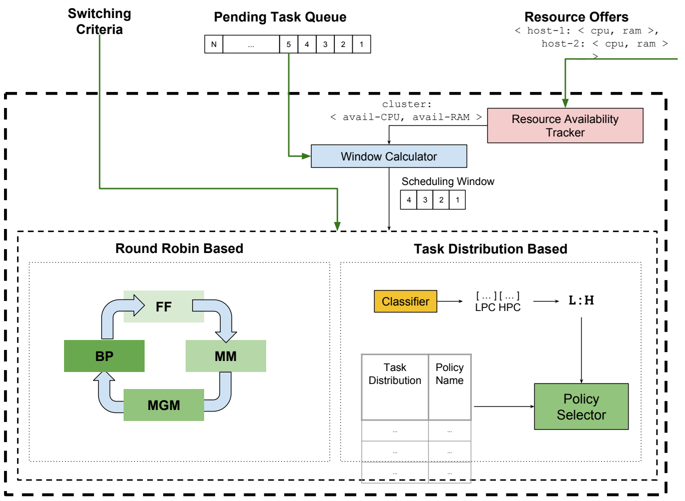


Fig. 2. **Scheduling Policy Selector**: Clusterwide resource availability and the pending tasks are used to determine the scheduling window. Policy selection uses this window to select the next scheduling policy based on the switching criteria (task distribution based or round-robin).

III. Scheduling Policies

In our previous work, we studied the effects of four different scheduling policies, namely First-Fit, Bin-Packing, Max-Min, and Max-GreedyMins, on peak power consumption and total energy consumption for different types of workloads [7]. These policies greatly differ in their task assignment heuristics and the distribution of L_{pc} and H_{pc} tasks across the worker nodes and thus, selecting an appropriate scheduling policy is crucial in making trade-offs between peak power consumption and makespan. For Bin-Packing and its variants, Max-Min and Max-GreedyMins, the tasks are first sorted in non-decreasing order of their estimated power consumption (described in Section V-C).

A. First-Fit (FF)

Tasks are scheduled on a first-come-first-serve basis. If a task fits a resource offer, then that task is scheduled on the node corresponding to the consumed offer. First-Fit results in an even distribution of tasks across the nodes in the cluster. When a majority of the tasks that need to be scheduled are power-intensive, First-Fit evenly distributes them across the nodes in the cluster, thereby reducing the chances of a degradation in performance due to resource contention. Therefore, in such scenarios, policies like First-Fit can greatly help trade-off a reduction in peak power consumption for makespan.

B. Bin-Packing (BP)

Bin-Packing packs as many tasks into an offer as possible, until no more resources are available within the offer. This aggressive nature of collocating tasks facilitates the reduction in peak power consumption across the cluster. Hence, in scenarios where there is a lesser chance of a large degradation in performance due to resource contention, Bin-Packing can help lower peak power consumption while having a minimal impact on makespan.

C. Max-Min (MM)

For each resource offer, tasks are scheduled by alternating between selecting a task from the back of the queue and from the front of the queue. Max-Min's ability to perform early scheduling of power-intensive tasks can help prevent having a residue of all power-intensive tasks towards the tail end of the pending task queue. When the pending task queue has a larger proportion of power-intensive tasks, Max-Min can help lower peak power consumption and yet not have a detrimental impact on makespan.

D. Max-GreedyMins (MGM)

For each offer, Max-GreedyMins alternates between picking one task from the back of the queue and as many as possible from the front of the queue. MGM's ability to schedule L_{pc} tasks at a much faster rate than the H_{pc} tasks, makes it appropriate for scheduling workloads that comprise a larger proportion of L_{pc} tasks.

IV. Scheduling Policy Switching (SPS)

As mentioned in Section I, Cloud service providers typically handle continuously varying workloads. The cluster resource availability (*Cluster State*) and the *Task Distribution* are important factors considered when selecting and switching to a new scheduling policy, illustrated in Figure 3.

A. Factors

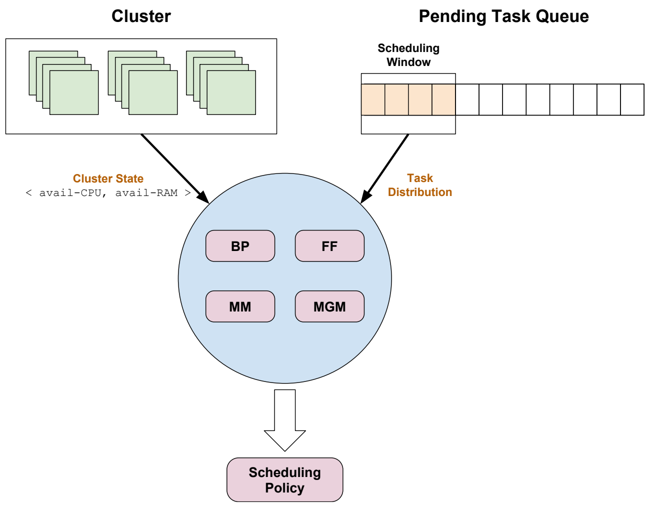


Fig. 3. Factors influencing the scheduling policy selection: *Cluster State* & *Task Distribution*.

1) Cluster State and Scheduling Window: When executing a continuous long queue of tasks, typically only a part of the entire workload in the queue is taken into account. As the state of the current set of pending tasks is volatile, we use a scheduling window to determine the set of tasks that are going to be considered when selecting the next scheduling policy. We make use of the resource offers and task status messages to keep track of the total and current resource availability on the cluster. We sequentially traverse the pending task queue until the aggregate resource requirement of tasks traversed exceeds the total available resources on the cluster. All the tasks that were traversed constitute the scheduling window, as shown in Algorithm 1. Once a policy schedules all the tasks contained in the current window, and there are still tasks pending execution, the scheduling window is re-determined for the next scheduling policy and the process is repeated. A potential limitation with this approach is the increase in average wait times for tasks, as a result of waiting for all the tasks in the current window to be scheduled before scheduling the next set of tasks.

2) Task Distribution: Some scheduling policies are more appropriate to lower peak power consumption, while some others are better to reduce the total energy consumption, given certain distributions of tasks in the workload. In order to mitigate peak power consumption and reduce total energy consumption, trade-offs must be made between co-scheduling power-intensive tasks and spreading them across the nodes in the cluster. Thus, taking the distribution of pending tasks into account will facilitate the selection of a more appropriate scheduling policy.

Algorithm 1 Determining Scheduling Window

```
1: $availRes_{cw}$ ⇐ Clusterwide availability of resources.
2: $tasks_{pndg}$ ⇐ Pending tasks that need to be scheduled
3: $CPU_{agg}$ ⇐ Aggregate CPU requirement of all the tasks
   traversed
4: $MEM_{agg}$ ⇐ Aggregate Memory requirement of all the
   tasks traversed
5: procedure SCHEDWINDOW($availRes_{cw}$, $tasks_{pndg}$)
6:     for task in $task_{pndg}$ do
7:         if CANSCHEDULE(task, $availRes_{cw}$) then
8:             $CPU_{agg}$ += $task$.CPU
9:             $MEM_{agg}$ += $task$.RAM
10:            $schedWindow[...] \leftarrow task$
11:        else
12:            $break$
13:        end if
14:    end for
15: end procedure
16: procedure CANSCHEDULE($task$, $availRes_{cw}$)
17:    $used_{CPU} \leftarrow CPU_{agg}$ + $task$.CPU
18:    $used_{MEM} \leftarrow MEM_{agg}$ + $task$.RAM
19:    if $used_{CPU} \leq CPU_{agg}$ then
20:        if $used_{MEM} \leq MEM_{agg}$ then
21:            return $true$
22:        else
23:            return $false$
24:        end if
25:    else
26:        return $false$
27:    end if
28: end procedure
```

B. Switching Mechanisms

The scheduling policies are stored in a non-decreasing order, keyed by the distribution of tasks that they are most appropriate for (mentioned in the SP-Config file), to aid in a fast lookup and the switching of policies in a round-robin fashion.

1) Dynamic switching (SPS-D): For every switch, the scheduling window is determined based on the total compute resource availability in the cluster. Policy selection is then based on the task distribution. To do this, we first categorize the tasks into Low Power Consuming (L_{pc}) and High Power Consuming (H_{pc}) using the well-known k-means clustering algorithm. The ratio of the number of L_{pc} and H_{pc} tasks is determined to be the task distribution. The task distribution is then mapped to the most appropriate scheduling policy using a modified binary search. In the scenarios where all the tasks get classified into just one cluster, the default scheduling policy is selected. Bin-Packing was set as the default scheduling policy for our experiments as it helps mitigate peak power consumption by aggressively packing tasks onto worker nodes.

2) Round Robin (SPS-RR): The head policy is set to First-Fit, and the tail policy is set to Bin-Packing. Deploying the scheduling policies in a round-robin fashion automatically helps alternate between trading off a reduction in peak power consumption and makespan. When policies closer to the head are selected, we favor a more even distribution of H_{pc} tasks. On the other hand, when policies closer to the tail are deployed, we favor a more aggressive packing of tasks. In SPS-D, the scheduling window will constitute only a small number of pending tasks, when the resource reservation approaches the maximum compute capacity of the cluster. This might affect the scheduling decisions made by the deployed scheduling policy and have a detrimental impact on peak power and energy consumption. To allow a scheduling policy to work with a larger set of tasks, we also fix the size of the scheduling window when launching Electron. For our experiments, we fix the scheduling window to constitute 200 tasks. It is to be noted that this fixed window size resulted in the most beneficial trade-offs between peak power consumption and makespan for our cluster and the variation in our workload.

V. EXPERIMENTAL SETUP

A. Cluster Setup

All of our experiments were conducted on the Cloud and Big Data Laboratory's research cluster named Stratos.

- 2 *Class A* nodes - Two 10 core, 20 thread Intel Xeon E5-2650 v3 @ 2.30GHz and 128 GB RAM
- 1 *Class B* nodes - Two 8 core, 16 thread Intel Xeon E5-2640 v3 @ 2.60GHz and 128 GB RAM
- 1 *Class C* nodes - Two 8 core, 16 thread Intel Xeon E5-2640 v3 @ 2.40GHz and 64 GB RAM
- 4 *Class C* nodes - Two 6 core, 12 thread Intel Xeon E5-2620 v3 @ 2.40GHz and 64 GB RAM

Each of the nodes in the Stratos cluster runs 64-bit Linux 4.7.10 and shares an NFS server. We use Apache Mesos 1.5.0 as our cluster manager. For scheduling our workloads, we use Electron as the sole framework on Mesos. Each of our benchmarks are run inside docker containers to ensure consistency of runtime environments. We use Docker 17.12.0-ce as the container technology. For our experiments, we use Performance Co-Pilot [14] to collect metrics from each node in the cluster. The metrics monitored include CPU and memory usage, and power and energy measurements read from RAPL [15] hardware counters.

B. Power Classes

The nodes in our Stratos cluster were classified into four power classes, based on their Thermal Design Power (TDP). The higher the TDP, the more power (Watts) a processor is expected to consume as per the chip manufacturers. Class A nodes have the highest TDP value, followed by Class B, then Class C, and finally Class D.

Test suites	Description	Type
Audio Encoding	Runtime measurement to encode WAV file to different audio formats.	CPU
Avrora	Multithreaded AVR microcontrollers simulator.	CPU
Batik	Produces Scalable Vector Graphics images.	Memory
Cryptography	Cryptography tests such as OpenSSL and GnuPG.	CPU
DGEMM [16]	Multi-threaded, dense-matrix multiplication.	CPU
Eclipse	Non-GUI jdt performance tests for the Eclipse IDE.	CPU
FOP	Multithreaded, in-memory benchmarks.	CPU
H2	Executes transactions against a model of a banking application.	Memory
Jython	Interprets the pybench Python benchmark.	CPU
luindex	Documents indexer, limited concurrency.	CPU
lusearch	Multithreaded keyword finder.	CPU
miniFE [17]	Finite element generation, assembly and solution for an unstructured grid problem.	CPU
Network Loopback	Computer's networking performance testing.	Network
Pmd	Multithreaded Java source code analysis.	CPU
STREAM [18]	Calculates sustainable memory bandwidth and the computation rate for simple vector kernels.	Memory
Sunflow	Renders a set of images using ray tracing.	CPU
Tomcat	Multithreaded server.	CPU
Tradebeans	Daytrader benchmark run on GERONIMO with an in-memory H2 DB.	Memory
Video Encoding	Video encoding tests, processor tests and system performance testing.	CPU
Xalan	Multithreaded XML to HTML converter.	Mixed

TABLE I
BENCHMARKS

Benchmarks that were used to create the workload.

C. Estimating Power Consumption

For each of the benchmarks listed in Table I, we approximated the power that was going to be consumed when the benchmark was run on any node in the cluster, using the Median of Medians of the Max Peak Power Usage algorithm [7]. We made ten runs on four different nodes in our cluster, each belonging to a different power class. For each power class, the median of the max-peak power usage values (MMPU) for each of the ten runs was considered as the approximate power consumption on that power class. The approximate power consumption of the benchmark was then estimated to be the median of the four MMPU values.

Batch #	L:H Ratio
1	133 : 67
2	172 : 28
3	97 : 103
4	120 : 80
5	200 : 0
6	125 : 75
7	141 : 59
8	56 : 0

TABLE II
WORKLOAD VARIATION

Variation in the task distribution given a batch size of 200 tasks.

D. Workloads

Table I shows the benchmarks that constitute our workload: MiniFE from Mantevo [17], stream and DGEMM from NERSC [16]. Other benchmarks were derived from the Dacapo Benchmark Suite [19] and the Phoronix Benchmark Suite [20]. Using these benchmarks, we create a workload with variation in the concentration of power-intensive tasks. Our workload is made up for 1456 tasks. Assuming a batch size of 200 tasks, the variation in the distribution of L_{pc} and H_{pc} tasks (L:H) is shown in Table II.

VI. PERFORMANCE ANALYSIS

A. Dynamic Switching (SPS-D)

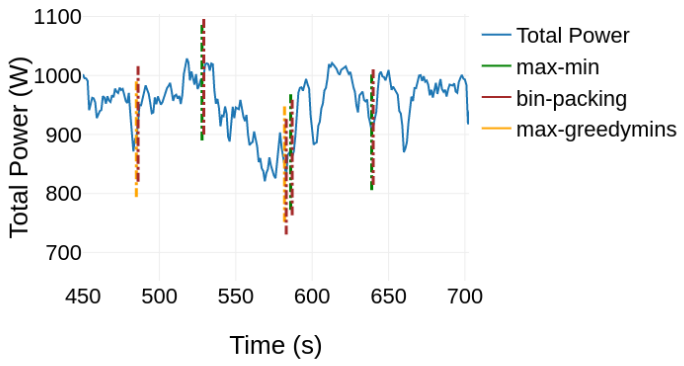
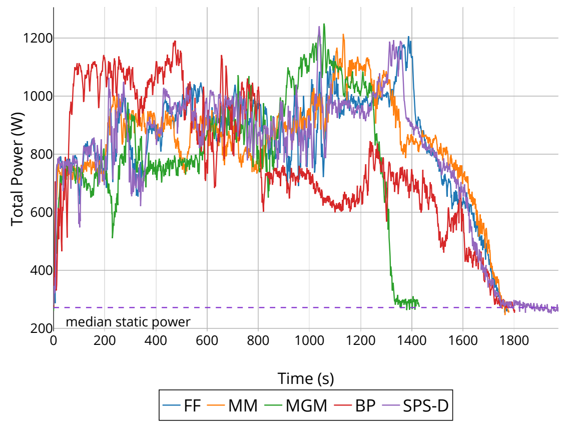
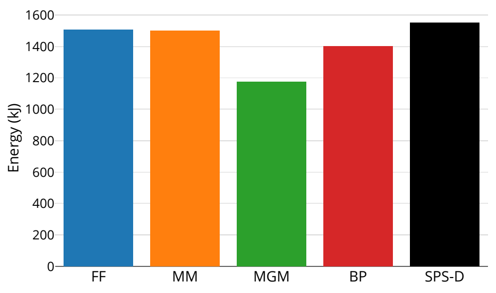


Fig. 4. Illustration showing the switch back to the default scheduling policy (BP) when the tasks in the scheduling window were classified into just one cluster.

Figures 5a and 5b compare the power consumption and total energy consumption when using Dynamic Switching versus just deploying a single scheduling policy.
Power: SPS-D improves in the 90th percentile in power consumption over BP, MM, and MGM, by 89 Watts, 39 Watts, and 90 Watts respectively. This improvement can be attributed to SPS-D being considerate to the distribution of tasks in the scheduling window, thereby reducing peak power consumption between the 800th and the 1250th second. FF has a similar



(a) Power Consumption



(b) Total Energy Consumption

Fig. 5. Figure 5a shows the total power consumption (CPU and DRAM) of Dynamic Switching (SPS-D) and when using just one scheduling policy. The increase in makespan for SPS-D is due to the increase in wait-times, as a result of small scheduling windows. Figure 5b compares the total energy consumption when using SPS-D and when just using a single scheduling policy. The increase in total energy consumption can be attributed to the detrimental impact on makespan, as seen in Figure 5a.

90th percentile power consumption when compared with SPS-D with the latter only improving by 11 Watts. With regard to average power consumption, SPS-D improves over FF, MM, and MGM by 63 Watts, 57 Watts, and 35 Watts respectively. BP, however, shows a similar average power consumption when compared to SPS-D, improving by just about 9 Watts. BP experiences a lower average power consumption as it more aggressively collocates tasks onto worker nodes.

Makespan: SPS-D suffers from a degradation in makespan when compared to the rest of the scheduling policies, as seen in Figure 5a. SPS-D leads to an increase in makespan of 200 seconds, 193 seconds, 543 seconds, and 170 seconds when compared to FF, MM, MGM, and BP respectively. This can be attributed to an increase in wait-times of tasks outside the scheduling window. When the resource reservations approach the maximum capacity of the cluster, the scheduling window constitutes only a few number of tasks. In such a scenario, if the tasks in the scheduling window cannot fit a resource offer, then that offer is declined even though resource requirements of one or more tasks outside might be satisfied by that very same offer. This increases the average number of offer cycles required to launch each task and thus leads to a detrimental impact on makespan. It is important to note that although MGM leads to a noticeable reduction in makespan, the improvement will start to reduce as the variation in the workload and the number of power-intensive tasks increases.

Energy: SPS-D experiences a higher total energy consumption when compared to FF, MM, MGM, and BP of 45.52 kJ, 51.373 kJ, 377.95 kJ, and 150.93 kJ. This can be attributed to the detrimental impact on makespan.

When all the tasks in the scheduling window are similar, the task classification returns only a single cluster of tasks. In such scenarios, SPS-D will immediately switch back to the default scheduling policy (Bin-Packing), as shown in Figure 4, thereby preventing the other scheduling policies from contributing to the reduction in peak power consumption and/or energy consumption. However, it is to be noted that SPS-D is not workload specific and does show comparable results to the some of the other scheduling policies.

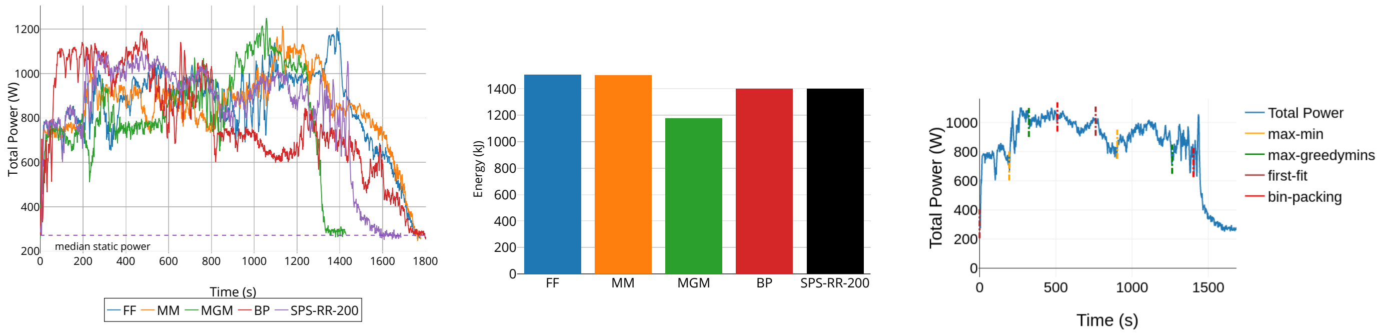
B. Round Robin (SPS-RR)

Figure 6a and 6b compare the total power and energy consumption, respectively, when using a Round-Robin based policy selection (SPS-RR-200) and when using just a single scheduling policy.

Power: SPS-RR-200 experiences an improvement in the 95th percentile of power consumption of 40.89 Watts, 69.04 Watts, and 54.86 Watts over MM, MGM, BP respectively. These improvements are marginal as SPS-RR-200 cycles through trading off peak power consumption and trading off makespan.

Makespan: From the switch trace shown in Figure 6c we can see the power profile of SPS-RR-200 following a wave-like pattern. During the first half of the round-robin cycle, SPS-RR-200 experiences an increase in peak power consumption due to a bias towards makespan. On the other hand, during the second half of the round-robin cycle, from 500th second to approximately 770th second, SPS-RR-200 experiences a reduction in peak power consumption as the deployed policies are more aggressive in packing tasks onto worker nodes. Overall, SPS-RR-200 experiences a noticeable improvement in makespan of 86 seconds, 93 seconds, and 116 seconds over FF, MM, and BP.

Energy: SPS-RR-200 experiences an improvement in the total energy consumption of 104.92 kJ, and 99.06 kJ over FF and MM respectively. In addition, the total energy consumption of SPS-RR-200 is comparable to BP, with BP improving by just 0.49 kJ. Notice that even though the total energy consumption of BP is similar to SPS-RR-200, BP can result in an increase



(a) Power Consumption (b) Total Energy Consumption (c) Switch Trace

Fig. 6. Figures 6a and 6b show the total power consumption (CPU and DRAM) and total energy consumption of Round-Robin based switching (with the scheduling window fixed to 200 tasks) and when using just a single scheduling policy (FF, MM, MGM, or BP). Figure 6c shows the trace of the scheduling policy switching. We can clearly see the policies being selected in a round-robin fashion.

in resource contention, and hence can lead to a detrimental impact on makespan, as is seen in Figure 6a.

VII. Related Work

Energy efficiency in heterogeneous clouds and datacenters is a major problem and has been studied and explored by many researchers in many different ways including, but not limited to, using power capping techniques, analyzing state transitions of compute nodes, turning on/off idle nodes, and bin-packing based task consolidation. However, apart from our previous works done by DelValle et al. [7], [9], [11], we our not familiar with any other work that deals with power and energy optimizations on heterogeneous clusters managed by Apache Mesos.

Hartog et al. [21] used CPU temperature as a metric to estimate the current power consumption of the cluster and developed a MapReduce framework that is capable of dynamically scheduling jobs by adapting to the changes in CPU temperature. In their work, jobs are scheduled on nodes that are operating at temperatures below a user-defined threshold, with the goal of shifting work towards more energy efficient nodes. In this work, we adapt to the variation in the workload and switch to a scheduling policy that is more energy efficient in scheduling the tasks in the pending queue.

Josep Ll. Berral et al. [22] proposed a framework that provides an intelligent consolidation methodology. The authors used a combination of machine learning, turning on/off nodes and power-aware consolidation algorithms to improve power and energy efficiency for different types of workloads such as Grid and Service oriented workloads. In contrast we focus on the variation in the power-intensity of tasks constituting the workload. In addition, shutting down nodes in a large datacenter is not practical, and can impact the availability of the system during periods of high demand. The authors' work also considered minimizing the number of worker nodes that are relatively idle, by migrating the tasks onto other nodes in the cluster. The basic ideology and considerations made in this work is similar to ours. However, in our work, we consider the assignment of tasks to nodes as fixed, as migration can lead to an overhead in network traffic.

R. Ge et al. [23] proposed a distributed performance-directed dynamic voltage scaling (DVS) strategy for power-aware clusters. The authors analyzed the performance of benchmarks for different operating points, such as the Energy Delay Product (EDP) for different CPU frequencies. They then performed DVS scheduling to run nodes at optimal operating points so as to improve energy efficiency. Yang et al. [24] proposed a knapsack based scheduling policy to mitigate energy consumption. Their policy involved scheduling of high power consuming jobs during off-peak periods, and scheduling low power consuming jobs during peak periods. The authors' approach to scheduling high power consuming jobs during off-peak periods can help reduce the possibility of stragglers. However, scheduling all the power-intensive jobs during the off-peak periods can affect peak power consumption. We, however, propose a design for dynamically switching between different scheduling policies to meet energy budgets. The scheduling policy proposed by Yang et. al can be plugged into Electron and then be selected by the Scheduling Policy Selector when appropriate.

H.V Raghu et al. [12] proposed a power-aware algorithm that focuses on energy reduction and load balancing in HPC systems. A knowledge base is constantly updated, with the system resource and energy utilization for each application. Using data from the knowledge base, DVFS knobs are tweaked to optimize makespan and thus reduce total energy consumption. Their work does not consider the impact that co-scheduled applications have on peak power consumption. Although reducing makespan can result in energy savings, the increase in Coincidence Peak power can drive up the running costs of datacenters. In our work, we make trade-offs between reducing peak power and preventing a detrimental impact on makespan.

Chunling Cheng et al. [25] proposed an energy saving task scheduling strategy to cater to the variability in the incoming workload. The authors study the latencies of compute nodes transitioning between running state, idle state, and sleep state. Being sensitive to the variance in service times, a compute node is tagged as either being energy efficient or not. Task scheduling then takes as input the average service times of nodes, deadlines of pending tasks and the transition latencies of compute nodes. On the other hand, we consider the total makespan of a continuously varying workload, and do not consider the latencies of each task.

VIII. CONCLUSIONS AND FUTURE WORK

Given a set of scheduling policies that are proven to be energy efficient for specific kinds of workloads, we introduced and analyzed the feasibility of a mechanism of dynamically switching between these scheduling policies to achieve acceptable results for a varying workload.

Dynamic Switching is able to adapt to the variation in the workload. Although there is a slight increase in energy consumption for our workload, it is to be noted that Dynamic Switching does not require prior knowledge of the workload and in the average case, will perform better than using a single scheduling policy for the entire workload.

Round-Robin based switching results in an alternation between trading off peak power consumption and makespan. Granting that it does not result in the largest reduction in peak power consumption, the trade-offs between peak power and makespan results in an improvement in energy utilization.

A more thorough analysis of the pending queue to make the best use of every offer cycle is subject to experimentation and future work. Currently, the scheduling policy selection does not consider the impact on the already running tasks. A more fine-grained monitoring of cluster state can improve the decision making and is subject to future work.

REFERENCES

[1] C. A. Stewart, D. C. Stanzione, J. Taylor, E. Skidmore, D. Y. Hancock, M. Vaughn, J. Fischer, T. Cockerill, L. Liming, N. Merchant, T. Miller, and J. M. Lowe, "Jetstream," in *Proceedings of the XSEDE16 on Diversity, Big Data, and Science at Scale - XSEDE16*. New York, New York, USA: ACM Press, 2016, pp. 1–8.

[2] "Chameleon Cloud." [Online]. Available: https://www.chameleoncloud.org/

[3] V. K. Vavilapalli, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, E. Baldeschwieler, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, and H. Shah, "Apache Hadoop YARN," in *Proceedings of the 4th annual Symposium on Cloud Computing - SOCC '13*. New York, New York, USA: ACM Press, 2013, pp. 1–16.

[4] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes, "Large-scale cluster management at Google with Borg," in *Tenth European Conference on Computer Systems, EuroSys*. Bordeaux: ACM New York, NY, USA ©2015, 2015.

[5] Garrick Staples, "Toque Resource Manager," in *SC '06 Proceedings of the 2006 ACM/IEEE conference on Supercomputing*. ACM New York, NY, USA ©2006, 2006.

[6] B. Hindman, A. Konwinski, and M. Zaharia, "Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center." *NSDI*, 2011.

[7] R. DelValle, P. Kaushik, A. Jain, J. Hartog, and M. Govindaraju, "Exploiting Efficiency Opportunities Based on Workloads with Electron on Heterogeneous Clusters," in *Proceedings of the10th International Conference on Utility and Cloud Computing*, ser. UCC '17. New York, NY, USA: ACM, 2017, pp. 67–77. [Online]. Available: http://doi.acm.org/10.1145/3147213.3147226

[8] Z. Liu, A. Wierman, Y. Chen, B. Razon, and N. Chen, "Data center demand response: Avoiding the coincident peak via workload shifting and local generation," *Performance Evaluation*, vol. 70, no. 10, pp. 770–791, 2013.

[9] R. DelValle, G. Rattihalli, A. Beltre, M. Govindaraju, and M. J. Lewis, "Exploring the Design Space for Optimizations with Apache Aurora and Mesos," in *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*. IEEE, 6 2016, pp. 537–544.

[10] D. V. Silvano Martello, David Pisinger, "Multi-Dimensional Bin-Packing," *Operations Research*, 2000.

[11] R. DelValle, P. Kaushik, A. Jain, J. Hartog, and M. Govindaraju, "Electron: Towards Efficient Resource Management on Heterogeneous Clusters with Apache Mesos," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*. IEEE, 6 2017, pp. 262–269.

[12] H. V. Raghu, Sumit Kumar Saurav, and Bindhumadhava S. Bapu, "PAAS: Power Aware Algorithm for Scheduling in High Performance Computing," in *IEEE/ACM 6th International Conference on Utility and Cloud Computing (UCC)*. Dresden: IEEE, 2013.

[13] Kyong Hoon Kim, Rajkumar Buyya, and Jong Kim, "Power Aware Scheduling of Bag-of-Tasks Applications with Deadline Constraints on DVS-enabled Clusters," in *CCGrid*, 2007. [Online]. Available: http://www.cloudbus.org/papers/power-aware-scheduling-ccgrid07.pdf

[14] "Performance Co-Pilot," 2016. [Online]. Available: http://www.pcp.io/

[15] "Running Average Power Limit RAPL," 2014. [Online]. Available: https://01.org/blogs/2014/running-average-power-limit--rapl

[16] Sandia National Laboratories, "DGEMM," 2016. [Online]. Available: http://www.nersc.gov/research-and-development/apex/apex-benchmarks/dgemm/

[17] Michael A Heroux, Douglas W Doerfler, Paul S Crozier, James M Willenbring, H Carter Edwards, Alan Williams, Mahesh Rajan, Eric R Keiter, Heidi K Thornquist, and Robert W Numrich, "Improving performance via mini-applications," Sandia National Laboratories, Tech. Rep., 2009.

[18] John D. McCalpin, "Memory Bandwidth and Machine Balance in Current High Performance Computers," *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, pp. 19–25, 1995.

[19] S. M. Blackburn, S. Z. Guyer, M. Hirzel, A. Hosking, M. Jump, H. Lee, J. Eliot, B. Moss, A. Phansalkar, D. Stefanović, T. VanDrunen, R. Garner, D. von Dincklage, B. Wiedermann, C. Hoffmann, A. M. Khang, K. S. McKinley, R. Bentzur, A. Diwan, D. Feinberg, and D. Frampton, "The DaCapo benchmarks," *ACM SIGPLAN Notices*, vol. 41, no. 10, p. 169, 10 2006.

[20] "Phoronix Test Suite - Linux Testing and Benchmarking Platform, Automated Testing, Open-Source Benchmarking," 2017. [Online]. Available: http://www.phoronix-test-suite.com/

[21] J. Hartog, Z. Fadika, E. Dede, and M. Govindaraju, "Configuring a MapReduce Framework for Dynamic and Efficient Energy Adaptation," in *2012 IEEE Fifth International Conference on Cloud Computing*. IEEE, 6 2012, pp. 914–921.

[22] Josep Ll. Berral, Íñigo Goiri, Ramón Nou, Ferran Julià, Jordi Guitart, Ricard Gavaldà, and Jordi Torres, "Towards energy-aware scheduling in data centers using machine learning," in *1st International Conference on Energy-Efficient Computing and Networking*. ACM New York, NY, USA ©2010, 2010.

[23] R. Ge, Xizhou Feng, and K.W. Cameron, "Performance-constrained Distributed DVS Scheduling for Scientific Applications on Power-aware Clusters," in *2005 ACM/IEEE Conference on Supercomputing*. IEEE, 2005.

[24] X. Yang, Z. Zhou, S. Wallace, Z. Lan, W. Tang, S. Coghlan, and M. E. Papka, "Integrating dynamic pricing of electricity into energy aware scheduling for HPC systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '13*. New York, New York, USA: ACM Press, 2013, pp. 1–11.

[25] Chunling Cheng, Jun Li, and Ying Wang, "An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing," in *Tsinghua Science and Technology, Issue 1, Volume 20*. TUP, 2015.