Exploiting Efficiency Opportunities Based on Workloads with Electron on Heterogeneous Clusters

Renan Del Valle  
SUNY at Binghamton  
rdelval1@binghamton.edu 

Pradyumna Kaushik  
SUNY at Binghamton  
pkaushik1@binghamton.edu 

Abhishek Jain  
SUNY at Binghamton  
ajain13@binghamton.edu 

Jessica Hartog  
SUNY at Binghamton  
jhartog1@binghamton.edu 

Madhusudhan Govindaraju  
SUNY at Binghamton  
mgovinda@binghamton.edu

ABSTRACT

Resource Management tools for large-scale clusters and data centers typically schedule resources based on task requirements specified in terms of processor, memory, and disk space. As these systems scale, two non-traditional resources also emerge as limiting factors: power and energy. Maintaining a low power envelope is especially important during Coincidence Peak, a window of time where power may cost up to 200 times the base rate. Using Electron, our power-aware framework that leverages Apache Mesos as a resource broker, we quantify the impact of four scheduling policies on three workloads of varying power intensity. We also quantify the impact of two dynamic power capping strategies on power consumption, energy consumption, and makespan when used in combination with scheduling policies across workloads. Our experiments show that choosing the right combination of scheduling and power capping policies can lead to a 16% reduction of energy and a 37% reduction in the 99th percentile of power consumption while having a negligible impact on makespan and resource utilization.

KEYWORDS

Apache Mesos, Power, Energy, Efficiency, RAPL, Heterogeneous

1 INTRODUCTION

Resource management in large heterogeneous clusters is essential both to effectively use the available resources (such as processors, memory, and storage) and to reduce the cost in terms of the power envelope and energy usage. Apache Mesos [16] has emerged as the leader in the resource management space in the open source community. Mesos is akin to a distributed operating system, pooling together the available cores, system memory, and disk space for consumption by the applications on the cluster. Mesos’ two-level scheduling scheme, along with its fair resource distribution policy, has shown to be successful for massive workload execution. Other efforts, such as Hadoop’s YARN [30], work in a similar manner. These cluster management tools have generated interest in the science and academic computing environments due to the recent availability of virtualized clouds for production use such as JetStream [29] and Chameleon [1]. However, Mesos and YARN do not have support for considering energy budgets and power envelope in their off-the-shelf packages.

Mesos, Hadoop’s YARN, and other tools in this space, are designed to allow co-scheduling of workloads on worker nodes. As tasks are co-scheduled, the power they draw from the node is dependent on how efficiently co-scheduled tasks use hardware resources and how the peak power draws of tasks align. Coincident Power is the total power drawn from the cluster at any time. This value is dependent on the power consumed by all the tasks executing in the same instant. This includes the supporting software stack and hardware components. Ensuring that the cluster’s desired power envelope is not breached requires workload shifting to ensure that the power peaks of various tasks do not align. Maintaining low power usage is especially important during the Coincidence Peak, a window of time where power may cost up to several times the base rate [23].

It is known that optimally co-scheduling applications to minimize peak power usage can be reduced to a multi-dimensional Bin-Packing problem and is NP-Hard [24]. Previously [13], we used Mesos and Aurora [7] to demonstrate how a policy driven approach, involving Bin-Packing workloads, can effectively reduce peak power consumption and energy usage. In our previous work [12], we introduced a pluggable power aware framework for Mesos, Electron. In this work, we deploy Electron with three different workloads, four different scheduling algorithms, and two different power capping strategies to quantify the effects that the different combinations of these three components have on power consumption, total energy consumption, and makespan.

Our workloads are composed of the DaCapo benchmarks [8], Phoronix benchmarks [6], and Scientific micro-benchmarks. The DaCapo benchmark suite is a set of open source, real-world applications that exercise the various resources within a compute node, while the Phoronix and scientific workloads are microbenchmarks. Our approach measures and monitors the power usage of CPU and Memory for each node using fine-grained power profiles provided by Intel’s Running Average Power Limit (RAPL) [11] counters via the Linux Powercapping framework [5]. We use the power profiling data for a given benchmark and determine the approximate power usage.

We make the following contributions in this paper:

- We profile several different, well understood benchmarks and classify them using k-means clustering into low power consuming tasks and high power consuming tasks based on their power consumption.
- In contrast to the single type of workload used in our previous work [12], we use the benchmark classification to construct three kinds of workloads, each varying in power consumption. The different workloads are then used to
quantify the power, energy, and makespan characteristics of the combinations of various scheduling policies and power capping strategies.

- We include two new scheduling policies and analyze their effect on power consumption, energy consumption, and makespan when used to schedule the different categories of workloads.
- We introduce a new power capping strategy to overcome certain limitations of the power capping strategies discussed in our previous work [12] and to further dampen large fluctuations in power consumption.
- We make recommendations based on our findings, on how different scheduling policies and power capping strategies should be used to satisfy Coincident Peak constraints and energy consumption requirements for a Mesos-based cluster.

## 2 MESOS

Apache Mesos is a fast evolving cluster manager capable of providing scalability to data center and cloud applications. Mesos currently powers several cloud infrastructures across the industry. Software such as Apple’s Siri, Bloomberg’s data analytics, and PayPal’s Continuous Integration are built on top of Mesos [2].

Mesos combines resources such as CPU, memory, and storage into a shared pool. Resources from this pool are offered to applications that run on Mesos called frameworks [16]. Frameworks view the Mesos layer as a cluster-wide, highly-available, fault tolerant, and distributed operating system.

Mesos works as follows: (1) A daemon runs on each worker node, known as a Mesos Agent, discovers available resources from the worker and advertises them to the Mesos Master. (2) The Mesos Master makes these resources available to registered frameworks by sending course-grained Resource Offers. (3) A framework can choose to refuse an offer if it does not suit its needs or the framework can choose to use the offer to launch tasks.

### 3 ELECTRON

Electron [12] was built with both pluggable scheduling policies and pluggable power capping strategies. A high level view of Electron’s architecture is shown in Figure 1. Electron is comprised of three main components: Task Queue, Scheduler, and Power Capper.

**Task Queue:** Maintains the tasks that are yet to be scheduled.

**Scheduler:** Checks whether the resource requirements for one or more tasks, in the Task Queue, can be satisfied by the resources available in the Mesos resource offers. If yes, those tasks are scheduled on the nodes corresponding to the consumed resource offer.

**Power Capper:** Responsible for power capping one or more nodes in the cluster through the use of RAPL [11]. The Power Capper monitors the power consumption of the nodes in the cluster which is retrieved through the use of Performance Co-Pilot [4]. A power capping policy that is plugged into Electron uses this information to make the decision to power cap or power uncap one or more nodes in the cluster.

### 3.1 Power Classes

We categorized the machines in our cluster into four power classes: A, B, C, and D based on their Thermal Design Power (TDP). Each node advertises its power class to the framework through the Mesos master. The specifications of the machines belonging to each power class are described in Section 4.1.

### 3.2 Consuming Mesos Offers

The core scheduling logic for a Mesos Framework’s scheduler is defined based on how the framework consumes Mesos Offers. These offer selecting heuristics determine which jobs are co-located, and are therefore key in reducing Coincident Peak power. In this work we compare four such scheduling algorithms.

- **3.2.1 First-Fit (FF).** For each offer the framework receives, it finds the first task in the job queue whose resource constraints are satisfied by the resources available in the offer. If a match is made between an offer and a task, the offer is consumed in order to schedule the matching task. Otherwise, it moves on to a new resource offer and the process of finding a suitable task is repeated.

- **3.2.2 Bin-Packing (BP).** For each offer that is received by the framework, tasks are matched from a priority queue keyed by an approximation of the worst case power consumption using Median of Medians Max Power Usage (M^3PU) (described in Section 4.3). If a task’s resource requirements match with the resources contained in a resource offer, the resources the task will use are subtracted from the available resources in this offer. If a task’s resource requirements are not satisfied by the remaining resources, the next task in the queue is evaluated for fitness. We repeat this process until no task from the queue fits in the remaining resources. The offer is then consumed and the set of tasks evaluated to fit are scheduled to run.

- **3.2.3 Max-Min (MM).** Although Bin-Packing reduces peak power consumption compared to a First-Fit policy, BP commonly
leads to excessive co-location of high power consuming tasks, in
turn increasing resource contention. This increase in resource
contention can potentially result in stragglers and therefore have a
significant impact on makespan. Max-Min is able to address this
issue by picking a mixed set of tasks from the queue. Max-Min
uses a double-ended queue (deque) sorted in non-decreasing MWPU
values, alternating between attempting to schedule tasks from the
front and the back of the deque. If a task fits in the offer, the count
of available resources is reduced and the process is continued until
there are no more tasks that fit in an offer from the deque. Max-Min
then moves on to the next offer.

The distribution of tasks when Max-Min is used as the schedul-
ing policy for a Moderate Power Consuming Workload (described
in Section 4.2) is shown in Figure 2b. Max-Min results in the re-
sources contained in an offer to be consumed in quicker succession,
thereby leading to a better distribution of the workload across the
cluster. Compared to Figure 2a, the workload for Max-Min is better
distributed among the power classes. However, Max-Min does not
show a noticeable improvement in the distribution of high power
consuming tasks when compared to Bin-Packing.

3.2.4 Max-GreedyMins (MGM). Through experimentation
we found that Max-Min has a significant impact on makespan
when there is a higher proportion of low power consuming tasks.
We created Max-GreedyMins (MGM) to counter MM’s impact on
makespan, and to further reduce peak power and energy consump-
tion. MGM consumes offers by packing the low power consuming
tasks at a faster rate than the high power consuming tasks. Like
MM, unscheduled tasks are stored using a deque sorted in non-
decreasing MWPU. MGM, as shown in Algorithm 1, attempts to pack
tasks into an offer by picking one task from the back of the deque
followed by as many tasks as possible from the front of the deque,
stopping when no more tasks can be fit into the offer. At this point,
the policy moves on to the next Mesos offer and repeats the process.

The distribution of the workload when MGM is used as the schedul-
ing policy for a Moderate Power Consuming Workload (described
in Section 4.2) is shown in Figure 2c. When compared against MM,
MGM distributes the workload across the cluster more evenly as
it also spreads out high power consuming tasks amongst power
classes. This improvement in the distribution of high power con-
suming tasks helps reduce starvation and thereby limits negative
effects on makespan.

3.3 Power Capping Policies

3.3.1 Extrema. In our previous work, we presented a dynamic
power capping strategy, Extrema [12], which is able to make trade-
offs between makespan and power consumption. Extrema reacts
to power trends in the cluster and seeks to restrain the power
consumption of the cluster to a power envelope defined by a high
and low threshold. If the cluster’s Average Historical Power (AHP)
exceeds the high threshold, the node consuming the highest power
is power capped to half its Thermal Design Power (TDP). TDP is the
maximum power (turned to heat) that is expected to be dissipated,
as listed by the chip manufacturer. On the other hand, if the cluster’s
AHP is lower than the low threshold, a node is uncapped. Nodes
are fully uncapped in the reverse order in which they were capped.
Extrema is successful in maintaining the power profile within a

```
Algorithm 1 Max-GreedyMins
1: \textbf{sortedTasks} $\leftarrow$ Tasks to schedule sorted in non-decreasing
   order by their corresponding MWPU value.
2: \textbf{offer} $\leftarrow$ Mesos offer.
3: \textbf{procedure} \textsc{Max-GreedyMins}(\textbf{offer}, \textbf{sortedTasks})
4:  for task in \textbf{sortedTasks}.Reverse() do
5:   if \textbf{fit}(\textbf{offer}.\textbf{UnusedResources}(), task) then
6:     \textbf{offer}.\textbf{schedule}(task)
7:     \textbf{sortedTasks}.remove(task)
8:     break
9:   end if
10: end for
11: for task in \textbf{sortedTasks} do
12:   for instance in task.\textbf{Instances}() do
13:     if \textbf{fit}(\textbf{offer}.\textbf{UnusedResources}(), task) then
14:       \textbf{offer}.\textbf{schedule}(task)
15:       \textbf{sortedTasks}.remove(task)
16:     else
17:       break
18:   end if
19: end for
20: end procedure
```

defined envelope, reducing power peaks while having a subdued
impact on makespan.

3.3.2 Progressive Extrema. By observing the outcome of ex-
periments using Extrema we have identified a few of its limitations:

1. If every node in the cluster has already been capped but the
   AHP is still above the High threshold, Extrema is unable
   to perform any new action that may bring the AHP down.
2. As Extrema caps nodes to 50% of their TDP and uncaps
   them to their full TDP, large differences in these values
   can result in nodes experiencing high power fluctuations.
3. Extrema requires high and low thresholds to be manually
   predefined. It follows that prior knowledge of the work-
   load greatly benefits the configuration of the high and low
   thresholds and by extension, the efficacy of Extrema.

While the last drawback still remains an open problem, we address
the first two drawbacks through a modified version of Extrema
named Progressive Extrema. Progressive Extrema, described in Al-
gorithm 2, is similar to Extrema except for one key difference: power
capping is applied in phases. Whereas picking a previously capped
node as a victim in Extrema resulted in a no-op, in Progressive Ex-
trema the same scenario results in a harsher capping value for the
victim. In the initial phase of Progressive Extrema’s design, capping
history is maintained for each node in an in-memory data structure.
When a victim node is chosen for uncapping, the previous cap value
in the node’s history is used. Finally, Progressive Extrema also uses a
\texttt{CapLimit} that defines the floor value beyond which a node should
not be capped.
Figure 2: Distribution of tasks by classification for a Moderate Power Consuming Workload. The number of nodes for each power class is shown in parentheses. Figure 2a shows Class A nodes processing more power intensive tasks than Class D nodes despite having half as many workers when using the Bin-Packing strategy. Figure 2b shows Class D bearing a larger burden than the rest of the classes with Max-Min while Figure 2c shows Class A and Class D completing about the same number of tasks with Class D processing more power intensive tasks.

Algorithm 2 Progressive Extrema Capping

```text
procedure PROGEXTREMA(Threshold, InitCap, CapLimit)
    ClusterAvg ← AvgRunning(ClusterPower)
    if ClusterAvg > Threshold.Hi then
        Victims ← Sortnon-inc(AvgPowerNode[...])
        uncappedVictimFound ← false
        for victim in Victims do
            if victim not in CappedVictims then
                Cap(victim, InitCap)
                CappedVictims[victim.Host] ← InitCap
                uncappedVictimFound ← true
            end if
        end for
        if uncappedVictimFound == false then
            for victim in CappedVictims do
                if victim.curCap > CapLimit then
                    newCap ← victim.curCap ÷ 2
                    Cap(victim, newCap)
                    CappedVictims[victim.Host] ← newCap
                end if
            end for
        end if
    end if
    if ClusterAvg < Threshold.Low then
        victim ← MaxCapped(CappedVictims)
        uncapValue ← CappedVictims[victim.Host] * 2
        Uncap(victim, uncapValue)
        CappedVictims[victim.Host] ← uncapValue
        if victim.curCap == 100 then
            delete(CappedVictims, victim.Host)
        end if
    end if
end procedure
```

4 EXPERIMENTS

4.1 Setup

Our experiments were conducted on a research cluster which comprises the following components:

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

Each node runs a 64-bit Linux 4.4.0-64 kernel and shares an NFS server. Apache Mesos 1.1.0 is deployed as the cluster manager. The Electron framework is used as the sole Mesos framework to schedule workloads. Docker 1.13.1 is used as the container technology. Benchmarks are run inside Docker containers to ensure environment consistency across worker nodes. Performance CoPilot [4] is deployed across the cluster to collect metrics from all worker nodes. Metrics collected from worker nodes include energy measurements from RAPL1 counters and various statistics about CPU and memory usage from each worker node’s Linux kernel. No metrics are collected from our Master nodes as they do not run any workload and thus have limited impact on variable power and energy consumption.

4.2 Workloads

The benchmarks with which we created our Light, Moderate, and Heavy Power Consuming Workloads were derived from the DaCapo Benchmark suite [8], Phoronix Benchmark suite [6], MiniFE from Mantevo [26], and Stream and Dgemm from NERSC [3]. Benchmarks like HiBench[17] were not used as the current focus is only

---

1RAPL only supports monitoring CPU and DRAM. Thus, any references to power and energy should be understood to mean energy consumed by CPU and DRAM.
Table 1: Workload benchmarks

<table>
<thead>
<tr>
<th>Test suites</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio Encoding*</td>
<td>Runtime measurement to encode WAV file to different audio formats.</td>
<td>CPU</td>
</tr>
<tr>
<td>Video Encoding†</td>
<td>Video encoding tests, processor tests and system performance testing.</td>
<td>CPU</td>
</tr>
<tr>
<td>Cryptography†</td>
<td>Cryptography tests such as OpenSSL and GnuPG.</td>
<td>CPU</td>
</tr>
<tr>
<td>Network Loopback*</td>
<td>Computer’s networking performance testing.</td>
<td>Network</td>
</tr>
<tr>
<td>Avrorra*</td>
<td>Multithreaded AVR microcontrollers simulator.</td>
<td>CPU</td>
</tr>
<tr>
<td>Batik*</td>
<td>Produces Scalable Vector Graphics images.</td>
<td>Memory</td>
</tr>
<tr>
<td>Eclipse*</td>
<td>Non-GUI jdt performance tests for the Eclipse IDE.</td>
<td>CPU</td>
</tr>
<tr>
<td>Jython*</td>
<td>Interprets the pybench Python benchmark.</td>
<td>CPU</td>
</tr>
<tr>
<td>Pmd†</td>
<td>Multithreaded Java source code analysis.</td>
<td>CPU</td>
</tr>
<tr>
<td>Tradebeans*</td>
<td>Daytrader benchmark run on GERONIMO with an in-memory H2 DB.</td>
<td>Memory</td>
</tr>
<tr>
<td>H2*</td>
<td>Executes transactions against a model of a banking application.</td>
<td>Memory</td>
</tr>
<tr>
<td>Xalan†</td>
<td>Multithreaded XML to HTML converter.</td>
<td>Mixed</td>
</tr>
<tr>
<td>Sunflow†</td>
<td>Renders a set of images using ray tracing.</td>
<td>CPU</td>
</tr>
<tr>
<td>miniFE[26]*</td>
<td>Finite element generation, assembly and solution for an unstructured grid problem.</td>
<td>CPU</td>
</tr>
<tr>
<td>DGEMM[27]†</td>
<td>Multi-threaded, dense-matrix multiplication.</td>
<td>CPU</td>
</tr>
<tr>
<td>STREAM[18]*</td>
<td>Calculates sustainable memory bandwidth and the computation rate for simple vector kernels.</td>
<td>Memory</td>
</tr>
</tbody>
</table>

Table 4.1: Workload benchmarks

The † symbol indicates a High Power Consuming benchmark while the * symbol indicates a Low Power Consuming benchmark as determined through profiling and k-means clustering.

4.3 Median of Medians Max Power Usage

There are many ways of calculating a suitable global value to be used as an estimation of the power consumption for each benchmark. For this set of experiments we opted to use the Median of Medians of the Max Power Usage (M3PU) value for each benchmark as an approximation of the power consumption in our workloads (described in Algorithm 3).

Algorithm 3 Median Median Max Power Usage (M3PU)

1: \( R \leftarrow \) Number of individual runs.
2: \( P \leftarrow \) Power.
3: \( \text{Peaks} \leftarrow \) Power peaks per run.
4: \( \text{PC} \leftarrow \) Power Classes.
5: \( \text{procedure} \ M3PU(Benchmarks[...], PC[...]) \)
6: \( \text{for} \ bm \ \text{in} \ Benchmarks \text{do} \)
7: \( \text{MMPU} \leftarrow \text{List}(\) \)
8: \( \text{for} \ pc \ \text{in} \ PC \text{do} \)
9: \( \text{peaks} \leftarrow bm.\text{getPeaks}(pc) \)
10: \( \text{mmpuPc} \leftarrow \text{BENCHMARK-MMPU}(\text{peaks, pc}) \)
11: \( \text{MMPU}[... \] ⩾ mmpuPc \)
12: \( \text{end for} \)
13: \( M3PU[\text{bm}] \leftarrow \text{Median}(\text{MMPU}[...]) \)
14: \( \text{end for} \)
15: \( \text{end procedure} \)
16: \( \text{procedure} \ \text{BENCHMARK-MMPU}(\text{Peaks}[R][P], PC) \)
17: \( \text{MaxPeaks} \leftarrow \text{List}(R) \)
18: \( \text{for} \ i \ \text{in} \ 0 \ \text{to} \ R-1 \text{do} \)
19: \( \text{MaxPeaks}[i] \leftarrow \text{MaxPeak}([\text{Peaks}[1]) \)
20: \( \text{end for} \)
21: \( \text{return} \ \text{Median}([\text{MaxPeaks}) \cdot \text{StaticPower}_{PC} \)

Since our cluster is heterogeneous, the estimated values varied between machines belonging to the four different power classes described in Section 4.1. For each benchmark, ten profiling runs were recorded on four nodes, one for each class. The max peak was found for each of the ten runs. Each power class then had ten max peaks from which the median was calculated. From the median we subtracted the median static power for each power class, generating a Median Max Power Usage (M3PU) of the benchmark for each power class. We used these values as an approximation of the worst case power consumption of the benchmark on any node in that power class. The four MMPU values were used as observations for our task classification described in Section 4.4.

In order to be able to build the data structures required for our scheduling policies we required a single value as a point of comparison for sorting. We opted to use the Median of the four MMPU values which represents a cluster-wide central tendency of power usage for each benchmark, resulting in a Median of Medians Max Power Usage (M3PU) for each benchmark.

4.4 Task Classification

Using the well known k-means clustering algorithm with the four MMPU values of each benchmark as observations, we classified benchmarks into two categories: low power consuming and high power consuming. As a benchmark can be scheduled on any node in the cluster, the power consumption of the benchmark on different power classes needs to be considered. For this reason, the MMPU values for each power class were used as the observations instead of the global approximation M3PU value. The classification of our benchmarks can be seen in Table 1. Using this classification we created three kinds of workloads: Light, Moderate, and Heavy. Each workload has a different ratio of low power consuming tasks to
high power consuming tasks: Light (20:3), Moderate (10:6), and Heavy (5:12).

5 PERFORMANCE ANALYSIS
In this section, we analyze the performance of different strategies for consuming Mesos offers: First-Fit (FF), Bin-Packing (BP), Max-Min (MM), and Max-GreedyMins (MGM). We also study the effect of two power capping strategies, Extrema and Progressive Extrema, when used with each scheduling policy. To further discover strengths and weaknesses of a combination, we run three workloads: Light, Moderate, and Heavy. We quantify each combination of scheduling policy, power capping strategy, and class of workload based upon the following aspects: (1) ability to reduce peak power consumption, (2) ability to reduce energy consumption, and (3) impact on makespan.

5.1 Performance of Scheduling Policies
Tables 2, 3, and 4 compare the energy, makespan, and the 95th percentile in power consumption, for different scheduling policies for a Light, Moderate, and Heavy Power Consuming Workload respectively.

5.1.1 Light Power Consuming Workload. Figure 3 shows the power profiles of the execution of the Light Power Consuming Workload (LPCW) when using the previously mentioned scheduling policies.

<table>
<thead>
<tr>
<th>Power (W)</th>
<th>FF</th>
<th>BP</th>
<th>MM</th>
<th>MGM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power (W)</td>
<td>1072.75</td>
<td>1033.1</td>
<td>1039.2</td>
<td>1094.8</td>
</tr>
<tr>
<td>M k espan (s)</td>
<td>1319</td>
<td>1417</td>
<td>1544</td>
<td>1122</td>
</tr>
<tr>
<td>Energy (kJ)</td>
<td>858</td>
<td>844.6</td>
<td>761.9</td>
<td>793</td>
</tr>
</tbody>
</table>

Table 2: Comparison of the effects of different scheduling policies for a Light Power Consuming Workload.

5.1.2 Moderate Power Consuming Workload. Figure 4 shows the power profiles of the execution of the Moderate Power Consuming Workload (MPCW) when using the previously mentioned scheduling policies.
### Table 3: Comparison of the effects of different scheduling policies for a Moderate Power Consuming Workload.

<table>
<thead>
<tr>
<th>Power (W)</th>
<th>FF</th>
<th>BP</th>
<th>MM</th>
<th>MGM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1032.9</td>
<td>957</td>
<td>980</td>
<td>1049.4</td>
<td></td>
</tr>
<tr>
<td>Makespan (s)</td>
<td>1521</td>
<td>1602</td>
<td>1575</td>
<td>1450</td>
</tr>
<tr>
<td>Energy (kJ)</td>
<td>1183.8</td>
<td>1031</td>
<td>918.5</td>
<td>891.7</td>
</tr>
</tbody>
</table>

### Table 4: Comparison of the effects of different scheduling policies for a Heavy Power Consuming Workload.

<table>
<thead>
<tr>
<th>Power (W)</th>
<th>FF</th>
<th>BP</th>
<th>MM</th>
<th>MGM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1098.2</td>
<td>1006.5</td>
<td>1110.7</td>
<td>1028.2</td>
<td></td>
</tr>
<tr>
<td>Makespan (s)</td>
<td>1630</td>
<td>1683</td>
<td>1626</td>
<td>1697</td>
</tr>
<tr>
<td>Energy (kJ)</td>
<td>1546.4</td>
<td>1380.1</td>
<td>1259.1</td>
<td>1226.9</td>
</tr>
</tbody>
</table>

**Power:** Table 3 shows an improvement in the 95th percentile of power consumption for BP when compared to FF, MM, and MGM. However, MGM experiences a substantial reduction in the 90th percentile (not shown in the table) of power consumption, improving over FF, BP, and MM by 304 Watts, 88.8 Watts, and 39.8 Watts respectively. This discrepancy between 90th and 95th percentile in power consumption for MGM can be attributed to early execution of high power consuming tasks, leading to a high initial spike of power consumption. Throughout the rest of execution, MGM maintains a lower power profile in comparison to the other scheduling policies.

**Makespan:** BP suffers an increase in makespan when compared to FF, MM, and MGM, which can be attributed to BP co-locating several high power consuming tasks late in the task allocation process. This excessive co-location leads to an increase in contention for resources, thus increasing the completion times for these high power consuming tasks. MGM’s reduction in makespan can be attributed to better distribution of high power consuming tasks across the worker nodes.

**Energy:** Although BP reduces the power envelope, the increase in makespan reduces the impact it has on the energy consumption due to the static power penalty. However, BP still consumes 152.8 kJ less than FF. On the other hand, MM and MGM are able to achieve a more heterogeneous mix of low power consuming and high power consuming tasks, thereby reducing energy consumption. As MGM results in a further increase in the distribution of high power consuming tasks across the cluster, it experiences a 26.6 kJ reduction in energy consumption when compared to MM.

**5.1.3 Heavy Power Consuming Workload.** Figure 5 shows the power profiles of execution of the Heavy Power Consuming Workload (HPCW), using various scheduling policies.

**Power:** As the HPCW contains an increased number of high power consuming tasks, we can see a clear increase of power envelopes for all the policies. Table 4 shows that BP experiences a substantial reduction in the 95th percentile of power consumption when compared to FF. Furthermore, BP is better than MM by 104.2 W in the 95th percentile of power consumption. Although MM improves the distribution of tasks across the cluster, it does not have a substantial impact in reducing the excessive co-location of high power consuming tasks for this power intensive workload. MGM, however, shows an improvement in the 95th percentile when compared to FF, BP, and MM, and this reduction can be attributed to a decrease in the co-location of high power consuming tasks, leading to a reduction in coincident peaks.

**Makespan:** Although MGM shows an improvement in power consumption, it delays the start time of execution of the high power consuming tasks. This increase in latency for power intensive tasks leads to an increase in makespan, as seen in the data shown in Table 4. In addition, as the workload gets more power intensive, MGM’s detrimental impact on makespan might become more prominent. MGM shows a similar makespan to BP, posting just a 14 second difference, but the increase in makespan for BP can be attributed to resource contention of excessively co-located high power consuming tasks.

**Energy:** BP experiences a significant reduction in energy consumption when compared to FF, as seen in Table 4. Furthermore, MM and MGM experience a decrease in energy consumption when compared to BP. Although MM experiences a slight increase in energy consumption of around 32 kJ when compared to MGM, MM would be a more appropriate choice when scheduling a higher ratio of high power consuming tasks as MGM is more likely to have a negative impact on makespan as the ratio of high power consuming benchmarks to low power consuming benchmarks increases. Eventually, the increase in makespan would lead to the static power penalty nullifying energy decrease from maintaining a lower power envelope.

### 5.2 Power Capping Impact On Scheduling

In this section we quantify the impact of our set of Power Capping Strategies {Extrema, Progressive Extrema} across our set of different scheduling policies \{FF, BP, MM, MGM\}.

#### 5.2.1 Extrema

Figure 7 shows the effect of using the Extrema power capping strategy when used alongside different scheduling policies for the Light, Moderate, and Heavy Power Consuming Workloads (LPCW, MPCW, and HPCW respectively).

**Power:** Figure 7a shows the power profiles when Extrema is run
Figure 6: Figure 6a shows the comparison of Cluster-wide Energy Consumption of different scheduling policies when used to schedule different classes of workloads (Light, Moderate and Heavy). Figures 6b & 6c compare Cluster-wide Energy Consumption of different scheduling policies in combination with the Extrema and the Progressive Extrema power capping policies for scheduling different classes of workloads.

Figure 7: Cluster-wide Power Consumption for Extrema Capping for Light, Moderate and Heavy Power Consuming Workloads.

Figure 8: Cluster-wide Power Consumption for Progressive Extrema Capping for Light, Moderate and Heavy Power Consuming Workloads.

alongside the different scheduling policies for the LPCW. Compared to their uncapped runs FF, BP, MM, and MGM experience a reduction in the p95 of power consumption of 231 W/t-ts, 190 W/t-ts, 193 W/t-ts, and 251 W/t-ts, respectively, when running under the Extrema capping policy. For the MPCW, shown in Figure 7b, FF, BP, and MGM experience an improvement of 220 W/t-ts, 74 W/t-ts, and...
7 Watts, respectively, in the 90th percentile of power consumption compared to the uncapped runs. On the other hand, MM with Extrema experiences an increase of 23 Watts in the 90th percentile of power consumption for the MPCA compared to its uncapped run. The effect of Extrema on MM is likely due to the size of tasks in the queue and the time Extrema needs to place power caps on each node. When all higher energy jobs remain in the queue after the lower energy jobs have been exhausted a higher percentage of nodes are capped and remain capped for a longer period of time. Further investigation into the exact circumstances that lead to a power increase in this instance is the subject of future work. Finally, when scheduling the HPCA the scheduling policies FF, BP, MM, and MGM show an improvement in the 90th percentile of power consumption (relative to their uncapped runs) of 118.5 Watts, 94.2 Watts, 116 Watts, and 35 Watts respectively.

Makespan: Extrema does not impact the makespan of FF as the workload is well distributed across the cluster. However, when Extrema is used alongside BP, the makespan is increased by 128 seconds, 67 seconds, and 76 seconds for LPCW, MPCA, and HPCA respectively. The makespan is not affected when Extrema is deployed on MM for the LPCW and MPCA. On the other hand, when Extrema and MM are used for the HPCA it experiences an increase in makespan of 94 seconds. This increase is due to RAFL lowering CPU clock speeds to stay within the power budget which has an adverse effect on the larger number of high power consuming tasks in contention for system resources. For the combination of Extrema and MGM there is an increase in makespan of 236 seconds for the LPCW and 179 seconds for the MPCA. This indicates that Extrema combined with MGM is not a good fit for systems that want to maintain a high Service Level Agreement (SLA) for processing LPCW and MPCA. There is no impact on the makespan when Extrema is combined with MGM for the HPCA as shown in Figure 7c as MGM is better at distributing the high power consuming tasks across the cluster compared to BP and MM.

Energy: In general, Extrema's reduction in peak power consumption is much more prominent than Extrema's increase in makespan. This leads to Extrema lowering the energy consumption for FF, BP, MM, and MGM for MPCA and HPCA. Figure 6b shows that when Extrema is used for the MPCA, the scheduling policies FF, BP, and MM experience a reduction in energy consumption of 171$k$, 53$k$, and 159$k$ respectively when compared to their uncapped runs while MGM experiences an increase of 77$k$ in energy consumption compared to its uncapped run. When Extrema is used for the HPCA, our results show that compared to their uncapped runs FF, BP, and MM experience a reduction of 191$k$, 73$k$, and 25$k$ respectively, while MGM experiences an increase of 24$k$. The increase in energy consumption for MGM for MPCA as well as HPCA can be attributed to the delayed start time for the high power consuming tasks, thus incurring a heavier static power penalty.

5.2.2 Progressive Extrema. The graphs in Figure 8 show the effect of using the Progressive Extrema power capping strategy alongside the scheduling policies described above for the Light (LPCA), Moderate (MPCA), and Heavy (HPAC) Power Consuming Workloads.

Power: Compared with the uncapped runs and against the Extrema capped runs Progressive Extrema reduces the initial power draw on a cluster. The initial power peaks are around 1200 Watts for the uncapped runs, around 830 Watts for the Extrema power capped runs, and have been reduced to around 600 Watts for the LPCW and MPCA as shown in Figures 8a and 8b. This reduction can be attributed to Progressive Extrema being able to more aggressively cap the already capped nodes and thereby quickly bringing down the power envelope closer to the predefined high and low thresholds. Analyzing the results shown in Figure 8, we observe that when compared to their corresponding uncapped runs, shown in Figures 3, 4, and 5, FF, BP, MM, and MGM experience a substantial reduction in the p95 of power consumption. FF experiences a p95 reduction in power consumption of 368, 374, and 141 Watts for LPCW, MPCA, and HPAC respectively. BP experiences a p95 reduction in power consumption of 405, 297, and 133 Watts for the LPCW, MPCA, and HPAC respectively. MM experiences a p95 reduction in power consumption of 457, 260, and 251 Watts for the LPCW, MPCA, and HPAC respectively. Finally, MGM experiences a p95 reduction in power consumption of 387, 401, and 155 Watts for the LPCW, MPCA, and HPAC respectively.

Makespan: Progressive Extrema has a negative impact on makespan as it is aggressive in power capping the nodes and therefore decreases CPU resources available to workloads. For the LPCW, FF, MM, and MGM experience an increase in makespan of 394, 293, and 396 seconds, respectively, when compared to their corresponding energy consumptions shown in Figure 6a. However, BP does not experience a significant impact on makespan for the LPCW when compared to its uncapped run. FF, BP, MM, and MGM experience an increase in makespan of 336, 53, 189, and 341 seconds, respectively for the MPCA when compared to their corresponding uncapped runs. BP and MM experience an increase in makespan of 102 and 52 seconds, respectively, for the HPAC. As FF and MGM distribute the high power consuming tasks more evenly, they do not experience an impact in makespan for the HPAC due to the decreased availability of CPU resources.

Energy: Progressive Extrema proves to be beneficial in significantly reducing the power envelopes and the power fluctuations across all runs. For the HPAC, Progressive Extrema reduces energy consumption for FF, BP, MGM, and MM by 120$k$, 37$k$, 45$k$, and 83$k$ respectively, when compared to their uncapped runs. For the MPCA, all but MGM experience a reduction in energy consumption. FF, BP, and MM improve by 126$k$, 85$k$, and 33$k$ respectively. However, for the LPCW the significant impact on makespan leads to Progressive Extrema not having a substantial improvement in energy consumption when compared to uncapped runs with BP being the only exception where Progressive Extrema reduces the energy consumption of BP by 83$k$.

6 RELATED WORK

Apart from the authors’ initial work [12], we are not aware of any other work that discusses power and energy optimization for Mesos based clusters. There exist several complementary publications that focus on energy management via microarchitecture level optimizations, turning off or suspending nodes, and using Dynamic-Voltage and Frequency-Scaling (DVFS) schemes.

In [20, 22], the cluster is divided into hot and cold zones and a Covering Subset is determined that has at least one replica of
each data block, and then nodes that can be suspended or shut down are chosen. Lang and Patel [21] identify when it is feasible to keep nodes off-line and boot them up once a job arrives. These approaches are not practical in Data Centers, which is where Apache Mesos is primarily used, as they negatively affect the responsiveness of the system for unexpected workloads [25].

Prior to the adoption of Mesos in the Apache community and the readily available nature of RAPL[11], Hartog et al. [14] leveraged the unified nature of MapReduce job scheduling to study scheduling of tasks in a cluster. To achieve this power consumption was constrained by scheduling using the constraint of keeping the CPU within a temperature range. In this work, we leverage the unified scheduler of Aurora on Mesos and RAPL metrics, in order to employ a similar algorithm which takes into account the power and energy consumption of the entire cluster. The goals here is also to maintain the power envelope within defined thresholds. Hartog et al. [15] also focus on fine grained task splitting in a MapReduce setup by using smaller subtasks to increase the opportunities to react to clusters with heterogeneous processing nodes, Similarly, we take into account the size of the tasks that we are trying to schedule and focus our scheduling based around task size.

Bodas et al. [9], developed a power-aware scheduler for SLurm that is based on monitoring the power usage and then using a uniform frequency mechanism to limit power. Just like our approach, they advocate for a policy-driven approach for high and low power consumers. Our work however, assumes that CPU frequency scaling mechanisms based on individual benchmarks are not practical for co-scheduled workloads in virtualized clusters.

Karakoyunlu et al. [19] developed a 2-approximation solution for minimizing energy consumption and balancing system load. Their schemes (Fixed Scheme, Dynamic Greedy Scheme, and Correlation-Based Scheme), are designed to link cloud storage needs to the cloud users. These policies can be adapted depending on the priorities set by the cloud storage system. Our work takes a similar approach but our policies are applied to affect the scheduling on Mesos clusters using CPU, memory, and power usage.

Sarood et al. [28], propose a software-based online resource management system and a scheme that uses an adaptive runtime system that can dynamically change the resource configuration of a running job. As a result resources allocated to currently executing jobs can be modified. While this work also allows power capping, it assumes job malleability such that resources can be dynamically changed for a running job. We assume that the resources allocated to a job cannot be changed as that is a tenet of Mesos’s design. In [10], Chandraseka et al. present the Power-check framework, which uses smart data-funneling mechanisms along with power capping to reduce the CPU utilization during the I/O phase of checkpointing applications. In comparison, we apply dynamic power-capping during all phases to maintain a power envelope.

7 CONCLUSION

There are trade-offs that must be made in order to operate a cluster under specific power and energy constraints. Some policies favor lowering the height of power peaks in the cluster, others favor a reduced makespan, while still others favor lower energy consumption and Bin-Packing in the metrics analyzed in this work, we acknowledge these scheduling policies may cause starvation for tasks in workloads that are between Light and Heavy Power Consuming. Guarding against task starvation is not addressed in this paper and will be the subject of future work. We have developed a few heuristics to enable Mesos-powered clusters to schedule with the aim of avoiding Coincident Peak power consumption, decreased power consumption, and decreased energy consumption while avoiding a large increase in makespan.

• Max-GreedyMins should be used when a workload requires high throughput and the scheduling time does not fall within the Coincidence Peak, regardless of the power intensity of the workload.

• When the workload consists of a higher proportion of high power consuming tasks, scheduling policies similar to Max-Min, are an appropriate choice. On the other hand, if the workload consists of a higher proportion of low power consuming tasks, then scheduling policies similar to Max-GreedyMins, are a more appropriate choice.

• To decrease power peaks and energy consumption, Extrema is best deployed as the power capping strategy to schedule Light and Moderate Power Consuming Workloads while Progressive Extrema is best suited for Heavy Power Consuming Workloads.

In our future work, we plan to develop a policy capable of switching between different scheduling policies and power capping strategies in order to adapt to a continuously changing workload.

REFERENCES


