# Exploring the Potential of using Power as a First Class Parameter for Resource Allocation in Apache Mesos Managed Clouds

Pradyumna Kaushik, Srinidhi Raghavendra, Madhusudhan Govindaraju
*Cloud and Big Data Lab*
*Binghamton University, SUNY, USA*
{*pkaushi1,sraghav2,mgovinda*}*@binghamton.edu*

Devesh Tiwari
*Dept. of Electrical and Computer Engineering*
*Northeastern University, USA*
*d.tiwari@northeastern.edu*

***Abstract*—We propose a resource allocation policy that uses (a) Power as a first class parameter as an indicator of the computational intensity of a task and its potential impact on peak power draw, and (b) Power Tolerance as an indicator of a task's sensitivity towards degradation of performance as a result of resource contention. Through experimentation and analysis, we present coarse-grained and fine-grained Power Tolerance assignment methods that can be employed to make smarter peak power performance trade-offs. Our experiments show that (a) cloud operators can benefit from a uniform workload-wide setting of Power Tolerance to achieve significant reduction in peak power consumption, (b) fine-grained Power Tolerance assignment methods show potential in making smarter peak power and performance trade-offs.**

***Keywords*-Power-Aware, Apache Mesos, Resource Contention, Resource Allocation, Peak Power, Power Tolerance**

## I. Introduction

With the proliferation of virtualization technologies and the adoption of container technology in the industry and academic clouds, execution of many application instances on a single physical host (termed as co-location) has become the norm. Academic cloud providers such as Chameleon [1] and Jetstream [2] provide easy access to large-scale compute resources and programmable cloud services. Cloud resource management software such as Google's Borg [3], Apache Mesos [4], and Hadoop's YARN [5] have proven successful in improving resource utilization and have widely been incorporated in the industry to execute diverse workloads on large heterogeneous clusters. However, these resource management software do not make peak power considerations when co-locating applications [6]. Executing heterogeneous workloads poses challenges in mitigating peak power consumption of clusters. Large influx of jobs with compute intensive applications can quickly ramp up the peak power draw leading to periods of *Coincidence Peak*, a phenomenon where multiple nodes in a datacenter draw large amounts of power in quick succession, *increasing the total cost of powering the datacenter orders of magnitude over the base rate [7].*

While Bin-Packing solutions have been known to improve resource efficiency [8], excessive co-location can lead to performance degradation as a result of resource contention [9]. Although energy efficiency techniques proposed earlier in the literature [10], [11], [12] can be exercised, they do not address peak power concerns. Such energy efficiency techniques can, however, still be applied in addition to the work proposed in this paper.

When co-locating applications it is important to ensure that (a) surges in peak power draw are contained, (b) applications do not affect performance of one another by dominating CPU time via the *noisy neighbor* phenomenon and (c) when contention for resources is low, applications have the ability to consume idle CPU cycles. A task's potential impact on peak power draw of a machine and its sensitivity towards resource contention are key pieces of information that help factoring the above mentioned criteria into resource allocation. Currently, the popular open-source frameworks built on Mesos [13] do not provision for ways to specify the above information when submitting jobs.

The power consumed by a machine as a result of executing a task can be used to indicate the task's potential impact on peak power draw. **We thus introduce *Power* as a first class parameter to represent the computational intensity of a task and in turn connote its potential impact on peak power draw**. Although the resource allocation of a task can be updated, it comes with additional overhead of performing checkpoint restart or migration and is out of scope of this work. In this work, we consider a task to be non-malleable in the context of traditional resources such as CPU and memory. The amount of computation performed by a task influences its sensitivity towards resource contention. **We term a task's sensitivity towards co-location and in turn resource contention as *Power Tolerance***. Power, unlike CPU and memory, is considered a malleable resource and the adjustment is facilitated by Power Tolerance. In our work, Power Tolerance is interpreted as the acceptable percentage reduction of the requested Power that can be allocated to a task (discussed later in Section II-B). Power Tolerance can be used to regulate the extent of co-location to prevent performance degradation while attempting to mitigate peak power consumption. Similar to traditional resources such as CPU and memory, Power and Power Tolerance are included as part of tasks' resource requirements.

The work presented in this paper will allow managers of cloud allocations to provide informed values on the Power Tolerance of tasks constituting each workload and achieve desired effects on peak power and resulting performance trade-off. Ideally, a middleware tool would be responsible for estimating the Power requirement for a task and analyzing the workload to assign Power Tolerance before submitting it to the scheduler for execution. In this paper, we explore the opportunities in making smarter peak power and performance trade-offs when using Power and Power Tolerance to allocate resources on a cluster managed by Apache Mesos. We make the following contributions.

- Propose, implement and analyze a resource allocation policy that uses Power as a first class parameter, as an indicator of a task's potential impact on peak power draw, along with Power Tolerance as an indicator of the task's sensitivity to resource contention.
- Propose Power Tolerance assignment methods that project managers (principal investigators) and datacenter managers can employ in different scenarios to make smarter peak power and performance trade-offs.
- Analyze the influence of the Power Tolerance assignment methods on performance, peak power and energy consumption, and quantify the trade-offs.

*Note that the applicability of the proposed work towards meeting strict power budgets (maintaining cluster power consumption within certain limits) has not yet been explored and is subject of future work.*

### A. Mesos

Mesos [4] has evolved to become a leader in the cloud resource management and scheduling space. Large corporations such as PayPal, Netflix, Apple, Twitter, and Bloomberg deploy Mesos to manage clusters spanning hundreds to thousands of nodes [14]. All the experiments in this paper were conducted on a cluster managed by Apache Mesos.

A Mesos cluster is comprised of master nodes and worker nodes. Mesos agent daemons run on worker nodes and master daemons run on master nodes. Mesos agents advertize their available resources, such as CPU and memory, to the master nodes. Apart from resources, Mesos agents can also advertize attributes that are forwarded by the Mesos master to applications that run on top of it called $frameworks$ [4]. The agents in our cluster are configured to advertise the category of machines that they belong to called a *power-class* (discussed later in Section III-A). Mesos uses a two level scheduling system. In the first level, the Mesos master pools together advertised resources and offers them in the form of coarse-grained resource offers to frameworks. In the second level, the framework either consumes the resources in an offer by allocating them to pending tasks, in part or in full, or declines the offer. Tasks are launched on the host bound to the consumed offer. Unused resources are put back into the resource pool and offered to the framework in subsequent offer cycles.

In Mesos, CPU resource limits can be set using the `cgroups/cpu` isolator [15] that offers two ways of limiting the amount of CPU time used. By default this isolator uses *CFS shares limiting* that translates to allocating a weighted share of CPU time but not limiting the amount of CPU time used when the resource contention is low. This can lead to co-located tasks affecting the performance of one another. The `cgroups/cpu` isolator offers another method for CPU resource limiting: *CFS Bandwidth Control* that sets a hard cap on the CPU time allotted within a scheduling period. This can prevent applications from utilizing the full compute potential of the CPU, thereby impacting performance when the resources are under-estimated. Neither CFS shares nor CFS Bandwidth Control address the *noisy neighbor* phenomenon while still allowing tasks to consume idle cycles when resource contention is low.

### B. Electron

Electron is a power-aware, pluggable and configurable Mesos framework [16]. Leveraging the resource abstraction provided by Mesos, Electron allows different scheduling algorithms to participate in consuming resource offers. Electron has been enhanced to accept a stream of jobs and monitor co-located tasks' metrics. The proposed resource allocation policy has been added to Electron. Electron is used as the Mesos framework for all the experiments conducted in this work. Running Average Power Limit (RAPL) [17] and Performance Co-Pilot (PCP) [18] are used to monitor system-wide utilization metrics and power readings.

## II. Power as a Resource (WaaR)

### A. Power

Knowledge of the induced power consumption of a machine when a task is launched on it, although important to gauge the potential impact on peak power draw, is difficult to estimate. In addition, estimating the resultant power draw when a task is co-located with others is infeasible as it requires prior knowledge of the set of applications to be scheduled.

We consider *PoWer* as another resource that tasks can reserve. In this paper, we use PCP [18] and the estimation method proposed in [6] (which is discussed later in Section III-B) to estimate the Power requirement of a task on a category of machines.

Along with the traditional resources such as CPU and memory, the Power requirement of a task is also included as part of its resource requirements.

### B. Power Tolerance (WTol)

It is important to note that the added constraint (Power) for allocating resources can reduce the probability of a task's resource requirements from being satisfied by the

next available resource offer, thereby impacting wait-times. On the other hand, neglecting the computational intensities of tasks can lead to performance degradation of co-located tasks as a result of resource contention.

Along with the *Power* requirement, *PoWer Tolerance* (WTol) is included to indicate the sensitivities of tasks towards resource contention and in turn help regulate the extent of co-location. Increase in the number of co-located tasks reduces the CPU time allotted to each task, in turn leading to a smaller share of the processing power. To allow WTol to dictate a task's resistance towards co-location, it is interpreted as the acceptable percentage reduction of the requested Power that can be allocated to a task. **A task with a higher WTol is inferred to have lesser resistance towards co-location than compared to a task with a lower WTol.** Note, that this work assumes that WTol of a task is fixed throughout its execution. Variable tolerance requires further study into task behavior and is subject of future work. In subsequent sections, we discuss different methods of WTol assignment and analyze the impact on peak power, performance and energy.

The minimum acceptable power that can be allocated to a task 't' on a host 'h' of power-class 'PC' is given by function (1).

$$\text{MinAccPower(t, h)} = \text{Power(t, PC(h))} * (1 - \text{WTol(t)}) \quad (1)$$

As Power and Power Tolerance are used to regulate co-location and limit the impact on performance, we use the *CFS shares resource limiting* mechanism to allow for tasks to consume idle cycles, if any. When there is contention for CPU, a task's CPU requirement is a factor in determining the proportional share of the host machine's CPU cycles that the task can utilize [19], [15]. Considering $T(h) = \{t_1, t_2, t_3, ..., t_k\}$ as the set of 'k' co-located tasks on host 'h', the actual share of CPU resources that a new task 't' would get, if launched on that host, is given by function (2).

$$\text{Cr(t,h)} = \text{CPU(t)} / \{\sum_{i=1}^{k} \text{CPU(T(h)}_i) + \text{CPU(t)}\} \quad (2)$$

The Thermal Design Power (TDP) gives an estimate of the power consumed by the CPU that is being fully utilized. It has been shown that the power consumption has a linear relationship with CPU utilization [20], [21] and therefore, allocating a share of CPU resources can be assumed to be equivalent to allocating the same share of the TDP. Note that although the TDP of a machine is a loose upper bound, it gives a reasonable estimate of the consumed power when the CPU is completely utilized. The allocated share of power for a task 't' on host 'h' is calculated using function (3).

$$\text{P(t,h)} = \text{Cr(t,h)} * \text{TDP(h)} \quad (3)$$

Note that Power Tolerance only applies to allocated CPU, a soft constraint, and not to memory as it is a hard constraint.

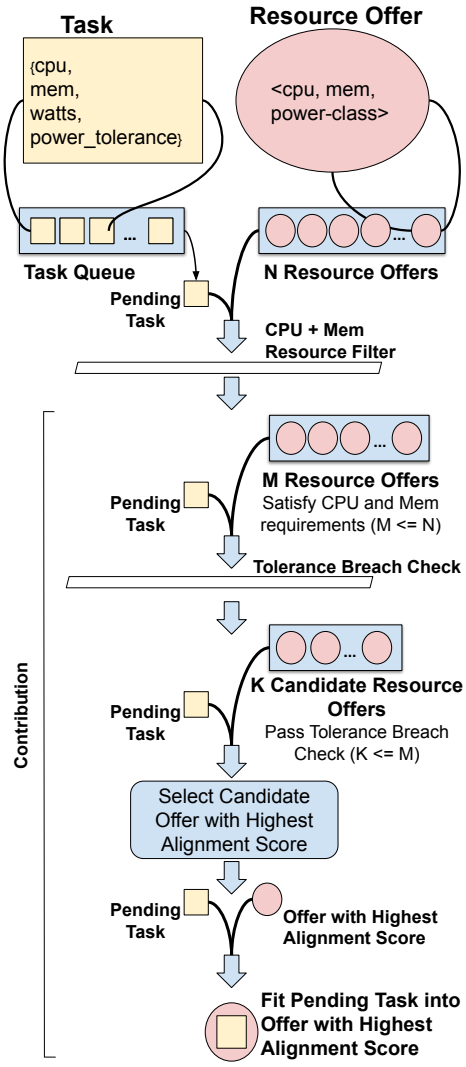



Figure 1. Task-Offer matching cycle in WaaRAlloc (Algorithm 1).

### C. Resource Allocation with WaaR (WaaRAlloc)

The proposed resource allocation policy ($WaaRAlloc$) that uses Power as a First Class Parameter to allocate resources to tasks is shown in Algorithm 1. Figure 1 outlines the series of steps involved in WaaRAlloc to match a pending task with a resource offer. For every task $t_{new}$, resource offers that satisfy both memory and CPU requirements go through a tolerance breach check where equation (4) is evaluated. If there is no tolerance breach ((4) holds true), then the offer is tagged as a candidate.

$$\text{P(t,h)} \geq \text{MinAccPower(t,h)} \ \forall \ \text{t} \in \{\text{T(h)}, \text{t}_{\text{new}}\} \quad (4)$$

A task could be launched on a host corresponding to any of the candidate offers. To limit the surge in peak power draw when a task is launched, we factor in the slack in total power consumption. We define the slack in total power consumption as the difference between the max CPU and

DRAM power specification, and the past 5 second running average of the consumed power (shown in function (5)). In our work, we select the offer corresponding to the host that is experiencing the least slack in total power consumption. Optionally, alignment methods like the ones proposed by Grandl et al. [22] can be used to rank the candidate offers.

$$\begin{aligned}\text{PowerSlack(h)} = (\text{TDP(h)} + \text{DramMaxPow(h)}) \\ - (\text{RunAvgPow}_{5\text{sec}}(\text{h}))\end{aligned} \quad (5)$$

**Algorithm 1** Resource allocation with WaaR (WaaRAlloc)

```
1: O[..] ← received resource offers
2: CO[..] ← candidate offers (no
   tolerance breach)
3: T_pend[..] ← tasks pending allocation
4: T_run {host => [..]} ← tasks running on
   host
5: FitTasks{offer => [..]} ← tasks fit
   into offers
6: function WAAR_ALLOC(O[..], T_pend[..])
7:     for t in T_pend[..] do
8:         for o in O[..].withSuffMem(t) do
9:             if o.CPU ≥ t.CPU then
10:                if !TOL_BREACH(t, o.host) then
11:                    CO.add(o)
12:                end if
13:            end if
14:        end for
15:        o ← leastPowerSlack(CO[..])
16:        FitTasks[o] ← t
17:    end for
18:    for o, tasks in FitTasks do
19:        LAUNCHTASKS(o.host, tasks)
20:    end for
21:    DECLINEUNUSEDOFFERS(O[..])
22: end function
23:
24: function TOL_BREACH(t_pend, h)
25:    return P(t,h) < MinAccPower(t,h)
26:                ∀ t ∈ {T_run(h), t_pend}
27: end function
```

## III. Experiments and Evaluation

### A. Cluster Setup

We conducted our experiments on a local cluster, the hardware specifications of which are tabulated in Table I. The nodes in our cluster were categorized into three power-classes based on their TDP specifications with class A nodes having the highest and class C nodes having the lowest.

Each node runs 64-bit Ubuntu 18.04 (Bionic) as the operating system with Linux Kernel v4.15.0-39-generic. Docker v18.09.0 was used as the container technology. Performance Co-Pilot (PCP) [18] v4.0.1 was used to monitor system metrics across the cluster. Power readings were retrieved from Intel's Running Average Power Limit (RAPL) [17] hardware counters. Apache Mesos v1.6.0 was used as the cluster resource manager and Electron was used as the Mesos framework to schedule tasks in docker containers. *Note that Electron was enhanced for the work presented in this paper. The rest of the tools were off-the-shelf versions, thereby ensuring portability.*

Table I
CLUSTER HARDWARE SPECIFICATION

| | **Power-Class** | | |
|---|---|---|---|
| | A | B | C |
| **Sockets** | 2 | 2 | 2 |
| **Cores/ Socket** | 10 | 8 | 6 |
| **Threads/ Socket** | 20 | 16 | 12 |
| **Family** (Intel Xeon Haswell) | E5-2650 v3 @ 2.30GHz | E5-2640 v3 @ 2.60GHz | E5-2620 v3 @ 2.40GHz |
| **Memory (GB)** | 128,256 | 128,64 | 64 |
| **TDP (Watts)** | 105 | 90 | 85 |
| **Nodes** | 2 | 2 | 4 |

### B. Power Estimation

Delvalle et. al. [6] proposed a method to estimate the worst case power consumption of a task on a category of machines. Ten runs of the task were executed on each power class. However, in this work, instead of using the max peak power usage from each run, the median power usage was used as the estimated power consumption of a task as it provides a better approximation of the computational intensity. The median of medians of power consumption across the ten runs was then considered as the typical power consumption of the task on any host belonging to that power-class. We call this estimation MMPU. For each power-class of nodes the MMPU value was used as the Power requirement of the task. Ideally, an online Power requirement estimation algorithm would need to be developed and is subject of future work.

### C. Workload

The benchmarks in our workload have been taken from NERSC [23] and Mantevo [24], as well as the DaCapo [25] and Phoronix [26] benchmark suites. Table II lists the benchmarks along with their description. The power-intensive and non power-intensive benchmarks are tagged PI and NPI respectively. The categorization of tasks into PI and NPI was performed using $k - means$ classification based on the estimated power consumption (MMPU value) when the tasks were run isolated.

Apart from the traditional resource requirements such as CPU and Memory, the tasks submitted to Electron also

included Power requirement and Power Tolerance specification.

Each **task** in our workload specifies the number of **instances** that need to be executed. Each **job** in our workload is a collection of tasks submitted together. Our workload comprises of different sections. Each workload **section** consists of a set of jobs that have similar characteristics, each containing a mix of PI and NPI tasks. For experimentation, we created a workload (characteristics shown in Table III) that consists of four sections in the order: non power-intensive section, power-intensive section, sparse section and dense section. These workload sections differ in the following.

- Interval (sec) - Time interval between job submissions.
- Ratio of number of NPI to PI tasks as used in [27].
- Total number of instances across all tasks in each job.

While these workload sections do not constitute an exhaustive set, the varied characteristics are representative of workloads in clouds or large cluster settings and provision a good environment for experimentation and analysis. A total of 2514 task instances with the following split up constituted the workload.

- Non Power-Intensive section - 629 Task Instances.
- Power-Intensive section - 621 Task Instances.
- Sparse section - 65 Task Instances.
- Dense section - 1199 Task Instances.

We assign WTol at different levels of granularity (as explained in subsequent sections), analyze the impact on peak power, performance and energy, and compare the results with two traditional resource allocation policies: First-Fit and Bin-Packing.

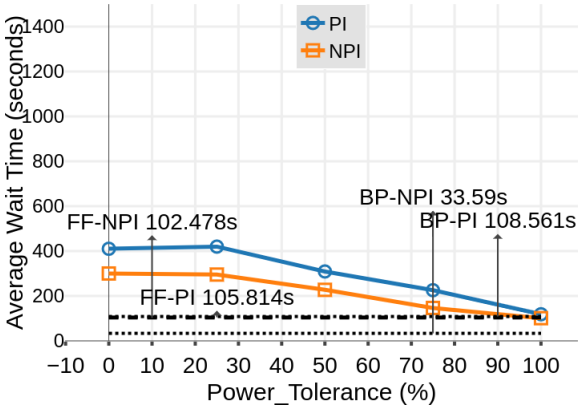
- **First-Fit (FF)** - FF assigns resources to pending tasks on a first-come-first-serve basis from resource offers. Once a task is assigned resources, it is scheduled to execute on the host corresponding to the consumed resource offer.
- **Bin-Packing (BP)** - Tasks are sorted in non-decreasing order of their respective CPU resource requirement. BP assigns resources from a resource offer to as many pending tasks as possible until there are no more resources available.

For the remainder of the paper, we use the following abbreviations in addition to those that are already mentioned.

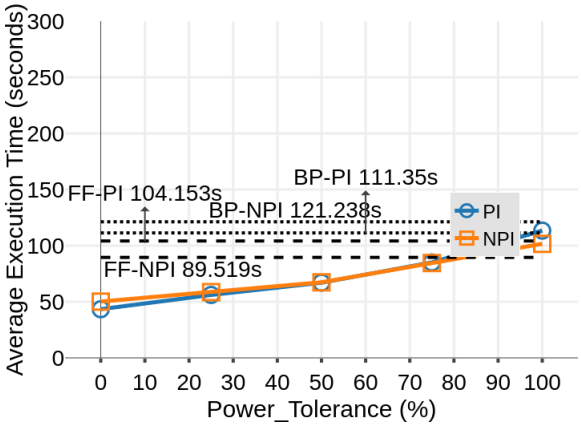
- ⟨ X ⟩ pWTol - X% Power Tolerance.
- wT, eT - Wait Time and Execution Time respectively.

### *D. Workload wide Uniform Power Tolerance Assignment (U-WTol)*

All tasks in the workload are assigned a uniform value of WTol. We experimented with uniform WTol values of 0%, 25%, 50%, 75% and 100%, and analyzed the effects on peak power consumption, task performance and energy usage. This method of WTol assignment can be employed




(a) Average wT (PI vs NPI Tasks)




(b) Average eT (PI vs NPI Tasks)

Figure 2. Impact on average wait-times and execution-times when varying uniform Power Tolerance. Dashed and dotted lines represent the average wait-times and execution-times for FF and BP respectively.

in scenarios where the workload is primarily being viewed as a black box and limited information is available on its constituents.

Figures 2a, 2b show variations in the average wT and average eT respectively for PI and NPI tasks with respect to variations in WTol.

*1) **Evaluation**:* From figure 2a we can see that the tasks predominantly experience a much lower average wT for FF and BP when compared to WaaRAlloc with U-WTol. This is due to FF and BP being oblivious to the potential increase in resource contention and launching tasks at a faster rate. The large impact on average wT for WTol less than 75pWTol can be attributed to under-utilization of resources and a larger backlog of pending tasks. For a given reduction in WTol, PI tasks experience a larger impact on average wT when compared to NPI tasks as a higher power requirement is to

Table II
BENCHMARKS USED TO CREATE WORKLOADS. **NPI** - NON POWER-INTENSIVE. **PI** - POWER-INTENSIVE.

| | Benchmark | Description | Tag |
|---|---|---|---|
| **Mantevo** | miniFE | Proxy application for unstructured implicit finite element codes. | NPI |
| **NERSC** | stream | Measure sustainable memory bandwidth in MB/s. | NPI |
| | dgemm | Dense multi-threaded matrix multiplication. | PI |
| **DaCapo** | jython | Interpret pybench Python benchmark. | NPI |
| | h2 | In-memory benchmark for transactions against a banking application. | NPI |
| | batik | Create Scalable Vector Graphics (SVG) image for Apache Batik unit tests. | NPI |
| | avrora | Simulate programs run on AVR micro-controllers grid. | NPI |
| | eclipse | non-gui performance testing of Eclispe IDE | NPI |
| | fop | Parse and format XSL-FO file into PDF file. | NPI |
| | luindex | Create document indices using lucene. | NPI |
| | lusearch | Text document searching using lucene. | PI |
| | pmd | Java source code problem analysis. | NPI |
| | sunflow | Use ray tracing for image rendering. | PI |
| | tomcat | Tomcat server testing by querying and analyzing served webpages. | PI |
| | tradebeans | Day-trader benchmark run with GERONIMO backend and h2 in-mem DB. | NPI |
| | xalan | Transformation of XML into HTML documents. | PI |
| **Phoronix** [28] | network-loopback | Network adapter performance measurement using micro-benchmark. | NPI |
| | audio-encoding | Measure performance of WAV file to other audio format conversions. | NPI |
| | video-encoding | System performance, processor and video encoding tests. | PI |
| | cryptography | John the Ripper, OpenSSL and GnuPG testing. | PI |

Table III
CHARACTERISTICS OF WORKLOAD SECTIONS. **#NPI/#PI** IS THE RATIO OF NUMBER OF NON POWER-INTENSIVE TASKS TO NUMBER OF POWER-INTENSIVE TASKS.

| Workload Section | Intervals (sec) | #NPI/ #PI | Jobs | Instances/ Job |
|---|---|---|---|---|
| Power Intensive | [5,300] | 5/12 | 5 | [100,150] |
| Non-Power Intensive | [5,300] | 20/3 | 5 | [100,150] |
| Sparse Jobs | [200,300] | - | 5 | [1,25] |
| Dense Jobs | [1,120] | - | 5 | [200,300] |

be satisfied by the resource offers.

*If a workload primarily consists of non power-intensive tasks, Power Tolerance can be further lowered to obtain peak power gains as the impact on average wait-times is less severe.*

Figure 2b shows that tasks experience an improvement in execution times for WaaRAlloc with U-WTol when compared to FF and BP. This improvement validates the importance of Power and Power Tolerance in moderating resource contention.

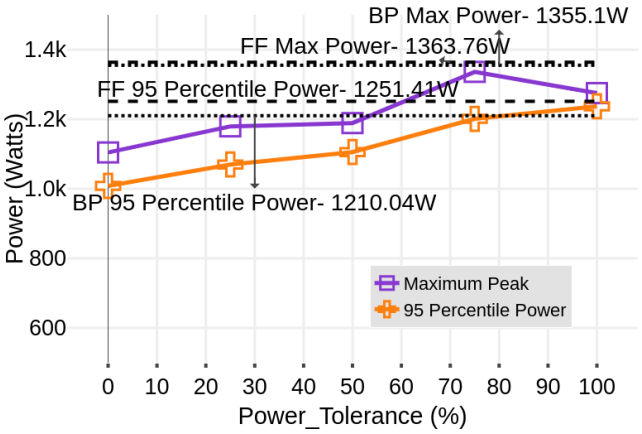
Although lowering of Power Tolerance reduces overall resource contention and improves execution-times, significant lowering of Power Tolerance will impact wait-times.

As mentioned earlier in Section I, surges in peak power consumption can considerably ramp up the operational costs and impact hardware. Instead of using average or median as the comparison metric, we therefore use the $95^{th}$ percentile in power consumption to also factor in temporary surges in power draw. Figure 3a compares the max peak power draw and $95^{th}$ percentile of power consumption, and figure 3b compares the overall energy usage when using WaaRAlloc with U-WTol compared to FF and BP. WaaRAlloc with uniform 100pWTol experiences a 6.5% and 5.9% reduction in max peak power consumption compared to FF and BP respectively. This can be attributed to selecting the offer with the least slack in total power consumption, thereby reducing the induced peak power draw when tasks are launched. We can also see that larger gains, up to about 19% in max peak and about 19.4% in $95^{th}$ percentile of power consumption, can be achieved with lower values of WTol compared to FF and BP respectively. However, the one size fits all policy of uniform WTol assignment fails to adhere to workload heterogeneity resulting in lower values of uniform WTol incurring significant losses in overall energy usage when compared to FF and BP. In addition, the significant impact on wait-times for lower values of Power Tolerance could have a cascading effect leading to an even larger impact on overall energy usage.

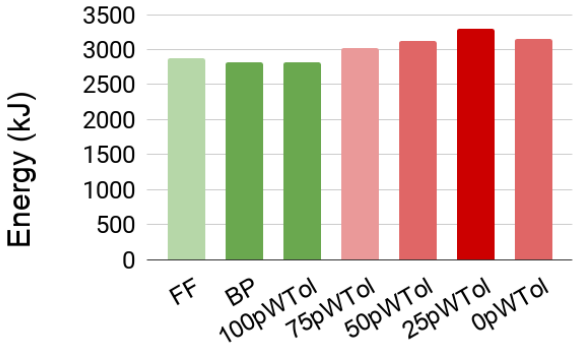
*With limited information on workload constituents, lowering the uniform Power Tolerance can help significantly lower peak power draw. However, heterogeneity in the workload causes uniform Power Tolerance values below 75% to suffer from a noticeable impact on energy usage.*

### *E. Fine-Grained Power Tolerance Assignment*

We propose two fine-grained methods of assigning WTol to tasks. These experiments, unlike those in section III-D, require a deeper understanding of the workload. These experiments were conducted to study if an increase in the granularity of WTol assignment shows potential in making smarter trade-offs between peak power consumption, task performance and energy usage.




(a) Max Peak and $95^{th}$ percentile Power.
Dashed and dotted lines represent max peak and $95^{th}$ percentile of power consumption when using FF and BP respectively.




(b) Energy

Figure 3. Low uniform Power Tolerance achieves noticeable reduction in peak power consumption. However, the large impact on wait-times as seen in 2a out-weights the improvements in execution times thereby affecting net energy usage for lower values of Power Tolerance.

*1)* ***Workload Section based WTol Assignment (WlS-WTol)****:* The overall characteristic of a workload section can dictate the impact on peak power draw, average wait-times (wT) and the average execution-times (eT) of tasks. All tasks within a given workload section are assigned the same value of WTol, supported with an understanding of its characteristics. This method of WTol assignment can be incorporated in scenarios where it is feasible to monitor large variations in workload characteristics.

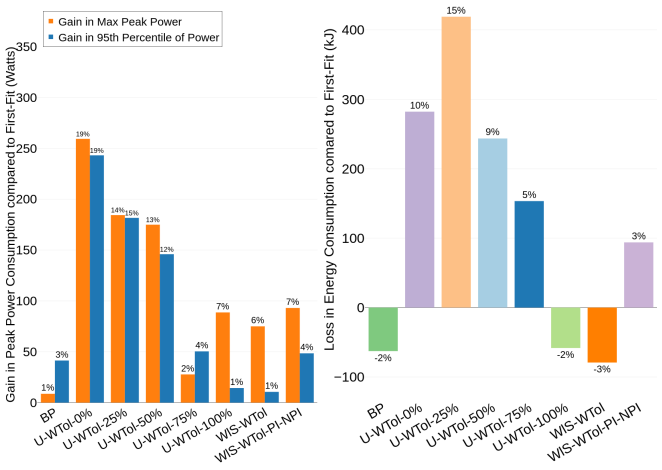
- Non Power-Intensive Section - Most of the tasks in this section are non power-intensive (NPI tasks). Given that lowering the assigned WTol value results in a smaller impact on average wT of NPI tasks (shown earlier in Figure 2a), all tasks in this section were assigned a 50pWTol to favor a reduction in peak power consumption.
- Power-Intensive Section - Most of the tasks in this section have a higher Power requirement (PI tasks) reducing their chances of being fit into the next available resource offer. Given that lowering the assigned WTol value to PI tasks largely affects their average wT (as seen in Figure 2a), all tasks in this section are assigned a 100pWTol.
- Sparse Section - Given that there are relatively fewer tasks that can suffer from an impact to their respective wT, all tasks in this section are assigned a 50pWTol.
- Dense Section - Given that a larger number of tasks can suffer from an impact to their respective wT, all tasks in this section are assigned a 100pWTol.

*Workload section based Power Tolerance assignment assumes a broad understanding of the tasks in a given workload section and ignores heterogeneity in characteristics of the workload section's constituent tasks.*

*2)* ***Workload Section + PI/NPI Task Category based WTol Assignment (WlS-WTol-PI-NPI)****:* This method of WTol assignment not only factors in the characteristic of the workload section but also its constituent tasks. This method of WTol assignment demands deeper insight into the constituent tasks. All PI tasks in a workload section are assigned the same WTol value, and all the NPI tasks in a workload section are assigned the same WTol value.

- Non Power-Intensive Section - All NPI and PI tasks are assigned a 50pWTol and 100pWTol respectively to reduce the collective impact of PI tasks on the wT of pending tasks.
- Power-Intensive Section - NPI and PI tasks are assigned a 100pWTol and 50pWTol respectively. This is done in order to prevent PI tasks, being more in number, from hoarding CPU time and degrading the performance of co-located tasks.
- Sparse Section - As the collective impact on average wT is lower, PI tasks are assigned a 50pWTol to favor a reduction in peak power draw, while NPI tasks are assigned a 100pWTol.
- Dense Section - NPI and PI tasks are assigned a 50pWTol and 100pWTol respectively. Jobs in this section are not only comprised of a large number of task instances but are also submitted at a faster rate. Assigning a higher WTol value to PI tasks would help consume the task queue more quickly, as additional tasks can be co-located with them, than if a higher WTol value were to be assigned to NPI tasks.

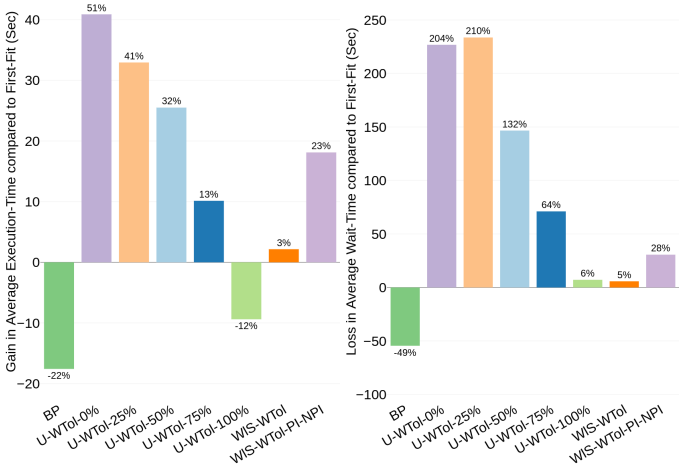
*3)* ***Evaluation****:* We compare the fine-grained WTol assignment methods with U-WTol, FF and BP with respect to peak power consumption, task performance and energy usage. The experiments conducted using the proposed fine-grained WTol assignment methods demonstrate that increasing the granularity of WTol assignment reduces the impact on average wT while still showing gains in average eT. In




(a) Gain in Power Consumption (Watts) with respect to First-Fit

(b) Loss in Energy Consumption (kilojoules) with respect to First-Fit

Figure 4. Gains in peak power consumption and corresponding losses in energy usage with respect to First-Fit. Percentages are rounded to the nearest integer.




(a) Gain in Average Execution-Time (seconds) with respect to First-Fit

(b) Loss in Average Wait-Time (seconds) with respect to First-Fit

Figure 5. Gains in average execution-times and losses in average wait-times with respect to First-Fit. Percentages are rounded to the nearest integer.

addition, these fine-grained WTol assignment methods lead to improvements in peak power consumption with a limited impact on energy usage. However, it is important to note that these methods for WTol assignment demand a thorough analysis of the previously conducted experiments.

Figures 4a, 4b, 5a and 5b illustrate the gains in peak power consumption, losses in net energy usage, gains in average eT and the losses in average wT respectively, of the experiments conducted in this work with respect to

Table IV
AVERAGE TURNAROUND TIMES, MAX PEAK PEAK AND $95^{th}$ PERCENTILE OF POWER CONSUMPTION, ENERGY USAGE AND MAKESPAN OF ALL THE EXPERIMENTS CONDUCTED IN THIS WORK (THE NUMBERS HAVE BEEN ROUNDED TO THE NEAREST INTEGER).

| | Avg TaT (s) | Power (Watts) | | Energy (kJ) | Make-span (s) |
|---|---|---|---|---|---|
| | | Max Peak | $95^{th}$ Per-centile | | |
| FF | 191 | 1364 | 1251 | 2875 | 4092 |
| BP | 155 | 1355 | 1210 | 2812 | 4389 |
| U-WTol-0% | 377 | 1105 | 1008 | 3157 | 4838 |
| U-WTol-25% | 392 | 1179 | 1070 | 3294 | 4596 |
| U-WTol-50% | 313 | 1189 | 1106 | 3119 | 4437 |
| U-WTol-75% | 252 | 1336 | 1201 | 3029 | 4256 |
| U-WTol-100% | 208 | 1275 | 1237 | 2817 | 4025 |
| WlS-WTol | 195 | 1289 | 1241 | 2796 | 4016 |
| WlS-Wtol-PI-NPI | 204 | 1271 | 1203 | 2969 | 4302 |

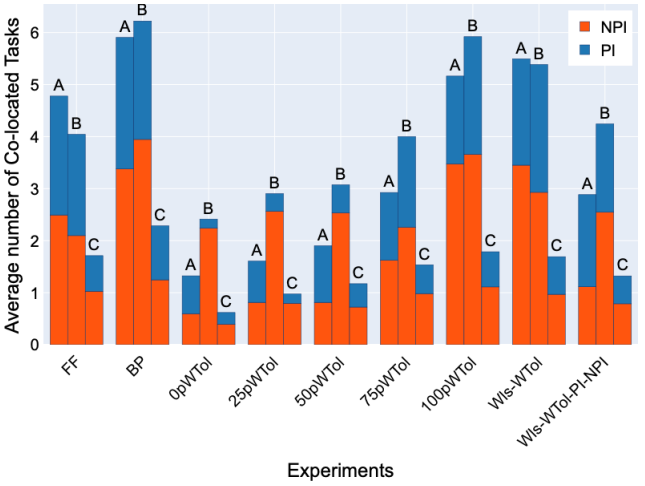



Figure 6. Types of tasks that were co-located on machines, categorized by their power-class (A, B or C). Co-locating too many tasks on a machine degrades performance and co-locating too few leads to resource wastage. Co-locating a good mix of PI and NPI tasks promotes better peak power performance trade-offs.

FF. Table IV tabulates the average turn-around-time (TaT), peak power consumption, net energy usage and makespan of all the experiments conducted in this work. Although uniform WTol (U-WTol) assignment can achieve significant improvements in peak power consumption (for lower values of Power Tolerance), the one size fits all policy results in a large impact on average wT affecting energy usage. When the characteristics of different workload sections are factored into the assignment of WTol (WlS-WTol) noticeable improvements are seen in peak power and energy usage, and yet the impact on average wT is substantially lower when compared to U-WTol. We see that WlS-WTol experiences

a 5.4% and 2.7% reduction in the max peak power and net energy usage respectively, compared to FF, while still exhibiting a similar $95^{th}$ percentile of power consumption (improving by around 1%) and average TaT (increase of around 2%). A WTol assignment that also considers the computational intensities of tasks (WlS-WTol-PI-NPI) leads to a 6.8% and 3.9% improvement in max peak power and $95^{th}$ percentile of power consumption respectively when compared to FF. WlS-WTol-PI-NPI demonstrates an increase of 6.5% and 3.2% in average TaT and net energy usage respectively. This can be attributed to WlS-WTol-PI-NPI being ignorant to variations in task execution times.

Figure 6 illustrates the types of tasks that were co-located, on average, on each of the power-classes in each of the experiments conducted in this work. From the figure, it is evident that Power Tolerance facilitates the regulation of co-location. With a one size fits all policy in U-WTol, we can see that although reduction in Power Tolerance reduces the extent of co-location, significant lowering of uniform Power Tolerance leads to uneven distribution of tasks across the cluster and under-utilization of resources. On the other hand, WlS-WTol and WlS-WTol-PI-NPI not only reduce the extent of co-location but also co-locate a better mix of PI and NPI tasks and thereby promote a better peak power performance trade-off.

## IV. Related Work

Wang et al. [29] proposed a scheduling algorithm that reduces the energy consumption and limits the increase in execution times of tasks on homogeneous clusters. Our work, however, focuses on peak power performance trade-offs on heterogeneous clusters. Kotla et al. [30] analyze the execution characteristics of currently running tasks and alter the frequency of the processor to provide the required performance. Unlike both [29] and [30], we do not alter the voltage and frequency of processors as it would be unsustainable and impractical when large number of applications are co-located in virtualized environments.

Wang et al. [31] proposed a power scheduling mechanism to improve performance considering the power-cap, hardware usage and application sensitivity to power. Our work does not employ power capping and considers the performance sensitivities of applications to resource contention.

Jivaranji et al. [32] proposed a scheduling algorithm that groups together similar tasks and allocates resources on a machine, that is predicted to be idle, using a best fit approach. Similarly, we categorize machines into power-classes. However, instead of maintaining a list of machines that are predicted to be idle our proposed algorithm works in a reactive manner using Mesos resource offers.

Fan et al. [12] proposed a mechanism to better the energy-latency trade-off by putting nodes in deeper sleep states based on the variations in the workload and fluctuations in system-wide utilization. In our work, we do not adjust the P and C states of the servers and leave that to the governor. In addition, we also factor in peak power consumption and turnaround times.

Felter et al. [33] proposed a mechanism of reducing peak power consumption by estimating the amount of activity of the CPU and memory and throttling those components. Our work however, is targeted towards portability across any Mesos managed cluster and assumes no change to the working of the server components.

Zhang et al. [34] use K-means clustering to group together tasks with similar resource and performance objectives. Similar to their work, we too consider workload heterogeneity and categorize tasks based on the computational intensity.

Delimitrou et al. [35] proposed Paragon, a heterogeneity and interference aware data-center scheduler that improves QoS guarantees and reduces the impact of interference on application performance. Applications are profiled on all server configurations. To reduce the profiling overhead for new applications, the authors use collaborative filtering to derive the missing information. Similarly, we too profile the tasks on all classes of servers (power-classes) as shown in Section III-B. In addition to considering the impact of resource contention on performance, we also explore opportunities to make better peak power, performance and energy trade-offs.

Tarahomi et al. [36] proposed a prediction based power-aware virtual machine allocation algorithm in three-tier cloud data-centers. The authors predict the load on each physical host and then make a decision to either place or migrate a virtual machine with the aim of improving energy efficiency. In our work, we focus on improving peak power consumption, performance and energy trade-offs.

To attain energy savings, [35], [36], [37] proposed turning idle servers off during periods of low load. Our work, on the other hand, assumes that the servers are always up and running as turning servers off can potentially affect responsiveness of the system during surges in compute demands [27].

Qureshi [38] used application profiles including power usage data and proposed a power-aware framework for efficient placement of application workloads. Similarly, we also perform offline profiling of tasks and factor in induced power consumption of applications. However, we also factor in the sensitivities of applications towards performance degradation with the use of Power Tolerance.

## V. Conclusion

We proposed a resource policy (WaaRAlloc) that considers (1) a task's potential impact on peak power draw (Power) and (2) its sensitivity towards resource contention (Power Tolerance). Through experimentation we observe that WaaRAlloc shows promising results in regulating the trade-off between peak power and performance.

- $Power$ proves viable to represent the computational intensity of a task.
- *Power Tolerance* helps regulate the extent of co-location and provisions for making smarter peak power and performance trade-offs.

We also proposed two fine-grained Power Tolerance assignment methods that consider the heterogeneity of the workload at difference levels of granularity. We make the following recommendations.

- When limited insight of the workload is available and stricter peak power constraints have to be enforced, the proposed workload-wide uniform Power Tolerance assignment can be used to obtain up to 19.4% and 16.7% reduction in the $95^{th}$ percentile of power consumption compared to First-Fit (FF) and Bin-Packing (BP) respectively. This gain is associated with a 9.8% and 12.3% increase in energy usage compared to FF and BP respectively.
- With a better understanding of workload heterogeneity, the proposed fine-grained Power Tolerance assignment methods can be utilized to obtain better peak power and performance trade-offs. These Power Tolerance assignment methods account for heterogeneity in the workload and are therefore more robust towards workload fluctuations, unlike uniform Power Tolerance assignment.
  - A Power Tolerance assignment method that factors in large variations in workload characteristics (WlS-WTol) can be adopted to realize gains of approximately 6% and 5% in max peak power consumption and a similar $95^{th}$ percentile of power consumption compared to FF and BP respectively. In addition, WlS-WTol results in approximately a 3% reduction in overall energy consumption compared to FF, and a similar energy consumption compared to BP.
  - A Power Tolerance assignment method that also factors in the tasks' potential impact on peak power draw (WlS-WTol-PI-NPI) shows promise in making better trade-offs. WlS-WTol-PI-NPI demonstrates a 7% reduction in max peak power and 4% reduction in $95^{th}$ percentile of power consumption when compared to a FF policy. These gains are accompanied by a 3.2% increase in energy usage and 6.5% increase in average turnaround times. Compared to BP, WlS-WTol-PI-NPI shows a 6% reduction in max peak power consumption while also improving average task execution times by 37%. However, oblivious to heterogeneity in task execution times, WlS-WTol-PI-NPI experiences a slight increase in net energy usage.

## ACKNOWLEDGMENT

We thank Vipul Chaskar, Santosh Hegde and Kunal Kulkarni for helping with workload generation and data visualization. In addition, we thank Renan Delvalle and Jessica Hartog for their insightful feedback.

## REFERENCES

[1] "Chameleon: A Large-scale, Reconfigurable Experimental Environment for Cloud Research," 2014. [Online]. Available: https://www.chameleoncloud.org

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] Pawel Janus and Krzysztof Rzadca, "SLO-aware Colocation of Data Center Tasks Based on Instantaneous Processor Requirements," in *2017 Symposium on Cloud Computing*. Santa Clara: ACM, 2017. [Online]. Available: https://dl.acm.org/citation.cfm?doid=3127479.3132244

[9] P. Kaushik, A. Kothawale, R. DelValle, A. Jain, and M. Govindaraju, "Analysis of Dynamically Switching Energy-Aware Scheduling Policies for Varying Workloads," in *IEEE 11th International Conference on Cloud Computing (CLOUD)*. San Francisco, CA, USA: IEEE, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8457792

[10] Y. Lee and A. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomputing*, p. 268–280, 2012. [Online]. Available: https://link.springer.com/article/10.1007/s11227-010-0421-3

[11] 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.

[12] Yao Fan, Wu Jingxin, Subramaniam Suresh, and Venkataramani Guru, "WASP: Workload Adaptive Energy-Latency Optimization in Server Farms Using Server Low-Power States," in *IEEE 10th International Conference on Cloud Computing (CLOUD)*. Honolulu, CA, USA:

IEEE, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8030586

[13] "Mesos frameworks," 2019. [Online]. Available: http://mesos.apache.org/documentation/latest/frameworks/

[14] "Scaling Mesos at Apple, Bloomberg, Netflix and more - Mesosphere," 2015. [Online]. Available: https://mesosphere.com/blog/2015/08/25/scaling-mesos-at-apple-bloomberg-netflix-and-more/

[15] "Mesos-Cgroups-Isolators," 2020. [Online]. Available: http://mesos.apache.org/documentation/latest/isolators/cgroups-cpu/

[16] R. DelValle, P. Kaushik, A. Jain, J. Hartog, and M. Govindaraju, "Elektron." [Online]. Available: https://github.com/spdfg/elektron

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

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

[19] docker, "Runtime options with Memory, CPUs, and GPUs," 2020. [Online]. Available: https://docs.docker.com/config/containers/resource_constraints/

[20] Vasan Arunchandar, Sivasubramaniam Anand, Shimpi Vikrant, Sivabalan T, and Subbiah Rajesh, "Worth their watts? - an empirical study of datacenter servers," in *The Sixteenth International Symposium on High-Performance Computer Architecture*. Bangalore, India: IEEE, 2010.

[21] Waltenegus Dargie, "A Stochastic Model for Estimating the Power Consumption of a Processor," *IEEE Transactions on Computers*, 2014. [Online]. Available: https://www.researchgate.net/publication/261958210_A_Stochastic_Model_for_Estimating_the_Power_Consumption_of_a_Processor

[22] Robert Grandl, Ganesh Ananthanarayanan, Srikanth Kandula, Sriram Rao, and Aditya Akella, "Multi-Resource Packing for Cluster Schedulers," in *2014 ACM conference on SIGCOMM*. Chicago, Illinois, USA: ACM New York, NY, USA ©2014, 2014. [Online]. Available: https://dl.acm.org/citation.cfm?id=2626334

[23] "NeRSC Benchmark Distribution and Run Rules," 2016. [Online]. Available: http://www.nersc.gov/research-and-development/apex/apex-benchmarks/

[24] 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.

[25] 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.

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

[27] 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

[28] Phoronix, "OpenbenchmarkingPhoronix," 2008. [Online]. Available: https://openbenchmarking.org/suites/pts

[29] Lizhe Wang, Gregor von Laszewski, Jay Dayal, and Fugang Wang, "Towards Energy Aware Scheduling for Precedence Constrained Parallel Tasks in a Cluster with DVFS," in *CCGRID '10 Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. Washington, DC: IEEE Computer Society, 2010. [Online]. Available: https://dl.acm.org/citation.cfm?id=1845153

[30] R. Kotla, S. Ghiasi, T. Keller, and F. Rawson, "Scheduling processor voltage and frequency in server and cluster systems," in *19th IEEE International Parallel and Distributed Processing Symposium*. Denver, CO, USA: IEEE, 2005. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1420157/authors#authors

[31] B. Wang, D. Schmidl, C. Terboven, and M. S. Muller, "Dynamic Application-Aware Power Capping," in *5th International Workshop on Energy Efficient Supercomputing*. Denver, CO, USA: Association for Computing Machinery, New York, NY, USA, 2017, pp. 1–8. [Online]. Available: https://doi.org/10.1145/3149412.3149413

[32] Aarti Jivrajani, Dhanya Raghu, Apoorva KH, H L Phalachandra, and Dinkar Sitaram, "Workload Characterization and Green Scheduling on Heterogeneous Clusters," in *22nd Annual International Conference on Advanced Computing and Communication*. Bangalore, India: IEEE, 2016. [Online]. Available: https://ieeexplore-ieee-org.proxy.binghamton.edu/document/8385595

[33] Wes Felter, Karthick Rajamani, Tom Keller, and Cosmin Rusu, "A performance-conserving approach for reducing peak power consumption in server systems," in *19th annual international conference on Supercomputing*. Cambridge, Massachusetts, USA: ACM New York, NY, USA ©2005, 2005. [Online]. Available: https://dl.acm.org/citation.cfm?id=1088188

[34] Qi Zhang, Mohamed Faten Zhani, Raouf Boutaba, and Joseph L. Hellerstein, "Harmony: Dynamic Heterogeneity-Aware Resource Provisioning in the Cloud," in *IEEE 33rd International Conference on Distributed Computing Systems*. Philadelphia, PA, USA: IEEE, 2013. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6681620/authors#authors

[35] C. Delimitrou and C. Kozyrakis, "Paragon: QoS-aware scheduling for heterogeneous datacenters," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 1, pp. 77–77, 5 2013.

[36] Mehran Tarahomi and Mohammad Izadi, "A prediction-based and power-aware virtual machine allocation algorithm in three-tier cloud data centers," *International Journal of Communication Systems*, vol. 32, no. 3, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/dac.3870

[37] Dinh-Mao Bui, YongIk Yoon, Eui-Nam Huh, SungIk Jun, and Sungyoung Lee, "Energy efficiency for cloud computing system based on predictive optimization," *Journal of Parallel and Distributed Computing*, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731516301708

[38] Basit Qureshi, "Profile-based power-aware workflow scheduling framework for energy-efficient data centers," *Future Generation Computer Systems*, vol. 94, pp. 453–467, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0167739X18318491