Comparing GPU Power and Frequency Capping: A Case Study with the MuMMI Workflow

Tapasya Patki, Zachary Frye, Harsh Bhatia, Francesco Di Natale, James N. Glosli,
Helgi I. Ingólfsson, Barry Rountree
Lawrence Livermore National Laboratory, 7000 East Ave. Livermore, CA
{patki1, frye7, bhatia4, dinatale3, glosli1, ingolfsson1, rountree4}@llnl.gov

***Abstract*—Accomplishing the goal of exascale computing under a potential power limit requires HPC clusters to maximize both parallel efficiency and power efficiency. As modern HPC systems embark on a trend toward extreme heterogeneity leveraging multiple GPUs per node, power management becomes even more challenging, especially when catering to scientific workflows with co-scheduled components. The impact of managing GPU power on workflow performance and run-to-run reproducibility has not been adequately studied.**

In this paper, we present a first-of-its-kind research to study the impact of the two power management knobs that are available on NVIDIA Volta GPUs: frequency capping and power capping. We analyzed performance and power metrics of GPU's on a top-10 supercomputer by tuning these knobs for more than 5,300 runs in a scientific workflow. Our data found that GPU power capping in a scientific workflow is an effective way of improving power efficiency while preserving performance, while GPU frequency capping is a demonstrably unpredictable way of reducing power consumption. Additionally, we identified that frequency capping results in higher variation and anomalous behavior on GPUs, which is counterintuitive to what has been observed in the research conducted on CPUs.

I. Introduction

The supercomputing community is at a major turning point: exascale computing is diverging from traditional HPC where it is driven not only by peak FLOPs, but also by the need for higher system efficiency [1]. Future systems are expected to exhibit extreme heterogeneity in terms of the architectural components they build upon. This trend is already evident with several Top500 systems [2], such as Sierra and Summit [3], [4], which leverage multiple GPUs to achieve higher performance and parallel efficiency. In response to this trend, scientific workflows and application codes are adapting to newer node architectures, resulting in complex interplays, added dependencies, and portability challenges.

This transition toward extremely heterogeneous architectures faces the *utilization* challenge. Traditionally, supercomputing sites measured utilization by looking at the number of nodes or cores in use. As we venture toward exascale, utilization of other hardware components (such as GPUs or burst buffers) and other resources (such as power, I/O, and network bandwidth, etc.) is coming into focus. Improving power and energy efficiency is a critical research area for exascale computing, especially because heterogeneous nodes (such as those with multiple GPUs) can prove to be extremely power hungry and can also result in temperature hotspots in machine rooms [5]. Additionally, large supercomputers can incur high electricity and cooling costs, making it important to consider mechanisms that allow for power capping in certain scenarios [6]–[9].

A key concern when managing power is the impact of power capping or selection of lower frequencies on the execution times and performance of the underlying science being accomplished. Significant advances have been made in the field of power and energy efficiency in the past decade in order to make it possible to ensure minimal or no loss of performance when capping power [10]–[18]. Techniques such as hardware overprovisioning, configuration selection at runtime, dynamic power balancing, and critical path analysis are being used actively to ensure improved performance through intelligent power capping – with the main idea being that of directing power to where it is needed most.

Most of the aforementioned research in power and performance optimization has focused primarily on HPC applications executing on many-core CPUs. Little research exists in the area of how power can be distributed intelligently for complex scientific workflows or on how power capping impacts performance of science applications that leverage multiple GPUs on a node. Furthermore, no current research studies the well-known issue of chip-level manufacturing variation in GPUs, or the differences between the options of frequency capping and power capping on GPUs [19]–[23].

This paper presents a first-of-its-kind analysis in this direction on a large-scale supercomputer. We analyze over 5,300 power and performance profiles of ddcMD [24], a well-known molecular dynamics (MD) code, in the context of the Joint Design of Advanced Computing Solutions for Cancer (JDACS4C) Pilot 2 MuMMI workflow [25] developed for cancer research on the Sierra and Lassen supercomputers at Lawrence Livermore National Laboratory (LLNL). Sierra is currently the second-fastest and Lassen is the tenth-fastest supercomputer in the world [2]. They are based on the IBM Power9 and NVIDIA Volta architecture. We explore two power management knobs to facilitate a detailed analysis: GPU power capping and GPU frequency capping [26]–[31]. We discuss the impact of both of these knobs on execution times as well as performance reproducibility. Specifically, we show that, for the MuMMI workflow:

- GPU power capping is an effective way of improving HPC cluster power efficiency,
- Counterintuitive to what we know about CPUs, performance variation resulting from chip lithography

differences is lower when setting power caps as opposed to setting direct frequencies on GPUs,
- Setting GPU frequencies can cause unreliable and unpredictable power usage,
- Setting GPU frequencies can result in anomalous behavior, which in our case is observed at 1005 Mhz,
- Reducing power from 300 W to 170 W on the GPU has no impact on ddcMD performance,
- Reducing power to the lowest value of 105 W (1/3rd of maximum) results in an effective frequency reduction of 26% and a slowdown of only 17%.

We organize the remainder of the paper as follows. Section II introduces the MuMMI workflow, and emphasizes on the importance of the ddcMD code as part of this workflow. The section also discusses the challenges faced when conducting an end-to-end analysis of such a complex workflow, and establishes directions for development of low-level monitoring and control technologies for power and performance. Section III presents the necessary background on power measurement and control as well as on setting frequencies on the NVIDIA Volta GPUs. This section also describes how our experiments were set up and run, and how ddcMD profiles were collected. Section IV presents our results that describe details of execution times, present the relationships between GPU power and frequency, and capture the observed variation and anomalies. Section V presents related work, and Section VI concludes the paper.

II. Massively Parallel Workflow for Multiscale Modeling of RAS Biology

Modern scientific applications require large-scale, complex workflows with several components, which must work together cohesively towards effective utilization of heterogenous resources. The intricate design of such workflows and their components require special focus on their power consumption. In this work, we study a massively parallel infrastructure for adaptive multiscale simulations developed to explore the role of RAS protein in the initiation of cancer [25].

A. Multiscale Modeling of RAS Biology

The JDACS4C program is a joined partnership between the Department of Energy (DOE) and the National Cancer Institute (NCI) to advance cancer research using emerging exascale HPC computing. The JDACS4C Pilot 2 project, is designed to exploit modern heterogenous computational resources to uncover the detailed characterizations of the behavior of RAS on cellular membranes. About 90.5 million people in the world had been diagnosed with cancer as of 2015 [32]. Over 30% of these cancers have been attributed to the RAS family of cancer-causing genes [33]. Yet, the true role of RAS in the cancer initiation is not well understood, limiting the ability to design drugs targeting RAS. Exploring RAS biology on cell membranes requires unraveling molecular interactions at high resolutions but at biologically relevant sizes and time scales. Ongoing experimental research in this field is limited by the lack of suitable technologies to analyze large molecular complexes on cell membranes at sufficient resolution. Molecular dynamics (MD) computer simulations can provide the require resolution, but existing models suffer from limitations of accessible size and time scales, and are often unable to achieve the large simulation times required to capture biologically relevant scales.

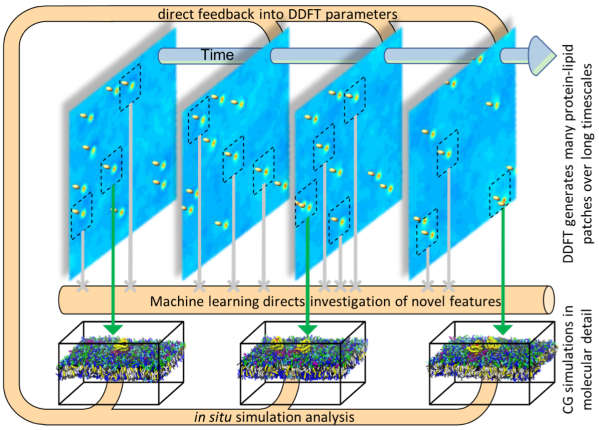
Towards understanding the underlying characteristics of RAS-RAS and RAS-membrane interactions, the Pilot 2 multiscale simulations utilize two scales: a *macro model*, which uses a continuum lipid description and a simplified RAS model, to describe RAS dynamics at larger length and time scales, and a *micro model*, based on high-fidelity molecular dynamics (MD) simulations. On the one hand, micro model, coarse-grained (CG) MD simulations [34] provide sufficient accuracy, but they are computationally challenging and are limited in both size and duration. On the other hand, the macro model offers a promising avenue for scaling to biologically-relevant time and length scales, but the continuum descriptions is often too simplistic, and phenomenological approximations are necessary to capture the complexity needed to model biological systems [35].

The Pilot 2 multiscale framework (Figure 1a) couples the two scales through novel use of machine learning (ML). Here, ML investigates a single, continuously running macro model simulation that spans micrometers and hundreds microseconds and selects "regions of interest" for which the framework instantiates CG MD simulations covering local behavior spanning tens of nanometers for a few microseconds. Several *in situ* analysis and processing capabilities are connected to analyze the thousands of concurrently running MD simulations, including an active feedback loop to update the parameterization of the macro model.

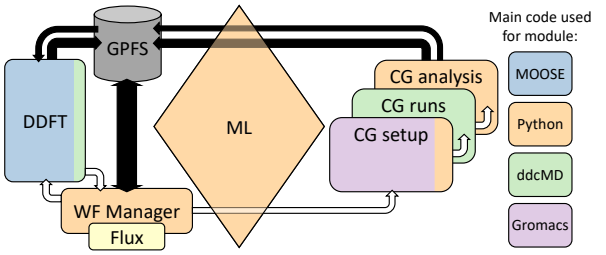
B. MuMMI Workflow and GPU-Enabled MD Simulations

The Multiscale Machine-Learned Modeling Infrastructure (MuMMI) was developed as part of the Pilot 2 project to enable the multiscale simulations described above. As illustrated in Figure 1b, the workflow contains several diverse components. The macro model simulation is developed using MOOSE [36], which is a scalable, MPI-enabled, finite-element solver that can utilize hundreds to thousands of CPU cores. The workflow itself, including the ML module, is written in Python. Upon availability of computational resources, the workflow uses ML to identify regions of interest in the macro model simulation, and maps them to an MD configuration. The resulting configurations are equilibrated using GROMACS [37] using 24 CPU cores each. Once equilibrated, the dynamics are simulated using ddcMD [24] modified for GPU performance, and *in situ* analysis utilities are executed along side, running on CPU cores on the nodes shared with the corresponding ddcMD.

A key design of this framework is that the MD simulations, which provide the true data of interest, run almost exclusively on GPUs, whereas all other components work entirely on CPU cores. The performance of ddcMD on CPU cores has previously acknowledged twice with Gordon Bell Prize [38], [39]. However, in order to support the multiscale simulations



(a) Overview of the MuMMI multiscale simulations.



(b) Module diagram of MuMMI highlighting diverse components.

Fig. 1: Multiscale Machine-Learned Modeling Infrastructure (MuMMI) has been designed to enable multiscale simulations of RAS biology to understand its role in the initiation of cancer. MuMMI framework couples a single macro scale simulation that spans large time- and length-scales with several thousand MD simulations that provide high-resolution data, but for small spatiotemporal regions. The framework is driven using ML, which evaluates macro simulation and selects regions of interest to spawn the corresponding MD simulations. ML-based sampling allows exploring the configuration space significantly more effectively resulting in a scope of exploration that is not achievable using only brute force calculations. Finally, *in situ* analysis of the CG simulations provide feedback improving the parameterization of the macro simulation on the fly.

targeted by MuMMI, significant extensions were made to ddcMD to support GPU-enabled high-throughput MD simulations and to minimize its CPU utilization.

This GPU-enabled ddcMD implements the Martini force field [40] as well as a number of improvement to improve GPU performance (e.g. improve the thread scheduling, enforce coalesced memory accesses, and rearrange data structures). This implementation offloads the entire computation to the GPU. Every non-constant time calculation step necessary for Martini now runs on the GPU via CUDA kernels, including the integrator and constraint solver, such that particles are only communicated back to the host for I/O purposes and never for calculations of the particle forces or movement. This leaves the CPUs tasked with only managing the order and launches of the aforementioned kernels.

While the MD simulations are running, analysis modules are executed every two seconds of wall time to continuously accumulate data of interest. When run on 4,000 nodes of *Sierra*: 1,000 nodes (CPU only) are used for the macro model, a single node for machine learning and workflow management system, and GROMACS simulations (on CPU only) run as needed the remaining 3,000 nodes. While those are running, four separate ddcMD simulations were run on each node using the GPUs as well as online analysis for each simulation. During a MuMMI run at least 8 logically separate jobs run on each node. Flux [41] and Maestro [42] are used to coordinate these co-scheduled components.

C. Challenges in Performance Monitoring

Traditional performance monitoring tools and libraries (such as TAU [43], HPCToolkit [44], or PAPI [45]), node monitoring frameworks (such as LDMS [46]), or power management and balancing tools (such as `libmsr` [47], GEOPM [48], Conductor [15]) are not suitable for analyzing or optimizing such complex workflows. A lot of existing tools require independent binaries for instrumentation, source code annotations, or reliance on MPI's PMPI layer. Integrated workflows through software such as Flux [41] or Maestro [42] are difficult to instrument or modify for use with the aforementioned tools. Additionally, these tools assume dedicated node access: that is, only a single application or component of a workflow is executing on a node at one point in time. Complex exascale workflows such as MuMMI have several co-scheduled components. High-level measurements done at the node-level become inaccurate for detailed analysis in these scenarios, as performance counters or power measurements cannot be clearly attributed to the contributing co-scheduled component. At present, tools for measuring end to end performance metrics (including power) of science workflows are non-existent — this is an open research and development area in the community. As a result, we take a more direct approach to power measurement on GPUs so as to ensure that we get accurate data through command-line interfaces. We do not take CPU or node power into consideration, and we solely focus on the GPU execution times of ddcMD. We describe this in the next section in detail.

III. Experimental Setup

We describe our experimental setup in this section. In our work, we consider two scenarios:

- Setting GPU frequencies and observing power usage and application performance, and,
- Setting GPU power caps and observing average frequency and application performance

This allows us to compare our options for tuning power directly and indirectly, and also allows us to compare variations and anomalies observed in each scenario. The following subsections describe our test system, tools, and data collection process.

A. Our Test System: Lassen

Lassen is a scaled-down version of the LLNL flagship cluster Sierra [3]. There are 684 nodes in Lassen with each node consisting of twin IBM Power9 processors, quad Nvidia Tesla V100 (Volta) GPUs, and 256 GB of DDR4 main memory. Being an IBM Power9 architecture system, Lassen uses the proprietary IBM Job scheduler LSF which is centered around allocating resource sets. A resource set is a user-defined collection of resources requested by a job in order to run. So, to better replicate performance of ddcMD within the MuMMI workflow, we ran our tests on the same resource sets that were used to run ddcMD within MuMMI: one GPU and two CPU cores per instance of ddcMD.

B. GPU Power and Frequency Capping

NVIDIA provides a tool named `nvidia-smi` [49] to enable setting as well as querying aspects of GPU performance. The tool has an easy-to-use command-line interface that works by designating a numerical value for either the power or frequency limit that needs to be enforced across the GPUs of a node, or by specifying the performance metric to be queried. The syntax is:

```
nvidia-smi <options> <value or metric>
```

The option to set a power limit is `-pl`, and the option to set a frequency limit is `-ac`. The only restriction to setting a power or frequency limit is that the value must be a supported value for the model of GPU. There is a list of supported frequencies visible by running the command:

```
nvidia-smi -q -d SUPPORTED_CLOCKS
```

The supported power limit range is between 100W and 300W. These two modes were used independently from another to ensure that the collected profile data provided an accurate analysis of how each kind of limit affects both power consumption and performance. We can also use the same tool to monitor the clock rate and power consumption of the GPUs. This was done by using the following command:

```
nvidia-smi -q -d POWER
```

Setting frequencies as well as measuring power did not require administrative privileges on Lassen. However, setting power caps required the use of a limited version of `sudo`, which was enabled for us through operations support with the help of a utility called `nv_powercap`.

C. Data Collection

Data collection was performed in two main steps:

- Running ddcMD and collecting profiles on Lassen, and,
- Post-processing the data to extract relevant information locally

The workflow on Lassen begins with a version of ddcMD that is compiled to support the IBM Power9 and Nvidia GV100 GPU architecture, and then setting up the data needed to run ddcMD. In order to simultaneously execute ddcMD and set a power or frequency limit, we request 2 CPUs and 1 GPU for a run of ddcMD as part of the LSF resource set. This resource set allows us to poll the power or frequency information during the ddcMD execution.

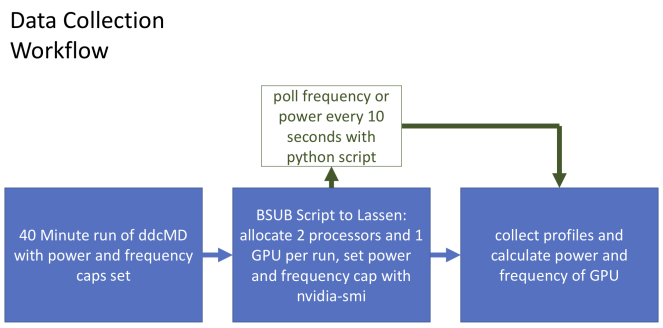


Fig. 2: Data Collection Process

ddcMD starts with a series of particle positions and performs MD calculations in order to simulate the evolution of their positions. Every 500,000 steps through the MD simulation, ddcMD outputs a profile that has information regarding memory usage, particle energies, and calculation statistics. For our purposes, we used the data corresponding to the number of milliseconds per calculation per particle in the simulation. In our tests, we ran ddcMD for a total of 40 minutes, typically generating two to four such profiles per run. The information regarding time in milliseconds per particle was extracted from these profiles and placed into a *summary* file using both python and bash scripts.

The ddcMD process ran in conjunction with a program that polled power and frequency of the GPU throughout the runtime of the application. We created a python application that polled GPUs using the `nvidia-smi` tool discussed previously. When a GPU power cap is set, the program polls the GPU for its current operational frequency every ten seconds, and then calculates the average frequency of the entire run at the end of the 40 minute job. We then append this average frequency value to its corresponding profile data in the profile summary files described earlier. This process is replicated in the same way for polling power values when running ddcMD under a frequency cap. All of this data is then compiled into one master profile summary for the batch of jobs running ddcMD and used for graphing and analysis. Analysis was performed using python to convert the master profiles from Lassen into CSV (comma separated value) files that hold all of the information pertaining to running ddcMD under power and frequency caps. These CSV files are graphed and analyzed in order to draw conclusions about how frequency and power caps affect performance of the workflow and power efficiency of the GPU.

We collected a total of 5,353 ddcMD profiles, with 753 profiles with frequency capping and 4,600 profiles with power capping. The counts appear nonuniform because each 40

minute could generate anywhere between two to four profiles, depending on the power or frequency cap that was enforced. We chose 17 frequency values and 30 power caps (in the ranges of 877 MHz to 1530 MHz, and 105 W to 300 W, respectively).

IV. Results

As discussed in Section III, we collected 5,353 ddcMD profiles and experimented two scenarios: setting GPU frequency (753 profiles) and power caps (4,600 profiles). We organize this data in three subsections. First, we discuss the relationship between power and frequency in the two scenarios. The next subsection takes a detailed look at the variation observed in both scenarios and presents a deeper dive into the data. We highlight an anomalous scenario that we encountered and were unable to resolve. Finally, we focus on application performance, where we present data on execution time slowdowns as well as data on variation in both the scenarios that we studied. For evaluating performance, we consider the calculations and the number of MD steps from the ddcMD profiles (MDSteps). As discussed earlier in Section III, these values are measured for every 500,000 steps. The performance metric we use represents time taken per particle per iteration, and while the metric values appear relatively small, we believe there is little measurement error in timing due to the sheer number of steps involved (500,000) in each measurement.

A. Relationship between GPU Power and GPU Frequency

Figure 5 depicts the relationship between power and frequency when running ddcMD on Lassen. In Figure 3a, we set 17 different frequencies in the range of 877 MHz to 1530 MHz using the `-ac` option described in Section III. For each frequency, we measure the power usage by polling every ten seconds.Figure 3a plots average power draw across the ddcMD profile against the selected frequency. Each color on the graph represents a certain selected frequency: note that there are multiple repeated observations for each frequency setting (about 100 per setting depending on how many ddcMD profiles were created at a certain frequency level in 40 minutes). The plot uses colors to clearly depict the variation in average power usage along the vertical axis for a certain frequency level.

As can be observed from Figure 3a, when we set the GPU frequency, the power usage varies significantly, and cannot be relied upon for providing any guarantees in a power-constrained scenario. The power usage is also unpredictable, depicting several anomalous scenarios. Typically, based on known power, voltage and frequency relationships, one would expect lower frequencies to consume lesser power. Whereas this is true for most of our data, we observe two anomalous scenarios at frequency settings of 1005 MHz and 1250 MHz. Here, the power usage is significantly higher and closer to the power draw of the maximum frequency of 1530 MHz. We also observe a high range of variation in power usage here, ranging from the lower end of about 90 W going up to 170 W. The reasons for this anomaly are currently unknown. Note that we are depicting a total of 114 profiles, which have been scheduled on different nodes (and thus different GPUs) of Lassen, making this anomaly statistically significant.

Figure 3b depicts the second scenario in our experiments, where we set a power cap on the GPU using the `-pl` option, and poll the frequency every ten seconds. The horizontal represents the power cap (range of 100 W to 300 W), and the vertical axis represents the average frequency observed over the profile. We have a total of 30 values of power caps, each with about 180–200 observations depending on how many profiles were created under a certain power cap. In this data, we clearly observe that when ddcMD is executed, there is no frequency throttling occurring under a power cap until about the 180 W power cap. It is only when we set the power below 180 W, that we observe a change in frequency of operation (and in turn, slowdown in performance). There is little-to-no variation at higher power caps. However, lower power caps, such as those at 115 W or 120 W, depict a lot of variation in effective frequency.

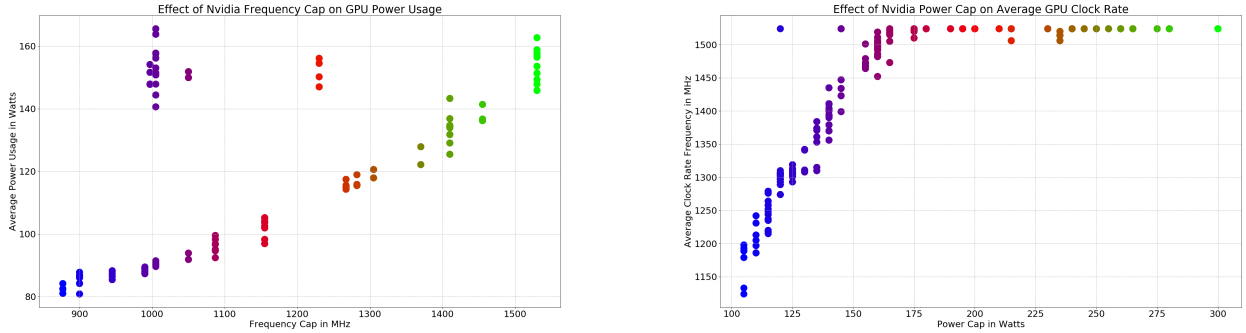
We attribute the variation in power usage (from Figure 3a) as well as the variation in frequency (from Figure 3b) to manufacturing differences between different GPU devices. We have not, however, exhaustively run experiments to quantify such variations across diverse benchmarks and to conclusively identify slower and faster GPU chips on our cluster. The presented study depicts ddcMD profiles only, and may not be generalized across different GPU benchmarks.

B. Anomalous Behavior and Observed Variation

We now present a detailed look at two specific knobs: (a) Setting the GPU frequency to 1005 MHz and observing power usage, and (b) Setting the GPU power cap to 120 W and observing frequency. Figure 4a shows the data across 114 profiles. The horizontal axis represents the profile identifier, and the vertical axis shows the power usage, sorted from minimum to maximum. As can be observed from the figure, the range of variation for power draw is quite high — ranging from 90 W to 170 W, about a $2\times$ difference in power usage at the exact same frequency across different GPUs. This indicates that setting frequencies is an unreliable way to achieve consistent power usage in a real cluster. This technique cannot be utilized in a power-constrained scenario as it can be quite unpredictable in terms of the power draw encountered. Figure 4b shows the scenario of a 120 W power cap across 196 profiles. Here too, we see a variation in effective frequencies ranging from 1250 MHz to 1530 MHz, a factor of $1.22\times$. It should be noted, however, that the range of variation observed here in effective frequencies is lower than that of the first scenario, which indicates that capping power could lead to relatively consistent performance in a power-constrained scenario.

C. Impact on Application Performance

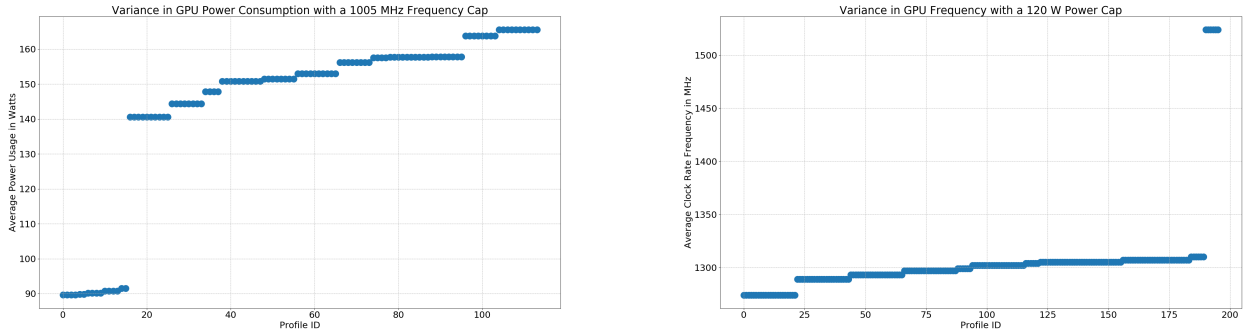
Finally, we discuss the impact of both frequency capping and power capping on application performance. We look at two performance metrics for the applications, the time taken for a molecular dynamics simulation step (MD step), and the



(a) Setting Frequencies

(b) Setting Power Caps

Fig. 3: Relationship between GPU Power and GPU Frequency in the two scenarios under test. In the first figure, we set GPU frequencies and measure GPU power usage. In the second figure, we set GPU power caps and measure GPU frequency.



(a) Variation in Power Usage

(b) Variation in Effective Frequency

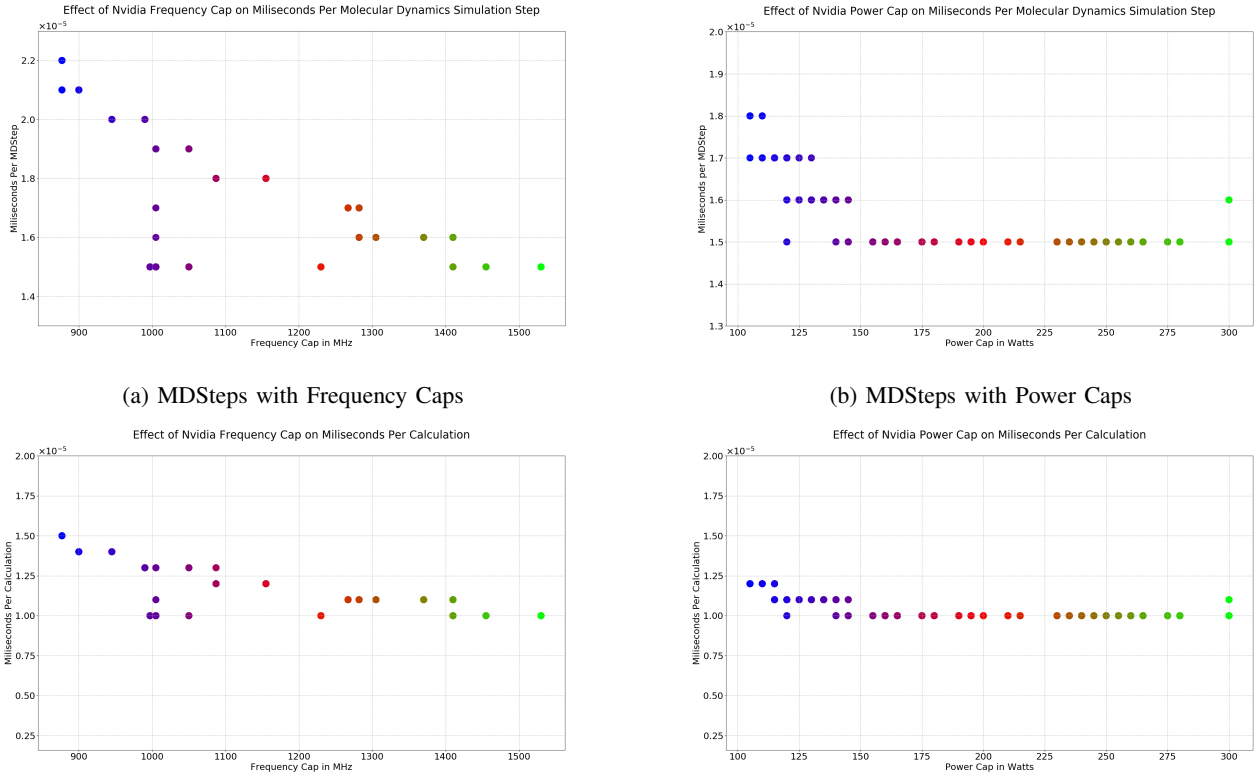
Fig. 4: Detailed look at the extent of variation at (a) 1005 Mhz and (b) 120 W. The first figure shows 114 profiles, and the second figure shows 196 profiles.

time taken for a particle's calculation. More specifically, the MD step metric is the amount of time in milliseconds that the program took to complete a molecular dynamics simulation step for a single particle. Likewise, the time to perform a calculation metric is the amount of time in milliseconds that the program took to perform a single particles energy or location calculation (which is a part of the MD step process).

Figure 5a and Figure 5b show the observed timings for MD steps in both the scenarios. The horizontal axes represent the selected frequency and power cap, respectively, and the vertical axes show time in milliseconds. Figure 5c and Figure 5d, show the calculations. A key observation from these plots is that when we set frequencies, we observe significant variation in the reported time. It is currently unclear why this is the case. Typically, when we set frequency, we expect lesser variation when compared to the scenario of setting power caps. When setting power caps, we see more consistent performance, with lesser slowdown as well as lesser variation. Reducing power from 300 W to 170 W on the GPU had no impact on ddcMD performance in either metric. As we go below the power cap of 170 W, we see some impact on performance. Reducing power to the lowest value of 105 W (1/3rd of maximum) resulted in an effective frequency reduction of 26% and a slowdown of 17%. We also observed a higher slowdown on the MDStep metric than the calculations metric. In order to clearly quantify the reasons for the variation (and slowdowns), we need to collect additional data with performance counters when setting frequencies as well as when setting power caps. This is part of our future work, where we plan on exploring this further with `nvprof` performance counter data and correlating it with the power and frequency observations.

V. Related Work

Energy and power research has focused on advancements at both software and hardware levels. Among architectures, the trend has been toward leveraging accelerators such as GPUs or FPGAs that lead to higher power per watt [2]. Additional non Von-Neumann architectures are actively being researched as well [50]. In software, most research in HPC has focused on saving energy [8], [13], [17], [51] and maximizing performance under power constraints [52]–[54]. Analyzing critical paths of applications and distributing power intelligently based on application characteristics has been studied for runtime systems [15], [48], [55] as well as for job scheduling [11], [56]–[62]. This research



(a) MDSteps with Frequency Caps

(b) MDSteps with Power Caps

(c) Calculations with Frequency Caps

(d) Calculations with Power Caps

Fig. 5: Impact of frequency capping and power capping on application performance. The first set of figures represent MDSteps data, and the second set of figures represent calculations.

has primarily focused on simpler proxy applications and traditional manycore server CPUs. There has been limited work that has focused on energy savings and analysis for workflows [63]–[66], and limited research on GPU frequency scaling and GPU power capping [26]–[31]. Additionally, while manufacturing variation has been studied for CPUs [20]–[23], no studies have explored performance variation or reproducibility on GPUs. Our work is a first-of-its-kind to explore GPU power capping and frequency scaling while studying variation and performance for workflows, and research in performance tools, predictions, and improved utilization on modern architectures and diverse workflows is an open research area.

VI. Conclusions and Future Work

In this paper, we presented initial results on GPU power capping, GPU frequency capping, and variation in performance on the ddcMD code, which is a part of the MuMMI workflow. We showed that GPU power capping within a scientific workflow is an effective way of improving HPC cluster power efficiency, and showed that reducing power from 300W to 200W on the ddcMD application had no impact on its performance. Even at the low value of a GPU power cap of 105 W (1/3rd of maximum), we saw a slowdown of only 17% on ddcMD performance. We also observed that counterintuitive to what we know about CPUs, performance variation resulting from potential lithography differences is lower when setting power caps as opposed to setting direct frequencies on GPUs. In our observations, setting GPU frequencies resulted in unreliable and unpredictable power usage, increased variation, and anomalous behavior at certain frequencies such as 1005MHz. These results are useful from the point of view of understanding the performance of future workflows that are both complex and highly heterogeneous. Future work includes conducting a detailed analysis of the other components of the MuMMI workflow, measuring performance metrics with tools such as `nvprof`, and building regression models to predict performance of workflows under power caps.

VII. Acknowledgments

We thank Adam Berstch, Greg Lee, and, John Gyllenhaal from Livermore Computing for their help in getting power capping enabled through their `nv_powercap` utility (a wrapper for `nvidia-smi`) and for trusting us with limited `sudo` privileges that are required to conduct these complex power capping based experiments on the Lassen supercomputer. This research would have been impossible without their guidance. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-CONF-784983).

REFERENCES

[1] K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, J. Hiller, S. Karp, S. Keckler, D. Klein, R. Lucas, M. Richards, A. Scarpelli, S. Scott, A. Snavely, T. Sterling, R. S. Williams, K. Yelick, K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, J. Hiller, S. Keckler, D. Klein, P. Kogge, R. S. Williams, and K. Yelick, "Exascale Computing Study: Technology Challenges in Achieving Exascale Systems," 2008.

[2] E. Strohmaier, J. Dongarra, H. Simon, and M. Meuer, "Top500 Supercomputer Sites," November 2014.

[3] L. L. N. Laboratory, "Sierra," https://hpc.llnl.gov/hardware/platforms/sierra, Lawrence Livermore National Laboratory, August 2018, retrieved July 30, 2018.

[4] O. R. N. Laboratory, "Summit," https://www.olcf.ornl.gov/summit/, Oak Ridge National Laboratory, August 2018, retrieved July 30, 2018.

[5] InsideHPC, "Power Consumption is the Exascale Gorilla in the Room," 2010.

[6] N. Bates, G. Ghatikar, G. Abdulla, G. A. Koenig, S. Bhalachandra, M. Sheikhalishahi, T. Patki, B. Rountree, and S. W. Poole, "Electrical Grid and Supercomputing Centers: An Investigative Analysis of Emerging Opportunities and Challenges," *Informatik Spektrum*, vol. 38, no. 2, pp. 111–127, 2015.

[7] J. Meng, S. McCauley, F. Kaplan, V. J. Leung, and A. K. Coskun, "Simulation and Optimization of HPC Job Allocation for Jointly Reducing Communication and Cooling Costs," *Sustainable Computing: Informatics and Systems*, vol. 6, pp. 48 – 57, 2015, special Issue on Selected Papers from 2013 International Green Computing Conference (IGCC).

[8] T. Patki, N. Bates, R. Ghatikar, A. Clausen, S. Klingert, G. Abdulla, and M. Sheikhalishahi, "Supercomputing Centers and Electricity Service Providers: A Geographically Distributed Perspective on Demand Management in Europe and the United States," 06 2016, pp. 243–260.

[9] T. Patki, D. K. Lowenthal, B. L. Rountree, M. Schulz, and B. R. de Supinski, "Economic Viability of Hardware Overprovisioning in Power-constrained High Performance Computing," in *Proceedings of the 4th International Workshop on Energy Efficient Supercomputing*, ser. E2SC '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 8–15. [Online]. Available: https://doi.org/10.1109/E2SC.2016.12

[10] O. Sarood, A. Langer, L. V. Kale, B. Rountree, and B. R. de Supinski, "Optimizing Power Allocation to CPU and Memory Subsystems in Overprovisioned HPC Systems," in *IEEE International Conference on Cluster Computing*, Sept 2013, pp. 1–8.

[11] T. Patki, A. Sasidharan, M. Maiterth, D. Lowenthal, B. Rountree, M. Schulz, and B. de Supinski, "Practical Resource Management in Power-Constrained, High Performance Computing," in *High Performance Parallel and Distributed Computing (HPDC)*, Jun. 2015.

[12] P. Bailey, D. Lowenthal, V. Ravi, B. Rountree, M. Schulz, and B. de Supinski, "Adaptive Configuration Selection for Power-Constrained Heterogeneous Systems," in *International Conference on Parallel Processing*, ser. ICPP '14, 2014.

[13] B. Rountree, D. K. Lowenthal, B. R. de Supinski, M. Schulz, V. Freeh, and T. Bletch, "Adagio: Making DVS Practical for Complex HPC Applications," in *International Conference on Supercomputing*, Jun. 2009.

[14] D. A. Ellsworth, A. D. Malony, B. Rountree, and M. Schulz, "POW: System-wide Dynamic ReEallocation of Limited Power in HPC," in *High Performance Parallel and Distributed Computing (HPDC)*, June 2015.

[15] A. Marathe, P. Bailey, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski, "A Run-time System for Power-constrained HPC Applications," in *In International Supercomputing Conference (ISC)*, July 2015.

[16] B. Li, H.-C. Chang, S. Song, C.-Y. Su, T. Meyer, J. Mooring, and K. W. Cameron, "The Power-Performance Tradeoffs of the Intel Xeon Phi on HPC Applications," in *Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International*. IEEE, 2014, pp. 1448–1456.

[17] R. Cochran, C. Hankendi, A. K. Coskun, and S. Reda, "Pack & Cap: Adaptive DVFS and Thread Packing Under Power Caps," in *Proceedings of the 44th annual IEEE/ACM international symposium on microarchitecture*. ACM, 2011, pp. 175–185.

[18] S. Wallace, V. Vishwanath, S. Coghlan, J. Tramm, Z. Lan, and M. Papkay, "Application Power Profiling on IBM Blue Gene/Q," in *Cluster Computing (CLUSTER), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1–8.

[19] T. Patki, E. Ates, A. Coskun, and J. Thiagarajan, "Understanding Simultaneous Impact of Network QoS and Power on HPC Application Performance," in *Computational Reproducibility at Exascale (CRE'18), Supercomputing Workshop 2018*, Nov 2018.

[20] B. Rountree, D. H. Ahn, B. R. de Supinski, D. K. Lowenthal, and M. Schulz, "Beyond DVFS: A First Look at Performance under a Hardware-Enforced Power Bound," in *IPDPS Workshops (HPPAC)*. IEEE Computer Society, 2012, pp. 947–953.

[21] Y. Inadomi, T. Patki, K. Inoue, M. Aoyagi, B. Rountree, M. Schulz, D. Lowenthal, Y. Wada, K. Fukazawa, M. Ueda, M. Kondo, and I. Miyoshi, "Analyzing and mitigating the impact of manufacturing variability in power-constrained supercomputing," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '15, 2015.

[22] J. W. Tschanz, J. T. Kao, S. G. Narendra, R. Nair, D. A. Antoniadis, A. P. Chandrakasan, and V. De, "Adaptive Body Bias for Reducing Impacts of Die-to-die and Within-die Parameter Variations on Microprocessor Frequency and Leakage," *Solid-State Circuits, IEEE Journal of*, vol. 37, no. 11, pp. 1396–1402, Nov 2002.

[23] S. Jilla, "Minimizing The Effects of Manufacturing Variation During Physcial Layout," *Chip Design Magazine*, 2013, http://chipdesignmag.com/display.php?articleId=2437.

[24] F. H. Streitz, J. N. Glosli, and M. V. Patel, "Beyond finite-size scaling in solidification simulations," *Phys. Rev. Lett.*, vol. 96, p. 225701, Jun 2006. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.96.225701

[25] F. Di Natale, H. Bhatia, T. S. Carpenter, C. Neale, S. K. Schumacher, T. Oppelstrup, L. Stanton, X. Zhang, S. Sundram, T. R. W. Scogland, G. Dharuman, M. P. Surh, Y. Yang, C. Misale, L. Schneidenbach, C. Costa, C. Kim, B. D'Amora, S. Gnanakaran, D. V. Nissley, F. Streitz, F. C. Lightstone, P.-T. Bremer, J. N. Glosli, and H. I. Ingólfsson, "A massively parallel infrastructure for adaptive multiscale simulations: Modeling RAS initiation pathway for cancer," in *Supercomputing '19: The International Conference for High Performance Computing, Networking, Storage, and Analysis*. New York, NY, USA: ACM, November 2019, to appear.

[26] Q. Zhu, B. Wu, X. Shen, L. Shen, and Z. Wang, "Co-Run Scheduling with Power Cap on Integrated CPU-GPU Systems," in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2017, pp. 967–977.

[27] M. Peres, "Reverse Engineering Power Management on NVIDIA GPUs - Anatomy of an Autonomic-ready System," in *ECRTS, Operating Systems Platforms for Embedded Real-Time applications 2013*, Paris, France, Jul. 2013. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00853849

[28] R. A. Bridges, N. Imam, and T. M. Mintz, "Understanding GPU Power: A Survey of Profiling, Modeling, and Simulation Methods," *ACM Comput. Surv.*, vol. 49, no. 3, pp. 41:1–41:27, Sep. 2016. [Online]. Available: http://doi.acm.org/10.1145/2962131

[29] B. Dutta, V. Adhinarayanan, and W.-c. Feng, "GPU Power Prediction via Ensemble Machine Learning for DVFS Space Exploration," in *Proceedings of the 15th ACM International Conference on Computing Frontiers*, ser. CF '18. New York, NY, USA: ACM, 2018, pp. 240–243. [Online]. Available: http://doi.acm.org/10.1145/3203217.3203273

[30] X. Mei, Q. Wang, and X. Chu, "A Survey and Measurement Study of GPU DVFS on Energy Conservation," *Digital Communications and Networks*, vol. 3, no. 2, pp. 89 – 100, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352864816300736

[31] R. Nath and D. Tullsen, "The CRISP Performance Model for Dynamic Voltage and Frequency Scaling in a GPGPU," in *Proceedings of the 48th International Symposium on Microarchitecture*, ser. MICRO-48. New York, NY, USA: ACM, 2015, pp. 281–293. [Online]. Available: http://doi.acm.org/10.1145/2830772.2830826

[32] GBD 2015 Disease and Injury Incidence and Prevalence Collaborators, "Global, regional, and national incidence, prevalence, and years lived with disability for 310 diseases and injuries, 1990?2015: a systematic analysis for the global burden of disease study 2015," *Lancet*, vol. 388, no. 10053, pp. 1545–1602, 2016.

[33] I. A. Prior, P. D. Lewis, and C. Mattos, "A comprehensive survey of ras mutations in cancer," *Cancer Research*, vol. 72, no. 10, pp. 2457–2467, 2012.

[34] T. S. Carpenter, C. A. López, C. Neale, C. Montour, H. I. Ingólfsson, F. Di Natale, F. C. Lightstone, and S. Gnanakaran, "Capturing phase behavior of ternary lipid mixtures with a refined martini coarse-grained force field," *Journal of chemical theory and computation*, vol. 14, no. 11, pp. 6050–6062, 2018.

[35] G. S. Ayton, J. L. McWhirter, P. McMurty, and G. A. Voth, "Coupling Field Theory with Continuum Mechanics: A Simulation of Domain

Formation in Giant Unilamellar Vesicles," *Biophysical Journal*, vol. 88, no. 6, pp. 3855–3869, 2005.

[36] "MOOSE," https://moose.inl.gov/SitePages/Home.aspx.

[37] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess, and E. Lindahl, "Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers," *SoftwareX*, vol. 1-2, pp. 19 – 25, 2015.

[38] F. H. Streitz, J. N. Glosli, P. M. V., C. B., Y. R. K., de Supinski B. R., S. J., and G. J. A., "100+ tflop solidification simulations on bluegene/l," in *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, ser. SC '05. New York, NY, USA: ACM, 2005. [Online]. Available: http://www.cresco.enea.it/SC05/schedule/pdf/pap307.pdf

[39] J. N. Glosli, D. F. Richards, K. J. Caspersen, R. E. Rudd, J. A. Gunnels, and F. H. Streitz, "Extending stability beyond cpu millennium: A micron-scale atomistic simulation of kelvin-helmholtz instability," in *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, ser. SC '07. New York, NY, USA: ACM, 2007, pp. 58:1–58:11. [Online]. Available: http://doi.acm.org/10.1145/1362622.1362700

[40] S. J. Marrink, H. J. Risselada, S. Yefimov, D. P. Tieleman, and A. H. de Vries, "The MARTINI Force Field: Coarse Grained Model for Biomolecular Simulations," *The Journal of Physical Chemistry B*, vol. 111, no. 27, pp. 7812–7824, Jul. 2007. [Online]. Available: https://pubs.acs.org/doi/10.1021/jp071097f

[41] D. H. Ahn, J. Garlick, M. Grondona, D. Lipari, B. Springmeyer, and M. Schulz, "Flux: A Next-Generation Resource Management Framework for Large HPC Centers," in *43rd International Conference on Parallel Processing Workshops*, Sept 2014.

[42] F. D. Natale, "Maestro workflow conductor (maestrowf)," https://github.com/LLNL/maestrowf, Lawrence Livermore National Laboratory, August 2018, retrieved Aug 11, 2018.

[43] S. Shende and A. D. Malony, "The Tau Parallel Performance System," *IJHPCA*, vol. 20, no. 2, pp. 287–311, 2006.

[44] L. Adhianto, S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey, and N. R. Tallent, "HPCTOOLKIT: Tools for Performance Analysis of Optimized Parallel Programs," *Concurr. Comput. : Pract. Exper.*, vol. 22, no. 6, pp. 685–701, Apr. 2010. [Online]. Available: http://dx.doi.org/10.1002/cpe.v22:6

[45] H. Jagode, A. YarKhan, A. Danalis, and J. Dongarra, "Power Management and Event Verification in PAPI," in *Tools for High Performance Computing 2015*, A. Knüpfer, T. Hilbrich, C. Niethammer, J. Gracia, W. E. Nagel, and M. M. Resch, Eds. Cham: Springer International Publishing, 2016, pp. 41–51.

[46] A. Agelastos, B. Allan, J. Brandt, P. Cassella, J. Enos, J. Fullop, A. Gentile, S. Monk, N. Naksinehaboon, J. Ogden, M. Rajan, M. Showerman, J. Stevenson, N. Taerat, and T. Tucker, "The Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '14, 2014.

[47] B. Rountree and S. Labasan, "Libmsr," https://github.com/LLNL/libmsr.

[48] J. Eastep, S. Sylvester, C. Cantalupo, B. Geltz, F. Ardanaz, A. Al-Rawi, K. Livingston, F. Keceli, M. Maiterth, and S. Jana, "Global Extensible Open Power Manager: A Vehicle for HPC Community Collaboration on Co-Designed Energy Management Solutions," in *High Performance Computing*, J. M. Kunkel, R. Yokota, P. Balaji, and D. Keyes, Eds. Cham: Springer International Publishing, 2017, pp. 394–412.

[49] NVIDIA, "NVIDIA System Management Interface," 2018. [Online]. Available: https://developer.nvidia.com/nvidia-system-management-interface

[50] E. J. Fuller, S. T. Keene, A. Melianas, Z. Wang, S. Agarwal, Y. Li, Y. Tuchman, C. D. James, M. J. Marinella, J. J. Yang, A. Salleo, and A. A. Talin, "Parallel Programming of an Ionic Floating-gate Memory Array for Scalable Neuromorphic Computing," *Science*, vol. 364, no. 6440, pp. 570–574, 2019.

[51] B. Rountree, D. K. Lowenthal, S. Funk, V. W. Freeh, B. de Supinski, and M. Schulz, "Bounding Energy Consumption in Large-Scale MPI Programs," in *Supercomputing*, Nov. 2007.

[52] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski, "Exploring Hardware Overprovisioning in

[54] D. Ellsworth, T. Patki, M. Schulz, B. Rountree, and A. Malony, "A Unified Platform for Exploring Power Management Strategies," in *2016* Power-constrained, High Performance Computing," in *International Conference on Supercomputing*, June 2013.

[53] O. Sarood, A. Langer, A. Gupta, and L. V. Kale, "Maximizing Throughput of Overprovisioned HPC Data Centers Under a Strict Power Budget," in *Supercomputing*, Nov. 2014.
4th International Workshop on Energy Efficient Supercomputing (E2SC), Nov 2016, pp. 24–30.

[55] A. Marathe, R. Anirudh, N. Jain, A. Bhatele, J. Thiagarajan, B. Kailkhura, J.-S. Yeom, B. Rountree, and T. Gamblin, "Performance Modeling Under Resource Constraints Using Deep Transfer Learning," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '17. New York, NY, USA: ACM, 2017, pp. 31:1–31:12. [Online]. Available: http://doi.acm.org/10.1145/3126908.3126969

[56] A. K. Coskun, R. Strong, D. M. Tullsen, and T. Simunic Rosing, "Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 1. ACM, 2009, pp. 169–180.

[57] A. K. Coskun, T. S. Rosing, and K. Whisnant, "Temperature aware task scheduling in mpsocs," in *Proceedings of the conference on Design, automation and test in Europe*. EDA Consortium, 2007, pp. 1659–1664.

[58] Z. Zhang, M. Lang, S. Pakin, and S. Fu, "Trapped Capacity: Scheduling under a Power Cap to Maximize Machine-room Throughput," in *Proceedings of the 2nd International Workshop on Energy Efficient Supercomputing*. IEEE Press, 2014, pp. 41–50.

[59] D. Bodas, J. Song, M. Rajappa, and A. Hoffman, "Simple Power-aware Scheduler to Limit Power Consumption by HPC System Within a Budget," in *Proceedings of the 2nd International Workshop on Energy Efficient Supercomputing*. IEEE Press, 2014, pp. 21–30.

[60] Z. Zhou, Z. Lan, W. Tang, and N. Desai, "Reducing Energy Costs for IBM Blue Gene/P via Power-Aware Job Scheduling," in *Job Scheduling Strategies for Parallel Processing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014, pp. 96–115.

[61] Y. Georgiou, T. Cadeau, D. Glesser, D. Auble, M. Jette, and M. Hautreux, "Energy Accounting and Control with SLURM Resource and Job Management System," in *Distributed Computing and Networking*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014, vol. 8314, pp. 96–118.

[62] N. Gholkar, F. Mueller, B. Rountree, and A. Marathe, "PShifter: Feedback-based Dynamic Power Shifting Within HPC Jobs for Performance," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '18. New York, NY, USA: ACM, 2018, pp. 106–117. [Online]. Available: http://doi.acm.org/10.1145/3208040.3208047

[63] E. Deelman, T. Peterka, I. Altintas, C. D. Carothers, K. K. van Dam, K. Moreland, M. Parashar, L. Ramakrishnan, M. Taufer, and J. Vetter, "The Future of Scientific Workflows," *The International Journal of High Performance Computing Applications*, vol. 32, no. 1, pp. 159–175, 2018.

[64] R. F. da Silva, T. Fahringer, J. J. Durillo, and E. Deelman, "A Unified Approach for Modeling and Optimization of Energy, Makespan and Reliability for Scientific Workflows on Large-Scale Computing Infrastructures," in *Workshop on Modeling and Simulation of Systems and Applications (MODSIM)*, 2014, funding Acknowledgements: DOE contract for dv/dt ER26110). [Online]. Available: http://pegasus.isi.edu/publications/2014/2014-modsim.pdf

[65] R. Ferreira da Silva, A.-C. Orgerie, H. Casanova, R. Tanaka, E. Deelman, and F. Suter, "Accurately Simulating Energy Consumption of I/O-Intensive Scientific Workflows," in *Computational Science – ICCS 2019*, J. M. F. Rodrigues, P. J. S. Cardoso, J. Monteiro, R. Lam, V. V. Krzhizhanovskaya, M. H. Lees, J. J. Dongarra, and P. M. Sloot, Eds. Cham: Springer International Publishing, 2019, pp. 138–152.

[66] Q. Zhu, J. Zhu, and G. Agrawal, "Power-Aware Consolidation of Scientific Workflows in Virtualized Environments," in *SC '10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2010, pp. 1–12.