Exploring the Optimal Platform Configuration for Power-constrained HPC Workflows

Kun Tang[1], Xubin He[1], Saurabh Gupta*[2], Sudharshan S. Vazhkudai[3], and Devesh Tiwari[4]

[1]Temple University [2]Intel Lab [3]Oak Ridge National Laboratory [4]Northeastern University

***Abstract*—In high-performance computing (HPC) workflows, data analytics is typically utilized to gain insights from scientific simulations. Approaching the era of exascale, online analysis is gaining popularity due to the savings of I/O to persistent storage. As computing capability keeps growing, power consumption is becoming critical to HPC facilities. Enforcing power limits is emerging as a practical trend for power-constrained HPC facilities. However, it remains unclear how to choose the appropriate power limits for various HPC workflows and how to distribute the power limit of a workflow between simulation and analysis. In addition, given a power limit, it is unclear what the optimal scales and power capping levels are for various workflows, especially when taking reliability into account. In order to resolve these issues in power-constrained HPC, in this paper, we propose a reliability-aware model to determine the aforementioned platform configurations for HPC workflows. We also validate our model and present model-driven studies for a wide range of real-system scenarios. Our study reveals interesting insights about how platform configuration affects the performance and energy efficiency of HPC workflows under power constraints.**

I. Introduction

High-performance computing (HPC) workflows comprise of large-scale scientific simulations and data analysis tasks, which produce and consume data, respectively. Scientific simulations run iteratively and produce raw data periodically [1]. In order to gain scientific insights, a series of data analysis tasks are performed on the raw data. Using offline analysis, data go through the persistent storage systems within the workflow, which incurs enormous overheads. By comparison, online analysis (in-situ and in-transit) is gaining popularity in extreme-scale scientific analysis, because it usually reduces the volume of raw data by several orders of magnitude [2]. In in-situ and in-transit workflows, analysis runs simultaneously with scientific simulation in a pipelined manner.

Approaching the era of exascale computing, power consumption becomes a critical concern to HPC facilities. As a result, the U.S. Department of Energy has set a power constraint of 20 MW to each exascale machine [3]. Enforcing power limits at all layers is emerging as a practical trend for power-constrained HPC facilities, especially for future exascale systems. Enforcing power limits directly impacts the performance and potentially impacts the energy efficiency. However, it is not clear how to choose the appropriate power limits for various HPC workflows. Furthermore, exascale storage I/O requirement is projected to have an O(100) increase as compared with petascale computing [4], which requires more powerful data analysis. In HPC workflows, assigning too much power to one part (either simulation or analysis) affects the other part negatively under power constraints. However, it remains unclear how to distribute the power limit between the simulation and analysis in a workflow. In addition, facing power constraints, utilizing hardware over-provisioning to improve performance is also gaining popularity [5]. Under a power limit, it is essential to choose the appropriate power capping levels and scales for simulation and analysis respectively. Both power capping and scaling affects the system reliability, which follow inverse variation under power constraint. For example, given a power limit, an increment in the power capping level leads to a decrement in scale. Lower power capping level can also lead to a higher MTBF (mean time between failures) due to reduced temperature level of the device. In addition, a smaller scale is always accompanied by a larger MTBF. Therefore, it is essential to explore the trade-off between power capping and scaling in order to improve the system reliability which has significant impact on application execution time and energy consumption. However, under power limits, it is unclear what the optimal power capping levels and scales are for HPC workflows.

To the best of our knowledge, no prior study has investigated the optimal platform configuration for power-constrained HPC workflows. In this paper, we propose a reliability-aware model to determine the optimal platform configuration, which includes power allocation and distribution, power capping levels, and scales for power-constrained HPC workflows. This work is based on real-system measurements, statistical techniques, data obtained from leadership computing facilities (i.e., supercomputers facilitated by the Department of Energy), and analytical models.

Contributions: First, we study and quantify the impact of power capping on execution time and energy consumption for a variety of scientific simulation and analysis applications. Second, we propose a reliability-aware model to predict the optimal platform configuration for power-constrained HPC workflows, considering both execution time and energy consumption. Third, we validate our model and present model-driven studies for a wide range of real-system scenarios. Our study reveals several interesting and new insights about how platform configuration affects the execution time and energy consumption of HPC workflows under power constraints.

The remainder of this paper is organized as follows. We

*Saurabh Gupta contributed to this work while working at ORNL.

discuss the related work and background in section II and section III, respectively. The impact of power capping and scaling on performance is studied in section IV. The optimal platform configuration model is introduced in section V. We present model validation and model-driven studies in section VI. Section VII summarizes the paper.

II. Related Work

The continuous growth in computing capability exacerbates the I/O bottleneck issues in HPC. Online analysis draws considerable interest in extreme-scale scientific analysis, since it usually reduces the volume of scientific raw data significantly. Deelman et al. [6] and Mandal et al. [7] proposed techniques to perform performance modeling and diagnosis for HPC workflows. Li et al. [8] and Zhang et al. [9] proposed to run both simulation and analysis workloads using separate cores on a chip. Besides, Bennett et al. [2] proposed to combine in-situ and in-transit processing to achieve the trade-off between performance and data movement cost.

The ever-increasing computing capability also significantly escalates the power demand. Power constraints are becoming a first-order concern for HPC facilities [10]. Existing research on power-aware scheduling mainly focus on energy savings of scientific simulations [11][12]. Bailey et al. [13] proposed a predictive model to estimate the performance and power for scientific simulations. It is alternative to the regression analysis on power capping impact, which our model bases on. Langer et al. [14] evaluated the performance and energy trade-off for scientific simulations under power constraints. In exascale computing, the O(100) increase in storage I/O requirement [4] gives prominence to the online analysis of HPC workflows. This paper targets HPC workflows and investigates the impacts of platform configurations on performance, energy, and energy efficiency under power constraints.

Facing power constraints, on one side, researchers propose to enforce power limits at all layers for power-constrained HPC facilities. At the processor layer, power capping is generally achieved through either Dynamic Voltage and Frequency Scaling [15], or throttling by idle cycle insertion [16]. Under power capping, processor manufacturing variability causes performance loss. Inadomi et al. [17] and Gholkar et al. [18] proposed to address this issue through applying non-uniform power capping. On the other side, hardware over-provisioning draws considerable research interests in power-constrained HPC. The common goals shared by many researchers are maximizing the power utilization and maximizing the throughput (or minimizing the turnaround time) [19][20]. These works are complementary to the optimal platform configuration model proposed in this paper. Patki et al. [5] examined the feasibility of hardware over-provisioning in power-constrained HPC. They showed that over-provisioning leads to an average speedup of more than 50% comparing with the worst-case provisioning. However, the authors did not propose a model to derive the optimal configurations. Researchers have also investigated the power allocation schemes between CPU and memory for power-constrained scientific simulations [21][22][23]. These schemes are complementary to our work which focuses on the power allocation between different components of HPC workflows. Overall, none of the existing research in power-constrained HPC involves HPC workflows and system reliability.

III. Background and Overview

This work aims to model a pair of supercomputer and analytics cluster where simulation part is run on accelerators (e.g., GPU) and analysis part is run on CPUs. One such example is the Titan supercomputer and corresponding analytics cluster Rhea. The Titan supercomputer consists of 18,688 compute nodes, which is equipped with AMD Opteron 6274 CPUs and Nvidia Tesla K20X GPUs. The analysis cluster Rhea is equipped with Intel Xeon E5-2650 CPUs.

TABLE I: Benchmark domain and problem size (Psize).

	Program	**Domain**	**Psize**	
Rodinia	Kmeans	Data Mining	256K	**Analysis**
	BFS (Breadth-First Search)	Graph Algorithms	8M	
	NN (k-Nearest Neighbors)	Data Mining	5M	
	B+tree	Search	10M	
	SC (Streamcluster)	Data Mining	64K	
	LUD (LU Decomposition)	Linear Algebra	2K	**Simulation**
	LavaMD	Molecular Dynamics	4K	
	CFD (CFD Solver)	Fluid Dynamics	97K	
NPB	LU (Gauss-Seidel)			
	SP (Scalar Pentadiagonal)	Fluid Dynamics	B	
	BT (Block Tridiagonal)			

It should be noted that performing power-related experiments with HPC workflows directly on supercomputing production machines is not viable, because power measurement and power capping are currently not supported on the platforms. To overcome the platform limitation, on hardware side, we choose platforms that support power capping from the same generation as Titan and Rhea. We have intensively profiled simulation and analysis programs of HPC workflows, and observed that simulation is more efficient on GPU and analysis is more efficient on CPU in terms of both performance and energy efficiency, because analysis programs typically have more conditional branches and less floating-point operations compared with simulation programs. For simulation programs, we choose the same GPU platform, Nvidia Tesla K20. For analysis programs, we choose Intel Xeon E5-2670 (8 cores, 2.6 GHz) and Xeon E5-2603 (4 cores, 1.8 GHz). Both Xeon platforms and the GPU platform have 32 GB main memory and run Linux 2.6.32 kernel with GCC 4.4.7 compiler. CUDA Toolkit 5.5 is installed on the GPU platform. The host processor of the GPU platform is Intel Xeon E5-2630. Power and energy consumption for the GPU platform include the host processor and main memory, unless otherwise mentioned. On software side, these platforms cannot host large-scale HPC workflows which depend on platform-specific libraries. Therefore, we employ a wide variety of scientific simulation and analysis applications from Rodinia benchmark suite and NPB benchmark suite to mimic HPC workflows and to study the impact of power capping and scaling on execution time. Applications in Table I represent scientific simulation and analysis tasks. We have also characterized the instruction composition of these applications using TAU profiling tool [24] and PAPI counters [25] to ensure that they represent typical HPC workflows.

"Optimal Configuration" refers to the platform configuration that leads to the near-minimal workflow execution time or energy consumption under specific power constraints. The optimal platform configuration model takes scaling coefficients, power capping coefficients, and baseline figures of merit (FOMs) as inputs. The scaling coefficients (in Eq. 1 and Eq. 2) and power capping coefficients (in Eq. 3 and Eq. 4) are introduced in section IV, which need to be profiled for each workflow. Baseline figures of merit include execution time, scale, and power consumption, which are obtained when power capping is not enforced. The coefficients and baseline figures of merit can be obtained through offline tests. Based on the model inputs, solving Eq. 10 and Eq. 20 in section V, we can derive the optimal platform configurations. The model outputs the optimal power limits, node counts, power capping levels, and platform types for simulation and analysis respectively.

Librapl [5] and NVML [26] are used to profile CPU and GPU power consumption, respectively. Intel Power Governor [27] is utilized to cap the package power consumption, and is only supported on the Xeon platforms. Per-processor power capping is uniform for an application. Power limit is set on a per-application basis.

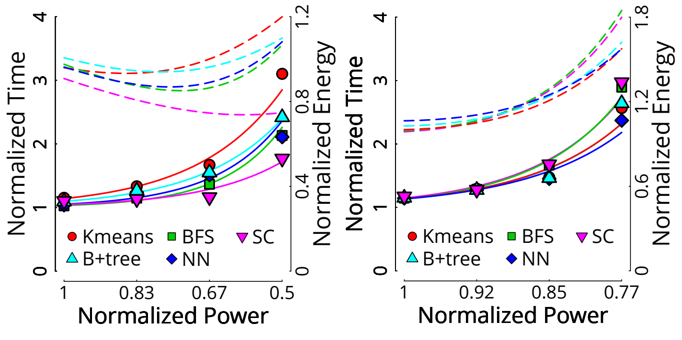
We recognize that our findings are bounded by the parameter settings. To mimic real-world scenarios, our simulation and model-driven studies employ parameters obtained from real-system experiments and large-scale application characteristics obtained from HPC facilities. We state our assumptions explicitly in an effort to increase reproducibility of the methodology, and clarify the scope of the work and the type of the system being modeled and evaluated. We believe that explicitly mentioning all (including hidden) assumptions can better serve the academic research community. Furthermore, as noted in the paper, these assumptions are realistic, grounded, and valid in real-world scenarios.

IV. Power Capping and Scaling

In this section, we statistically quantify the impact of power capping and scaling on execution time and energy consumption for scientific simulation and analysis.

First, we study the computation behaviors of scientific simulation and analysis under power capping. We evaluate the execution time and energy consumption of scientific analysis applications under different power capping levels on Xeon E5-2670. We find that, the average power consumption of each benchmark ranges from 63 to 78 watts. An effective power capping level for all benchmarks should be lower than 63 watts. The minimum package power that the power governor can enforce is 22 watts. Therefore, we choose power capping levels of 60, 50, 40, and 30 watts. Similarly, we choose effective power capping levels of 26, 24, 22, and 20 watts on Xeon E5-2603.

The impact of CPU power capping on simulation applications has been previously studied [22]. In this paper, we quantify the power capping impact on analysis applications, which typically have more conditional branches and less floating-point operations compared with simulation applications. The results show that the execution time of analysis applications increases dramatically as power capping level decreases. For example, the execution time on Xeon E5-2670 increases more rapidly when the power capping level reduces from 40 watts to 30 watts, as shown in Fig. 1a. The execution time of analysis applications on Xeon E5-2603 (Fig. 1b) shows similar increasing trends as on Xeon E5-2670, when power capping level decreases. We quantify the trends utilizing regression analysis, and observe that they can be fitted with exponential functions, as shown in Eq. 1.



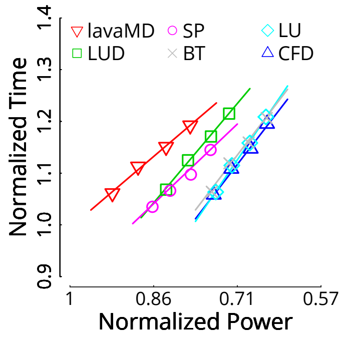
(a) Xeon E5-2670 (b) Xeon E5-2603

Fig. 1: Normalized execution time (solid lines) and normalized energy consumption (dotted lines) for analysis applications under power capping on Xeon platforms. X-axes are normalized to the highest (left most) power capping levels which are 60 watts and 26 watts respectively for E5-2670 and E5-2603 platforms.

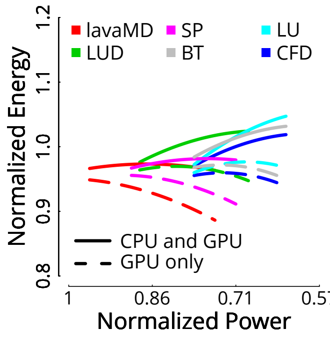
$$\overline{T_a}(P_i)/T_a = A \times e^{B \times Pi} + 1 \quad (1)$$

T_a is baseline execution time (without power capping), and $\overline{T_a}(P_i)$ denotes the execution time under power capping level P_i. e is Euler's number. A and B are coefficients of the regression function. The average R-squared value of regression functions for all the benchmarks is 0.96, which indicates statistically sound fit. In Fig. 1, energy consumption under various power capping levels are plotted as dotted lines. On Xeon E5-2670, we observe that the optimal power capping level optimized for minimal energy shifts from 50 watts (Kmeans) toward 30 watts (SC), as shown in Fig. 1a. By comparison, on Xeon E5-2603, the optimal power capping level optimized for minimal energy stabilizes at the highest power capping level for all benchmarks, as shown in Fig. 1b. Consequently, we do not apply power capping to Xeon E5-2603 for analysis applications.

Then, we evaluate the execution time and energy consumption under various power levels on GPU. The minimum power limit on GPU is 150 watts, which is too high to perform the power capping experiments. Alternatively, we mimic power capping based on frequency scaling, to obtain various execution time and power consumption pairs. As shown in Fig. 2a, execution time increases steadily as power consumption

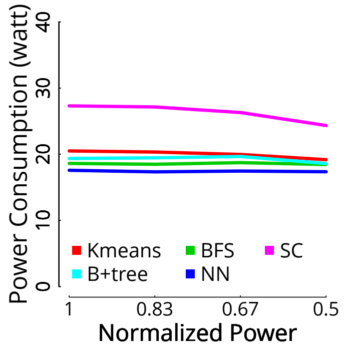


(a) Execution time

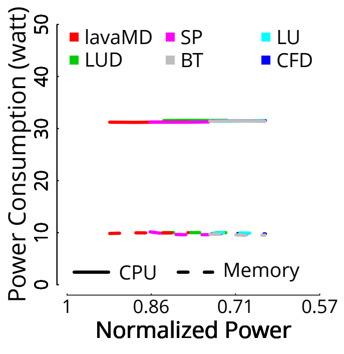


(b) Energy Consumption

Fig. 2: Execution time and energy consumption under various GPU power levels. Y-axis is truncated for better illustration. Time and energy are normalized to the highest frequency. X-axes are normalized to the highest (left most) power level which is 140 watts.



(a) CPU memory power



(b) GPU host power

Fig. 3: CPU memory power consumption and GPU host power consumption under various power capping levels. X-axes represent CPU power capping level and GPU power consumption which are normalized to the highest (left most) power levels 60 watts and 140 watts respectively in Fig. 3a and Fig. 3b.

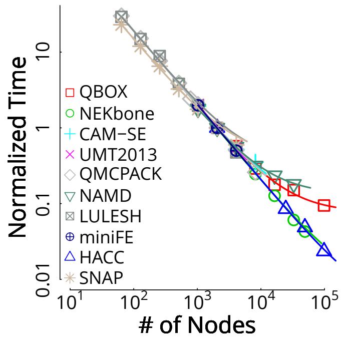
decreases. We quantify the trends with regression analysis, and observe that they can be fitted with linear functions, as shown in Eq. 2.

$$\overline{T_s}(P_i)/T_s = C \times Pi + D \tag{2}$$

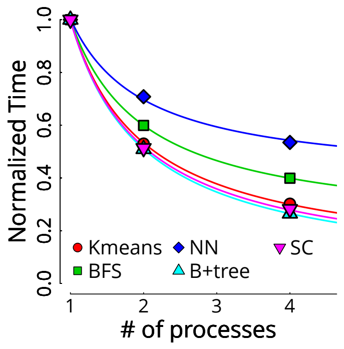
T_s is execution time under maximum power consumption, and $\overline{T_s}(P_i)$ denotes the execution time under power consumption P_i on GPU. C and D are coefficients of the regression function. The R-squared values of regression functions are all above 0.99, which indicates statistically sound fit. Fig. 2b shows the energy consumption under various GPU power levels. Energy consumption for only GPU is plotted as dotted lines. As GPU power drops, GPU energy consumption decreases for lavaMD and SP, and keeps relatively steady for the rest of benchmarks. However, in this paper, we also consider the host CPU and memory as a part of the GPU platform. The total energy consumption of the GPU platform is plotted as solid lines in Fig. 2b, which keeps relatively steady for lavaMD and SP, and increases for the rest of benchmarks.

Besides, we also study the impact of power capping on main memory and host power consumption. As shown in Fig. 3a, when processor power capping level decreases, the memory power consumption of Xeon E5-2670 decreases slightly for SC and remains steady for the other benchmarks. We profile the memory throughput and find that SC is at least 5 times the memory throughput of other benchmarks, which makes SC memory power consumption more sensitive to power capping. The power consumption of host CPU and memory keeps steady for all benchmarks under various GPU power levels, as illustrated in Fig. 3b.

Finding 1: *On GPU platforms, host side (CPU and memory) power consumption is independent of application characteristics and GPU power levels. When taking host CPU and main memory into account, running at lower power levels makes GPU less energy-efficient.*



(a) Simulation scaling results [28]



(b) Analysis scaling results

Fig. 4: Scaling results for simulation and analysis applications. Simulation scaling results are measured on BlueGene/Q and Cray XK7 with a maximum scale of 98,304 nodes, and are normalized to the scale of 2,048 nodes. In (a), axes are plotted in log scale for better illustration purpose. Analysis execution time is normalized to the scale of 1.

Finally, we quantify the scalability of scientific simulation and analysis. In terms of simulation applications, we exploit the CORAL scaling results [28] as shown in Fig. 4a. According to Amdahl's law, these data points can be fitted with rational functions as shown in Eq. 3.

$$S_s(x) = E/x + F \tag{3}$$

$S_s(x)$ is normalized (to the scale of 2,048 nodes) execution time and x is scale (i.e., number of nodes). E and F are coefficients of the regression function. The R-squared values of regression functions are all above 0.99, which indicates statistically sound fit.

With respect to online analysis, a group of analysis processors is usually assigned per unit of simulation nodes. We assume strong scaling within the group when processing data from one unit of simulation nodes, and weak scaling among groups when processing global data from the entire simulation. We admit that there are certain analysis stages that exhibit data dependencies among groups. Analysis applications can be decomposed into a massively parallel stage and a small-scale parallel or serial stage [2]. We focus on the massively parallel stage which is usually the main body of scientific analysis.

Afterwards, a small scale of processors can be utilized to finish up the task. As shown in Fig. 4b, execution time decreases in a sublinear manner for analysis applications when parallelism increases. These data points can be fitted with rational functions as in Eq. 4.

$$S_a(x) = G/x + H \tag{4}$$

$S_a(x)$ is normalized (to the scale of 1 process) execution time and x denotes scale (i.e., number of cores). G and H are coefficients of the regression function. R-squared values of these regression functions are all above 0.996.

V. Optimal Platform Configuration

In section IV, we have quantified the impact of power capping and scaling on execution time for both simulation and analysis. Based on these relationships, we develop a reliability-aware model to predict the optimal platform configurations for HPC workflows, which include power limits, power capping levels, and scales for scientific simulation and analysis respectively. The variables used in our model are listed in Table II.

TABLE II: Variables in the analytical model. Subscript 's' denotes simulation and subscript 'a' denotes analysis.

Variable	Description
$e_{s\|a}$	Variable energy consumption
$N_{s\|a}$	Baseline scale
n_{cores}	Number of cores per processor
$n_{s\|a}$	Variable scale
r_n	Ratio between n_a and n_s
$T_{s\|a}$	Baseline execution time
$t_{s\|a}$	Variable execution time
$p_{s\|a}$	Variable processor and memory power
P_{host}	Host power consumption for GPU
P_{mem}	Main memory power consumption
$\overline{P}$	Workflow power limit

A. Base Model

In order to facilitate the derivation of the reliability-aware model, first, we develop the base model in this subsection. Facing power constraints, workflows always run with a power limit ($\overline{P}$). Workflow power consumption comes from processor, memory, and network. Using optical links to replace copper links has become a practical trend in exascale computing, because power consumption of optical links remains steady as bit-rate scales up [29]. Therefore, the network power consumption remains constant and is excluded from power limit $\overline{P}$. The peak power consumption of workflows should always be under the power limit, as defined in formula 5.

$$n_s \times p_s + n_a \times p_a \leq \overline{P} \tag{5}$$

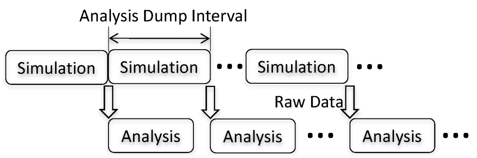


Fig. 5: Overview of workflow execution.

Simulation and analysis run in a pipelined manner, as illustrated in Fig. 5. In HPC workflows, online analysis reduces the amount of simulation data to persistent storage, which in return expedites the simulation. Therefore, analysis should finish earlier than simulation, rather than impeding simulation. The overall execution time of a workflow ($T_{workflow}$) is dominated by the part (either simulation or analysis) that takes longer time, as defined in Eq. 6.

$$T_{workflow} = \text{MAX}(t_s, t_a) \tag{6}$$

t_a is analysis execution time and t_s is simulation execution time, which are defined in Eq. 7 and Eq. 8, respectively. t_a and t_s are calculated based on the baseline execution time T_a and T_s, respectively, after scaling and power capping. In the optimal configuration sets, t_a is always smaller than t_s.

$$t_a = S_a(r_n \times n_{cores})/S_a(n_{cores}) \times \overline{T_a}(p_a - P_{mem}) \tag{7}$$

P_{mem} denotes the main memory power consumption, which is included in the energy calculation. Baseline analysis time (T_a) is the per-processor (n_{cores}) execution time without power capping, as shown in Eq. 1. Baseline analysis scale (N_a) is 1. Scaling factor is calculated based on Eq. 4, where r_n analysis processors are utilized to perform the same task as the baseline case.

$$t_s = S_s(n_s)/S_s(N_s) \times \overline{T_s}(p_s - P_{host}) \tag{8}$$

P_{host} denotes the host power consumption on the GPU platform, which consists of CPU and main memory power consumption. Host power consumption is included in the energy calculation. Baseline simulation time (T_s) is the execution time under baseline scale (N_s) without power capping, as shown in Eq. 2. Baseline simulation time and scale can be obtained from large-scale runs. Scaling factor is calculated based on Eq. 3.

The overall energy consumption of a workflow ($E_{workflow}$) consists of energy consumption from the simulation (e_s) and analysis (e_a), as shown in Eq. 9.

$$E_{workflow} = e_s + e_a = n_s \times p_s \times t_s + n_a \times p_a \times t_a \tag{9}$$

To be noted that network energy consumption is not included in $E_{workflow}$. As discussed earlier, power consumption of optical links remains steady as bit-rate scales up [29], which makes network energy consumption dependent on workflow execution time ($T_{workflow}$). As long as $T_{workflow}$ is minimized, network energy consumption is optimal.

The objective function $F(n_s, p_s, r_n, p_a)$ to achieve the minimal workflow execution time ($T_{workflow}$) is given in Eq. 10.

$$F(n_s, p_s, r_n, p_a) = minimize(t_s), \quad t_s \geq t_a \tag{10}$$

In order to solve the objective function, we should minimize t_s while ensuring that t_a is smaller than t_s. When analysis variables r_n and p_a are treated as constants in t_s, simulation time (t_s) is minimized when simulation scale (n_s) reaches the upper limit. According to formula 5, the upper limit of n_s can be expressed as a function of variable p_s, as shown in formula 11.

$$n_s \leq \overline{P}/(p_s + r_n \times p_a) \tag{11}$$

Based on formula 11, the minimum t_s is converted into a quadratic function of the variable p_s, as shown in Eq. 12.

$$t_s = S_s(\overline{P}/(p_s+r_n\times p_a))/S_s(N_s) \times \overline{T_s}(p_s-P_{host}) \tag{12}$$

Solving the quadratic function (Eq. 12), the x coordinate of the vertex (i.e., maximum t_s) is defined in Eq. 13.

$$x = -0.5 \times (D/C - P_{host} + r_n \times p_a + F/E \times \overline{P}) \tag{13}$$

As a result, p_s for minimal simulation execution time (t_s) is defined in Eq. 14.

$$p_s = \begin{cases} P_{max}, & \text{if } x \le (P_{max}+P_{min})/2; \\ P_{min}, & \text{if } x > (P_{max}+P_{min})/2. \end{cases} \tag{14}$$

P_{max} and P_{min} denote the maximum and minimum effective power capping level for simulation. In the solution above, analysis variables r_n and p_a are treated as constants at first, to derive the p_s for minimal t_s. In resolving the objective function $F(n_s,p_s,r_n,p_a)$, next, we enumerate the combinations of r_n and p_a to obtain the minimal t_s that satisfies the condition $t_s \ge t_a$.

B. Reliability-aware Model

In subsection V-A, we develop the base model to predict the optimal platform configurations for a failure-free environment. In reality, it is common to see failures happen, especially in large-scale HPC facilities. In this subsection, first, we revisit how the power-capping aware checkpointing model [10] calculates failure rate, wasted time, and OCI (Optimal Checkpointing Interval), as shown in Eq. 15 to Eq. 19. Then, we propose a reliability-aware model based on the base model, to consider failures in the real world.

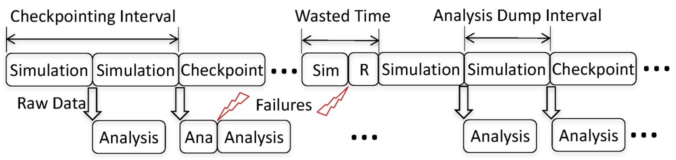


Fig. 6: Overview of workflow execution with failures, checkpoint, and restart.

Taking failures, checkpoint, and restart into account, the total execution time of scientific applications ($\overline{T_{total}}(P_i)$) includes computation time ($\overline{T_{comp}}(P_i)$), checkpointing time ($\overline{T_{chkp}}(P_i)$), and wasted time ($\overline{T_{waste}}(P_i)$), as illustrated in Fig. 6. Computation time is the time to perform scientific simulation (t_s) and data analysis (t_a). In order to tolerate failures, simulation applications perform checkpointing periodically with an interval σ. The analysis dump interval is independent of the checkpointing interval. Time to checkpoint (denoted as δ) is the time taken to store checkpoints into permanent storage systems. When failures occur, the simulation application rollbacks to the latest checkpoint, or the analysis application reprocesses the raw data. The lost work caused by such rollback is wasted time. The checkpointing time and wasted time are defined in Eq. 15 and Eq. 16, respectively.

$$\overline{T_{chkp}}(P_i) = (\frac{\overline{T_{comp}}(P_i)}{\sigma} - 1) \times \delta \tag{15}$$

$$\overline{T_{waste}}(P_i) = \frac{\overline{T_{comp}}(P_i)}{\sigma} \times (e^{\frac{\sigma+\delta}{\overline{MTBF}(P_i)}} - 1) \times(\tau \times (\sigma+\delta) + \tau) \tag{16}$$

τ represents the fraction of lost work due to failures. σ denotes OCI, which is derived by solving $\frac{d}{d\sigma}(\overline{T_{total}}(P_i)) = 0$, leading to the minimal total execution time ($\overline{T_{total}}(P_i)$). In this paper, in order to accurately predict OCI under all conditions, we utilize the MATLAB numeric solver "vpasolve" to calculate OCI.

In Eq. 16, $\overline{MTBF}(P_i)$ represents the MTBF (Mean Time Between Failures) at power capping level P_i, which is defined in Eq. 17.

$$\overline{MTBF}(P_i) = (MTTF_{base}/F_A(\overline{TEMP}(P_i)) + \gamma) \times N_s/n_s \tag{17}$$

$MTTF_{base}$ denotes the baseline mean time to fail. γ is the time to recover from failures. $F_A(x)$ is the acceleration factor of the Arrhenius Equation at temperature x, as defined in Eq. 18.

$$F_A(x) = e^{\frac{E_a}{k}\times(1/TEMP_{base} - 1/x)} \tag{18}$$

k is Boltzmann constant (8.617×10^{-5} eV/°K) and E_a is activation energy (0.7eV). $TEMP_{base}$ denotes the baseline temperature. In Eq. 17, $\overline{TEMP}(P_i)$ denotes processor temperature at power capping level P_i, which is defined in Eq. 19.

$$\overline{TEMP}(P_i) = I \times P_i + J \tag{19}$$

The sensitivity of processor temperature to changes in power capping (i.e., coefficients I and J) is dominated by the cooling infrastructure. Processor temperature increases linearly as power capping level grows, as illustrated in Eq. 19. When processor temperature gets higher, it becomes less reliable, as illustrated in Eq. 17.

We propose the reliability-aware model to explore various configuration sets and minimize the total execution time ($\overline{T_{total}}(P_i)$). Rather than minimizing the simulation time t_s as in the base model, the reliability-aware model minimizes the total execution time of simulation, which includes computation time, checkpointing time, and wasted time. Reliability plays a major role in the trade-off between checkpointing time and wasted time. In addition, under power constraints, power capping level and scale affect both reliability and computation time inversely. For example, an increment in power capping level is accompanied by decreased reliability level and shortened computation time. Under a power limit, this causes a decrement in scale, which occurs with increased reliability level and prolonged computation time. The reliability-aware model explores the various interplay among power capping level, scale, reliability, computation time, checkpointing time, and wasted time. The objective function is defined in Eq. 20.

$$F(n_s,p_s,r_n,p_a) = minimize(\overline{T_{total}}(p_s)), \quad t_s \ge t_a + \widetilde{t_a} \tag{20}$$

$\widetilde{t_a}$ is the wasted time of analysis applications. In order to avoid the interference between consecutive analysis tasks, the summation of analysis time (t_a) and analysis wasted time ($\widetilde{t_a}$) should not exceed the simulation time (t_s).

Total execution time ($\overline{T_{total}}(p_s)$) is determined by $\overline{T_{comp}}(p_s)$ and σ, both of which can be expressed as a function of power capping level p_s. In solving the objective function, we utilize the same methodology as discussed in subsection V-A. The difference lies in how to derive the optimal p_s to achieve minimal $\overline{T_{total}}(p_s)$. Solving $\frac{d}{dp_s}(\overline{T_{total}}(p_s)) = 0$ leads to the optimal power capping level p_s. In this paper, we utilize the MATLAB numeric solver "vpasolve" to calculate the optimal configurations.

VI. Evaluation and Model-driven Studies

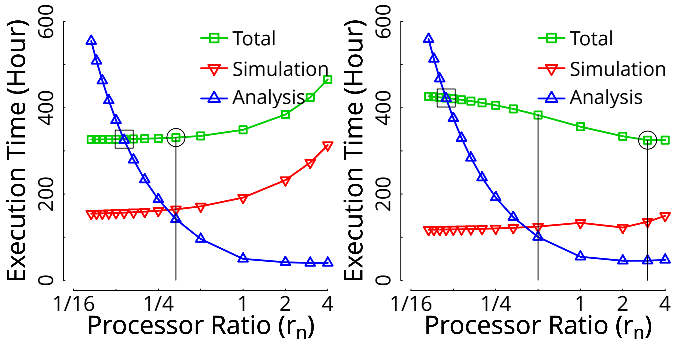
In this section, we evaluate the model in terms of scale, power capping level, and power limit. In addition, we also study the sensitivity of our model to various parameters, compared with the base model.

In real world, n_s and r_n have facility-dependent boundaries. p_s and p_a have platform-dependent boundaries. In solving the objective function $F(n_s,p_s,r_n,p_a)$, we enumerate the possible values of r_n and p_a since they have small ranges. In this way, the 4-tuple (n_s, p_s, r_n, p_a) can be converted into 2-tuple (n_s, p_s), which can be further converted into 1-tuple p_s based on Eq. 11. Then we follow the steps as discussed above to solve the objective function for each r_n-p_a pair. Finally, we traverse the enumeration set to obtain the optimal solution.

In the baseline setup, power consumption and coefficients are the average of measured data across all the applications on real platforms. The baseline setup assumes a 120 hour job (scientific simulation) running on a Titan-like supercomputer using 20,000 nodes. We assume that this scientific simulation produces raw data at 1 TB/s, which equals to the peak I/O bandwidth of the Spider Lustre filesystem at Oak Ridge National Laboratory. This is a valid assumption since not all the data will reach the Lustre filesystem after online analysis. The job consumes 120 watts power on each GPU, with an upper (P_{max}) and lower (P_{min}) power limit of 120 watts and 90 watts, respectively. Online analysis is performed on a separate group of Xeon processors, whose baseline per-processor analysis throughput is 140 MB/s. Per-processor analysis throughput is chosen based on the geometric mean instead of arithmetic mean, due to the significant variation across different applications. In the baseline setup, analysis time is calculated as simulation time multiplied by the throughput ratio between simulation and analysis.

When running simulation applications on CPUs, reducing active cores relieves memory contention and improves performance [5]. Unlike CPU applications, the parallelism of GPGPU applications is orders of magnitude larger than the available CUDA cores. GPGPU is designed to serve such huge parallelism. Therefore, it is not viable to reduce the cores in exploring the optimal configuration.

Fig. 7 shows the minimal total simulation time ($\overline{T_{total}}(P_i)$) which includes simulation computation time, checkpointing time, and wasted time. Processor ratio refers to the number of analysis processors used to process the data generated by one simulation processor. Under a power limit of 2 MW, as shown in Fig. 7a, both total simulation time and simulation computation time increase as processor ratio increases. With power under-provisioned, increasing the number of analysis processors means decreasing the power limit of simulation, which leads to the decreased simulation scale and increased simulation time. The minimal total simulation time is marked with a circle when analysis time ($t_a + \widetilde{t_a}$) becomes slightly smaller than simulation computation time (t_s). Under a power limit of 6 MW, as shown in Fig. 7b, although simulation computation time increases slightly, the total simulation time decreases as processor ratio increases. The reason is that, with power over-provisioned, smaller simulation scale and lower power capping level result in a higher reliability level (or larger MTBF), which in return reduces the total checkpointing time and wasted time. Consequently, the total simulation time decreases as analysis scale increases. The minimal total simulation time is achieved (marked with a circle) when analysis time ($t_a + \widetilde{t_a}$) becomes much smaller than simulation computation time (t_s), rather than the point where $t_a + \widetilde{t_a}$ becomes slightly smaller than t_s. Running a workflow under power constraints, it is intuitive to assign less power to analysis (letting analysis match the simulation time), and distribute more power to simulation in order to boost performance. However, this can raise the checkpointing time and wasted time, when reliability is taken into account.

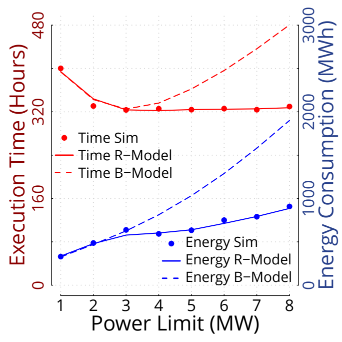


(a) Power limit 2 MW (b) Power limit 6 MW

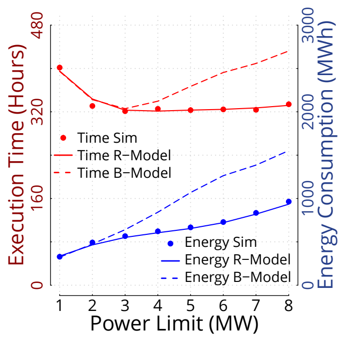
Fig. 7: Minimal total simulation time ($\overline{T_{total}}(P_i)$), and associated workflow computation time t_s (simulation) and t_a (analysis), with various processor ratios and under analysis power capping level 60 watts. They show similar trends under different analysis power capping levels. X-axis is plotted in a log scale to clearly show the data points when $r_n < 1$.

Finding 2: *Distributing excessive power to simulation can prolong the total simulation time. Matching analysis and simulation in terms of execution time does not always lead to the minimal total time.*

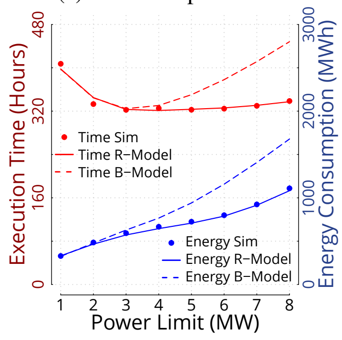
To be noted that, extending the upper limit of analysis time from simulation computation time (t_s) to total simulation time ($\overline{T_{total}}(P_i)$) can reduce the total simulation time by 1.1%, marked with a square in Fig. 7a. However, our model does not treat such configuration as optimal for the following two reasons. First of all, the performance gain is limited. However, it may cause significant performance loss as illustrated in Fig. 7b. In addition, extending the analysis time causes interferences between consecutive analysis tasks. For example, an application performs checkpointing every σ time, the time taken to store a checkpoint is δ, and analysis data dump frequency is σ/n. If analysis takes up to $(\sigma+\delta)/n$ time, when the second analysis dump starts at time σ/n, the analysis resources are still busy processing the first data dump. To be worse, such delay propagates to the consecutive $n-1$ analysis tasks.



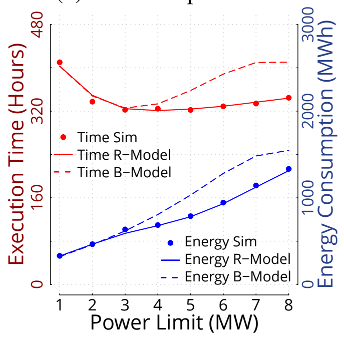
(a) Power Cap 60 watts



(b) Power Cap 50 watts



(c) Power Cap 40 watts



(d) Power Cap 30 watts

Fig. 8: Minimal workflow execution time and energy consumption under various power limits. Solid lines represent the reliability-aware model (R-Model) and dotted lines are for the base model (B-Model). Solid dots are simulation results. Color red is utilized to plot execution time (left y-axis) and blue is used for energy consumption (right y-axis).

In Fig. 7, we show the minimal workflow execution time under power limits 2 MW and 6 MW. Next, we validate the reliability-aware model and explore the minimal execution time and energy consumption under various power limits (ranging from 1 MW to 8 MW) for HPC workflows.

In order to validate the reliability-aware model, the model results are compared against simulation results. We developed an event-driven simulator which simulates different phases (i.e., computation, checkpointing, and rollback) and failure events of simulation and analysis in HPC workflows. The simulator generates random failures which follow a Poisson process, and the intervals between failures follow the exponential distribution. In the simulation, execution time is adjusted based on the power capping level and scale in accordance with the relationships derived in section IV.

Simulation results for both execution time (red dots) and energy consumption (blue dots) are shown in Fig. 8. We observe that our model results (solid lines) closely match the simulation results under various power limits and platform configurations. We also show the results of the base model in Fig. 8 to compare with the reliability-aware model. First, we observe that these two models are close to each other when power limit is small (smaller than 3 MW). As power limit increases, applying the base model results in larger execution time and energy consumption, and the gap is enlarging. Finally, we observe that workflow execution time decreases first, keeps steady, and then increases slightly as power limit increases. Therefore, the optimal power limit for performance is the starting point of the steady phase. Concerning workflow energy consumption, it increases significantly as power limit increases.

Finding 3: *The reliability-aware model can accurately predict the optimal platform configurations for minimal workflow execution time and energy consumption. Energy consumption increases as power limit increases, but an excessive high power limit may degrade the workflow performance.*

Although processor ratio strongly affects the workflow execution time (as illustrated in Fig. 7), different (r_n, p_a) pairs (e.g., $(60, 1/3)$, $(50, 1/3)$, $(40, 1/2)$ and $(30, 1)$ under 2 MW power limit) lead to similar minimal total execution time, as shown in Fig. 8. However, using the lowest analysis power capping level requires 3 times analysis processors as using the highest power capping level. Moreover, as shown in Fig. 8, employing a lower analysis power capping level results in higher workflow energy consumption, especially with a larger power limit. Although a lower power capping level tends to improve the reliability level, it improves the analysis scale, which in return reduces the overall MTBF. However, in Fig. 1a, we observe that, to achieve the minimal energy consumption, the optimal analysis power capping levels for most applications are between 40 watts and 50 watts. In contrast, when taking scalability and reliability into account, a higher analysis power capping level typically leads to lower energy consumption.

Finding 4: *The combination of analysis power capping level and scale offsets their individual impact on execution time within a range. The optimal analysis power capping level should be the highest effective level. Otherwise, it leads to resource over-utilization and excessive energy consumption.*

To be noted that, under power constraints, the platform configuration for minimal execution time does not necessarily lead to minimal energy consumption, as illustrated in Fig. 8. Under power limits, execution time reflects the performance per power limit. Conventionally, energy efficiency is measured as performance per watt, which can be reflected by energy consumption.

Finding 5: *Execution time and energy consumption show distinct trends as power limit changes. Under power constraint, performance per power limit does not reflect real energy efficiency which is typically measured as performance per watt.*

While determining the optimal power limit, users usually pursue minimizing execution time. In contrast, facility owners typically want to minimize the electricity bills (energy consumption). The different goals result in a dilemma in deciding the optimal power limit. To resolve this, we can resort to the metric, energy-delay product (or performance per joule), when perform comparison among various workflow power limits, in order to achieve the trade-off between different goals.

VII. Conclusion and Acknowledgement

In this paper, we propose a reliability-aware optimal platform configuration model to determine the optimal combination of power limit, processor power capping level, and scale for simulation and analysis respectively. Validation results show that our model is accurate in predicting the optimal platform configurations. Furthermore, we present model-driven studies and make several insightful findings on how platform configurations impact the reliability, execution time, and energy consumption of HPC workflows under power constraints.

The work performed at Temple university is partially sponsored by the U.S. National Science Foundation grants CCF-1717660, CNS-1702474 and CCF-1547804. This work was supported by, and used the resources of, the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at ORNL, which is managed by UT Battelle, LLC for the U.S. DOE, under the contract No. DE-AC05- 00OR22725. This work was also partially supported by Northeastern University.

References

[1] Y. Liu, R. Gunasekaran, X. Ma, and S. S. Vazhkudai, "Automatic identification of application i/o signatures from noisy server-side traces," in *Proceedings of the 12th USENIX Conference on File and Storage Technologies*, 2014, pp. 213–228.

[2] J. C. Bennett, H. Abbasi, P.-T. Bremer, R. Grout, and A. Gyulassy, "Combining In-situ and In-transit Processing to Enable Extreme-Scale Scientific Analysis," in *International Conference for High Performance Computing, Networking, Storage and Analysis*, 2012, pp. 49:1–49:9.

[3] DOE ASCAC Subcommittee Members, "Top ten exascale research challenges," DOE ASCAC Subcommittee, Tech. Rep., 2014.

[4] DOE ASCAC Subcommittee Members, "The Opportunities and Challenges of Exascale Computing," DOE ASCAC Subcommittee, Tech. Rep., 2010.

[5] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski, "Exploring Hardware Overprovisioning in Power-constrained, High Performance Computing," in *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing*, 2013, pp. 173–182.

[6] E. Deelman, C. Carothers, A. Mandal, B. Tierney, J. S. Vetter, I. Baldin, C. Castillo, G. Juve, D. Król, V. Lynch, B. Mayer, J. Meredith, T. Proffen, P. Ruth, and R. F. da Silva, "Panorama: An approach to performance modeling and diagnosis of extreme-scale workflows," *The International Journal of High Performance Computing Applications*, vol. 31, no. 1, pp. 4–18, 2017.

[7] A. Mandal, P. Ruth, I. Baldin, D. Król, G. Juve, R. Mayani, R. F. D. Silva, E. Deelman, J. Meredith, J. Vetter, V. Lynch, B. Mayer, J. Wynne, M. Blanco, C. Carothers, J. Lapre, and B. Tierney, "Toward an end-to-end framework for modeling, monitoring and anomaly detection for scientific workflows," in *IEEE International Parallel and Distributed Processing Symposium Workshops*, 2016, pp. 1370–1379.

[8] M. Li, S. S. Vazhkudai, A. R. Butt, F. Meng, X. Ma, Y. Kim, C. Engelmann, and G. Shipman, "Functional Partitioning to Optimize End-to-End Performance on Many-core Architectures," in *International Conference for High Performance Computing, Networking, Storage and Analysis*, 2010, pp. 1–12.

[9] F. Zhang, C. Docan, M. Parashar, S. Klasky, N. Podhorszki, and H. Abbasi, "Enabling In-situ Execution of Coupled Scientific Workflow on Multi-core Platform," in *IEEE 26th International Parallel and Distributed Processing Symposium*, 2012, pp. 1352 – 1363.

[10] K. Tang, D. Tiwari, S. Gupta, P. Huang, Q. Lu, C. Engelmann, and X. He, "Power-capping Aware Checkpointing: On the Interplay among Power-capping, Temperature, Reliability, Performance, and Energy," in *46th Annual IEEE/IFIP Int'l Conference on Dependable Systems and Networks*, 2016, pp. 311–322.

[11] D. Li, D. S. Nikolopoulos, K. Cameron, B. R. de Supinski, and M. Schulz, "Power-aware MPI task aggregation prediction for high-end computing systems," in *IEEE International Symposium on Parallel Distributed Processing*, 2010, pp. 1–12.

[12] D. Li, B. R. de Supinski, M. Schulz, K. Cameron, and D. S. Nikolopoulos, "Hybrid mpi/openmp power-aware computing," in *IEEE International Symposium on Parallel Distributed Processing*, 1-12, Ed., 2010.

[13] P. E. Bailey, D. K. Lowenthal, V. Ravi, B. Rountree, M. Schulz, and B. R. d. Supinski, "Adaptive Configuration Selection for Power-Constrained Heterogeneous Systems," in *43rd International Conference on Parallel Processing*, 2014, pp. 371–380.

[14] A. Langer, H. Dokania, L. V. Kalé, and U. S. Palekar, "Analyzing energy-time tradeoff in power overprovisioned hpc data centers," in *IEEE International Parallel and Distributed Processing Symposium Workshop*, 2015, pp. 849–854.

[15] C. Lefurgy, X. Wang, and M. Ware, "Power capping: a prelude to power shifting," *Cluster Computing*, vol. 11, no. 2, pp. 183–195, 2008.

[16] A. Gandhi, M. Harchol-Balter, R. Das, J. O. Kephart, and C. Lefurgy, "Power Capping Via Forced Idleness," in *Proceedings of Workshop on Energy-Efficient Design*, 2009.

[17] 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*, 2015, pp. 78:1–78:12.

[18] N. Gholkar, F. Mueller, and B. Rountree, "Power Tuning HPC Jobs on Power-Constrained Systems," in *Proceedings of the 2016 International Conference on Parallel Architectures and Compilation*, 2016, pp. 179–191.

[19] O. Sarood, A. Langer, A. Gupta, and L. Kale, "Maximizing throughput of overprovisioned hpc data centers under a strict power budget," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2014, pp. 807–818.

[20] T. Patki, D. K. Lowenthal, A. Sasidharan, M. Maiterth, B. L. Rountree, M. Schulz, and B. R. de Supinski, "Practical resource management in power-constrained, high performance computing," in *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*, 2015, pp. 121–132.

[21] Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini, "CoScale: Coordinating CPU and Memory System DVFS in Server Systems," in *Proceedings of the 45th Annual IEEE/ACM International Symposium on Microarchitecture*, 2012, pp. 143–154.

[22] O. Sarood, A. Langer, L. Kalé, B. Rountree, and B. de Supinski, "Optimizing power allocation to cpu and memory subsystems in overprovisioned hpc systems," in *IEEE International Conference on Cluster Computing*, 2013, pp. 1–8.

[23] R. Ge, X. Feng, Y. He, and P. Zou, "The case for cross-component power coordination on power bounded systems," in *45th International Conference on Parallel Processing*, 2016, pp. 516–525.

[24] S. S. Shende and A. D. Malony, "The Tau Parallel Performance System," *International Journal of High Performance Computing Applications*, vol. 20, no. 2, pp. 287–311, 2006.

[25] P. J. Mucci, S. Browne, C. Deane, and G. Ho, "PAPI: A Portable Interface to Hardware Performance Counters," in *Proceedings of Department of Defense HPCMP Users Group Conference*, 1999.

[26] NVIDIA, *NVML API REFERENCE MANUAL*, NVIDIA Corporation, 2012.

[27] M. Dimitrov, C. Strickland, S.-W. Kim, K. Kumar, and K. Doshi, "Intel Power Governor," https://software.intel.com/en-us/articles/intel-power-governor, July 2012.

[28] C. B. Codes, *https://asc.llnl.gov/CORAL-benchmarks*.

[29] J. Shalf, S. Dosanjh, and J. Morrison, "Exascale Computing Technology Challenges," in *Proceedings of the 9th International Conference on High Performance Computing for Computational Science*, 2011, pp. 1–25.