2018 30th International Symposium on Computer Architecture and High Performance Computing

Exploring Power Budget Scheduling Opportunities and Tradeoffs for AMR-based Applications

Yubo Qin, Ivan Rodero, Pradeep Subedi, Manish Parashar
Rutgers Discovery Informatics Institute (RDI^2), Rutgers University
Piscataway, New Jersey, USA
{yubo.qin, irodero, pradeep.subedi, parashar}@rutgers.edu

Sandro Rigo
Universidade Estadual de Campinas
Campinas, Sao Paulo, Brazil
sandro@ic.unicamp.br

***Abstract*—Computational demand has brought major changes to Advanced Cyber-Infrastructure (ACI) architectures. It is now possible to run scientific simulations faster and obtain more accurate results. However, power and energy have become critical concerns. Also, the current roadmap toward the new generation of ACI includes power budget as one of the main constraints. Current research efforts have studied power and performance tradeoffs and how to balance these (e.g., using Dynamic Voltage and Frequency Scaling (DVFS) and power capping for meeting power constraints, which can impact performance). However, applications may not tolerate degradation in performance, and other tradeoffs need to be explored to meet power budgets (e.g., involving the application in making energy-performance-quality tradeoff decisions). This paper proposes using the properties of AMR-based algorithms (e.g., dynamically adjusting the resolution of a simulation in combination with power capping techniques) to schedule or re-distribute the power budget. It specifically explores the opportunities to realize such an approach using checkpointing as a proof-of-concept use case and provides a characterization of a representative set of applications that use Adaptive Mesh Refinement (AMR) methods, including a Low-Mach-Number Combustion (LMC) application. It also explores the potential of utilizing power capping to understand power-quality tradeoffs via simulation.**

I. Introduction

High-Performance Computing (HPC) plays an essential role in the field of computational science. From the design perspective, HPC systems are built to maximize performance irrespective of power and energy consumption. However, as HPC system scale approaches exascale computing, power consumption is shifting from an optimization goal to a critical operational constraint. For example, the U.S. Department of Energy (DOE) has set a boundary of 20MW for an exascale system [1]. This power constraint poses a serious research challenge for both hardware and software. The fastest supercomputer, Sunway TaihuLight, as of November 2017, has a maximum performance of 93.0 PetaFLOPS and consumes 15.4 MW, which is 6.04 GigaFLOPS per watt [2]. Though Sunway's power efficiency has improved three times more than Tianhe-2 [3], it still needs more than an eight-fold power efficiency improvement to meet the expected exascale computing power constraint. While architecture improvements promise significant energy efficiency improvements at the system level, it is clear that this 20MW whole-system power constraint will be filtered down to job- and/or workflow-level power constraints, and as a result, optimizing power budget management is paramount.

While a significant number of research efforts have studied power and performance tradeoffs, most of these energy models or strategies are based on runtime (e.g., leveraging MPI slack) for power capping or power capping techniques such as Dynamic Voltage and Frequency Scaling (DVFS). However, power and performance are on two sides of a balance scale, and therefore it is difficult to improve one side without sacrificing the other. Although most of the existing techniques attempt to reduce power consumption while minimizing performance degradation (e.g., execution time), the scientific applications involved may not tolerate any degradation in performance (e.g., when they are part of a workflow).

We thus believe that the application should be involved in making power-related tradeoff decisions. Otherwise, the workflow might experience either an elongated execution time or reduction in the final quality of data and/or solutions. In this paper, we target applications that use Adaptive Mesh Refinement (AMR) methods. AMR-based applications consider a hierarchy of grids with different computational resolutions (*mesh*) ranging from coarsest to finest. These applications can focus computational resources in regions of interest while decreasing the mesh resolution in regions with less interest. This flexible resolution property gives us an opportunity to control the workload and reorganize the power budget during the execution of workflow.

The overarching goal of this work is to investigate tradeoff between power, performance, and quality, and to build a new task scheduling model considering AMR properties for managing power budgets at scale. To understand the applicability of the proposed approach, this paper first studies the mechanisms and policies that control AMR properties and then characterizes AMR properties in terms of performance, power consumption, and system impact for a representative set of AMR-based applications. Finally, it studies the potential of power capping techniques for re-distributing the power budget to other components of a workflow (i.e., a checkpointing task) by trading-off power and quality metrics. To this end, the main contributions of this paper include: (1) providing an empirical evaluation of different configurations of application to provide insight into the power-performance-quality tradeoff for AMR-based scientific data-driven workflow; and (2) developing

978-1-5386-7769-8/18/$31.00 ©2018 IEEE
DOI 10.1109/SBAC-PAD.2018.00023

an understanding, via simulation, of the potential of using power capping to re-distribute the power budget to workload components when resolution degradation is tolerable.

The rest of this paper is organized as follows: Section II briefly introduces some related background information. Section III describes the experiment setup of the exploration study. Section IV presents the evaluation methodology and the results of our study. Section V concludes the paper and outlines ongoing and future research.

II. Background and Related Work

A. Adaptive Mesh Refinement

The AMR method was introduced by Berger et al. [4] as a powerful tool to solve partial differential equations (PDEs), and it has been widely used in large-scale simulations. The AMR method constructs a nested grid hierarchy architecture and adaptively adds new refinement grids to the region where the error is estimated to be large. The refined grids are aligned with the underlying coarser grids. The base grid is defined as level zero, and subsequent grids are added to level 1 and so on. Each coarser grid on level *L-1* is refined into grids on level *L* by a refinement ratio *r*, which usually is 2 or 4. During the refining process, the coarser grid is divided into r^2 finer grids in 2-dimension or r^3 finer cubes in 3-dimension.

In contrast to the refining process, once the grids are outside existing refining regions, or are marked as coarsening regions, a coarsening process occurs. The field information obtained in level *L* will then be stored back to level *L-1* by the coarsening process, and the grids at level *L* are automatically eliminated [5]. By alternatively refining and coarsening grids, the AMR method can dynamically adjust computing resources and tolerate resolution changes.

Numerous grand challenge problems (e.g., cosmological hydrodynamics [6], fluid flow simulations [7], etc.) are modeled by PDEs, and the AMR method is widely used to support their execution at an extreme scale. For example, Burstedde et al. [8] implemented parallel AMR algorithms targeting petascale HPC systems. In this paper, we use the simulation resolution level as a knob to control the power versus quality of the solution tradeoff.

B. Power capping

Since the launch of Sandy Bridge family processors, Intel provides the Running Average Power Limit (RAPL) mechanism for controlling the power constraint on processors and memory [9]. RAPL has combined the automatic DVFS and clock throttling techniques. Unlike most popular power capping mechanisms that maintain instantaneous power limits, RAPL estimates and maintains an average power limit over a sliding time window. Several studies have evaluated the RAPL performance such as the work from Zhang et al. [10]. It provided a systematic evaluation of RAPL behavior including stability, settling time, overshoot, etc. In addition, RAPL supports high-accuracy power capping performance and has been used in many research studies [11], [12], for instance, application runtime variability and power optimization for Exascale computing. Rountree et al. [13], [12] evaluated power consumption for package and memory subsystems to explore RAPL as a replacement for DVFS in HPC systems.

RAPL also provides mechanisms for measuring power/energy consumption. Existing research work has leveraged these capabilities [11], [14]. For example, Vignesh et al. [15] used RAPL to measure the CPU and DRAM power consumption in order to study the "greenness" of in-situ and post-processing visualization pipelines and claimed an average RAPL measurement error rate of less than 1%.

C. Power management and scheduling

Power management has become one of the critical concerns for HPC applications. Existing work has combined DVFS with predictions to reduce power consumption during non-critical sections in MPI applications [16]. Freeh et al. [17] presented a Jitter system that addresses slack time in MPI programs. Li et al. [18] presented the thrifty barrier technique, which uses low-power sleep states during barrier synchronization. Rountree et al. [19] presented the Adagio system to dynamically determine optimal CPU power levels during the runtime of MPI codes. Closer to this work, Piga et al. [20] proposed a technique to power cap the waiting processes and shift the remaining power budget to processes in the critical execution path.

In addition to leveraging opportunities for better use of energy in MPI executions, power management has also been considered at the component and cluster levels. For example, Rodero et al. [21] studied the potential of application-centric aggressive power management of HPC scientific workloads considering different subsystems and virtualized environments [22]. Wang et al. [23] explored an energy-aware task scheduling leveraging DVFS during the slack time for non-critical jobs. Etinski et al. [24] proposed a power budget-guided job scheduling policy that maximizes overall job performance for a given power budget. Existing work has also addressed power management in simulation workflows; for example, Gamell et al. [25] presented a power model to analyze the behavior of simulation workflows and explored data-related energy/performance tradeoff at an extreme scale.

Power budget scheduling has also been addressed in the context of multi-core systems. Sartori et al. [26] proposed a peak power management technique for each core to meet power constraints. Cebrian et al. [27] proposed borrowing the power budget from cores that consume lower power to dynamically adjust the per-core power budget. Gandhi et al. [28] provided a power capping strategy to meet the power budget by inserting idle cycles during execution. Moreover, the power budget can be constrained by dynamically reconfiguring processor resources; for example, Meng et al. [29] and Konotorinis et al. [30] proposed to dynamically reconfigure the cache configuration and the load-store queue in the floating point unit.

Our work is unique from these studies in the following ways: 1) we consider the properties of AMR as a knob to tune power-performance tradeoff, 2) we proactively create a power budget based on the tolerance of resolution degradation while

maintaining performance, and 3) we opportunistically use the power budget to enhance the overall workflow performance (e.g., providing checkpointing capabilities without requiring further power resources).

III. Experimental Methodology

A. Hardware platform

Two different RDI^2 [31] platforms, Caliburn and CAPER, have been utilized for conducting our experimental evaluation. Caliburn is a 560-node cluster based on SuperMicro's FatTwin SuperServer solution, containing 20,160 cores, 140 TB of DRAM memory, 218 TB of non-volatile memory, and 100 Gbps Omni-Path fabric (OPA) interconnection, which delivers a peak performance of 677 TFLOPS. An Intelligent Platform Management Interface (IPMI) and SLURM workload management system have been installed.

CAPER is an 8-node cluster based on SuperMicro SYS-4027GR-TRT systems and is configured with power-monitoring instruments that provide both coarse- and fine-grained power metering at a per-server level. A Raritan iPDU PX2-4527X2U-K2 provides system-wide power measurements at 1 Hz, and a Yokogawa DL850E ScopeCorder data acquisition recorder provides power measurements at a higher rate. While CAPER is a flexible system that allows us to directly log into nodes to implement and evaluate the power management experiment, Caliburn allows us to evaluate the tradeoff strategies on a larger scale.

B. Applications and their input configurations

In this study, we focus on LMC [32] to study the mechanisms that control AMR properties. We also explore ENZO [33], FLASH [34], and RAMSES [6] applications to further characterize the AMR properties in terms of performance, power consumption, and system impact. We leveraged their built-in test problems as test cases. In order to keep a reasonable execution time, we tuned their input configurations to adapt to the testbed. Table I lists the mesh size and dimensions used in our evaluation of the four test applications.

C. Power capping and system utilization measurement

Similar to existing research studies mentioned in Section II-B, we used Intel RAPL in our experiments for power capping. This functionality is integrated in the Intel Power Governor software utility, which was available in each node, and the power capping value was set as the input parameter for each experiment.

We measured CPU-wide and system-wide power consumption as well as CPU and DRAM utilization. The CPU-wide power consumption was measured using RAPL power metering, which obtains data by reading Machine-Specific Registers (MSRs). The system-wide power consumption was measured through external metering in CAPER and via SMCIPMITOOL [35] in Caliburn. For both platforms, we collected CPU and DRAM utilization via the Unix System monitor *sar*. As all of the profiling data was collected from the control node, the impact on result accuracy was negligible.

Application	Test case	Mesh size	Dimensions
LMC	Flame in a box	32x64	2d
ENZO	Cosmology	128x128	2d
FLASH	Cellular nuclear burning	128x128	2d
RAMSES	Sedov 2D	512x512	2d

Table I: *Configurations of AMR-based applications*

IV. Exploration of Power-Performance-Quality Tradeoffs

When a workflow is in execution, different tasks associated with the workflow (e.g., analytics, visualization, etc.) can be scheduled based on the control flow and user requirements. These additional tasks are likely to impact the execution time and power consumption of the workflow. If the power budget is limited, a tradeoff between performance and quality might be required. In this study, we define application performance as the total execution time of the application/workflow and quality as the resolution of the mesh in AMR-based applications. We aim to opportunistically obtain a power budget from the running application and use it to perform additional tasks while maintaining the original application's performance.

In addition, we target AMR-based applications as they can tolerate resolution degradation by nature. As described in Section II-A, AMR methods can dynamically tune their resolutions based on computational requirements. They are able to increase the resolution in the area of interest while maintaining or degrading the resolution in other areas without reducing the quality of the final result. This property makes AMR-based applications able to tolerate arbitrary resolution degradation. We leverage this fact to dynamically tune the computation resolution based on the required power budget without sacrificing performance.

A. Power, performance, and quality tradeoffs

Figure 1 illustrates the power consumption of an AMR-based application and its tuning to obtain either idle time slots or the power budget to run additional tasks. These extra tasks can be any work that is part of the workload, such as checkpointing. Figure 1.a represents the AMR-based application executing at full resolution and power. If we run the same application with a lower resolution but at the same power, it can finish faster. In this case, there are some idle time slots available to execute additional tasks. However, if the

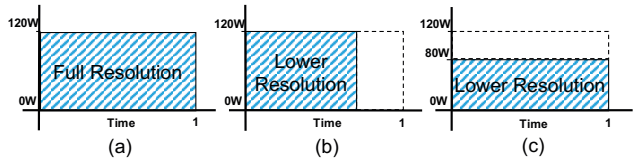


Figure 1: *Power consumption of an AMR-based application in different execution scenarios for one time step (a) executing at full resolution and power, (b) executing at a lower resolution where the blank area is the idle time slot that is available for executing additional tasks (e.g., checkpointing), and (c) executing under power capping and resolution degradation where the blank area is the available power budget that can be redistributed to execute additional tasks.*

additional tasks introduced are long-running tasks but consume low power, the available idle time slots will not be sufficient to complete the tasks on hand. This could lead to elongation of the total workload execution time. Idle time slots can be intelligently managed; however, our empirical evaluation is concentrated on the approach described below.

We thus propose a second strategy in Figure 1.c, which combines resolution degradation with power capping to create the available power budget. The available power budget is the blank area in Figure 1.c. Because this extra power budget is available for a longer time, additional tasks can be scheduled in available hardware resources. In this strategy, we reduce the resolution and apply power capping only to the extent that the total execution time is no longer than when the application runs with full resolution and no power capping.

While degrading the resolution can reduce the execution time, in order to implement the second strategy, users need to find out the appropriate power capping value to make sure that execution times under different resolutions are approximately similar to each other. This value can be found and/or adjusted by running the application iteratively. In each iteration, the resolution setting and power capping value is tuned until the execution time under each degraded resolution setting matches the execution time at full resolution. Since AMR-based applications are typically used to process complex problems, they tend to repeatedly run on the same HPC system. Once we find the right resolution setting and power capping value, we can reuse them for subsequent executions.

In order to demonstrate the process of finding adequate power capping values, we first ran the LMC combustion simulation application [32] on the CAPER platform. We categorized the LMC application resolution into three levels: $High$, $Medium$, and Low. Because CAPER's socket Thermal Design Power is 96W, we used RAPL to cap down CPU power at 16W per level. Figure 2 plots the relationship between power capping and its corresponding execution time. Once the approximate range of expected power capping values (the intersection points of the dashed line and curves of each resolution in Figure 2) were determined, the power capping intervals were made finer until the execution time of each resolution level matched the execution time at full resolution with a maximum 5% variability. This determines the appropriate power capping value for each resolution level.

This power capping value can then be used to run the simulation application and determine the available power budget. Figure 3 illustrates the LMC application run-time system-wide power consumption during three time steps. In the first time step, the application runs on full power with high resolution. In subsequent time steps, we lower the resolution and apply corresponding power capping to them. The average power consumption drops by 36.9% and 51.1% from the first to the second and from the first to the third time steps, respectively. The red backslash area is the available power budget that can be routed to other applications. This strategy helps us to proactively create the available power budget by dynamically tuning the application resolution based on the workflow needs.

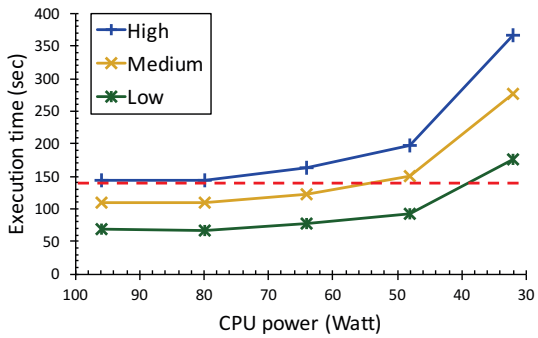


Figure 2: *Impact of power capping on LMC application execution using RAPL to cap down CPU power at 16W per level. The power capping configurations that meet the targeted execution time are the intersection points of the dashed line and curves for each resolution.*

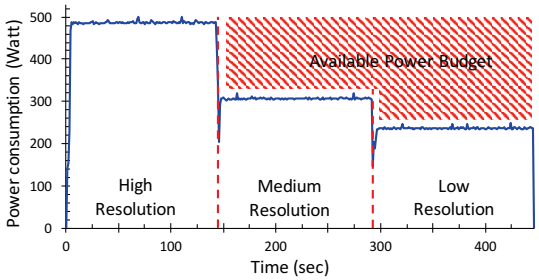


Figure 3: *Impact of resolution degradation on power consumption. This tradeoff strategy can create the available power budget, which is highlighted by the red backslash area (time steps 2 and 3), without sacrificing performance (i.e., total execution time).*

Although we ran our tests for just three levels of resolution, it can be easily tweaked to any number of resolution levels for more control over the available power budget.

B. Power budget scheduling

For the first strategy in Figure 1.b, the power budget is represented as the idle time slot. Thus, we can insert the additional tasks to run within the idle time slot. This strategy is suitable for co-located extra tasks, which run on the same hardware as the AMR-based applications did. For the second strategy in Figure 1.c, the power budget in the blank area can be shifted to other available hardware resources to execute additional tasks. In the most common scenario, the power budget is the system-wide power consumption constraint, and the AMR-based applications will not reserve all computational resources in the high-end system. Thus, it is feasible to allocate additional computational resources to execute extra tasks while under the same system-wide power consumption constraint.

To demonstrate the applicability of the second power scheduling strategy, we used two sets of nodes on CAPER and then measured the system-wide power consumption as shown in Figure 4. While one set ran the LMC application, the other one ran the additional tasks, which could be any work (e.g., LMC checkpointing in this case) that is part of the workload. We scheduled the additional task execution in every other time

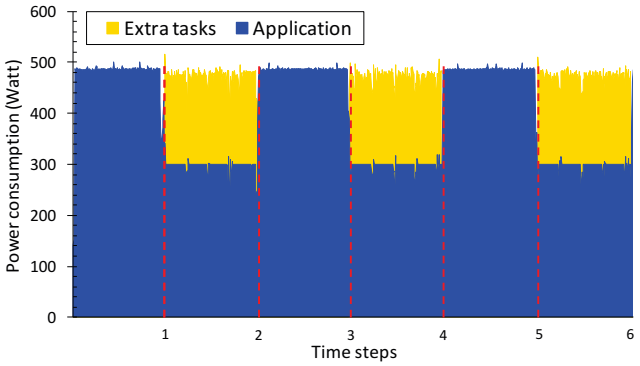


Figure 4: *The LMC application can use the available power budget to run additional tasks (LMC checkpointing). The tradeoff strategy has reduced the application's system-wide power consumption (blue area) via degrading its resolution from $High$ to $Medium$ and applying power capping. The extra tasks run on an additional set of nodes (power in yellow area).*

step. When additional tasks were scheduled, the LMC application initiated the tradeoff strategy to lower the resolution from $High$ to $Medium$ and simultaneously launched these scheduled additional tasks in another set of nodes. The blue and yellow areas in Figure 4 represent the power consumption of the LMC application and the additional tasks, respectively. As can be observed in the figure, the system-wide power consumption remains constant, irrespective of the additional tasks. The total execution time of the simulation application also remains constant for all time steps.

C. Tuning the power capping and computational resolution

In order to have a more comprehensive understanding of the potential tradeoff strategies, the selected large-scale simulation applications (ENZO, FLASH, and RAMSES) were configured with the parameters listed in Table I and were evaluated on Caliburn at a scale of 512 cores (15 nodes). As in the LMC experiment, their resolutions were divided into three levels: $High$, $Medium$, and Low. Table II lists the power capping value for these sets of experiments.

Figure 5 illustrates the power consumption behavior of the selected AMR-based applications and their execution times. The red line is the execution time under each resolution level. In the experiments, it can be observed that the execution times at *High*, *Medium*, and *Low* resolution levels were within 5% of each other. With power capping and reduction in computational resolution, we can see that there was extra power budget available. The light-blue backslash bar represents the system-wide power consumption, but it does not include the CPU power consumption. The dark-blue slash bar represents the CPU power consumption. With the reduction in simulation resolution, the available power budget increased. It can be observed that 36.2%, 31.4%, and 39.5% of the system-wide power budget was made available by tuning the resolution from $High$ to $Medium$ for FLASH, ENZO, and RAMSES, respectively. Similarly, we observed that 22.0%, 22.3%, and

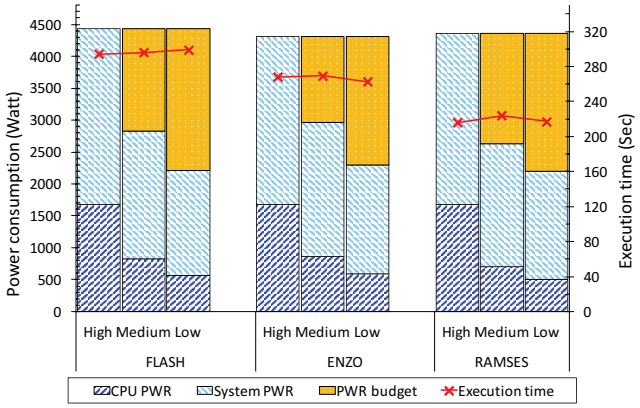


Figure 5: *Power consumption behavior of the three AMR-based applications using the proposed tradeoff strategy.*

SIM	RES	TIME W/O	PC	TIME W	RW	CPU %	DRAM
FLASH	High	294.21	120	294.21	1	94.68	2471.93
	Medium	87.76	59	296.11	0.25	89.54	2373.24
	Low	36.89	41	299.08	0.0625	79.46	2160.11
ENZO	High	267.86	120	267.86	1	94.81	3755.87
	Medium	61.65	62	269.39	0.25	89.00	1939.49
	Low	21.29	42	262.56	0.0625	88.23	1217.92
RAMSES	High	216.045	120	216.04	1	99.29	2772.95
	Medium	88.81	52	223.99	0.25	98.66	2695.21
	Low	46.89	36	217.06	0.0625	96.89	2417.60

Table II: *CPU and DRAM utilization for the three AMR-based applications. RES: resolution level; TIME W/O: application execution time in one time step without power capping; PC: power capping value; TIME W/: application execution time in one time step with power capping; RW: resolution weight, users define this based on application; CPU: CPU utilization in percentage; DRAM: memory usage in MB. The PC and RES values were pre-determined to ensure that the application execution time remains the same for each resolution level.*

16.5% of the power budget was freed by tuning the resolution from $Medium$ to Low for these applications.

The CPUs consume around one-third of the system-wide power consumption for all AMR-based applications. However, by lowering resolution, CPU power contributes more than half of the freed power budget. From resolutions $High$ to $Medium$, CPU power capping contributes 53.3%, 59.9%, and 57.0% of the freed power budget for FLASH, ENZO, and RAMSES, respectively. The rest of the freed power budget came from other system components, such as DRAM memory, disk, network, etc. This is mainly due to the reduction in component utilization, which directly relates to the computational workload. Table II shows the CPU and memory utilization observed during the evaluation. Please note that the power capping values were pre-determined for each resolution level to maintain similar execution times as compared to the *High* resolution level.

All three AMR-based applications showed similar trends for freeing up the power budget by applying the tradeoff strategy, which can be used to schedule new tasks on separate nodes or computational resources.

D. Quantifying the impact on resolution

Although we can free up some of the power budget by reducing the computational resolution of AMR-based applica-

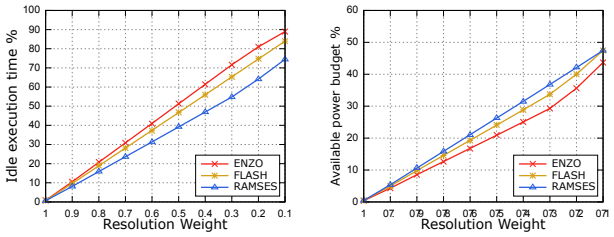
tions, it is important to quantify the amount of power budget obtained and the corresponding resolution degradation. If we can build a relationship between power budget and resolution degradation before long simulation runs, AMR applications can use this relationship to initiate the tradeoff strategies dynamically. To this end, we built a C++ quantifier to emulate the two tradeoff strategies described in Section IV-A and used it to quantify their applicability.

This quantifier takes a set of input data from each simulation application, which includes the simulation resolution level RS, execution time ET, corresponding power capping value (PC), and time step TS. It then evaluates the amount of power budget that can be obtained by degrading resolution. To quantify the loss or gain, we introduce Resolution Weight (RW), which ranges from 1 to 0 and represents the resolution degradation of the simulation result. For the AMR-based applications, the RW equals the proportion of the number of grids remaining after applying the tradeoff strategies. $RW = 1$ means no resolution degradation, thus a smaller RW value represents lower resolution in the simulation. However, defining RW value is subjective because the AMR method adaptively refines or coarsens grids only in certain regions, and each simulation object has different characteristics. Therefore, RW value can vary from application to application and in our quantifier, RW is considered a user input. The proportion of the total grid numbers between two consecutive resolution levels equals r^{dims}. If we assume the AMR refinement ratio r is 2 and the AMR grid dimension $dims$ is also 2, the resolution weight RW is reduced by 2^2 times in each resolution degradation process.

Algorithm 1 describes the quantifier that is used to quantify the loss and gain of each tradeoff strategy. The quantifier operates under the constraint that the application performance should remain the same (i.e., the total execution time must not change). At the end of each time step, the quantifier chooses a configuration for the next time step. It compares the current average RW with the user-given target RW_{target} value. If $RW_i > RW_{target}$, the resolution in the next time step will be reduced, and vice versa. In the end, it calculates the corresponding number of idle time slots for the strategy in Figure 1.b and the amount of available power budget for the strategy in Figure 1.c. This is the amount of the power budget that the tradeoff strategy can obtain from the given resolution weight RW value.

E. Evaluating the quantifier

Figure 6 plots the resolution weight versus the percentage of available idle time slots and the power budget for the three AMR-based applications (ENZO, FLASH, and RAMSES) when running for 100 time steps. The inputs for the quantifier are taken from Table II. The quantifier runs iteratively and tunes the resolution configurations to match the given resolution weight value and then generates the corresponding average available idle time slots and power budget. A nearly linear correlation between the resolution weight and the available power budget was observed.



(a) *Available idle time slot* **(b)** *Available power budget*

Figure 6: *Percentage of available idle time slots and system-wide power budget freed from dynamic reduction in computational resolution.*

Algorithm 1 Quantifier algorithm

Input: ET //Execution time
PC //Power capping value
RW //Resolution weight
RS //Resolution level
TS //Time step
RW_{target}//Targeted resolution weight
Output:
$\%PB_{avail}$ //Percentage of available power budget
$\%TM_{idle}$ //Percentage of available idle time slot

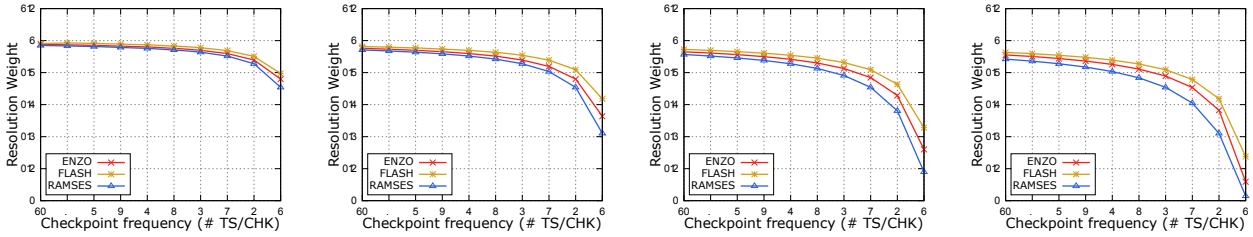
for ITR $\leq$ N **do**
  while i $\leq TS$ **do**
    $\{PW_i, TM_i, RW_i\}$ = exec($ET[RS_i]$, $RW[RS_i]$, $PC[RS_i]$);
    if $RW_i > RW_{target}$ **then**
      RS_{i+1} = decrease(RS_i);
    else if $RW_i < RW_{target}$ **then**
      RS_{i+1} = increase(RS_i);
    else
      $RS_{i+1} = RS_i$;
    end if
  end while
  $\%TM_{idle} = (1 - \frac{\sum_{i=1}^{TS} TM_i}{ET_{high} * TS})\%$;
  $\%PB_{avail} = (1 - \frac{\sum_{i=1}^{TS} PW_i}{PC_{high} * TS})\%$;
end for

When the resolution weight was set to 0.25, which represents a $Medium$ resolution level, more than 60% of the idle time slots and 30% of the system-wide power budget were available for executing additional tasks. We assume, in real scenarios, $RW = 0.8$ or 20% resolution loss should be acceptable. Even with a lower value, the tradeoff strategies can still provide 20.8%, 18.9%, and 15.9% of available idle time slots and 8.6%, 9.8%, and 10.7% of the available power budget in ENZO, FLASH, and RAMSES simulations, respectively.

Our evaluations reveal that these three AMR-based applications demonstrate similar power consumption behavior when using the tradeoff strategy. This also provides a reference for opportunistically inserting additional tasks by implementing the tradeoff strategy.

F. Checkpointing as a use-case

In real scenarios, the simulation application and additional tasks have complex run-time behaviors. For example, the inserted extra tasks may not finish within the given time slot or power budget. In order to satisfy the application power budget and execution time constraints, the tradeoff strategy should be able to compensate for this additional cost. In this section, we use a checkpointing task as a use-case to study the impact of additional workload on the AMR-based application resolution weight.



(a) *Duration: 50 seconds* **(b)** *Duration: 100 seconds* **(c)** *Duration: 150 seconds* **(d)** *Duration: 200 seconds*

Figure 7: *The impact on the simulation resolution weight when implementing the tradeoff strategy to insert a varying checkpointing workload. Single checkpointing task duration ranges from 50 to 200 seconds.*

Algorithm 2 Impact of various checkpointing task workloads on AMR-based simulation application resolution weight

Input: ET, RW, TS,RS,
CHK_{freq} //Write checkpoint frequency
CHK_{time} //Single checkpoint task duration
Output: RW_{avg} //Average resolution weight

1: **for** ITR $\leq$ N **do**
2: **while** i $\leq TS$ **do**
3: **if** i % CHK_{freq} == 0 **then**
4: $runtime_i$ = write_chkpoint(CHK_{freq},CHK_{time},RS_i)
5: **end if**
6: $\{runtime_i, RW_i\}$ = exec($ET[RS_i]$,$RW[RS_i]$);
7: **if** $runtime_i > ET[RS_i]$ **then**
8: RS_{i+1} = decrease(RS_i);
9: **else if** $runtime_i < ET[RS_i]$ **then**
10: RS_{i+1} = increase(RS_i);
11: **else**
12: $RS_{i+1} = RS_i$;
13: **end if**
14: RW_{avg} = average(RW_i)
15: **end while**
16: **end for**

Checkpointing is a mechanism for fault tolerance (or resiliency) that stores the current simulation status to a file (i.e., checkpoint). If any error occurs during runtime, the checkpoint file can be used to roll back the simulation to the most recent saved status. Recovering from a checkpoint file requires recomputing from the last checkpoint. Therefore, it is desirable to write the checkpoints as frequently as possible in order to reduce the impact of failures. However, writing checkpoints is costly because they consume power and bring I/O latency to the simulation. Usually, checkpoint frequency is pre-determined by finding the right balance between performance and resilience. In this work, we treat checkpointing as a use-case representing additional tasks.

The quantifier uses Algorithm 2 to quantify the impact of various checkpointing tasks on AMR-based applications' resolution weights. Because checkpoints are periodically written every few time steps, we use a tradeoff strategy in Figure 1.b to create an idle time slot for writing checkpoints. The checkpoint frequency CHK_{freq} determines the number of time steps between two consecutive checkpoints. For example, $CHK_{freq} = 10$ represents execution of a checkpoint task every 10 time steps. The checkpoint duration CHK_{time} represents a single checkpoint time. The smaller the CHK_{freq} and the larger the CHK_{time} values, the greater the number of additional tasks that can be inserted in conjunction with the AMR application. We also assume that the checkpointing task is executing at full power. The quantifier uses the input data from Table II and executes the checkpointing task according to the corresponding CHK_{freq}. At the end of each time step, it chooses a configuration for the next time step by comparing the current average runtime $runtime_i$ with the user-provided execution time $ET[RS_i]$. If $runtime_i > ET[RS_i]$, the resolution in the next time step will be reduced, and vice versa. In the end, the corresponding resolution weight RW is generated.

Figure 7 plots the impact of checkpointing tasks on the resolution weight. When the resolution was degraded from $High$ to $Medium$, it was observed that FLASH has the largest difference in execution time among the evaluated AMR applications. FLASH has a higher capacity to tolerate the inserted additional tasks with the same resolution degradation. Thus, FLASH has a flatter curve. Because a 50-second workload is relatively small for these AMR-based applications, the tradeoff strategy can still maintain over a 0.7 resolution weight at point $CHE_{freq} = 1$. When the duration grew to 200 seconds, the resolution weight sharply dropped after the checkpoint frequency exceeded 2 TS/CHK. However, assuming that the AMR application can tolerate a resolution weight of up to 0.8, the tradeoff strategy can still support writing checkpoints over 6 TS/CHK at a duration of 200 seconds. This can greatly enhance the application's reliability.

V. Conclusion

In this paper, we studied AMR properties and explored the performance, quality, and power tradeoff of AMR-based applications. We presented an empirical evaluation of various power capping and resolution configurations for FLASH, ENZO, RAMSES, and LMC applications, which provided insights into the energy-performance-quality tradeoff. We evaluated the power-performance-quality tradeoff for these applications and used checkpointing as a use-case to quantify the tradeoff. Our experiments reveal that tradeoff strategies can create an available power budget to enhance system reliability with minimum resolution degradation. Our future work will include the integration of the techniques explored in this paper into the scheduling runtime to dynamically tune the simulation level of the resolution with the appropriate power capping configuration to match the targeted quality of service with acceptable resolution degradation in areas/steps of less interest.

ACKNOWLEDGMENTS

This work is supported in part by National Science Foundation via grants numbers ACI-1464317 and CNS-1305375, and was conducted as part of the Rutgers Discovery Informatics Institute (RDI^2). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. We acknowledge Weiqun Zhang and Marcus S. Day, from Lawrence Berkeley National Laboratory, for helping with LMC simulation application.

REFERENCES

[1] M. Tolentino and K. W. Cameron, "The optimist, the pessimist, and the global race to exascale in 20 megawatts," *Computer*, vol. 45, no. 1, pp. 0095–97, 2012.

[2] "Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway." [Online]. Available: https://www.top500.org/system/178764

[3] "Tianhe-2 (MilkyWay-2) - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P." [Online]. Available: https://www.top500.org/system/177999

[4] M. J. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic partial differential equations," *Journal of computational Physics*, vol. 53, no. 3, pp. 484–512, 1984.

[5] M. Matsumoto, F. Mori, S. Ohshima, H. Jitsumoto, T. Katagiri, and K. Nakajima, "Implementation and evaluation of an amr framework for fdm applications," *Procedia Computer Science*, vol. 29, pp. 936–946, 2014.

[6] R. Teyssier, "Cosmological hydrodynamics with adaptive mesh refinement-a new high resolution code called ramses," *Astronomy & Astrophysics*, vol. 385, no. 1, pp. 337–364, 2002.

[7] A. C. Calder, B. C. Curtis, L. Dursi, B. Fryxell, P. MacNeice, K. Olson, P. Ricker, R. Rosner, F. Timmes, H. Tufo *et al.*, "High performance reactive fluid flow simulations using adaptive mesh refinement on thousands of processors," in *Proceedings of the 2000 ACM/IEEE conference on Supercomputing*. IEEE Computer Society, 2000, p. 56.

[8] C. Burstedde, O. Ghattas, M. Gurnis, T. Isaac, G. Stadler, T. Warburton, and L. Wilcox, "Extreme-scale amr," in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society, 2010, pp. 1–12.

[9] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, "Rapl: memory power estimation and capping," in *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*. IEEE, 2010, pp. 189–194.

[10] H. Zhang and H. Hoffmann, "A quantitative evaluation of the rapl power control system," *Feedback Computing*, 2015.

[11] A. Porterfield, R. Fowler, S. Bhalachandra, B. Rountree, D. Deb, and R. Lewis, "Application runtime variability and power optimization for exascale computers," in *Proceedings of the 5th International Workshop on Runtime and Operating Systems for Supercomputers*. ACM, 2015, p. 3.

[12] 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 *Cluster Computing (CLUSTER), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1–8.

[13] 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 *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*. IEEE, 2012, pp. 947–953.

[14] M. Hähnel, B. Döbel, M. Völp, and H. Härtig, "Measuring energy consumption for short code paths using rapl," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 3, pp. 13–17, 2012.

[15] V. Adhinarayanan, W.-c. Feng, J. Woodring, D. Rogers, and J. Ahrens, "On the greenness of in-situ and post-processing visualization pipelines," in *Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International*. IEEE, 2015, pp. 880–887.

[16] R. Ge, X. Feng, W.-c. Feng, and K. W. Cameron, "Cpu miser: A performance-directed, run-time system for power-aware clusters," in *Parallel Processing, 2007. ICPP 2007. International Conference on*. IEEE, 2007, pp. 18–18.

[17] V. W. Freeh, N. Kappiah, D. K. Lowenthal, and T. K. Bletsch, "Just-in-time dynamic voltage scaling: Exploiting inter-node slack to save energy in mpi programs," *Journal of Parallel and Distributed Computing*, vol. 68, no. 9, pp. 1175–1185, 2008.

[18] J. Li, J. F. Martinez, and M. C. Huang, "The thrifty barrier: Energy-aware synchronization in shared-memory multiprocessors," in *Software, IEE Proceedings-*. IEEE, 2004, pp. 14–23.

[19] B. Rountree, D. K. Lownenthal, B. R. de Supinski, M. Schulz, V. W. Freeh, and T. Bletsch, "Adagio: making dvs practical for complex hpc applications," in *Proceedings of the 23rd international conference on Supercomputing*. ACM, 2009, pp. 460–469.

[20] L. Piga, I. Paul, and W. Huang, "Performance boosting opportunities under communication imbalance in power-constrained hpc clusters," in *Parallel Processing (ICPP), 2016 45th International Conference on*. IEEE, 2016, pp. 31–40.

[21] I. Rodero, S. Chandra, M. Parashar, R. Muralidhar, H. Seshadri, and S. Poole, "Investigating the potential of application-centric aggressive power management for hpc workloads," in *High Performance Computing (HiPC), 2010 International Conference on*. IEEE, 2010, pp. 1–10.

[22] I. Rodero, E. K. Lee, D. Pompili, M. Parashar, M. Gamell, and R. J. O. Figueiredo, "Towards energy-efficient reactive thermal management in instrumented datacenters," in *Proceedings of the 2010 11th IEEE/ACM International Conference on Grid Computing, Brussels, Belgium, October 25-29, 2010*, 2010, pp. 321–328.

[23] L. Wang, G. Von Laszewski, J. Dayal, and F. Wang, "Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with dvfs," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE Computer Society, 2010, pp. 368–377.

[24] M. Etinski, J. Corbalan, J. Labarta, and M. Valero, "Optimizing job performance under a given power constraint in hpc centers," in *Green Computing Conference, 2010 International*. IEEE, 2010, pp. 257–267.

[25] M. Gamell, I. Rodero, M. Parashar, J. C. Bennett, H. Kolla, J. Chen, P.-T. Bremer, A. G. Landge, A. Gyulassy, P. McCormick *et al.*, "Exploring power behaviors and trade-offs of in-situ data analytics," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. ACM, 2013, p. 77.

[26] J. Sartori and R. Kumar, "Distributed peak power management for many-core architectures," in *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE'09.* IEEE, 2009, pp. 1556–1559.

[27] J. M. Cebrián, J. L. Aragon, and S. Kaxiras, "Power token balancing: Adapting cmps to power constraints for parallel multithreaded workloads," in *Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International*. IEEE, 2011, pp. 431–442.

[28] 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.

[29] K. Meng, R. Joseph, R. P. Dick, and L. Shang, "Multi-optimization power management for chip multiprocessors," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*. ACM, 2008, pp. 177–186.

[30] V. Kontorinis, A. Shayan, D. M. Tullsen, and R. Kumar, "Reducing peak power with a table-driven adaptive processor core," in *Proceedings of the 42nd annual IEEE/ACM international symposium on microarchitecture*. ACM, 2009, pp. 189–200.

[31] "Rutgers Discovery Informatics Institute (RDI^2)." [Online]. Available: https://rdi2.rutgers.edu/

[32] M. S. Day and J. B. Bell, "Numerical simulation of laminar reacting flows with complex chemistry," *Combustion Theory and Modelling*, vol. 4, no. 4, pp. 535–556, 2000.

[33] G. L. Bryan, M. L. Norman, B. W. O'Shea, T. Abel, J. H. Wise, M. J. Turk, D. R. Reynolds, D. C. Collins, P. Wang, S. W. Skillman *et al.*, "Enzo: An adaptive mesh refinement code for astrophysics," *The Astrophysical Journal Supplement Series*, vol. 211, no. 2, p. 19, 2014.

[34] M. Frankel, P. Gordon, and G. Sivashinsky, "On disintegration of near-limit cellular flames," *Physics Letters A*, vol. 310, no. 5-6, pp. 389–392, 2003.

[35] "End user license agreement." [Online]. Available: ftp://ftp.supermicro.com/utility/SMCIPMITool/SMCIPMITool_User_Guide.pdf