Exploring Power-Performance-Quality trade-offs for Exascale Combustion Simulation

Authors Name/s per 1st Affiliation (Author)
line 1 (of Affiliation): dept. name of organization
line 2: name of organization, acronyms acceptable
line 3: City, Country
line 4: Email: name@xyz.com

Authors Name/s per 2nd Affiliation (Author)
line 1 (of Affiliation): dept. name of organization
line 2: name of organization, acronyms acceptable
line 3: City, Country
line 4: Email: name@xyz.com

***Abstract*—The computational demand of high-performance computing (HPC) applications has brought major changes to the HPC system architecture. As a result, it is now possible to run simulations faster and get more accurate results. But behind this, power and energy are becoming critical concerns for HPC systems, e.g. Titans electric cost is about $9 million per year[?]. Power efficiency has become a critical challenge for the exascale research challenges, and U.S. Department of Energy (DOE) has set the goal to achieve exascale performance with a power budget of 20MW[?].**

Current research efforts have studied power and performance tradeoffs, and how to balance these, e.g., using DVFS to meet power constraints, which significantly impacts performance. However, scientific 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 tradeoff decisions.

This research focuses on studying the properties and exploring the performance and powerenergy tradeoffs of Low-Mach-Number Combustion (LMC) application which is an Adaptive Mesh Refinement (AMR) algorithm. Our experimental evaluation provides an empirical evaluation of different application configurations that gives insights into the power-performance tradeoffs space for this LMC or AMR-based application workflows. The key contribution of this work is a better understanding of the running behavior of this AMR-based application and the power-performance tradeoffs for this application under power constraints, which can be used to better schedule power budgets across HPC systems.

***Keywords*-Power-Performance-Quality, trade-offs, Exascale**

I. Introduction

High performance computing (HPC) have been played an important role in the field of computational science in 21st century. However,as we are approaching the exascale era, power is turning from an optimization goal to a critical operation constraint. U.S Department of Energy (DOE) has currently set a bound of 20MW for an exascale system.[?] This strict power constraint poses a hard research challenge with current hardware and software.

In order to achieve this exascale system power constraint, many research efforts have studied power and performance tradeoffs, and most energy models or strategies are based on runtime (e.g., leveraging MPI slack) for power clamping or power capping techniques, like Dynamic Voltage and Frequency Scaling (DVFS) to constraint the power. However, power and performance are in the two sides of a balance scale, that it is hard to improve one side without scarifying the other one. Therefore, one of the key problems addressed in this research is keeping the power bound (or budget) without losing performance, which is challenging for real world application targeting exascale.

To address this challenge, we believe that the applications should be involved in making tradeoff decisions. This research targets a Center for Exascale simulation of Combustion in Turbulence (ExaCT) Low-Mach-number Combustion simulation code (LMC). [**?**] This combustion simulation is developed based on Adaptive-Mesh-Refinement (AMR) algorithm. The AMR method can be customized to resolve problems at different resolution levels. This property motivates us to find the potential tradeoffs to run applications in power-constrained environments without impacting performance by tolerating lower resolution/quality levels, together with power capping.

The contribution of this work are 1) present an empirical evaluation of different configurations of application that gives insights into the energy-performance-quality tradeoff for scientific data-driven workflows. 2) provide a comprehensive study of this LMC simulation performance, quality, and power and energy behavior. 3) present a proof-of-concept study of potential of power capping and power management to balance power-performance-quality tradeoffs.

The rest of this paper is organized as follows. Chapter 2 summarizes the related work. Chapter 3 presents background information related to the techniques used in this research. Chapter 4 describes the evaluation methodology and presents the results of the experimental evaluation. Chapter 5 concludes the chapter and outlines ongoing and future research.

II. Related works

A. *Energy efficiency*

Energy-efficiency has become a critical concern for HPC applications. There are many approaches have been proposed to obtaining energy savings during HPC application execution. Some of them are to focus on identifying stalls during the execution by measuring architectural parameters from performance counters as proposed in[?], [?], [?]. In addition to using performance counters, Rountree et al.[?] developed a runtime system called Adagio, by doing the critical path analysis, it can determine which tasks may be slowed down and also suitable opportunities to apply DVFS to minimize the performance loss in the parallel execution. This achieves significant energy saving in scientific application with negligible performance lose. However, this approach appears beneficial when applications have computation or communication imbalances among participating processes, which is typically not the case for a highly efficient parallel application nor suitable for the LMC program targeted in this work. Some approaches in[?], [?] are proposing to determine the communication phases to apply DVFS. Kandalla et al.[?] give algorithms to save energy in the collectives such as MPI_Alltoall and MPI_Bcase. Moreover, Ioannou et al.[?] describe a runtime system for the Intel Single-chip Cloud Computer (SCC) processor to detect repeatable communication phases followed by an application of frequency scaling. Donofrio et al.[?] have studied energy efficiency for extreme-scale science and developed Green Flash, an application-driven design to improve the kernels computational efficiency. Li et al.[?] developed a new power-aware performance prediction model of hybrid MPI/OpenMP programming model and combine memory and disk management techniques to provide performance guarantees for control algorithms. Rodero et al.[?] explored the potential of application-centric aggressive power management of HPC scientific workloads while considering power management mechanisms and controls available at different levels and different subsystems. Gmall et al.[?] explored data-related energy/performance trade-offs for end-to-end co-design simulation workflows on current and on-going high-end computing systems. Lively et al.[?] explored and investigated energy consumption and execution time of different parallel implementation of scientific applications on multi-cluster systems.

Most of the work described above are from system level and are based on tolerating performance loss and constrain the power consumption under that performance loss. The work presented here discusses the possibility to filter down power constraint to job-level, and then take advantage of applications specific properties, such as resolution, to keep the power budget while maintaining the level of performance.

B. *Power management*

The cost of provisioning power in data centers is a very large fraction of the total cost of operating a data center.[?], [?] Therefore the ability to cap peak power consumption is a desirable feature in modern data centers.[?] Many power management approaches have been proposed to provide performance guarantees while constraining power budget. Sartori et al.[?] describe a peak power management technique for multi-core systems by choosing the power state for each core that meets the power constraints. Cebrian et al.[?] develop a power balancing strategy that through borrowing power budgets from cores that consume lower power to dynamically adapts the per-core power budgets. While, Gandhi et al.[?] give a power capping strategy to meet the power budget by inserting idle cycles during execution. This approach aims at controlling the average power consumption, but cannot guarantee the peak power. So a number of other approaches are proposed through reconfiguring hardware to meet the power budget. Meng et al.[?] provide a power management strategy through dynamic reconfiguration of cores by cache resizing. Konotorinis et al.[?] propose a table-driven adaptive core reconfiguration technique that configures core resources such as load-store queues and floating point units to meet peak power budget. Most of the power management strategies are using DVFS. Since Intel SandyBridge family processors, Intel provide Running Average Power Limit (RAPL) for controlling the power constraint on processors and memory. Several studies have begun to evaluate the RAPL power management system. Rountree et al.[?] explore RAPL as a replacement for DVFS in HPC systems by evaluating power consumption for package and memory subsystem. Zhang et al.[?] give a systematic evaluation of RAPL behavior such as stability, setting time, overshoot and, etc. RAPL also been used to study application runtime variability and power optimization for exascale computing in work of Allan et al.[?][21] Sarood et al.[?] use RAPL to set power bounds on across an over-provisioned cluster running homogeneous application processes. This work also takes advantage of RAPL power controlling ability to dynamically manage power consumption.

C. *Power measurement systems*

Measuring the power/energy consumption of software components is the key for energy-aware scheduling, accounting and budgeting. RAPL measurement mechanism is described in[?] and Marcus et al.[?] has investigated the RAPL measurements performance and discussed the practical obstacles that existed in performing these measurements on complex modern CPUs. Vignesh et al.[?] studied the greenness of the in-situ and the post-processing visualization pipelines using RAPL to measure the CPUs and RAMs power consumption and with an average error rate of less than 1%. Venkatesh et al.[?] use RAPL to measure energy consumption in large message-passing applications.

III. Experimental evaluation

A. *Infrastructure overviews*

1) CAPER Cluster: CAPER (Computational And data-enabled Platform for Energy efficiency Research) is a unique and flexible instrument funded by The National Science Foundation that combines high performance Intel Xeon processors with a complete deep memory hierarchy, latest generation co-processors, high performance network interconnect and powerful system power instrumentation.

It is currently an eight-node cluster, which is capable of housing concurrently, in one node up to eight general-purpose graphical processing units (GPGPU), or eight Intel many-integrated-core (MIC) coprocessors - or any eight-card combination of the two; and up to 48 hard disk drives (HDD), or solid-state drives (SSD). A single-node configuration features a theoretical peak performance of 20TF/s (single precisions) or 10TF/s (double precision) and supports up to 32TB of storage and 24TB of flash-based non-volatile memory. A separate node serves as a login and storage server.

This hardware configuration is unprecedented in its flexibility and adaptability as it can combine multiple components into a smaller set of nodes to reproduce specific configurations. This platform also mirrors key architectural characteristics of high-end system, such as XSEDE's Stampede system at TACC, and provides several unique features to support critical research goals such as software/hardware co-design. CAPER provides a platform to validate models and investigate important aspects of data-centric and energy efficiency research.

2) DELL Cluster: DELL platform includes a 32 node Dell M610 blade cluster, consisting of two Dell M1000E Modular Blade Enclosures, necessary interconnect/management infrastructure, and a supervisory node. Each enclosure is maximally configured with sixteen blades, each blade having two Intel Xeon E5504 Nehalem family quad-core processors at 2.0 GHz, forming an eight core node. Each node has 24 GB RAM and 73 GB of local disk storage (10,000 RPM), twelve nodes have an additional 1 TB of local storage. The network infrastructure is comprised of an integrated 16-port Mellanox InfiniBand switch within each blade chassis, each switch linked to the switch in the other chassis. All blades have Mellanox Quad-Data-Rate (QDR) InfiniBand interface cards. There is also an integrated (redundant) 1 Gigabit Ethernet within each chassis, with two pairs of 10 Gigabit uplink capabilities in each chassis. In the aggregate, the cluster system consists of 32 nodes, 256 cores, 768 GB memory and 14.5 TB disk capacity, with a 20 Gigabit InfiniBand network and two 1 Gigabit Ethernet networks.

B. Methodology

1) Power measurement methodology: Our experiment platform CAPER is instrumented with both coarse- and fine-grained power metering at server level. On the one hand, an instrumented Raritan iPDU PX2-4527X2U-K2 provides power measurements at 1 Hz. While, on the other hand, a Yokogawa DL850E ScopeCorder provides voltage and current measurement for all nodes through 1 Ms/s modules, and can sample power data at rate of up to 10 KHz. In addition to those server level measurement, we have RAPL meter to provide power measurement at a up to 20Hz sampling rate in processor level.

Use of RAPL power metering
RAPL meter measuring power through reading Machine-Specific Registers (MSRs). Intel released Power Gadget API for using RAPL. This API is a framework that provides very comprehensive information including reading current estimated processor power, current processor frequency, base frequency, thermal design power (TDP), current temperature, timestamps and etc. Also, Intel gives a ready-to-use software-based power monitoring tool called Intel Power Gadget, this has a completed interface for using RAPL. We are using this to measure the processors power usage.

RAPL meter sampling frequency considerations
Thought, according to Intels manual,[**?**] RAPL MSRs would update at rate of once every 1 ms, using very lower frequency (e.g. less than 20 ms) may result in significant overhead and might also increase the power consumption, which would make the data less meaningful.[**?**] Also, since the instantaneous processor frequency would change very frequently, sampling data may be more useful if you sample often and average the samples overtime. So, Intel recommend sampling frequency of 50 ms or upper. For our experiment, we dont require high resolution power data, so we choose to sample twice per second.

Reading power measurement data from PDU
Raritan PDU keep measuring the whole servers power consumption and writing the power measurement data to power log files with timestamp. CAPER has eight nodes, so we extract real-time power measurement data and the timestamp from each nodes power log file after simulation started using Simple Network Management Protocol (SNMP) queries to the PDU from a side script running in an independent node.

2) AMR resolution adjustment: LMC simulation program is based on AMR algorithm, which uses a hierarchical grid structure, and fill finer patches on the region of interest. Therefore, the simulations resolution from the computational point of view is mainly directed from the AMR algorithm resolution configuration. There are many parameters controlling AMR solution, but in the work, only those four parameters listed in Table I have been used to tune the simulation resolution.

Parameter	**Definition**
amr.n_cell	Number of cells in each direction at the coareset level
amr.max_level	Number of levels of refinement above the coarsest level
amr.ref_ratio	Ratio of coarse to fine grid spacing between subsequent levels
amr.regrid_int	How often to regrid

Table I: AMR parameters which are used to tune simulation resolution

Each parameter has different impact on the resolution, which is illustrated with the example use case shown in Figure 1. The parameters of the example case are discussed as follows.

Example case:

- amr.n_cell = 32 32 32

This would define the domain size (at coarsest level) to have 32 cells in the x-direction, 32 cells in the y-direction and 32 cells in the z-direction. (If it is in the 2D input file, the last number will be ignored). As shown in Figure 1, there are 32 cells in both x and y direction.

- amr.max_level = 2

This would set a maximum of 2 refinement levels in addition to the coarse level. Within the calculation, the number of refinement level must be $\leqslant$ amr.max_level, but it can be change in time and it is not necessary always be equal to amr.max_level. Because these additional levels will only be created if the tagging criteria are such that cells are flagged as needing refinement. Figure 1 shows the mesh grids with maximum of 2 refinement levels.

- amr.ref_ratio = 4 2

Refinement ration means how many individual cells will a cell be divided into. For example in the left-hand side Figure 1, Setting amr.ref_ratio = 4 2 means dividing cell into 4 cells from levels 0 and 1, and dividing cell into 2 from levels 1 and 2.

- amr.regrid_int = 2

This would tell the code to regrid every 2 steps. This means level l+1 grids will be created every 2 level l time steps.

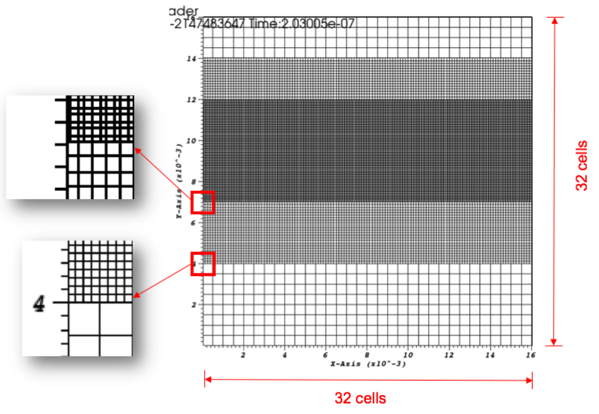

Figure 1: AMR mesh grids outcome of the example configuration

C. Results

1) Evaluation of different levels of refinement for different number of cores: This experiment is first run on CAPER cluster, with a maximum of 128 cores, from level 1 to level 4 on 2, 4, 8, 16, 32, 64 and 128 cores. The execution time is recorded simply with the system tool "time".

Configuration

Parameter	Value
Platform	CAPER and DELL cluster
Number of cores	2 to 128 quadratic growth
Level of refinement	level 1 to level 4
Time step	10
Time unit	Second

Table II: Configuration for experiment of execution time of different levels of refinement under variety number of cores

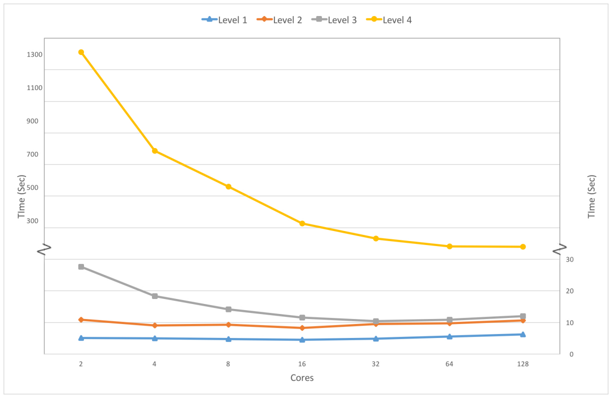

Figure 2: Execution time of running different levels LMC on CAPER cluster through 2 to 128 cores.

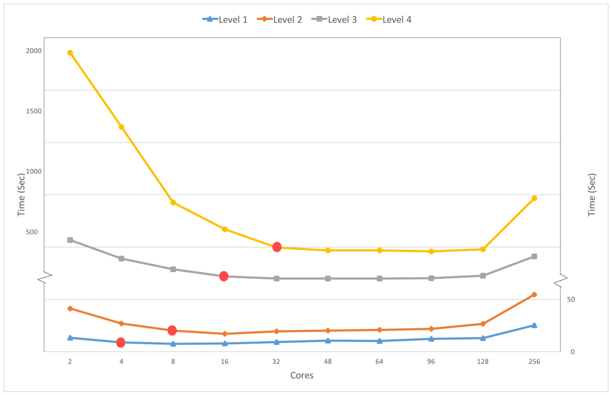

Figure 3: Execution time of running different levels LMC on DELL cluster through 2 to 256 cores.

The results of the first experiment using CAPER cluster are shown in Figure 2. As can be clearly seen from the level 4 line, which is yellow line with solid dot maker, the more computing resource we use, the faster the execution is. Comparing these four lines, we can also notice that the slope is decreasing from level 4 to level 1, and then the lines go to a relative flat zoom. From this notice, we can see that heavier workload need more computational resources, but provisioning more resources do not bring any performance increase. In Figure 2 it can be seen from level 1 to level

3 lines that their tails are tilting a little bit, but level 4 line doesnt have this trend. In order to show its trend more clearly, we ran this experiment on DELL cluster (256 cores cluster). The result is shown in Figure 3. All levels' curve is like a "U", that tilt in the head and tile. From level 1 to level 4, the head turning point, which marked by red dot, are respectively 4, 8, 16 and 32 cores. Then they tend to get into a flat zoom, while under the 256 core, the execution time are all increase. From these two experiments we can conclude that, to achieve optimal running time, we need to configure appropriate number of processors to execute certain level of quality for the LMC program and using more processors does guarantee better performance.

2) Evaluation of energy consumption of different levels of refinement under variety number of cores: In this experiment, we are aim at exploring the energy consumption of running LMC with different level of refinement under multiple number of cores.

Energy is the product of power and time, therefore, in this experiment, we need to measure the execution time and power respectively. To measure time, we still use Linux command TIME in the script file. And for the CPU power, we are using RAPL power meter to measure it. RAPL power meter is a sub-function of Intel Power Gadget program, which has been mentioned in previous background section. We are sampling power data every 0.5 second.

Configuration

Parameter	Value
Platform	CAPER cluster
Number of cores	2 128 quadratic growth
Level of refinement	level 1 to level 4
Time step	10
Time unit	Second
Power measurement object	8 nodes processors
RAPL sample rate	2 Hz

Table III: Configuration for experiment of power consumption of different levels of refinement under variety number of cores

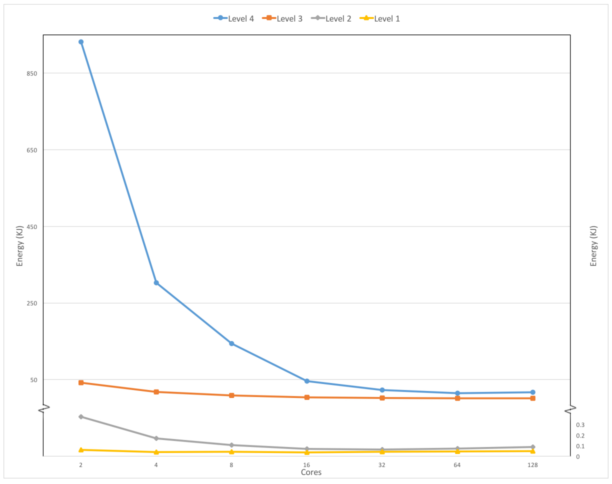

Figure 4: Power consumption of different levels of refinement under different number of cores

Figure 4 shows that there is a huge energy difference on the beginning of this curve. When running on 2 cores, the highest one (level 4) consumes almost 930 KJ while the lowest one (level 1) only take around 0.6 KJ. Since this experiment measure the all 8 node processors power status, the huge energy consumption of level 4 comes from the long execution time. It takes 1,314 seconds to run level 4 job while only 10 seconds to execute level 1. Most energy consumption comes from processors in idle state. But when assign appropriate processor resources to it, the curve reaches the flat zoom, the energy consumption difference is reasonable and stable. From this point of view, an appropriate number of cores for a certain workload is very crucial for energy consumption.

3) Evaluation of the impact of RAPL power capping on LMC performance : This experiment is aimed at studying the impact of power capping on LMC power-performance. And it would also give us a reference about the optimal power capping setting for different resolution levels.

We use level 1 to level 4 in LMC as inputs to measure the time and power consumption. Since CAPER CPU power is 95W (Intel Xeon E5-2650 V2), in this experiment, we cap the CPU power to 25W, 35W, 45W, 55W, 65W, 75W, 85W, and 95W and record the package energy consumption (PKG) and execution time respectively. Package energy consumption is the energy used by the CPU chip itself including cores, caches and graphics, etc.

Configuration

Parameter	**Value**
Platform	CAPER cluster
Number of cores	64 cores
Level of refinement	level 1 to level 4
Time step	10
Time unit	Second
Power measurement object	8 nodes processors
Power capping	95W to 25W decrease by 10
RAPL sample rate	2 Hz

Table IV: Configuration for experiment of exploring the affection of RAPL power capping on LMC performance

We use Intel *power_gadget* tool to cap the power. To use it, we assign value to the parameter *MY_POWER_LIMIT*, then compile and run it. We use linux command line "time" to measure the execution time. The execution command line is provided as follows:

```
time mpirun
-machinefile
hostfile.txt -np 64
.LMC2d.Linux.g++.gfortran.SDC.MPI.ex
inputfile
amr.max_lev=#level
```

For the power measurement, we also use the Intel *power_gadget* tool. When running it, it will give the real-time power consumption data every 0.5 second.

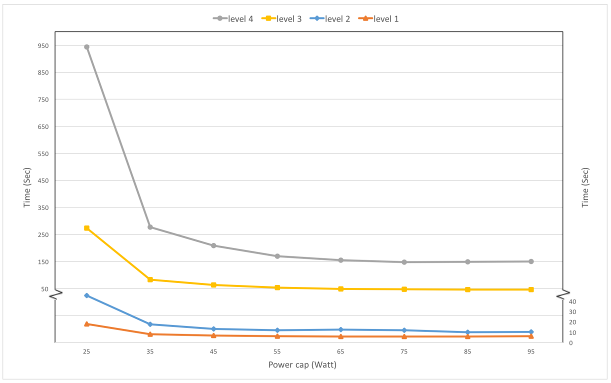

Figure 5: Relationship between different resolution of LMC and their execution time under different power capping levels

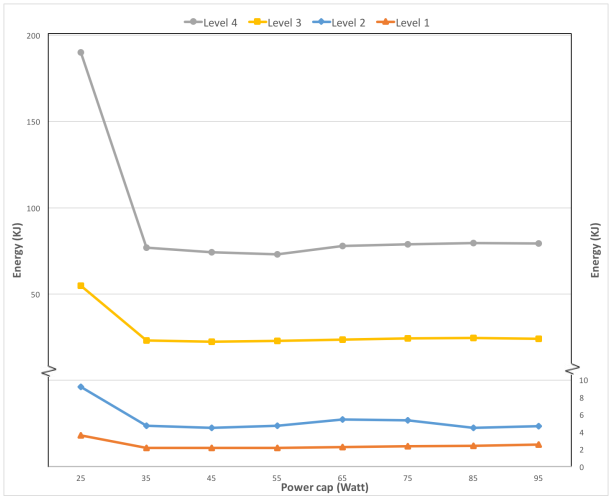

Figure 6: Relationship between different resolution of LMC and the energy consumption under different power capping levels

Figure presents the energy consumption result of executing level 1 to level 4 LMC program on certain number of cores under different power capping configurations. Those curves have the same pattern that sharply decreasing from 25 to 35 and becoming flat between 65 and 95. The lowest energy consumption is appearing when CPU power was capped to 45W to 55W. This can be explaining from execution data that without any CPU power capping, LMC program will boots CPU to 65W to 75W during execution whatever executing on how many cores. This also can be seem from the flat curve on segment 65W to 95W on each chart. Therefore, the advantage of power capping is working when CPU power down below 65W, and optimize between 45W to 55W.

4) Evaluation of power budget acquisition through power capping and resolution degradation: The previous experiments have characterized the behavior of LMC. It can be concluded that the execution time will increase as the level of refinement/resolution increases or CPU power is capped down. Also, the energy consumption trend for different power caps and refinement levels is shown in Figure 7. The energy consumption presents the same trend as the execution time, i.e., LMC will consume more energy as the levels of refinement/resolution increase or capping down CPU power. The question here is how to use this characterization to create available power budget. We propose to combine these two factors, adjusting resolution and applying appropriate power capping, to get available power budget for running other tasks (e.g., checkpointing).

Configuration

Parameter	Value
Platform	CAPER cluster
Number of cores	64 cores
Resolution	level 3 2 1
Time step	10
Time unit	Second
Power measurement	RAPL meter
Power cap	RAPL

Table V: Configuration for experiment of getting available power budget through power capping and resolution degradation

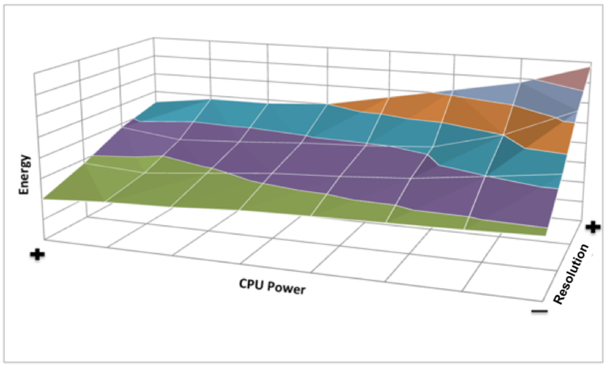

Figure 7: Energy consumption trend for different power caps and refinement levels

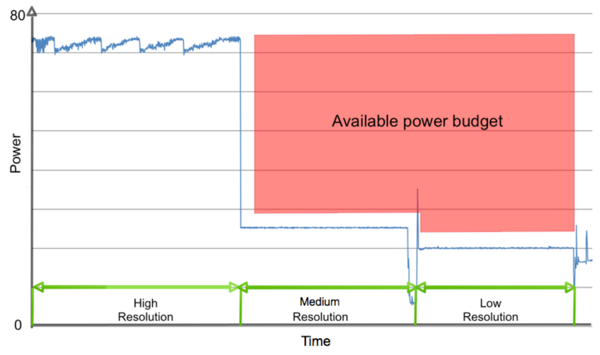

Figure 8: Available power budget from applying resolution degradation and appropriate power capping

In previous Figure **??**, we measured the power consumption of running LMC with different resolution. It shows that the execution time decreases when decreasing the resolution. Therefore, we apply appropriate power capping to the the medium and low resolution execution to make their execution equal the execution time in highest resolution. As shown in Figure 8, the red region is the available power budget extracted from the total power budget.

5) Evaluation of power budget management: We have tested the impact of power capping and level of refinement of LMC on the performance and energy consumption. We also observed the potential power saving from degrading resolution and implementing power capping. The goal of this experiment is being able to use power budgets opportunistically for running other tasks. In the experiment, we propose to use this power budget to do checkpointing, which will increase the resilience and make the LMC program running more reliably.

In this experiment, we are proposing to degrade LMC resolution and cap its running power, then use this power budget to do check pointing. However, there are some limitations due to the code characteristics: (i) we can not separate the checkpoint function from simulation program, and (ii) we can not dynamically control LMC resolution.

To address these two issues, we use two LMC executions running alternatively on two set of nodes to emulate one execution dynamically adjusting resolution and doing checkpoint. Each set of nodes contains three nodes for execution. When the first execution start doing checkpoint then the second set start another execution with lower resolution. We focus on system level power consumption.

We implement this using a flag to coordinate these two LMC executions. We've recompiled the first LMC set program to let it write flag when it start to do checkpoint, and the second LMC set keeps reading the flag to start running when the flag turn to be true. As shown in Figure 9, when the blue curve is the first instance (set 1) of LMC finishes its execution, it triggers the second instance (set 2), which is the orange curve. Once the second instance is completed the first one is triggered again and so on. By doing this, we can emulate a single LMC execution running and dynamically change the resolution while giving the available power budget for doing checkpoint.

Configuration

Parameter	Value
Platform	CAPER cluster
Number of cores	64 cores
Resolution	level 4 3
Time step	10
Time unit	Second
Power measurement object	3 nodes processors
Power cap	RAPL

Table VI: Configuration for experiment of exploring the power-performance tradeoffs

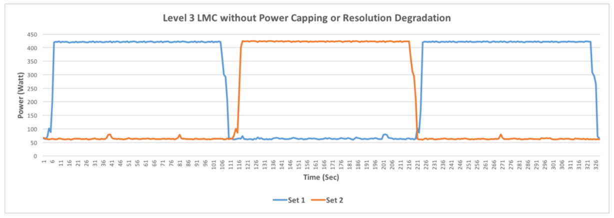

Figure 9: Level 3 LMC power consumption without power capping or resolution degradation

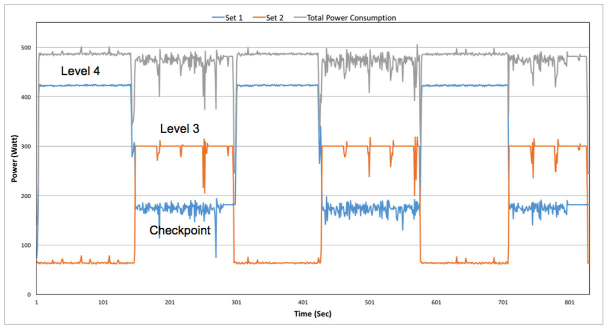

Figure 10: Level 4 LMC power consumption with power capping or resolution degradation

In Figure 10, the blue curve is the set 1 running the high resolution LMC (level 4). Once it starts doing checkpointing, the set 2 (in orange curve) is triggered to run LMC at level 3. Configuring power capping appropriately, the total power consumption, which is shown in gray curve on the top, is kept constant. This means we successfully constraint the power budget of LMC to perform other tasks (i.e., checkpointing).

IV. Conclusion and future work

In this work, we have focused on studying the properties and exploring the performance, quality and power/energy tradeoffs of Low-Mach-Number Combustion (LMC) application which is based Adaptive Mesh Refinement (AMR) algorithm. The key contributions of this work are (1) we present an empirical evaluation of different configurations of application that gives insights into the energy-performance-quality tradeoff for this work, (2) we provided a comprehensive study of this LMC simulation performance, power and energy behavior, and (3) we propose a power-performance-quality tradeoff for this application, which can be used to better schedule power budgets across HPC systems.

Our current work investigates how to leverage these insights to implement a runtime to manage power budgets dynamically. Moreover, instead of trading-off with resolution, we also plan to explore the possibility of managing resources dynamically (e.g., cores) to manage power budgets. Future work also includes the management and scheduling of power budgets at whole system scale for assigning power budgets to other applications, not necessarily to the LMC workflow.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.