Analysis of the Tradeoffs between Energy and Run Time for Multilevel Checkpointing

Prasanna Balaprakash[1,2], Leonardo A. Bautista Gomez[1], Mohamed-Slim Bouguerra[1], Stefan M. Wild[1], Franck Cappello[1,3], Paul D. Hovland[1]

[1] Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL
[2] Leadership Computing Facility, Argonne National Laboratory, Argonne, IL
[3] University of Illinois at Urbana-Champaign
{pbalapra,leobago,medslim,wild,cappello,hovland}@anl.gov

Abstract. In high-performance computing, there is a perpetual hunt for performance and scalability. Supercomputers grow larger offering improved computational science throughput. Nevertheless, with an increase in the number of systems' components and their interactions, the number of failures and the power consumption will increase rapidly. Energy and reliability are among the most challenging issues that need to be addressed for extreme scale computing. We develop analytical models for run time and energy usage for multilevel fault-tolerance schemes. We use these models to study the tradeoff between run time and energy in FTI, a recently developed multilevel checkpoint library, on an IBM Blue Gene/Q. Our results show that energy consumed by FTI is low and the tradeoff between the run time and energy is small. Using the analytical models, we explore the impact of various system-level parameters on run time and energy tradeoffs.

1 Introduction

Large-scale scientific simulations require larger supercomputers to produce more accurate results. In high-performance computing (HPC) researchers and engineers are pushing the envelops to increase scalability and performance. As systems scale, new challenges appear, in particular, two major challenges for next generation supercomputers consists of minimizing power/energy consumption and maximizing reliability. However, these two objectives are in conflict which each other because increased reliability comes at the expense of power and energy usage.

Researchers in the HPC community have developed various fault tolerance techniques to improve the reliability of current and future machines. Nevertheless, all these techniques involve overheads in terms of storage space, computation and their respective energy consumption, hinting at the existence of a tradeoff between execution run time and energy efficiency. Multilevel checkpointing is a promising approach to deal with reliability at extreme scale. The key idea of this approach consists in using layers of checkpointing, each one of them offering different levels of resilience and overheads. Low-cost levels offer limited fault tolerance while highly resilient levels involve large overheads. Consequently, the correct usage of the multiple levels should lead to substantial gains in performance, resilience, and energy consumption.

In this paper, we study the impact of optimal multilevel checkpointing intervals on the tradeoffs between run time and energy consumption. Our experimental study with FTI, a multilevel checkpointing library on an IBM Blue Gene/Q supercomputer shows that performance-energy tradeoffs are minimal, but may be significantly larger under certain future exascale HPC scenarios. The contributions of this paper are as follows:

- We derive analytical models for expected run time and energy consumption for multilevel checkpointing.
- We characterize the Pareto-optimal solution set and investigate the tradeoffs between time and energy consumption.
- We perform power consumption measurements of large-scale executions on an IBM Blue Gene/Q with several applications.
- We present an experimental study to analyze several system-level parameters for multilevel checkpointing that can potentially impact the tradeoffs.

The rest of the paper is organized as follows. Section 2 describes the main concepts used by multilevel checkpointing. Section 3 introduces models for time and energy for multilevel checkpointing strategies. We introduce the notion of Pareto optimality in Section 4. Section 5 presents the results of our empirical evaluation and several future tradeoff projections. Section 6 reviews related work, and Section 7 presents conclusions and a brief look at future work.

2 Multilevel Checkpointing

Long-running scientific simulations executed on large supercomputers are checkpointed periodically to stable storage in order to avoid having to restart from the beginning in case of failure. Traditionally, applications will stop, write all the required data to the parallel file system (PFS), and then continue. Checkpoint sizes have been constantly increasing with the exponential growth of supercomputers. Unfortunately, the speed at which one can write to the PFS has been increasing only linearly, leading to long checkpointing times and causing large overhead to the application.

To minimize the impact of checkpointing on run time, researchers have proposed multilevel checkpointing [4,13] which leverages multiple storage layers and limits the load on the PFS. This is achieved by using local storage in the compute nodes. However, local storage is not resilient against node crashes, even for persistent storage devices, as access to those devices might be lost after a failure. Therefore, local storage is usually coupled with data replication or erasure codes to guarantee that any unaccessible data can be reconstructed. We used the multilevel checkpointing library FTI [4] that provides four checkpoint levels, namely, Local checkpoint, Local checkpoint + Partner-copy, Local checkpoint + Reed-Solomon coding, and PFS-based checkpoint. Note that the model developed proposed in this paper can be used to analyze other multilevel checkpoint libraries.

Applications using FTI can perform checkpoints of different levels at different frequencies. Those frequencies can be easily configured through a configuration file. When a checkpoint of level i is done, FTI automatically removes all previous checkpoints of level j for $j \leq i$ because i is more recent and offers more reliability. Previous checkpoints of level k for $k > i$ are kept however, so that if a failure cannot be recovered by using level i, it can try to recover from a higher level. In addition to these four checkpointing levels, FTI offers features such as having dedicated processes that perform fault-tolerance tasks in the background, which speeds the checkpoints and limits the overhead imposed on the application's run. Dedicated processes could, for instance, copy a local checkpoint to the PFS in the background at the same time the application is running. In this way, applications are blocked only to perform the local checkpoint; all the rest of the work associated with addressing fault tolerance is hidden.

3 Energy and Checkpoint Models

A multilevel checkpoint strategy is defined by the intervals between checkpoints. We denote these intervals by the vector $\tau \in \mathbb{R}_+^L$, where L is number of different levels of checkpointing and the i^{th} component, τ_i, of the vector τ denotes the amount of time between checkpoints at level i. The checkpoint cost (in terms of time) at level i is denoted by c_i.

After a failure, the application uses the most recent checkpoint to restart the application. Suppose we have a failure at level i, the restart time is r_i and the down time is d_i. For a failure model we consider μ_i as the rate of failures affecting only level i. Hence, μ_1 corresponds to the rate of transient failures; μ_2 is the rate of permanent failures that affect many nodes but not two buddies at the same time; μ_3 represents the rate of failures affecting at least one partner node at the same time; and μ_4 is the rate of failures that occur at the same time and affect at least one group at the same time. Several derivations of μ_i are provided in [7,13]. Also, we note that $1/\mu_i$ can be interpreted as the mean time between failures at level i. The basic model notation is summarized in Table 1, with all times and powers taken in expectation.

Table 1: Summary of model notation.

	Description
τ_i	Time between level i checkpoints
c_i	Time for a level i checkpoint
r_i	Time for a restart from level i
T_a	Time for a failure-free computation without checkpointing
d_i	Downtime after a failure affecting level i
L	Number of levels
μ_i	Expected rate for failure affecting level i
$\mathcal{P}_i^c$	Power for a level i checkpoint
$\mathcal{P}_i^r$	Power for a restart from level i
$\mathcal{P}^a$	Power for a failure-free computation without checkpointing

3.1 Model for Run Time

We express the expected overall completion time as the sum of two times: the time for a failure-free execution of an application without checkpointing and the expected time wasted because of failures and/or checkpointing, $T_{\text{overall}} = T_a + T_{\text{wasted}} = T_a + \mathbb{W}T_{\text{overall}}$. The amount of waste per unit of time, $\mathbb{W}$, comprises the time to perform checkpointing, rework, and restart, as well as the downtime. We now examine the contributors to the wasted time: the checkpoint overhead per unit of time $\mathfrak{W}^{ch}$, the rework overhead per unit of time $\mathfrak{W}^{rew}$, and the restart per unit of time $\mathfrak{W}^{down}$.

Checkpoint overhead. We have two sources of overhead because of checkpointing. The first is based on the number of checkpoints performed in one unit of time. The number of checkpoints can be approximated by $\frac{1}{\tau_i}$. A tighter approximation is given by $\frac{1}{\tau_i + c_i}$, but $\frac{1}{\tau_i}$ is a good upper bound. The second term, $\mu_i \tau_i \sum_{j=1}^{i-1} \frac{c_j}{2\tau_j}$, represents the expected lost time due to extra checkpoints at levels $1, \ldots, i-1$ if a failure occurs at level i. The overall fraction of time spent in checkpointing is thus given by

$$\mathfrak{W}^{ch} = \sum_{i=1}^{L} \left(\frac{c_i}{\tau\ i} + \mu_i \tau_i \sum_{j=1}^{i-1} \frac{c_j}{2\tau_j} \right).$$

Rework time. We follow the classical first-order approximation and assume that a failure occurs at the half of the interval. The expected lost time due to re-execution (rework) is thus

$$\mathfrak{W}^{rew} = \sum_{i=1}^{L} \frac{\mu_i \tau_i}{2}.$$

Downtime and restart. The expected wasted time because of downtime and restart is

$$\mathfrak{W}^{down} = \sum_{i=1}^{L} \mu_i (r_i + d_i).$$

The total waste per unit time, $\mathbb{W}$, is thus given by

$$\sum_{i=1}^{L} \left(\frac{c_i}{\tau_i} + \frac{\mu_i \tau_i}{2} \left(1 + \sum_{j=1}^{i-1} \frac{c_j}{2\tau_j} \right) + \mu_i \left(r_i + d_i \right) \right). \tag{1}$$

3.2 Model for Energy

We now develop a model for the expected wasted energy per unit of time. We let $\mathcal{P}^a$, $\mathcal{P}_i^c$, and $\mathcal{P}_i^r$ denote respectively the amount of power (e.g., in watts) used by the user application to perform computation, checkpoint at level i, and restart from level i. Note that $\mathcal{P}^a$, $\mathcal{P}_i^c$, and $\mathcal{P}_i^r$ include the idle power as well.

We have the three sources of wasted energy:

$$\mathcal{E}^{ch} = \sum_{i=1}^{L} \left(\mathcal{P}_i^c \frac{c_i}{\tau_i} + \mu_i \tau_i \sum_{j=1}^{i-1} \frac{\mathcal{P}_j^c c_j}{2\tau_j} \right),$$

$$\mathcal{E}^{rew} = \sum_{i=1}^{L} \mathcal{P}^a \frac{\mu_i \tau_i}{2},$$

$$\mathcal{E}^{down} = \sum_{i=1}^{L} \mathcal{P}_i^r \mu_i (r_i + d_i),$$

corresponding to the checkpoint energy, the energy for rework because of failures, and the energy for restart, respectively.

3.3 Optimal Checkpoint Intervals

The optimal checkpoint intervals with respect to run time are obtained by minimizing (1) as a function of $\tau \in \mathbb{R}^L_+$. Similarly, the optimal intervals with respect to energy are obtained by minimizing the wasted energy during one unit of time,

$$\mathbb{E} = \sum_{i=1}^{L} \left(\frac{\mathcal{P}^c_i c_i}{\tau_i} + \mu_i \tau_i \left(\frac{\mathcal{P}^a}{2} + \sum_{j=1}^{i-1} \frac{\mathcal{P}^c_j c_j}{2\tau_j} \right) \right) + \sum_{i=1}^{L} \mathcal{P}^r_i \mu_i (r_i + d_i), \tag{2}$$

as a function of τ.

Under reasonable restrictions on the checkpoint intervals (based only on the failure rates μ; see the appendix), one can show that $\mathbb{W}$ and $\mathbb{E}$ are both convex over this restricted domain. Thus each has a unique optimal solution, which we can obtain, for example, using an iterative method such as Newton's method.

The first derivatives of equations (1) and (2) with respect to τ_i are given by

$$\frac{\partial \mathbb{W}}{\partial \tau_i} = \frac{\mu_i}{2} \left(1 + \sum_{j=1}^{i-1} \frac{c_j}{\tau_j} \right) - \frac{c_i}{\tau_i^2} \left(1 + \sum_{j=i+1}^{L} \frac{\mu_j \tau_j}{2} \right) \tag{3}$$

$$\frac{\partial \mathbb{E}}{\partial \tau_i} = \frac{\mu_i}{2} \left(\mathcal{P}^a + \sum_{j=1}^{i-1} \frac{\mathcal{P}^c_j c_j}{\tau_j} \right) - \frac{\mathcal{P}^c_i c_i}{\tau_i^2} \left(1 + \sum_{j=i+1}^{L} \frac{\mu_j \tau_j}{2} \right). \tag{4}$$

Setting these derivatives to zero, we note that the solutions for time and energy satisfy

$$\tau_i^{\mathbb{W}} = \sqrt{\frac{c_i (2 + \sum_{j=i+1}^{L} \mu_j \tau_j^{\mathbb{W}})}{\mu_i (1 + \sum_{j=1}^{i-1} \frac{c_j}{\tau_j^{\mathbb{W}}})}}$$

$$\tau_i^{\mathbb{E}} = \sqrt{\frac{\rho_i c_i (2 + \sum_{j=i+1}^{L} \mu_j \tau_j^{\mathbb{E}})}{\mu_i (1 + \sum_{j=1}^{i-1} \frac{\rho_j c_j}{\tau_j^{\mathbb{E}}})}},$$

respectively, with $\rho_i = \mathcal{P}^c_i / \mathcal{P}^a$.

When there is only a single level, the interval that minimizes run time is $\tau^{\mathbb{W}} = \sqrt{2c/\mu}$, while the interval that minimizes energy is $\tau^{\mathbb{E}} = \tau^{\mathbb{W}} \sqrt{\mathcal{P}^c / \mathcal{P}^a}$. Whenever $\mathcal{P}^c \neq \mathcal{P}^a$, we have that $\tau^{\mathbb{W}} \neq \tau^{\mathbb{E}}$, and hence the two objectives are conflicting, a subject we formalize next.

4 Tradeoffs Between Time and Energy

We now turn to the checkpoint-scheduling problem of minimizing *both* time and energy. Sometimes such *bi-objective optimization problems* have a single solution: there is a single decision that minimizes both objectives simultaneously. In other cases (such as seen at the end of Sec. 3), the objectives are conflicting, and many solutions may be "optimal" in the bi-objective sense.

The concept of two conflicting objectives is best illustrated by an example. Figure 1 shows the wasted time and energy per unit of time for a single-level checkpointing scheme (see Sec. 5 for details). The thinner curve illustrates the behavior of the objective pairs $(\mathbb{W}(\tau), \mathbb{E}(\tau))$. If the objective $\mathbb{W}$ [$\mathbb{E}$] is minimized in isolation, then we obtain the solution $\tau^{\mathbb{W}}$ [$\tau^{\mathbb{E}}$] and the corresponding

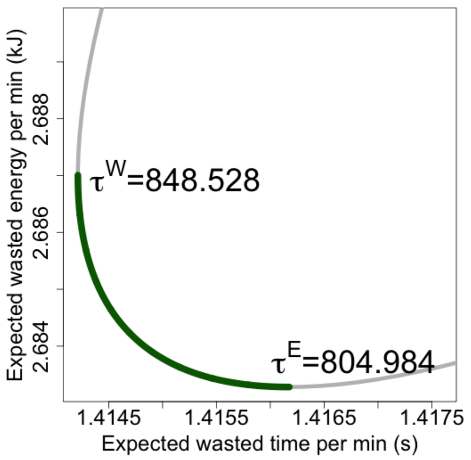


Fig. 1: Pareto front for single-level checkpointing for LAMMPS on BG/Q. The power for computing and checkpointing are 2 kW and 1.8 kW, respectively; μ=1/36000 Hz and the cost of a checkpoint is 10 s. The thin line shows the strategies dominated by the Pareto front (thick line).

point $(\mathbb{W}(\tau^{\mathbb{W}}), \mathbb{E}(\tau^{\mathbb{W}}))$ $\left[(\mathbb{W}(\tau^{\mathbb{E}}), \mathbb{E}(\tau^{\mathbb{E}}))\right]$ in Fig. 1. From a bi-objective perspective however, $\tau^{\mathbb{W}}$ and $\tau^{\mathbb{E}}$ provide only the boundary of the solution set: for any τ between the values $\tau^{\mathbb{W}}$ and $\tau^{\mathbb{E}}$, we obtain time and energy quantities that cannot both be improved upon. Formally, a point τ^j is dominated by a point τ^i when $\mathbb{W}(\tau^i) \le \mathbb{W}(\tau^j)$ and $\mathbb{E}(\tau^i) \le \mathbb{E}(\tau^j)$ (with at least one of these inequalities being strict). A point τ^i is said to be *Pareto-optimal* if it is not dominated by any other τ^j. The set of $(\mathbb{W}, \mathbb{E})$ values from all Pareto-optimal points is called the *Pareto front* (illustrated by the bold portion of the curve in Fig. 1); see [3,8] for further details.

In general, Pareto fronts can be nonconvex, and finding Pareto-optimal points can be a task significantly more challenging than optimizing a single objective. When the Pareto front is convex, any point on the front can be obtained by minimizing a linear combination of the objectives. This corresponds to minimizing the single objective

$$f_\lambda(\tau) = \lambda \mathbb{W}(\tau) + (1-\lambda)\mathbb{E}(\tau), \tag{5}$$

where $\lambda \in [0,1]$ represents the weight placed on $\mathbb{W}(\tau)$. For convex Pareto fronts, solving (5) for all $\lambda \in [0,1]$ yields the Pareto-optimal solutions, with the extreme case $\lambda = 1$ ($\lambda = 0$) corresponding to minimizing time (energy) in isolation.

Because $\mathbb{W}$ and $\mathbb{E}$ are convex, it follows that the function f_λ is convex for every $\lambda \in [0,1]$ and thus has a unique minimizer $\tau^*(\lambda)$. Using the derivatives in (3) and (4), one can easily show that the optimal $\tau_i^*(\lambda)$ satisfies

$$\tau_i^*(\lambda) = \sqrt{\frac{c_i(\lambda + (1-\lambda)\mathcal{P}_i^c)\left(2 + \sum\limits_{j=i+1}^{L} \mu_j \tau_j^*\right)}{\mu_i\left(\lambda + (1-\lambda)\mathcal{P}^a + \sum\limits_{j=1}^{i-1} (\lambda + (1-\lambda)\mathcal{P}_j^c)\frac{c_j}{\tau_j^*}\right)}}, \tag{6}$$

where each $\tau_j^* = \tau_j^*(\lambda)$ depends on λ. For example, in the single-level case, we have that

$$\tau^*(\lambda) = \tau^{\mathbb{W}} \sqrt{\frac{\lambda + (1-\lambda)\mathcal{P}^c}{\lambda + (1-\lambda)\mathcal{P}^a}}. \tag{7}$$

Equation (7) reiterates that tradeoffs are present in the single-level case whenever $\mathcal{P}^c \neq \mathcal{P}^a$. When $L > 1$, the situation is more complex; in the next section we investigate the behavior for specific values of the multilevel parameters.

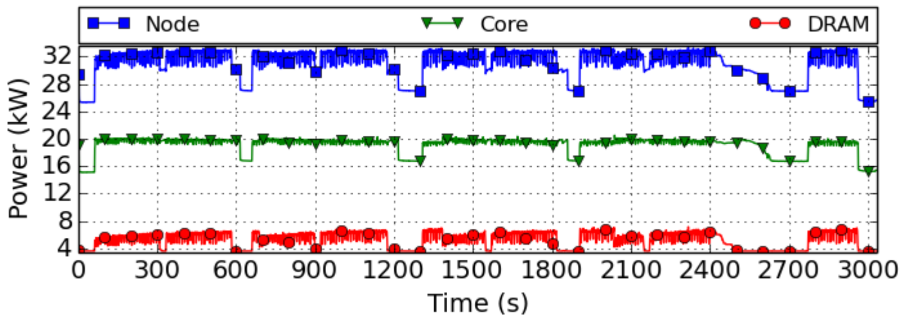
5 Experiments

Our evaluation was performed on MIRA, a 10-petaflops IBM Blue Gene/Q (BG/Q) system and Vesta, a developmental platform for Mira, at the Argonne Leadership Computing Facility. Mira has 48 racks with a total of 49,152 nodes, each one with 16 cores of 1.6 GHz PowerPC A2 and 16 GB of DDR3 memory. The compute nodes run on CNK, a proprietary, lightweight kernel that minimizes OS noise. A proprietary 5-D torus network connects all the compute nodes and the PFS. The machine is water-cooled for thermal efficiency. Vesta's architecture is the same as Mira's but with 2,048 nodes. For measuring power on BG/Q, we use MonEQ, a low overhead power-profiling library [16] that samples power readings at a frequency of 560 ms. The power measurements include the overall node consumption as well as core, DRAM and network. Further details on the power profiling used can be found in [16]. Because of control system limitations, MonEQ can collect power data only at the node-card level which includes 32 compute nodes. In addition, MonEQ only measures power consumption on the compute nodes, and does not provide data for the I/O power consumption. We revisit this issue in Sec. 5.2.

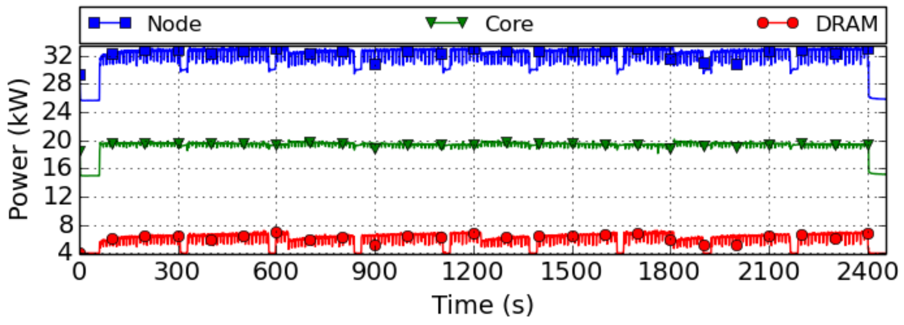
5.1 FTI on BG/Q

Our first set of experiments was done with LAMMPS, a production-level molecular dynamics application [14]. First, we measured the performance of LAMMPS on Mira to confirm that our setup was correct. We next ported LAMMPS to perform checkpoints with FTI and confirmed that the performance overhead imposed by FTI was low. We then added the MonEQ library to our setup and ran several tests to verify that the power measurements were being correctly logged. With this configuration, we ran a Lennard-Jones simulation of 1.3 billion atoms using 512 nodes and launching 64 MPI processes per node (32,678 ranks in total). Molecular dynamics applications such as LAMMPS are known to have a low memory footprint. Each rank used 16.2 MB of memory and checkpointed 2.9 MB of data. Thus, the checkpoint size per node is about 187 MB, and the total checkpoint size for the whole execution is roughly 93 GB. The checkpoint intervals for levels 1, 2, 3, and 4 were set to 4, 8, 16, and 32 minutes, respectively, producing the checkpoint order $\{1, 2, 1, 3, 1, 2, 1, 4\}$. This first experiment was done without using dedicated processes for fault tolerance. Thus, every process participated in the application, and the execution was blocked during the checkpoints.

Figure 2a shows the power consumption of LAMMPS checkpointing with FTI in a synchronous fashion. During normal execution, LAMMPS consumes about 32 kW on 512 nodes (32,678 processes). We introduce one minute idle phase (i.e. sleep) before the application starts, to measure the idle power consumption of the nodes. We observe that the idle phase consumes roughly 25 kW. The periodic drop (every four minutes) in power consumption is due to checkpointing. We can identify



(a) Synchronous multilevel checkpointing

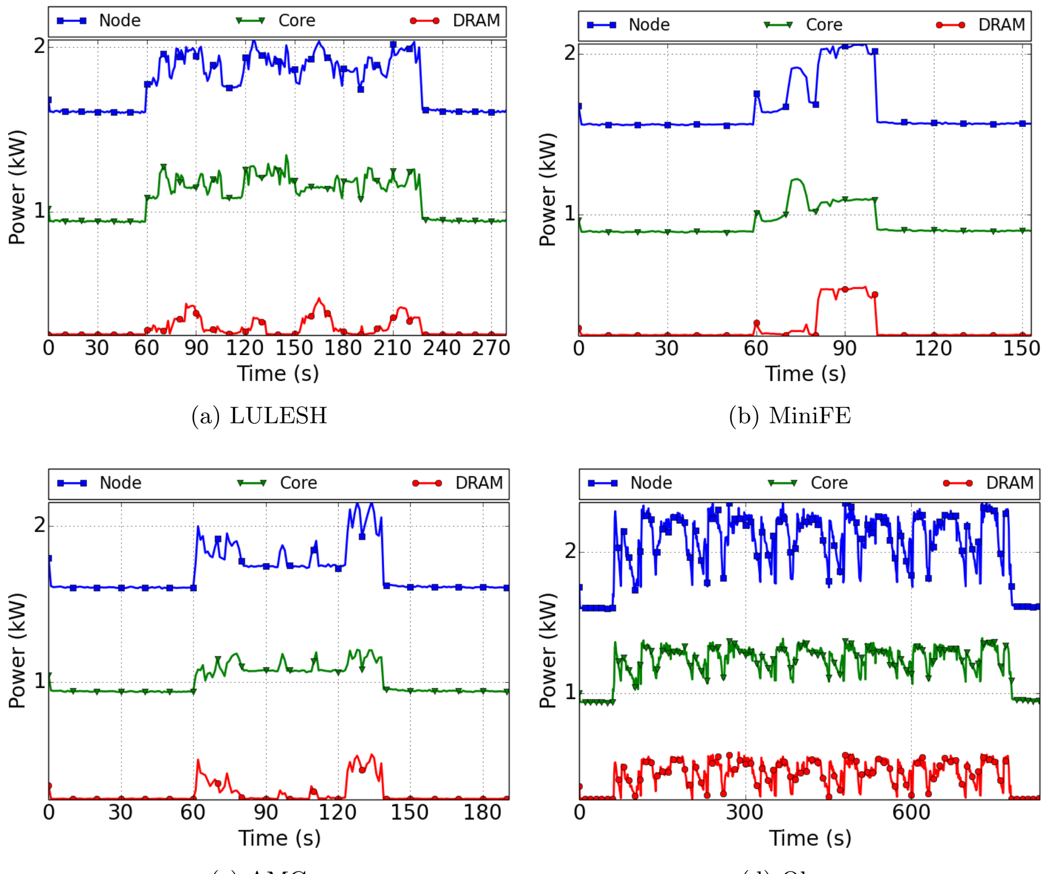


(b) Asynchronous multilevel checkpointing

Fig. 2: Power profile of LAMMPS running a 1.3 billion-atom Lennard-Jones simulation and checkpointing with FTI on BG/Q. Execution on 512 nodes running 64 MPI ranks per node (32,678 proc.). The power consumption of node is a sum of all power consumptions of the components.

the checkpoint levels by measuring the time that nodes spend in different power consumption regimes. Short drops in DRAM corresponds to the checkpoint level 1. Checkpoints level 2 and 3 expose two parts of checkpoint: DRAM power drop when the checkpoint data is being copied locally and core power drop where the checkpoint is either being transferred to a partner copy or a encoded with Reed-Solomon encoding, for level 2 and 3 respectively. Finally, PFS-based checkpoint is visible as a long drop in power consumption due to the time that it takes to transfer the checkpoint data to the PFS via I/O nodes and erase the previous local checkpoints. Since MonEQ provides only the power consumption of the participating compute nodes, the experiments do not allow us to accurately quantify the energy usage for level 4 PFS-based checkpointing. The power consumption of all other checkpoint levels vary between 27 kW and 30 kW. We note that although they have relatively similar power costs, their run times vary significantly. We verified that all node cards (set of 32 nodes) consume the same power, roughly 1.6 kW, 1.8 kW, and 2 kW during idle time, checkpointing, and execution, respectively.

The next experiment aims to test the asynchronous feature of FTI to speed the checkpoints. LAMMPS is a good candidate for this type of optimization because it does not require a particular number of MPI ranks. Therefore, one can easily dedicate one MPI process per node (out of 64) for fault tolerance. The same checkpoint frequencies are kept, producing the same checkpointing



(a) LULESH (b) MiniFE

(c) AMG (d) Qbox

Fig. 3: Power profile of CORAL benchmark applications on a BG/Q node board of 32 nodes. Each application is run with a configuration of 16 MPI ranks per node (512 processes).

pattern as in the previous configuration. The results in Fig. 2b illustrate that the drops in power consumption are much shorter because the application is blocked only during the local copy; the rest of the work is done in the background by the dedicated processes (one per node), and does not involve a significant extra power cost. As a result of this optimization, the application runs about 20% faster than in the previous configuration.

We also study the power profile of four mini-applications from the CORAL benchmark suite developed for the procurement of pre-exascale systems [1]. Qbox is a first-principles molecular dynamics code used to compute the properties of materials from the underlying physics equations. AMG is a parallel algebraic multigrid solver for linear systems arising from problems on unstructured grids. LULESH performs hydrodynamics stencil calculations, and miniFE is a finite-element code.

We ran the four applications on a single-node board of 32 nodes of Vesta with 512 MPI ranks (16 MPI ranks per node). Figure 3 shows the power profile of the fault-free computations, $\mathcal{P}^a$, on a node card. Except for Qbox, on average, the observed $\mathcal{P}^a$ values are similar to those of LAMMPS; for Qbox, $\mathcal{P}^a$ reaches up to 2.2 kW.

5.2 Tradeoff Analysis

We now revisit the energy-performance models from Sec. 3 and use the power consumption and checkpointing cost observed on BG/Q and presented in Sec. 5.1. In particular, we examine several system-level parameters that can affect the energy-performance tradeoffs.

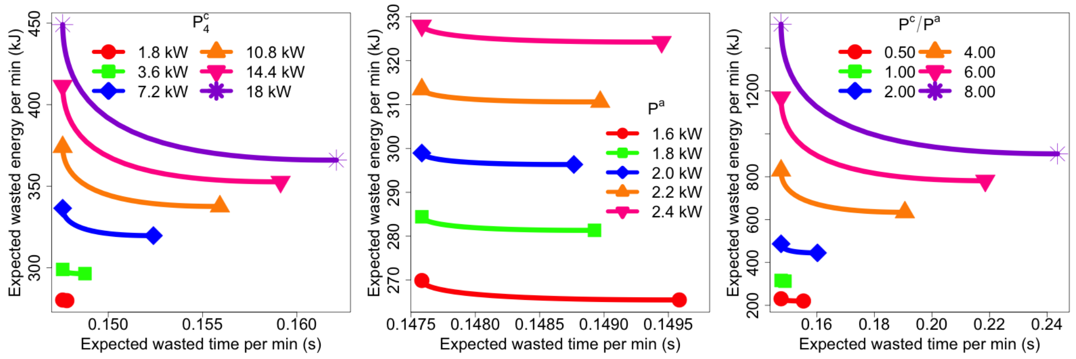
For this analysis, we consider values at a node-board level and applications with similar checkpoint sizes as the ones observed for LAMMPS. As a default, we use the configuration $c = [10, 30, 50, 150]$ s; $\mu = [1, 0.5, 0.25, 0.05]/36000$ Hz; $\mathcal{P}^a = 2$ kW; $\mathcal{P}^c_1 = \mathcal{P}^c_2 = \mathcal{P}^c_3 = 1.8$ kW; and, since there is no power monitoring infrastructure to measure the I/O power involved in level 4 checkpointing, we take $\mathcal{P}^c_4 = 2 \times \mathcal{P}^c_3$. We note that the default failure rates are those commonly used for petascale HPC systems [4,13,7]. Note that, given a fixed checkpoint size, the wasted time and energy consumption per unit time during checkpointing will be the same for different applications, because FTI performs the same amount of work (e.g., transfer) independently of the content of the checkpoint data. In what follows we report the *expected waste in time and energy per minute.*

With all other values held fixed, we first vary the number of levels considered for checkpointing (and at which failures can occur). Table 2 illustrates that the optimal checkpoint intervals depend on what is happening at *all* other levels. Despite the overall time between any failure (the final column) decreasing, the checkpoint intervals at a level actually increase because of the increases in the number of levels. Furthermore, differences in wasted time and energy between the two single-objective solutions $\tau^{\mathbb{W}}$ and $\tau^{\mathbb{E}}$ increase as the number of levels grows. Nevertheless, for a given number of levels, these differences are small.

Table 2: Optimal multilevel checkpoint intervals (s) for schemes with 1, 2, 3, and 4 levels.

Level	1	2	3	4	$\mathbb{W}(\tau), \mathbb{E}(\tau)$	$(\sum_{j=1}^{L} \mu_j)^{-1}$
$\tau^{\mathbb{W}}$	848.5	n/a	n/a	n/a	(1.41, 2.69)	36000 (s)
$\tau^{\mathbb{E}}$	805.0	n/a	n/a	n/a	(1.42, 2.68)	
$\tau^{\mathbb{W}}$	854.6	2066	n/a	n/a	(3.16, 6.00)	108000 (s)
$\tau^{\mathbb{E}}$	810.5	1961	n/a	n/a	(3.16, 5.99)	
$\tau^{\mathbb{W}}$	860.1	2080	3746	n/a	(4.76, 9.04)	252000 (s)
$\tau^{\mathbb{E}}$	815.4	1973	3556	n/a	(4.76, 9.02)	
$\tau^{\mathbb{W}}$	864.3	2090	3765	14417	(6.01, 12.53)	972000 (s)
$\tau^{\mathbb{E}}$	820.8	1986	3580	19362	(6.07, 12.37)	

Since we cannot measure I/O-intensive level 4 power consumption, we analyze the tradeoffs under various $\mathcal{P}^c_4$ scenarios. We consider $\mathcal{P}^c_4 = \alpha\mathcal{P}^c_3$, where $\alpha \in [1, 2, 4, 6, 8, 10]$ and the default $\mathcal{P}^c_3$. Figure 4a shows that increasing $\mathcal{P}^c_4$ relative to other levels has a significant impact on the observed tradeoff between $\mathbb{W}$ and $\mathbb{E}$. In particular, richer tradeoff is observed for $\alpha = 10$ (18kW). We also analyze the impact of different $\mathcal{P}^a$ values on time and energy. Figure 4b shows that varying $\mathcal{P}^a$ increases energy, but the tradeoffs are insignificant.



(a) Level 4 power consumption $\mathcal{P}_4^c$ (b) Computation power $\mathcal{P}^a$ (c) Power ratio $\frac{\mathcal{P}^c}{\mathcal{P}^a}$

Fig. 4: Time-energy Pareto fronts for multilevel schemes as different parameters are varied. The two end points represent only the boundary of the solution set: all values between them correspond to non-dominated points.

The projected low power consumption and high failure rate for next-generation systems can have a significant impact on energy-performance tradeoffs. Here, we characterize power consumption by the ratio $\frac{\mathcal{P}^c}{\mathcal{P}^a}$. We set the $\mathcal{P}^a$ and $\mathcal{P}_1^c$ values to obtain $\frac{\mathcal{P}^c}{\mathcal{P}^a} \in \{0.5, 1.0, 2.0, 4.0, 6.0, 8.0\}$, with all other default values unchanged. Recall that $\mathcal{P}_1^c = \mathcal{P}_2^c = \mathcal{P}_3^c$ and $\mathcal{P}_4^c = 2\mathcal{P}_3^c$. In Fig. 4c, we see that $\frac{\mathcal{P}^c}{\mathcal{P}^a}$ has a significant impact on the tradeoffs between $\mathbb{W}$ and $\mathbb{E}$, with these tradeoffs increasing as ρ increases. This suggests that power for computation should be significantly less than that for checkpointing in order for richer tradeoffs to exist. This situation could happen for several reasons. For instance, applications could be significantly more aware of data locality than what multilevel checkpointing techniques could achieve, because resilience can be achieved only through data dispersion across space, which requires communication. We also analyzed the tradeoffs by increasing μ values, but we did not observe significant tradeoffs.

6 Related Work

A rich body of literature exists for computing an optimal checkpoint period with respect to run time for various checkpoint protocols [5,6,7]. However, energy models and analysis of tradeoffs in current and future HPC systems are still in their infancy. Diouri et al. [9] modeled and evaluated the energy consumption of checkpointing, task coordination, and message logging components of a fault tolerance protocol. They showed that neither of these tasks significantly increases the power draw of a node and that minimizing execution time will minimize energy consumption. Later, they developed the ECOFIT framework [10] using component power models, and studied energy consumption of an application using coordinated, uncoordinated, and hierarchical protocols. Meneses et al. [12] developed models for expected run time and energy consumption for global recovery, message logging, and parallel recovery protocols. They observed tradeoffs in message logging due to significant run time overhead but faster recovery. They applied these models in an exascale scenario and showed that parallel recovery is more effective than a checkpointing protocol since parallel recovery reduces the rework time. A limitation of the model is that it considers failures at a single node level. Moreover, the RAPL API used to report the power consumption measures only the energy consumption at a processor-level and does not cover the I/O, or the communication [11]. Aupy et al. [2] developed performance and energy models and applied them to analyze the minimizers of each objective in isolation. Under an expensive I/O scenario with a low idle power of 10 mW/node, the authors showed different tradeoffs. However, the proposed models do not take into account multilevel checkpointing and are not used to assess the tradeoffs more generally. The authors considered the power consumption values from elsewhere [15]: the checkpointing power consumption was set to 10 times the computer power, a primary reason for the significant differences in time and energy.

7 Conclusions

We developed analytical models of performance and energy for multilevel checkpoint schemes. We went beyond minimizing the two corresponding objectives in isolation and examined them simultaneously. We proved that both models—and hence their shared Pareto front—are convex and used this result to analyze the performance-energy tradeoffs for the FTI multilevel checkpoint library on BG/Q. We ran a well-known molecular dynamics application (LAMMPS) over 32,000 ranks as well as other CORAL applications and performed detailed power measurements on them. The empirical results and analysis showed that the relative energy overhead due to the adoption of FTI is small on the studied applications and thus the tradeoffs between the run time and the energy consumption is not significant. This is due to the fact that the difference between power consumption during computation and multilevel checkpointing is minor. The exploratory analysis showed the existence of richer tradeoffs where the power consumption of checkpointing is significantly higher than that of the computation such a situation can be observed when using I/O-intensive and/or data-intensive checkpoint strategies.

Our future work includes analyzing power profile of different fault tolerance protocols such as full/partial replication and message logging. We plan to develop performance and energy models for replication and checkpointing in order to assess the viability of both protocols with respect to the power cap of future exascale platforms.

Appendix

We first formalize our assumption on the checkpoint intervals of interest.

Assumption (A1). *We consider checkpoint intervals $\tau \in \mathbb{R}^L_+$ that satisfy (for $i = 1, \ldots, L$): (i) $\tau_i > 0$; (ii) $\tau_j > \tau_i/2$ whenever $j > i$; and (iii) $\tau_i < 4/\sum_{j=1}^{i-1} \mu_j$.*

The second condition says that the checkpoint at level j cannot be that frequent relative to checkpoints at lower levels. The third condition says that the time between checkpoints needs to be sufficiently smaller than the expected time between *any* failure at a lower level.

Theorem 1. *If (A1) holds, then the time $\mathbb{W}$ and energy $\mathbb{E}$ are convex functions of $\tau \in \mathbb{R}^L$.*

Proof. Following (3) and (4), the second-order derivatives of $\mathbb{W}$ are given by

$$\frac{\partial^2 \mathbb{W}}{\partial \tau_i^2} = \frac{c_i}{\tau_i^3}\left(2 + \sum_{j=i+1}^{L} \mu_j \tau_j\right)$$

$$\frac{\partial^2 \mathbb{W}}{\partial \tau_i \partial \tau_j} = -\frac{c_i \mu_j}{2\tau_i^2}, \qquad j \neq i.$$

We then have

$$\begin{aligned} &\tfrac{\partial^2 \mathbb{W}}{\partial \tau_i^2} - \textstyle\sum_{j \neq i} \left| \tfrac{\partial^2 \mathbb{W}}{\partial \tau_i \partial \tau_j} \right| \\ &= \tfrac{c_i}{\tau_i^2} \left(\sum_{j=i+1}^{L} \mu_j \left(\tfrac{\tau_j}{\tau_i} - \tfrac{1}{2} \right) + \tfrac{2}{\tau_i} - \sum_{j=1}^{i-1} \tfrac{\mu_j}{2} \right), \end{aligned} \tag{8}$$

which is positive by (A1). Equation (8) being positive for all i means that the Hessian $\nabla^2_{\tau\tau}\mathbb{W}(\tau)$ is diagonally dominant, and thus $\mathbb{W}$ is a convex function of τ over the domain prescribed by (A1).

The convexity of $\mathbb{E}$ follows by a similar argument, with the derivatives of $\mathbb{E}$ given by

$$\begin{aligned} \frac{\partial^2 \mathbb{E}}{\partial \tau_i^2} &= \frac{\mathcal{P}_i^c c_i}{\tau_i^3} \left(2 + \sum_{j=i+1}^{L} \mu_j \tau_j \right) \\ \frac{\partial^2 \mathbb{E}}{\partial \tau_i \partial \tau_j} &= -\frac{\mathcal{P}_i^c c_i \mu_j}{2\tau_i^2}, \qquad j \neq i. \end{aligned}$$

As a result, there are unique minimizers $\tau^{\mathbb{W}}$ and $\tau^{\mathbb{E}}$ over the domain prescribed by (A1).

References

1. CORAL. `http://asc.llnl.gov/CORAL-benchmarks/`.
2. G. Aupy, A. Benoit, T. Hérault, Y. Robert, and J. Dongarra. Optimal checkpointing period: Time vs. Energy. In *Proc. 4th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS13)*, 2013.
3. P. Balaprakash, A. Tiwari, and S.M. Wild. Multi-objective optimization of HPC kernels for performance, power, and energy. In *Proc. 4th Int. Wksp. Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS13)*, 2013.
4. L. Bautista-Gomez, S. Tsuboi, D. Komatitsch, F. Cappello, N. Maruyama, and S. Matsuoka. FTI: high performance fault tolerance interface for hybrid systems. In *Proc. 2011 Int. Conf. High Performance Computing, Networking, Storage and Analysis (SC11)*, pages 32:1–32:32. ACM, 2011.
5. M.-S. Bouguerra, D. Trystram, and F. Wagner. Complexity analysis of checkpoint scheduling with variable costs. *IEEE Trans. Computers*, 62(6):1269–1275, 2013.
6. J.T. Daly. A higher order estimate of the optimum checkpoint interval for restart dumps. *Future Generation Computer Systems*, 22(3):303–312, 2006.
7. S. Di, M.S. Bouguerra, L. Bautista-Gomez, and F. Cappello. Optimization of multi-level checkpoint model for large-scale HPC applications. *Int. Parallel and Distributed Processing Symp.*, 2014. To appear.
8. M. Ehrgott. *Multicriteria Optimization.* Springer-Verlag, 2nd edition, 2005.
9. M. el Mehdi Diouri, O. Gluck, L. Lefèvre, and F. Cappello. Energy considerations in checkpointing and fault tolerance protocols. In *2012 IEEE/IFIP 42nd Int. Conf. Dependable Systems and Networks Workshops (DSN-W)*, pages 1–6, 2012.
10. M. el Mehdi Diouri, O. Gluck, L. Lefevre, and F. Cappello. ECOFIT: A framework to estimate energy consumption of fault tolerance protocols for HPC applications. In *13th IEEE/ACM Int. Symp. Cluster, Cloud and Grid Computing (CCGrid13)*, pages 522–529, 2013.
11. D. Hackenberg, T. Ilsche, R. Schone, D. Molka, M. Schmidt, and W.E. Nagel. Power measurement techniques on standard compute nodes: A quantitative comparison. In *2013 IEEE Int. Symp. Performance Analysis of Systems and Software (ISPASS13)*, pages 194–204, 2013.
12. E. Meneses, O. Sarood, and L.V. Kalé. Energy profile of rollback-recovery strategies in high performance computing. *Parallel Computing*, 2014.

13. A. Moody, G. Bronevetsky, K. Mohror, and B.R. de Supinski. Design, modeling, and evaluation of a scalable multi-level checkpointing system. In *Proc. 2010 Int. Conf. High Performance Computing, Networking, Storage and Analysis (SC10)*, pages 1–11, 2010.
14. S. Plimpton, P. Crozier, and A. Thompson. LAMMPS: large-scale atomic/molecular massively parallel simulator. *Sandia National Laboratories*, 2007.
15. J. Shalf, S. Dosanjh, and J. Morrison. Exascale computing technology challenges. In *High Performance Computing for Computational Science (VECPAR10)*, pages 1–25. Springer, 2011.
16. S. Wallace, V. Vishwanath, S. Coghlan, J. Tramm, Z. Lan, and M.E. Papka. Application power profiling on IBM Blue Gene/Q. In *2013 IEEE Int. Conf. Cluster Computing (CLUSTER13)*, pages 1–8, 2013.

Acknowledgment

This work was supported by the SciDAC and X-Stack activities within the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research program under contract number DE-AC02-06CH11357.