Modeling and Analysis of Checkpoint I/O Operations

Sarala Arunagiri[1], John T. Daly[2], and Patricia J. Teller[1]

[1] The University of Texas at El Paso
{sarunagiri,pteller}@utep.edu
[2] The Center for Exceptional Computing
john.t.daly@ugov.gov

Abstract. The large scale of current and next-generation massively parallel processing (MPP) systems presents significant challenges related to fault tolerance. For applications that perform periodic checkpointing, the choice of the checkpoint interval, the period between checkpoints, can have a significant impact on the execution time of the application and the number of checkpoint I/O operations performed by the application. These two metrics determine the frequency of checkpoint I/O operations performed by the application and, thereby, the contribution of the checkpoint operations to the demand made by the application on the I/O bandwidth of the computing system. Finding the optimal checkpoint interval that minimizes the wall clock execution time has been a subject of research over the last decade. In this paper, we present a simple, elegant, and accurate analytical model of a complementary performance metric - the aggregate number of checkpoint I/O operations. We present an analytical model of the expected number of checkpoint I/O operations and simulation studies that validate the analytical model. Insights provided by a mathematical analysis of this model, combined with existing models for wall clock execution time, facilitate application programmers in making a well informed choice of checkpoint interval that represents an appropriate trade off between execution time and number of checkpoint I/O operations. We illustrate the existence of such propitious checkpoint intervals using parameters of four MPP systems, SNL's Red Storm, ORNL's Jaguar, LLNL's Blue Gene/L (BG/L), and a theoretical Petaflop system.

1 Introduction

As Massively Parallel Processing (MPP) systems scale to tens of thousands of nodes, reliability and availability become increasingly critical. Scientists have predicted that three of the most difficult and growing problems in future high-performance computing (HPC) installations will be - avoiding, coping with, and recovering from failures. With the increase in the scale of computing systems, element failures become frequent, making it increasingly difficult for long running applications to make forward progress in the absence of fault tolerance mechanisms [5].

K. Al-Begain, D. Fiems, and G. Horváth (Eds.): ASMTA 2009, LNCS 5513, pp. 386–400, 2009.

Checkpoint restart is a common technique to provide fault tolerance for applications running on MPP systems. Checkpointing can be either application-directed or system-directed. An application's *checkpoint data* is data that represents a consistent state of the application that can be saved and then, in the event of a failure, restored and used to resume execution at the saved state. A checkpoint is generally stored to persistent media (e.g., a file system). *Checkpoint latency* is the amount of time required to write checkpoint data to persistent storage and a *checkpoint interval* is the application execution time between two consecutive checkpoint operations. *Checkpoint overhead* is the increase in the execution time of an application due to checkpointing.

In a disk-based periodic checkpointing system, selecting an appropriate checkpoint interval is important especially since the storage system is physically separated from the processors used for execution of the scientific application. If the checkpoint interval is too small, the overhead created by network and storage transfers of a large number of checkpoints can have a significant impact on performance, especially when other checkpointing applications share the network and storage resources. Conversely, if the checkpoint interval is too large, the amount of work lost in the event of a failure can significantly increase the time to solution. Deciding upon the optimal checkpoint interval is the well known *optimal checkpoint interval* problem. Most solutions attempt to minimize total execution time (i.e., the application time plus the checkpoint overhead) [18] [3] [15]. In this paper we focus on another performance metric, the number of checkpoint I/O operations performed during an application run.

1.1 Motivation

The rate of growth of disk-drive performance, both in terms of I/O operations per second and sustained bandwidth, is smaller than the rate of growth of the performance of other components of computing systems [15]. Therefore, in order to attain good overall performance of computing systems, it is important to design applications to use the I/O resources efficiently, bearing in mind the limitations posed by them. There are several scientific papers that elaborate on this problem, an example of a recent paper is [15].

I/O operations performed by an application can be segregated into productive I/O and defensive I/O. Productive I/O is the component that is performed for actual science such as visualization dumps, whereas defensive I/O is the component used by fault tolerance mechanisms such as checkpoint/restart. In large applications, it has been observed that about 75% of the overall I/O is defensive I/O [1]. As indicated by [5] and other scientific literature, the demand made by checkpoint (defensive) I/O is a primary driver of the sustainable bandwidth of high performance filesystems. Hence, it is critical to manage the amount and rate of defensive I/O performed by an application. In a recent paper [15] extensive results are presented showing that as the memory capacity of the system increases so does the I/O bandwidth required to perform checkpoint operations at the optimal checkpoint interval that attains the minimum execution time. An example presented in the paper is for a system with an MTBF of 8 hours and

memory capacity of 75TB. When the checkpoint overhead is constrained to be less than or equal to 20% of application solution time, there is no solution for the optimal checkpoint interval unless the I/O bandwidth is larger than 29GB/sec. They define *utility in a cycle* as the ratio of time spent doing useful calculations to the overall time spent in a cycle and show that the I/O bandwidth required to achieve a utility of 90% is higher than what is available for present systems. Thus, while performing checkpoints at the optimal checkpoint interval that minimizes execution time, if we either restrict the checkpoint overhead to less than or equal to 20% of solution time or expect a utility greater than or equal to 90%, the I/O bandwidth required is often larger than what is available at present.

Our efforts are focused towards enhancing an understanding of the variation of the volume of generated defensive I/O, as a function of the checkpoint interval. The contributions of this paper are:

- In Section 3, we present a simple and elegant analytical model of the aggregate number of checkpoint I/O operations and a mathematical analysis of its properties that have a bearing on system performance.
- In Section 4, we present results of Monte Carlo simulations that were performed to validate the analytical model. The results show that
  - The model is accurate by demonstrating that it has a small relative error.
  - The idealization used in our analytical modeling is reasonable and it does not introduce large errors.
- In Section 5 we discuss the performance implications inferred from the mathematical analysis of Section 3.
- In Section 6, based on Poisson Execution Time Model, described next, and the modeling studies presented in this paper, we show the existence of propitious checkpoint intervals using parameters of four MPP systems, Red Storm, Jaguar, BlueGene/L, and the Petaflop machine.

Finally, in Sections 7 and 8 we present related work and future work, respectively.

2 The Poisson Execution Time Model (PETM)

The work presented in this paper is based on and complementary to the following execution time model formulated by John Daly. The total wall clock time to complete the execution of an application, the optimal checkpoint interval, and an approximate optimal checkpoint interval are given by:

$$T = Me^{R/M}\left(e^{(\tau+\delta)/M} - 1\right)\frac{T_s}{\tau} \quad \text{for } \delta << T_s$$
$$\tau_{\text{opt}} = M\left(1 + ProductLog\left(-e^{-\frac{\delta+M}{M}}\right)\right)$$

The approximation to τ_{opt}, τ_{appx}, is given by

$$\tau_{\text{appx}} = \sqrt{2\delta M}\left[1 + \frac{1}{3}\left(\frac{\delta}{2M}\right)^{\frac{1}{2}} + \frac{1}{9}\left(\frac{\delta}{2M}\right)\right] - \delta \quad \text{for } \delta < 2M$$
$$= \text{M} \quad \text{for } \delta \geq 2M$$

where

T_s = application solution time,

τ = checkpoint interval,
δ = checkpoint latency,
M = mean time between interruptions (MTTI) of the application, and
R = restart time.

In this paper, for the sake of convenience, we refer to the execution time model and the model of the optimal checkpoint interval presented above as the Poisson Execution Time Model (PETM) and the ProductLog Optimal Checkpoint Interval Model w.r.t Execution time (POCIME), respectively. Note that in the original literature [3], which presents these models, the terms PETM and POICME are not used to refer to the models. We introduce these terms with permission from the author of that literature.

3 Modeling the Number of Checkpoint I/O Operations: ProductLog Optimal Checkpoint Interval Model w.r.t I/O(POCIMI)

The set of I/O operations performed by a checkpoint/restart mechanism is comprised of reads and writes. In a periodic checkpointing system we know that checkpoint writes are performed periodically at every checkpoint interval and, therefore, the number of checkpoint write operations is given by the solution time of the application divided by the checkpoint interval.

$$\text{Expected number of checkpoint writes} = T_{\text{s}}/\tau$$

When a failure occurs in a periodic checkpointing system, the last checkpoint data that was successfully written needs to be read to restart the application. Therefore, the number of checkpoint read operations is given by the expected number of failures.

$$\text{Expected number of checkpoint reads} =$$
$$\frac{\text{Expected execution time}}{M} = \frac{T_{\text{s}} e^{R/M}(e^{\frac{\delta+\tau}{M}}-1)}{\tau}.$$

Expected number of aggregate checkpoint I/O operations,

$$N_{\text{I/O}} = \frac{T_{\text{s}}}{\tau}\left[1 + e^{R/M}\left(e^{\frac{\delta+\tau}{M}} - 1\right)\right] \tag{1}$$

For values of parameters MTTI = 24 hours, checkpoint latency = 5 minutes, restart time = 10 minutes, and solution time = 500 hours, using the expression for the number of checkpoint I/O operations from POCIMI and the expression for execution time from PETM, we obtain the plot shown in Fig. 1. From modeling studies in [3], we know that the execution time is a convex function of the checkpoint interval and it has a single minimum at $\tau_{\text{opt}} = 117$ minutes. From Fig. 1, it appears like $N_{\text{I/O}}$, the aggregate number of checkpoint I/O operations,

also is a convex function of the checkpoint interval, with a minimum value in the range $0 \leq \tau \leq M$. In this case, the minimum is 1,436 minutes, which is larger than the value of τ_{opt}, 117 minutes. It is important to know if these properties are invariant with respect to parameter values. In the rest of this section we present mathematical proof that the properties observed are, indeed, true for any given set of parameters.

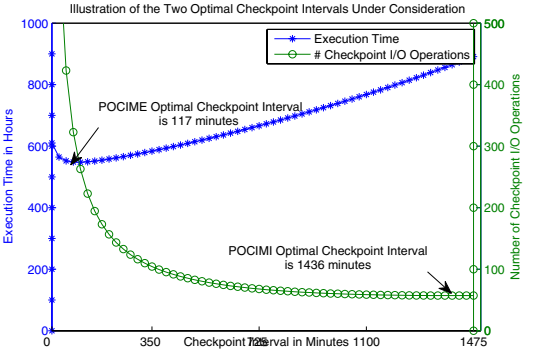

Fig. 1. Plots of Execution Time and the Number of Checkpoint I/O Operations as functions of checkpoint Interval. The parameters are MTTI, M = 24 hours, Checkpoint latency, δ = 5 minutes, and Restart time, R = 10 minutes, and Solution time, T_{s}=500hrs.

Theorem 1. *The function $N_{\text{I/O}}$ has a single minimum in the range $0 \leq \tau \leq M$; let us denote it by $\tau_{\text{I/O}}$. $N_{\text{I/O}}$ does not have any other stationary points in this range. $\tau_{\text{I/O}}$ is given by*

$$\tau_{\text{I/O}} = M\left(1 + ProductLog\left(-e^{-\frac{\delta+M}{M}} + e^{-\frac{R+\delta+M}{M}}\right)\right) \tag{2}$$

Proof. $N_{\text{I/O}}$ is given by Equation 1. We look for stationary points of $N_{\text{I/O}}$ w.r.t. τ , i.e., values of τ at which the first derivative of $N_{\text{I/O}}$ w.r.t τ is zero.

$$\frac{dN_{\text{I/O}}}{d\tau} = \frac{T_{\text{s}}}{\tau^2}\left[\frac{\tau}{M}e^{\frac{R}{M}}e^{\frac{\delta+\tau}{M}} - \left(e^{\frac{R}{M}}(e^{\frac{\delta+\tau}{M}} - 1)\right) - 1\right]$$

$$\left(\frac{dN_{\text{I/O}}}{d\tau} = 0\right) \Longrightarrow \left(e^{\frac{R}{M}}e^{\frac{\delta+\tau}{M}}\frac{\tau}{M} - e^{\frac{R}{M}}e^{\frac{\delta+\tau}{M}} + e^{\frac{R}{M}} - 1 = 0\right) \Longrightarrow$$

$$\left(e^{\frac{R}{M}}e^{\frac{\delta+\tau}{M}}\left(\frac{\tau}{M} - 1\right) = 1 - e^{\frac{R}{M}}\right) \Longrightarrow \left(e^{\frac{\delta+\tau}{M}}\left(\frac{\tau}{M} - 1\right) = -\left(1 - e^{-\frac{R}{M}}\right)\right) \Longrightarrow$$

$$\left(e^{\frac{\tau}{M}}\left(\frac{\tau}{M} - 1\right) = -e^{-\frac{\delta}{M}}\left(1 - e^{-\frac{R}{M}}\right)\right) \Longrightarrow \left(e^{\left(\frac{\tau}{M}-1\right)}\left(\frac{\tau}{M} - 1\right) = -e^{-\frac{\delta+M}{M}}\left(1 - e^{-\frac{R}{M}}\right)\right) \Longrightarrow$$

$$\left(\left(\frac{\tau}{M} - 1\right) = ProductLog\left(-e^{-\frac{\delta+M}{M}}\left(1 - e^{-\frac{R}{M}}\right)\right)\right) \Longrightarrow$$

$$\left(\frac{\tau}{M} = 1 + ProductLog\left(-e^{-\frac{\delta+M}{M}}\left(1 - e^{-\frac{R}{M}}\right)\right)\right) \Longrightarrow$$

$$\left(\tau = M\left(1 + ProductLog\left(-e^{-\frac{\delta+M}{M}}\left(1 - e^{-\frac{R}{M}}\right)\right)\right)\right)$$

$$\tau = M\left(1 + ProductLog\left(-e^{-\frac{\delta+M}{M}} + e^{-\frac{R+\delta+M}{M}}\right)\right) \tag{3}$$

There is a unique positive value of τ that satisfies the above equation; let us denote it by $\tau_{\mathrm{I/O}}$. The *ProductLog* term in Equation 3 is negative and its absolute value is less than one. Therefore, $\tau_{\mathrm{I/O}}$ is always less than M. We use the second derivative test in order to determine whether the stationary point $\tau_{I/0}$ is a minimum, maximum, or an inflexion point.

We know that

$$\begin{aligned} N_{\mathrm{I/O}} &= \frac{\text{Expected Execution Time}}{M} + \frac{T_{\mathrm{s}}}{\tau} = \frac{T}{M} + \frac{T_{\mathrm{s}}}{\tau} \\ \frac{dN_{\mathrm{I/O}}}{d\tau} &= \frac{1}{M}\frac{dT}{d\tau} - \frac{T_{\mathrm{s}}}{\tau^2} \\ \frac{d^2N_{\mathrm{I/O}}}{d\tau^2} &= \frac{1}{M}\frac{d^2T}{d\tau^2} + 2\frac{T_{\mathrm{s}}}{\tau^3} \end{aligned} \tag{4}$$

From [3] we know that $\frac{d^2T}{d\tau^2}$ is positive for all values of τ in the range $0 < \tau \leq M$. This makes the right-hand side of Equation 4 and, thus, $\frac{d^2N_{\mathrm{I/O}}}{d\tau^2}$ positive for all τ in the range $0 < \tau \leq M$. Therefore, the stationary point $\tau_{\mathrm{I/O}}$ is a minimum with respect to the number of I/O operations. □

We now investigate the relationship between $\tau_{\mathrm{I/O}}$ and τ_{opt} for any given set of checkpoint parameters.

Theorem 2. *The value of the checkpoint interval that minimizes the number of I/O operations, $\tau_{\mathrm{I/O}}$, is always greater than the value of the checkpoint interval that minimizes the expected execution time, τ_{opt}.*

Proof. Recall the expressions for τ_{opt} and $\tau_{\mathrm{I/O}}$;

$$\begin{aligned} \tau_{\mathrm{opt}} &= M\left(1 + ProductLog\left(-e^{-\frac{\delta+M}{M}}\right)\right) \\ \tau_{\mathrm{I/O}} &= M\left(1 + ProductLog\left(-e^{-\frac{\delta+M}{M}} + e^{-\frac{R+\delta+M}{M}}\right)\right) \end{aligned}$$

Consider arguments to the *ProductLog* function in the above equations for τ_{opt} and $\tau_{\mathrm{I/O}}$. They are both negative and the absolute value of the argument in the equation for τ_{opt} is larger than that of the equation for $\tau_{\mathrm{I/O}}$. Since $ProductLog(-1/e) = -1$ and the *ProductLog* function is monotonically increasing in the range ($-\frac{1}{e}$ to 0).

$$\begin{aligned} &|ProductLog\left(-e^{-\frac{\delta+M}{M}}\right)| > |ProductLog\left(-e^{-\frac{\delta+M}{M}} + e^{-\frac{R+\delta+M}{M}}\right)| \\ &\Longrightarrow \tau_{\mathrm{opt}} < \tau_{\mathrm{I/O}} \end{aligned}$$

□

Thus, as illustrated by Fig. 1, we have established that for checkpoint intervals τ in the range $\tau_{\mathrm{opt}} \leq \tau \leq \tau_{\mathrm{I/O}}$, the number of checkpoint I/O operations decreases with increasing checkpoint intervals.

Corollary 1. *For checkpoint intervals, τ, in the range $\tau_{\text{opt}} \leq \tau \leq \tau_{\text{I/O}}$, the expected value of the frequency of checkpoint I/O operations decreases as the checkpoint interval increases.*

Proof. We know from PETM that for values of checkpoint intervals, τ, in the range $\tau_{\text{opt}} \leq \tau \leq M$, the expected execution time increases as the checkpoint interval increases. Since $\tau_{\text{I/O}} < M$, it follows that the expected execution time increases as the checkpoint interval increases for τ in the range $\tau_{\text{opt}} \leq \tau \leq \tau_{\text{I/O}}$. This information and Theorem 2 together imply that for checkpoint intervals, τ, in the range $\tau_{\text{opt}} \leq \tau \leq \tau_{\text{I/O}}$, the expected value of the frequency of checkpoint I/O operations decreases as the checkpoint interval increases. □

In order to evaluate the accuracy of our analytical model, POCIMI, it is infeasible, in terms of system availability, execution time, and effort, to conduct repeated runs of experiments on the scale of systems that we are studying. Thus, the only feasible alternative for us is a simulation study, which we describe and discuss next.

4 Monte Carlo Simulation to Validate the Analytical Model, POCIMI

The goal of our simulation study was to validate the accuracy of the analytical model for the number of checkpoint I/O operations, POCIMI, by comparing the numbers estimated by POCIMI with those obtained using simulation of the execution of an application on an MPP system.

4.1 Details of Simulation

The simulator was coded using MATLAB to perform a discrete event simulation of the physical process of running an application on a 1,000-node system with each node having an exponential failure distribution. The events in the simulation were confined to those relevant to the process of checkpoint/restart. Failure times were generated using random number generators and, as time progresses, the number of checkpoint reads, number of checkpoint writes, execution time, and number of failures are counted until the application completes execution.

Six sets of simulations were performed, one for each of the following values of checkpoint latency: 5,10,15,20,25, and 30 minutes. The other parameter values were set as follows: solution time of the simulated application: 500 hours, restart time: 10 minutes, and MTTI of the parallel system: 24 hours or 1440 minutes. These parameter values were picked from examples in the published literature. Each set of experiments had five trials. The design variable was the checkpoint interval and the response variable was the number of checkpoint I/O operations. During each trial, the values of the response variable, i.e., the number of checkpoint I/O operations, were counted; each value corresponds to a different value of the design variable, i.e., the checkpoint interval. The range of interest for values of the checkpoint interval was 0 to 1440. We split this range into

three subintervals, low values, medium values, and high values, and picked six data points within each subinterval. Accordingly, the design points of our simulation study were the following 18 values of checkpoint intervals: {50,75,...,175, 650,675,...,775,1350,1375,...,1475}. For each trial, and at each chosen checkpoint interval, we simulated 100 runs of the application and recorded the number of checkpoint I/O operations, in addition to other data, such as execution time and number of failures. For each trial, we calculated the average values of the metrics of interest as an arithmetic mean over the 100 runs of the trial. For the plots presented in Fig. 2, we arbitrarily picked data from one trial, i.e., Trial 3, which has a checkpoint latency of 5 minutes. The decision to depict data from only one trial was made for the sake of clarity – the lines representing the simulated mean values of all trials were almost overlapping and cluttering the figure. Subplot(a) of Fig. 2 is a plot of 99% confidence interval of the mean simulated number of checkpoint I/O operations and the number estimated by the analytical model, POCIMI. For completeness sake, we present in Subplot(b) and Subplot(c)the execution time and inter-arrival times of checkpoint I/O operations, respectively. As can be seen from the plot, at the scale at which the figure is presented, the line representing the analytical model and the one representing the simulated mean almost overlap, and the 99% confidence interval is very small. When we did zoom into the figure, we were able to see that there was, indeed, an error bar showing the confidence interval. While the plots in Fig. 2 present the trends for checkpoint intervals varying over the whole range of interest, Figs. 3 and 4 show the details. Note that unlike Fig. 2, Figs. 3 and 4 use data from all trials belonging to all sets of experiments, i.e., 30 trials in total. Subplot(a) and Subplot(b) of Fig. 3 show bar graphs that represent the range of values of absolute errors and relative errors of the 30 trials. The absolute error and relative error are defined by,

$$\text{Absolute error} = \begin{array}{c}\#\text{ checkpoint I/O operations of POCIMI} \\ \text{-mean simulated }\#\text{ checkpoint I/O operations}\end{array}$$

$$\text{Relative error} = \frac{\text{Absolute error}}{\#\text{ checkpoint I/O operations of POCIMI}} * 100$$

For the simulated number of checkpoint I/O operations for all 30 trials, Subplot(a) of Fig. 4 presents the maximum value of the size of the 99% confidence interval.

4.2 Discussion of Results

- The value of the relative error of the estimates provided by the analytical model for all 30 trials lies within ±6%. This demonstrates the degree of accuracy of the model.
- The size of the 99% confidence interval of the number of simulated checkpoint I/O operations is no more than 8% of its mean value. This implies that the aggregate number of checkpoint I/O operations from the simulation runs has a small variance.

4.3 Addressing the Idealization in Analytical Modeling

Idealization is the process by which scientific models assume facts about the phenomenon being modeled that may not be entirely accurate. Often these assumptions are used to make models easier to understand or solve. One of the caveats of analytical modeling is the idealization used in order to make the model tractable or solvable, or mathematically elegant. With an intent to quantify the contribution of idealization to the error in the predictive accuracy of POCIMI, we performed the following experiment. Corresponding to every simulated run of the application at each chosen design point, i.e., value of checkpoint interval, we ran three versions of the simulation: the base version, the idealized version, and the minimally idealized version.

The details of this experiment are presented in [2]. Subplot(b) of Fig. 4 shows the difference in relative error between the idealized version of the simulation and the minimally idealized version of the simulation. We find that the contribution to the relative error made by the idealization used in our analytical model is within the range ±2%. This demonstrates that the idealization used in POCIMI is not too restrictive and, therefore, does not affect the accuracy of the model too much.

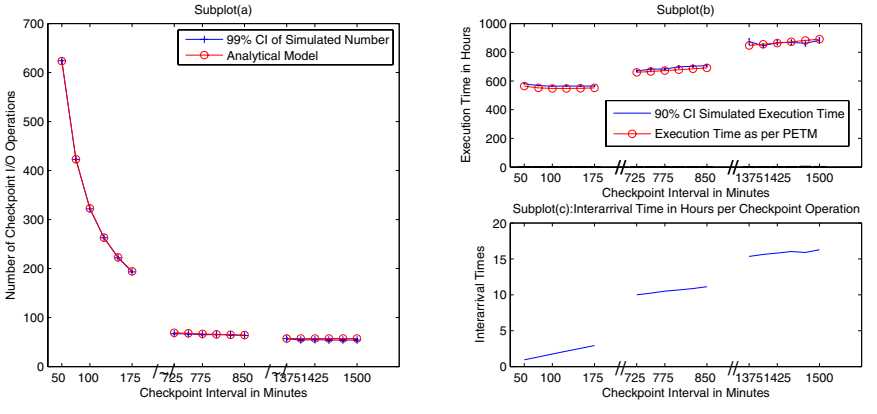

Fig. 2. Subplot(a): Number of checkpoint I/O operations as a function of the checkpoint interval. Subplot(b): Execution time versus checkpoint intervals. Subplot(c): Mean interarrival time, in hours, of checkpoint operations.

5 Performance Implications Inferred by Analyzing POCIMI

1. An insight provided by the model is that while τ_{opt} and $\tau_{\text{I/O}}$ are both functions of δ and M, $\tau_{\text{I/O}}$ is also a function of the restart time, R. $\tau_{\text{I/O}}$ decreases with increasing values of R.
2. Corollary 1 is key to promising avenues in performance improvement. For values of τ in the range $\tau_{\text{opt}} \leq \tau \leq \tau_{\text{I/O}}$, both the expected values of the frequency of checkpoint I/O operations and the number of checkpoint I/O operations decrease with increases in the checkpoint interval.

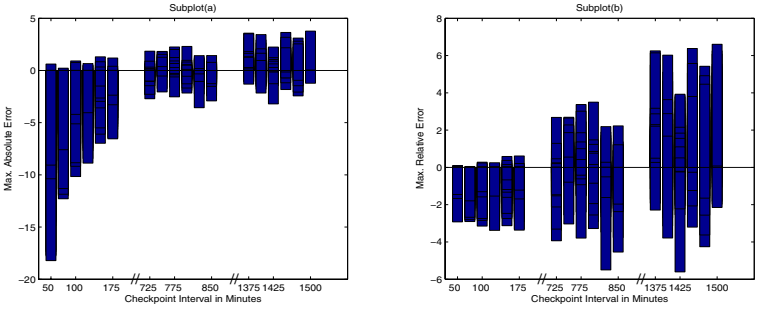

Fig. 3. Subplot(a): Absolute error of POCIMI. Subplot(b): Relative error.

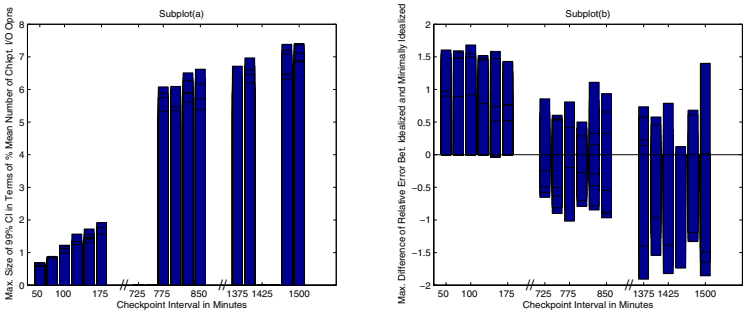

Fig. 4. Subplot(a): Size of the 99% confidence interval(CI) of the simulated number of checkpoint I/O operations. Subplot(b): Difference between relative errors of POCIMI w.r.t. the idealized and minimally idealized versions of the simulation.

3. For time-critical applications for which having a minimum wall clock execution time is important, using τ_{opt} as a checkpoint interval makes perfect sense. However, for all other applications it would be of interest to find out whether it is possible to choose a checkpoint interval that is larger than the τ_{opt} such that the corresponding execution time is marginally larger than the minimum execution time, while the corresponding number of checkpoint I/O operations is drastically smaller than its value at τ_{opt}.
4. If we explore clues from visual inspection of Fig. 1, we observe that for checkpoint intervals greater than and in the vicinity of τ_{opt}, the execution time curve rises slowly, while the curve of the number of checkpoint I/O operations falls steeply. This seems to indicate that, probably, in this region, there are checkpoint intervals such that the corresponding numbers of checkpoint I/O operations are drastically smaller than values corresponding to τ_{opt}, while the execution times are marginally larger than the minimum execution time. Whether or not this observation holds good, in general, for all values of parameters, is not clear. To know this requires a rigorous math-

ematical analysis involving gradients of the execution time function and the number of checkpoint I/O operations function, in the region of interest. This appears to be a non-trivial mathematical exercise and it could be prospective future work. Nonetheless, in the next section, we investigate this idea for specific cases using parameters from four MPP systems.

6 Investigation of Performance Improvement

In this section, using POCIME and POCIMI, we model the performance of four MPP architectures: SNL's Red Storm, ORNL's Jaguar, LLNL's Blue Gene/L (BG/L), and a theoretical Petaflop system. The values of parameters of these systems are presented in Table 1. For all experiments, we consider a representative application with a solution time, $T_s, of 500\ hours$ and a restart time, $R, of 10\ minutes$. For each of the four computing systems and the representative application, assume that the checkpoint interval is larger than τ_{opt} and the corresponding expected execution time is 105% of the minimum execution time, E_{min}, given by PETM, represented as $\tau_{1.05\text{E}_{\text{min}}}$. For the representative application running on the four MPP systems, we investigate the extent to which the number of checkpoint I/O operations corresponding to the checkpoint interval $\tau_{1.05\text{E}_{\text{min}}}$ is reduced, as compared to the number of checkpoint I/O operations at τ_{opt}. For each MPP system, assume that

- the application runs on all nodes of the system,
- the MTTI of each node is 5 years, and
- the application checkpoints half of each processor's memory at each checkpoint.

This set of assumptions is labeled *Standard*. We then consider three other variations of the standard assumptions. The first variation assumes that the application checkpoints 25% of its memory, instead of 50%. The second variation assumes that the MTTI of each node is 2.5 years, instead of 5 years. Finally, the third variation assumes that the application runs on 1/8th of the nodes of each system, instead of all the nodes. In this last case, while computing the checkpoint latency, the partition is considered to have 1/8th of the storage bandwidth available to it. These assumptions cover a few common cases.

For the sixteen cases discussed earlier, the impact of increasing the checkpoint interval, from τ_{opt} to $\tau_{1.05Emin}$, on the number of checkpoint I/O operations is

Table 1. Parameter values for the studied MPPs

Parameter	Red Storm	Blue Gene/L	Jaguar	Petaflop
$n_{max} \times cores$	$12,960 \times 2$	$65,536 \times 2$	$11,590 \times 2$	$50,000 \times 2$
d_{max}	1GB	0.25GB	2.0GB	2.5GB
M_{dev}	5 years	5 years	5 years	5 years
β_s	50GB/s	45GB/s	45GB/s	500GB/s

Table 2. Decrease in the number of checkpoint I/O operations of the representative application at $\tau_{1.05E_{min}}$

MPP System	Conditions	# checkpoint I/O operations for $\tau = \tau_{\text{opt}}$	# checkpoint I/O operations for $\tau = \tau_{1.05\text{E}_{\text{min}}}$	% decrease
Red Storm	Standard	962	587	38.94
	25% memory checkpointed	1248	669	46.35
	Partition Size: 1/8th n_{max}	***280***	***109***	***61.04***
	Node MTTI: 2.5years	1569	1091	30.48
Blue Gene/L	Standard	3407	2907	14.68
	25% memory checkpointed	3660	2773	24.24
	Partition Size: 1/8th n_{max}	***631***	***380***	***39.42***
	Node MTTI: 2.5years	6212	5571	10.25
Jaguar	Standard	712	482	32.53
	25% memory checkpointed	895	537	40.02
	Partition Size: 1/8th n_{max}	***194***	***86***	***55.63***
	Node MTTI: 2.5years	1215	924	23.94
Petaflop	Standard	2697	2100	22.35
	25% memory checkpointed	3166	2195	32.22
	Partition Size: 1/8th n_{max}	***615***	***324***	***47.22***
	Node MTTI: 2.5years	5568	4852	12.86

presented in Table 2. For each of the four systems considered, the case that has the largest decrease in the number of checkpoint I/O operations is shown in bold. The reduction in the number of checkpoint I/O operations was in the range of 10.25% to 61.07%.

7 Background and Related Work

There is a substantial body of literature regarding the optimal checkpoint problem and several models of optimal checkpoint intervals have been proposed. Young proposed a first-order model that defines the optimal checkpoint interval in terms of checkpoint overhead and mean time to interruption (MTTI). Young's model does not consider failures during checkpointing and recovery [18]. However, POCIME, which is an extension of Young's model to a higher-order approximation, does [3]. In addition to considering checkpoint overhead and MTTI, the model discussed in [16] includes sustainable I/O bandwidth as a parameter and uses Markov processes to model the optimal checkpoint interval. The model described in [11] uses useful work, i.e., computation that contributes to job completion, to measure system performance. The authors claim that Markov models are not sufficient to model useful work and propose the use of Stochastic Activity Networks (SANs) to model coordinated checkpointing for large-scale systems. Their model considers synchronization overhead, failures during checkpointing and recovery, and correlated failures. This model also defines the optimal number of processors that maximize the amount of total useful work. Vaidya models the checkpointing overhead of a uniprocess application. This model also considers failures during checkpointing and recovery [17]. To evaluate the performance and scalability of coordinated checkpointing in future large scale systems, [4] simulates checkpointing on several configurations of a hypothetical Petaflop system.

Their simulations consider the node as the unit of failure and assume that the probability of node failure is independent of its size, which is overly optimistic [6]. Yet another related area of research is failure distributions of large-scale systems. There has been a lot of research conducted in trying to determine failure distributions of systems. Failure events in large-scale commodity clusters as well as the BG/L prototype have been shown to be neither independent, identically distributed, Poisson, nor unpredictable [8] [10]. [12] presents a study on system performance in the presence of real failure distributions and concludes that Poisson failure distributions are unrealistic. Similarly, a recent study by Sahoo [14] analyzing the failure data from a large-scale cluster environment and its impact on job scheduling, reports that failures tend to be clustered around a few sets of nodes, rather than following a particular distribution. In 2004 there was a study on the impact of realistic large-scale cluster failure distributions on checkpointing [10]. Oliner et. al.[9] profess that a realistic failure model for large-scale systems should admit the possibility of critical event prediction. They also state that the idea of using event prediction for pro-active system management is a direction worth exploring [10][13]. Recently, there has been a lot of research towards finding alternatives for disk-based periodic checkpointing techniques [9] [7] and there have been some promising results. However, until these new techniques reach a level of maturity, disk-based periodic checkpointing technique will continue to be the reliable and time-tested method of fault tolerance [15]. Besides, a lot of important legacy scientific applications use periodic checkpointing and, therefore, issues related to periodic checkpointing still need to be addressed.

Note that PETM and POCIME do not make any assumptions on the failure distribution of the system for its **entire lifetime**. However, they assume an exponential failure distribution only for the **duration of the application run**, which might be a few days, weeks, or months. Note that this is drastically different from assuming an exponential failure distribution for the life of the system. This model offers the application programmer the flexibility to use whatever means is deemed right for the system to determine the value of MTTI, M, at the beginning of the application run. Given this value of M, the model then assumes that during the application run the failure distribution of the system is exponential. This makes the model mathematically amenable, elegant, and useful. The assumption of exponential failure distribution for the duration of the application run is validated by the observation that a plot of the inter-arrival times of 2,050 single-node unscheduled interrupts, gathered on two different platforms at Los Alamos National Laboratories over a period of a year, i.e., January 2003 to December 2003, fits a Weibull distribution with a shape factor 0.91/0.97. Since an exponential distribution is equivalent to a Weibull distribution with a shape factor 1.0, it is reasonable to assume an exponential failure distribution. Due to space constraints, we do not present the plot in this paper.

8 Conclusions and Future Work

We believe that the modeling work presented in this paper, based on the POCIMI model, is complementary to that associated with the PETM and POCIME

models. Together they provide pointers and insights for making an informed tradeoff between expected execution time and the number of checkpoint I/O operations. This facilitates an application programmer to chose a value of the checkpoint interval, a tunable parameter, that balances the frequency at which the application performs checkpoint I/O operations and expected execution time. To the best of our knowledge, at this time there is no quantitative guidance to facilitate such a tradeoff. Both models do not factor in the deterioration caused by resource contention. However, they model the general case, which can be used as a guidance for specific cases.

In an MPP system that has a system-wide view of all concurrently executing applications and has control over the checkpoint parameters of these applications, checkpoint intervals could be tuned to provide performance differentiation and performance isolation of concurrent applications. For example, the application with highest priority can be run with a checkpoint interval that is optimal w.r.t execution time, while applications with the lowest priorities can be set to run with checkpoint intervals that are closer to the value of the optimal checkpoint interval w.r.t total number of checkpoint I/O operations. The other applications can, perhaps, use checkpoint intervals that are between their two optimal values. For periodic checkpointing applications, both the expected wall clock execution time and the expected number of checkpoint I/O operations are important metrics to be considered in order to make decisions about checkpoint intervals. An important target of our future work is to provide specific guidelines about how to coordinate checkpoint operations of concurrently executing applications in order to achieve high system throughput.

Acknowledgments

We are pleased to recognize the support of this work by the Army High Performance Computing Research Center (AHPCRC) under ARL grant number W11NF-07-2-2007 and the helpful professional interactions we have had with Seetharami Seelam (IBM), Ron Oldfield and Rolf Riesen (Sandia National Laboratories), and Maria Ruiz Varela (UTEP).

References

1. Asci purple statement of work, lawrence livermore national laboratory, `http://www.llnl.gov/asci/purple/attachment_02_purplesowv09.pdf` (accessed: April 23, 2006)
2. Arunagiri, S., Daly, J.T., Teller, P.J.: Propitious checkpoint intervals to improve system performance. Technical Report UTEP-CS-09-09, University of Texas at El Paso (2009)
3. Daly, J.: A higher order estimate of the optimum checkpoint interval for restart dumps. Future Generation Computer Systems 22, 303–312 (2006)
4. Elnozahy, E.N., Plank, J.S.: Checkpointing for peta-scale systems: A look into the future of practical rollback-recovery. IEEE Transactions on Dependable and Secure Computing 1(2), 97–108 (2004)

5. Gibson, G., Schroeder, B., Digney, J.: Failure tolerance in petascale computers. CTWatch Quarterly (November 2007)
6. Kavanaugh, G.P., Sanders, W.H.: Performance analysis of two time-based coordinated checkpointing protocols. In: PRFTS 1997: Proceedings of the 1997 Pacific Rim International Symposium on Fault-Tolerant Systems, Washington, DC, USA, p. 194. IEEE Computer Society, Los Alamitos (1997)
7. Kim, Y., Plank, J.S., Dongarra, J.J.: Fault tolerant matrix operations for networks of workstations using multiple checkpointing. In: HPC-ASIA 1997: Proceedings of High-Performance Computing on the Information Superhighway, HPC-Asia 1997, Washington, DC, USA, p. 460. IEEE Computer Society, Los Alamitos (1997)
8. Liang, Y., Sivasubramaniam, A., Moreira, J.: Filtering failure logs for a bluegene/l prototype. In: Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN 2005), June 2005, pp. 476–485 (2005)
9. Oliner, A.J., Rudolph, L., Sahoo, R.K.: Cooperative checkpointing: a robust approach to large-scale systems reliability. In: ICS 2006: Proceedings of the 20th Annual International Conference on Supercomputing, Cairns, Queensland, Australia, pp. 14–23. ACM Press, New York (2006)
10. Oliner, A.J., Rudolph, L., Sahoo, R.K.: Cooperative checkpointing theory. In: Proceedings of IPDPS, Intl. Parallel and Distributed Processing Symposium (2006)
11. Pattabiraman, K., Vick, C., Wood, A.: Modeling coordinated checkpointing for large-scale supercomputers. In: Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN 2005), Washington, DC, pp. 812–821. IEEE Computer Society, Los Alamitos (2005)
12. Plank, J.S., Elwasif, W.R.: Experimental assessment of workstation failures and their impact on checkpointing systems. In: Proceedings of the The Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing, June 1998, pp. 48–57 (1998)
13. Sahoo, R.K., Bae, M., Vilalta, R., Moreira, J., Ma, S., Gupta, M.: Providing persistent and consistent resources through event log analysis and predictions for large-scale computing systems. In: SHAMAN Workshop, ICSY 2002 (June 2002)
14. Sahoo, R.K., Sivasubramaniam, A., Squillante, M.S., Zhang, Y.: Failure data analysis of a large-scale heterogeneous server environment. In: Proceedings of the International Conference on Dependable Systems and Networks (DSN 2004), June 2004, pp. 772–781 (2004)
15. Subramaniyan, R., Grobelny, E., Studham, S., George, A.D.: Optimization of checkpointing-related i/o for high-performance parallel and distributed computing. J. Supercomput. 46(2), 150–180 (2008)
16. Subramaniyan, R., Studham, R.S., Grobelny, E.: Optimization of checkpointing-related I/O for high-performance parallel and distributed computing. In: Proceedings of The International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 937–943 (2006)
17. Vaidya, N.H.: Impact of checkpoint latency on overhead ratio of a checkpointing scheme. IEEE Transactions on Computers 46(8), 942–947 (1997)
18. Young, J.W.: A first order approximation to the optimum checkpoint interval. Communications of the ACM 17(9), 530–531 (1974)