Propitious Checkpoint Intervals to Improve System Performance

Sarala Arunagiri [*] John T. Daly [†] Patricia J. Teller[*]

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 I/O bandwidth demand made by the application. In a computing environment where there are concurrent applications competing for access to the network and storage resources, the I/O demand of each application is a crucial factor in determining the throughput of the system. Thus, in order to achieve a good overall system throughput, it is important for the application programmer to choose a checkpoint interval that balances the two opposing metrics - the number of checkpoint I/O operations and the application execution time. 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 model this and present the optimal checkpoint interval that minimizes the total number of checkpoint I/O operations. We present extensive simulation studies that validate our analytical model. Insights provided by this model, combined with existing models for wall clock execution time, facilitate application programmers in making a well informed choice of checkpoint interval leading to 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.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems; D.4.5 [**Software**]: Reliability—*Checkpoint/restart*

1. INTRODUCTION

Current Massively Parallel Processing (MPP) systems typically have tens of thousands of processors that are partitioned into specialized sets of components for computation, storage, and system services [7]. Examples include the Cray XT systems at Sandia National Laboratories and Oak Ridge National Laboratory [3] and the IBM BlueGene/L system at Lawrence Livermore National Laboratory [23]. Simulations of environmental, biological, and seismic phenomena that address problems related to key scientific and social issues are a few examples of applications that run on these systems. Many of these applications require a large proportion of the MPP system, for months at a time, to achieve meaningful results.

However, it is well know that as MPPs 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 and recovering from failures. With increase in the scale of computing systems, element failures will become frequent making it increasingly difficult for long running applications to make forward progress in the absence of fault tolerance mechanisms [6].

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. A *checkpoint write operation* is the process of saving the computation state to a stable storage. *Checkpoint data* is the data that is sufficient to restart the computation, in the event of a failure. *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. In the event of a failure, a *checkpoint read operation* is performed in which the last checkpoint data is read. In this paper we assume that the time taken to read the checkpoint data is the same as the time taken to write checkpoint data which is given by the checkpoint latency. *Checkpoint overhead* is the increase in the execution time of an application due to checkpointing. A *checkpoint cycle* is comprised of a checkpoint write operation and the computation before the next checkpoint write operation, assuming no failures occur between the two checkpoint writes. The length of a checkpoint cycle is given

[*] The University of Texas at El Paso
[†] Center for Exceptional Computing

by the checkpoint interval plus the checkpoint latency.

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 increase the time to solution substantially. 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) [25, 4, 20].

In this paper we focus on another performance metric, the number of checkpoint I/O operations performed during an application run. It is our thesis that execution time and the number of checkpoint I/O operations are complementary metrics and both of them need to be considered in selecting a checkpoint interval. Using analytical modeling, we explore the impact of the choice of checkpoint interval on the number of checkpoint I/O operations performed during an application run.

1.1 Motivation and Background

The rate of growth of disk-drive performance, both in terms of I/Os per second and sustained bandwidth, is smaller than the rate of growth of the performance of other components in computing systems [20]. Therefore, in order to attain good overall performance of computing systems, it is important to design applications bearing in mind the limitations posed by I/O resources and to use them efficiently. There are several scientific papers that elaborate this problem, an example of a recent paper is [20].

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]. The demand made by checkpoint (defensive) I/O is a primary driver of the sustainable bandwidth of high performance filesystems as indicated by [6] and other scientific literature. Hence it is critical to manage the amount and rate of defensive I/O performed by an application. In a recent paper, [20], extensive results are presented showing that as the memory capacity of the system increases so does the I/O bandwidth required in order to perform checkpoint operations at the optimal checkpoint interval that minimizes the execution time and thus attain minimum execution time. An example presented in the paper is, for a system with 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.

In this paper we address this problem by suggesting that using the checkpoint interval that minimizes execution time may not always be the best choice. Sometimes, when we are dealing with applications that are not time critical, it might be better to use a checkpoint interval that leads to an execution time that is larger than the minimum value but reduces the demand on the I/O bandwidth. We perform analytical modeling studies and show that as the value of checkpoint interval increases the number of checkpoint I/O operations decreases. We also show that for values of checkpoint interval in the range 0 to M, the MTTI, there is a single value of checkpoint interval that minimizes the number of checkpoint I/O operations. However, as stated earlier, our thesis is that neither the checkpoint interval that minimizes execution time nor the one that minimizes the number of I/O operations might be an appropriate choice of checkpoint interval. This is because application execution time and the number of checkpoint I/O operations are opposing factors and therefore at the value of checkpoint interval that minimizes execution time the number of checkpoint I/O operations are prohibitive and vice versa. However, our observation is that there are checkpoint intervals between these two optimal values that result in reasonable values of both execution times and number of checkpoint I/O operations, although, at these checkpoint intervals both execution times and number of checkpoint I/O operations are not at their minimum values.

We present an example scenario in order to illustrate our main idea. Consider a partition of Blue Gene/L (BG/L) with 1024 nodes with 0.5GB of memory per node, an MTTI of one year per node, and a storage bandwidth of 45GB/s. Consider an application with a *solve time*, which is the time spent on actual computation cycles towards a final solution, of 500 hrs and a *restart time*, which is the time before an application is able to resume real computational work after a failure, of 10 minutes. Given that each of the 1024 nodes executing the application checkpoints half of its available memory (i.e., 0.25GB), the amount of checkpoint data generated by the application per checkpoint is 256GB. If we assume that the application gets the full storage bandwidth of 45GB/s, the checkpoint latency is 5.68 seconds. Given the set of parameters, the Poisson Execution Time Model [4] provides a way of computing the execution time and an approximation of the optimal checkpoint interval. Accordingly, the minimum expected execution time of this application is 519.76 hours, corresponding to the optimal checkpoint interval of 9.8 minutes.

In addition to the processors, checkpoint operations consume several resources, for e.g., network bandwidth, file system, storage bandwidth, etc. These resources are shared between checkpointing (defensive) I/O and the productive I/O of concurrently executing applications in an MPP. Contention at these shared resources can inhibit the applications from attaining the optimal execution time even when the applications are checkpointing at the optimal checkpoint interval. To illustrate this consider the example application running on BG/L. Now suppose that 64 such applications with similar checkpoint parameters, each executing on a 1024 node partition of BG/L, are running concurrently (this is possible in BG/L which has 64536 nodes). As stated before, the optimal checkpoint interval is 9.8 minutes. The total checkpoint data generated by the 64 applications during each checkpoint cycle is 16384 GB; if serviced at full bandwidth rate this takes 6.07 minutes of I/O system service time to write. In the best case scenario the checkpoints of the 64 applications are staggered to ensure that no two applications attempt to perform a checkpoint operation at the same time. This implies that for 6.07 minutes during every checkpoint cycle, which is 9.8 minutes, the I/O system is busy writing checkpoint data. In essence, the I/O system is busy performing checkpoint write operations 62% of its time.

If the communication system and the file system can handle data at the specified rate and the other I/O requirements of all 64 applications are small enough that the storage system can service all of them in less than 38% of its time, and this I/O does not ever contend with any checkpoint I/O, then all 64 of these applications, each having a solution time of 500 hours, can be concurrently executed to completion(in a statistical sense) , with checkpoints, in 519.76 hours - the expected minimum execution time according to the Poisson Execution Time Model. However, if any of these conditions are not satisfied then the specified minimum execution time cannot be attained. For example, if one or more of these applications are I/O intensive and together their I/O needs cannot be serviced by 38% of the I/O system's time, then the performance of such applications is bound to deteriorate. In the absence of I/O performance isolation, the performance of other applications that share the I/O resources with the I/O-intensive applications is bound to deteriorate and it is unlikely that any of them will complete their execution in 519.76 hours. The example presented was simplified for the purpose of illustration. But even if we had partitions of different sizes, similar concerns arise.

On encountering such a problem, at the outset it appears like increasing the checkpoint interval would increase the checkpoint cycle thereby reducing the relative time during which the I/O resources are busy performing checkpoint write operations during each checkpoint cycle. For instance, in our illustrative example if we increase the checkpoint interval from 9.8 minutes to 13 minutes, from the Poisson Execution Time Model, we know that the expected execution time increases to 520.16 hours, which is an increase of less than 30 minutes, and now the I/O resources are busy with checkpoint writes for a little less than 50% of their time. This looks encouraging. However, if the I/O demand of the application, excluding the checkpoint I/O, is high enough that it still cannot be handled in 50% of I/O resources' time, we consider increasing the checkpoint interval further. For example, if we increase the checkpoint interval to 67 minutes, the execution time increases to approximately 545 hours, which is about 5% larger than the minimum expected execution time, whereas, the I/O resources are now busy performing checkpoint writes for less than 10% of the times during each checkpoint cycle.

Note that, in the above example, as the checkpoint interval increases beyond 9.8 minutes, the execution time of the application increases. We know that as the execution time increases the expected number of failures increases thereby increasing the number of checkpoint read operations. Since I/O resources are consumed for both checkpoint read operations and checkpoint write operations, it is important to understand how the aggregate number of checkpoint I/O operations varies with increasing checkpoint intervals.

Contributions of this paper,

- In Section 3, we present an analytical model of aggregate number of checkpoint I/O operations and show how it varies over the range of checkpoint intervals from 0 to M, the MTTI of the system. We show that in this range there is one checkpoint interval that minimizes the number of checkpoint I/O operations, *the optimal checkpoint interval that minimizes number of checkpoint I/O operations.* We present the formula to compute this optimal checkpoint interval in terms of Product Log function.
- In Section 4, we present results of extensive stochastic simulation performed to validate the analytical model. The results show that
  - The model is accurate via demonstrating that the error is a small value.
  - In reality, we can expect the values of number of checkpoint I/O operations to have a small variance which is demonstrated by the fact that the simulated number of checkpoint I/O operations have a high degree of precision.
  - The idealization used in our analytical modeling is reasonable and it does not introduce large errors.
- In Section 5, we illustrate the existence of propitious checkpoint intervals using parameters of four MPP systems, Red Storm, Jaguar, BlueGene/L, and the Petaflop machine and a representative application.

In Sections 6 and 7 we present related work and future work, respectively.

The work presented in this paper is based on the following execution time model [4], formulated by John Daly.

2. THE POISSON EXECUTION TIME MODEL (PETM)

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 the following equations [4]. In this paper, for the sake of convenience, we refer to these models as The Poisson Execution Time Model (PETM) and the ProductLog Optimal Checkpoint Interval Model for Execution time (POCIME). Note that in the original literature, which presents these models, the terms PETM and POICME are not used to refer to these models. We introduce these terms with permission from the author of the original literature.

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

$$\tau_{opt} = M\left(1 + ProductLog\left(-e^{-\frac{\delta+M}{M}}\right)\right) \qquad (2)$$

POICME's approximation to $\tau_{opt}, \tau_{appx}$is given by

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

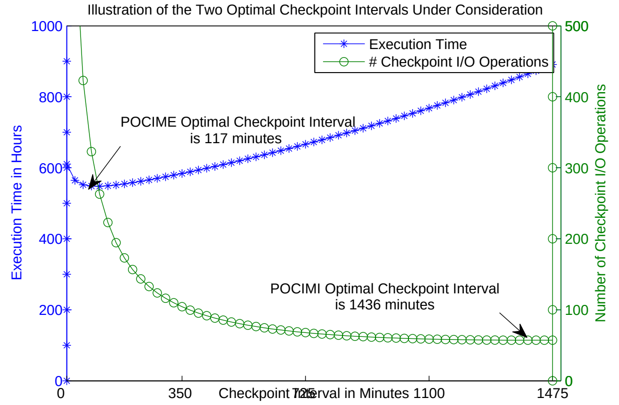

Figure 1: Plots of Execution Time and Number of Checkpoint I/O as a function of checkpoint Intervals. The parameters are MTTI, M = 24 hours, Checkpoint latency, δ = 5 minutes, and Restart time, R = 10 minutes, and Solution time, T_s=500hrs

$$\text{where,}\quad \begin{array}{lcl} T_s & = & \text{Solution time,} \\ \tau & = & \text{Checkpoint interval} \\ \delta & = & \text{Checkpoint latency,} \\ M & = & \text{Mean time between interruptions (MTTI) of} \\ & & \text{the application, and} \\ R & = & \text{restart time.} \end{array}$$

3. MODELING THE NUMBER OF CHECKPOINT I/O OPERATIONS: PRODUCTLOG OPTIMAL CHECKPOINT INTERVAL MODEL FOR I/O(POCIMI)

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

$$\begin{aligned} &\text{Expected number of checkpoint reads} = \\ &\frac{\text{Expected execution time}}{M} = \frac{T_s e^{R/M}(e^{\frac{\delta+\tau}{M}}-1)}{\tau} \\ &\text{Expected number of checkpoint writes} = T_s/\tau \\ &\text{Expected number of aggregate checkpoint I/O operations,} \end{aligned}$$

$$N_{I/O} = \frac{T_s}{\tau}\left[1 + e^{R/M}\left(e^{\frac{\delta+\tau}{M}} - 1\right)\right] \tag{4}$$

For values of parameters- MTTI = 24 hours, checkpoint latency = 5 minutes, and restart time = 10 minutes, 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 Figure 1. From modeling studies in [4] we know that the execution time is a convex function of checkpoint interval and it has a single minimum at $\tau_{opt} = 117$ minutes. From Figure 1, it appears like, $N_{I/O}$, the aggregate number of checkpoint I/O operations, is also a convex function of checkpoint interval with a minimum value in the range $0 \le \tau \le M$. The minimum in this case is 1436 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.

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

$$\tau_{I/O} = M\left(1 + ProductLog\left(-e^{-\frac{\delta+M}{M}} + e^{-\frac{R+\delta+M}{M}}\right)\right) \tag{5}$$

Proof. $N_{I/O}$ is given by Equation 4. We look for stationary points of $N_{I/O}$ w.r.t. τ , i.e., values of τ at which the first derivative of $N_{I/O}$ w.r.t τ is zero.

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

$$\frac{dN_{I/O}}{d\tau} = 0 \implies$$

$$e^{\frac{R}{M}} e^{\frac{\delta+\tau}{M}} \frac{\tau}{M} - e^{\frac{R}{M}} e^{\frac{\delta+\tau}{M}} + e^{\frac{R}{M}} - 1 = 0 \implies$$

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

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

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

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

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

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

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

$$\tau = M\left(1 + ProductLog\left(-e^{-\frac{\delta+M}{M}} + e^{-\frac{R+\delta+M}{M}}\right)\right) \tag{6}$$

There is a unique positive value of τ that satisfies the equation above, let us denote it by $\tau_{I/O}$. The $ProductLog$ term in Equation 6 is negative and its absolute value is less than one. Therefore, $\tau_{I/O}$ 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

$$\begin{aligned} N_{I/O} &= \frac{\text{Expected Execution Time}}{M} + \frac{T_s}{\tau} \\ &= \frac{T}{M} + \frac{T_s}{\tau} \\ \frac{dN_{I/O}}{d\tau} &= \frac{1}{M}\frac{dT}{d\tau} - \frac{T_s}{\tau^2} \\ \frac{d^2N_{I/O}}{d\tau^2} &= \frac{1}{M}\frac{d^2T}{d\tau^2} + 2\frac{T_s}{\tau^3} \end{aligned} \tag{7}$$

$\frac{d^2T}{d\tau^2}$ is known to be positive for all values of τ in the range $0 < \tau \le M$ [4]. This makes the right hand side of Equation 7 and thus $\frac{d^2N_{I/O}}{d\tau^2}$positive for all τ in the range $0 < \tau \le$

M. Therefore, the stationary point $\tau_{I/O}$ is a minimum with respect to the number of I/O operations. □

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

THEOREM 2. *The value of checkpoint interval that minimizes the number of I/O operations, $\tau_{I/O}$ 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 $\tau_{I/O}$;

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

Consider arguments to the $ProductLog$ function in the above equations for τ_{opt} and $\tau_{I/O}$. 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 $\tau_{I/O}$. Since $ProductLog(-1/e) = -1$ and since $ProductLog$ function is monotonically increasing in the range ($-\frac{1}{e}$ to 0).

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

□

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

COROLLARY 1. *For checkpoint intervals, τ, in the range $\tau_{opt} \leq \tau \leq \tau_{I/O}$, the expected value of frequency of checkpoint I/O operations decreases with increasing checkpoint intervals.*

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

In order to evaluate the accuracy of our analytical model, POCIMI, conducting repeated runs of experiments on the scale of systems we are looking into is infeasible in terms of system availability, execution time, and effort. This leaves us with simulation study as the only feasible alternative. We present details of our simulation studies in Section 4. Note that, for any given value of checkpoint interval, the mean frequency of checkpoint I/O operations is a derived quantity given by the expected number of checkpoint I/O operations divided by the expected execution time . Therefore, in the rest of this paper, we limit our discussion to the expected number of checkpoint I/O operations.

3.1 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. For example, in our analytical model, POCIMI, we made the following simplifying assumptions -

1. We made the assumption that the values of MTTI and checkpoint latency can be estimated accurately and they stay the same through the run of the experiment.
2. We model the number of checkpoint read operations as the expected execution time divided by MTTI. We use the expression for the expected execution time that is derived from the Poisson Execution Time Model which assumes that when a failure occurs, the system can restart right away and perform a checkpoint read and resume computation from the last successful checkpoint. In reality, however, there might be some repair times involved and the system might need to wait while repairs are conducted and proceed only after that.
3. We modeled the contribution of checkpoint write operations by -

$$\text{Expected number of checkpoint writes} = \frac{T_s}{\tau} \tag{8}$$

   There is an implicit assumption here that failures do not occur during checkpoint write operations. In reality, this is unlikely to be true and there is a non-zero probability that a failure occurs during a checkpoint write operation and therefore the write operation will need to be performed again. In reality the contribution of checkpoint write operations to the aggregate number of checkpoint I/O operations may include some unsuccessful write operations as well.

$$\begin{gathered}\text{Expected number of checkpoint writes} = (1+P)\tfrac{T_s}{\tau}\\ \text{Where } \ P \text{ is the probability of occurrence of a failure}\\ \text{during a checkpoint write operation}\end{gathered}$$

The question is, how does idealization affect the accuracy of our analytical model? This question is also addressed in our simulation studies in the next section.

4. STOCHASTIC SIMULATION TO VALIDATE ANALYTICAL MODEL

4.1 Goals and Experimental Plan

Two main goals of our simulation studies are

1. To evaluate the accuracy of POCIMI, the analytical model presented in Section 3. In order to achieve this goal, for a given set of parameters we simulate application runs for different values of checkpoint intervals covering the range of interest, i.e., 0 to M. For each value of checkpoint interval we simulate several runs of the application while counting the number of checkpoint I/O operations for each run. We then consider the mean value of the number of checkpoint I/O operations at each checkpoint interval and compare this mean value to the number of checkpoint I/O operations predicted by the analytical model. We present

the difference between simulated and predicted values of number of checkpoint I/O operations as the absolute error of the analytical model. We also present the relative error defined as the absolute error expressed as a percentage of the predicted analytical value.

2. To evaluate the effects of idealization on the accuracy of POCIMI. In order to achieve this goal, corresponding to every application run, we run two versions of simulations; an idealized version and a minimally idealized version. The idealized version of simulation performs simulation of the process including idealizations used to derive the analytical model. In the minimally idealized version, we relax several of these idealizations in order to more closely reflect the reality. In order to ensure that the difference of the outcome of these two versions can be attributed solely to the difference in the degree of idealization, we designed our simulation experiment such that all values of random variables used in the idealized version of simulation of an application run are the same as the ones used in the corresponding simulation of the minimally idealized version of the application run. We then present the difference in the relative errors associated with the two versions of simulations.

4.2 Details of Simulation

We simulated the process of running an application on a 1000 node system with each node having an exponential failure distribution. Each application run is simulated by accounting for execution time and marching through it while checking for failure of each node periodically and accounting for application progress during every failure free duration of length equal to the sum of a checkpoint interval and checkpoint latency. A checkpoint write operation is accounted for at the end of such a period. When a failure occurs, a checkpoint read operation is recorded and restart time and checkpoint read time are added to execution time. Failures are generated using random numbers and poisson distribution with mean equal to the node MTTI. We performed simulations for two sets of parameter values both of which were picked from examples in published literature. The solution time of the simulated application run was 500 hours, restart time was 10 minutes and the checkpoint latency was 5 minutes for both sets of experiments. However, the first experiment was for MTTI of 6 hours (360 minutes) and the second experiment was for a value of MTTI of 24 hours (1440 minutes).

For both experiments the design variable was the checkpoint interval and the response variable was the number of checkpoint I/O operations. For the first set of experiments, our design points were at 15 values of evenly spaced checkpoint intervals in the range 0 to 360 minutes, {25,50,75,...,350, 375}. In the second set of experiments, the range of interest for values of checkpoint intervals 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. So, the design points of our simulation were the following 18 values of checkpoint intervals - {50,75,...,175, 650,675,...,775,1350,1375,...,1475}. For each set of experiments, we conducted 5 trials. For each trial and at each chosen checkpoint interval we collected response data from simulation and computed the following

- The mean of the simulated number of checkpoint I/O operations
- The 99% confidence interval of the mean
- DIstance of the 99% confidence interval from the value of number of checkpoint I/O operations given by the analytical model
- Absolute error
- Relative error

Absolute and relative errors are defined by,

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

Due to space constraints, we present only the results of simulation on the second set of data because it spans a larger range of checkpoint intervals. However, we would like to report that the results for the first set of data were very similar to the results presented here. Subplot(a) of Figure 2 is a plot of 99% confidence interval of mean simulated number of checkpoint I/O operations and analytical values given by POCIMI. For this plot, we picked data from one of five trials, arbitrarily, and it happened to be Trial 3. This decision was made because the lines representing simulated mean values of all 5 trials were almost overlapping and cluttering the figure and therefore, for the sake of clarity, we picked data from one trial. 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 really small. When we did zoom into the figure we were able to see that there was indeed an error bar showing the confidence interval. However, this plot presents the larger picture for the range of checkpoint intervals. In order to present these details to the reader, we plotted the bar graphs of Subplot(b) and (c) of Figure 2 and Subplot(a) and (b) of Figure 3. Although, in Subplot(a) of Figure 2 we only used data from Trial 3, for all the bar graphs presented in this paper we use data from all 5 trials, as a result, the bars shown are an overlap of the bars for the 5 trials and thus the they represent largest possible positive and negative values. Subplots(b) and (c) of Figure 2 present details of Subplot(a) of Figure 2 which are significant but cannot be deciphered at the scale at which the figure is presented, i.e. the width and the distance of the error bars representing 99% confidence intervals from the line representing the analytical value given by POCIMI. Subplots(a) and (b) of Figure 3 are bar graphs of the absolute error and the relative error.

4.3 Discussion of Results

- The value of absolute error lies between -2 and 4. The relative error of the model is in the range -0.5% and 6%. This demonstrates a high degree of accuracy of the model.
- The size of 99% confidence interval of the mean value of simulated number of checkpoint I/O operations is less than 7. This implies that the aggregate number of checkpoint I/O operations from the simulation runs have a small variance. This in turn implies that, in reality, it is reasonable to expect the number of checkpoint I/O operations to be fairly precise. This finding, when combined with the fact that the relative error is small, implies that one can expect the number of

checkpoint I/O operations for each application run to be close to the value given by POCIMI.

- The distance between the narrow 99% confidence interval and the value of checkpoint I/O operations predicted by the analytical model was less than 2. When the 99% confidence interval of the mean simulated number of checkpoint I/O operations does not include the analytical value given by POCIMI, we consider how far the confidence interval is from it. This distance is a pointer to the extent of inaccuracy of the model because we can with a high probability expect the error to be at least as large as the distance of the 99% confidence interval from the value given by POCIMI. For the five trials of the simulation this value was no more than 2.

4.4 Idealized and Minimally Idealized Versions of Simulation

With an intent of quantifying the contribution of idealization to the error in predictive accuracy of POCIMI, we performed the following experiment. Corresponding to every run of the application at a chosen design point, i.e., value of checkpoint interval, we ran three versions of simulations-the base version, the idealized version, and the minimally idealized version. The idealized and the minimally idealized versions differ in the following ways;

1. The idealized version of simulation assumes that the checkpoint latency is the specified constant value throughout an application run. The minimally idealized version assumes that the checkpoint latency is a uniformly distributed random variable having a value centered around and varying between + or - 5% of the parameter value specified. Therefore, every time a checkpoint operation is performed in the minimally idealized version, a random value of checkpoint latency is picked from this range. Using such random values of checkpoint latencies is meant to account for inadequacies of the process of estimating checkpoint latency and the inherent variability of checkpoint latency in reality. In reality checkpoint latencies are not precise.
2. When a node fails, the idealized version of the simulator does not account for any repair times. It assumes that the system can restart right away and perform a checkpoint read and resume computation from the last successful checkpoint. The minimally idealized version, however, models the process where the system waits for the failed node to be repaired before performing restart. Repair time is assumed to be a random variable having a gamma distribution with a mean value of 60 minutes. This is based on anecdotal evidence from scientists at Los Alamos National Laboratories.
3. As a consequence of idealization number 2, every time a node failure occurs, a checkpoint read is recorded in the idealized version. However, in the minimally idealized version, a checkpoint read is accounted for only when a node failure occurs when the the system is in working condition, i.e., no other nodes have failed or the system is not waiting for repair.
4. In the idealized version of simulation, only checkpoint write operations that were successfully completed are accounted for. However, in the minimally idealized version if a failure occurs during a checkpoint write operation, the checkpoint write operation is accounted for. This more closely reflects the amount of effort invested in checkpoint writes because even when a checkpoint write does not succeed due to a node failure, some amount of I/O effort was invested and some amount of I/O resources were consumed by the unsuccessful checkpoint write operation.

In order to ensure that the idealized and the minimally idealized versions differ only in the above mentioned ways, they needed to be deterministic in all other manner. The events that introduce randomness in the simulation process are the failure times and repair times. For each application run, at every chosen checkpoint interval, we first ran a base version of simulation to generated failure times and repair times in conformance with the probability distribution of the corresponding random variable. We recorded these times and used them as failure times and repair times in the corresponding application run in the idealized and minimally idealized versions of simulation. Thus although failure times and repair times in the idealized and minimally idealized versions are still based on random variables, they have identical values for the corresponding runs of an application. The results presented earlier were for the minimally idealized version of simulation since it simulated conditions that are closer to reality.

We present absolute error of POCIMI with respect to minimally idealized and idealized versions of simulation in Subplot(a) of Figure 4. They are close to each other and therefore on cannot visually discern any difference between the two of them. Again, we present the difference in relative error between idealized version of simulation and the minimally idealized version in Subplot(b) of the same figure. We find that the contribution to the relative error made by the idealization used in our analytical model is within the range -2% and +1%. This demonstrates that the idealization used by us is not too restrictive and therefore does not affect the accuracy of the model too much.

For completeness sake, for the trial for which we presented the plot of mean simulated number of I/O operations in Subplot(a) of Figure 2, Trial 3, we present the plot of the mean of the simulated execution time and and the interarrival time of checkpoint I/O operations in Subplots(c) and (d) of Figure 4, respectively. The execution time shows trends that conform to PETM and the mean interarrival time of checkpoint I/O operations increases steadily with increase in the value of the checkpoint interval which is also as expected.

4.5 Implications of Analytical Modeling Studies of Section 3

- An insight provided by the model is that while τ_{opt} and $\tau_{I/O}$ are both functions of δ and M, $\tau_{I/O}$ is a function of the restart time, R, in addition. $\tau_{I/O}$ decreases with increasing values of R.
- Corollary 1 is key to promising avenues in performance improvement. For values of τ in the range $\tau_{opt} \leq \tau \leq \tau_{I/O}$, both the expected value of frequency of checkpoint I/O operations and the expected value of the number of checkpoint I/O operations decrease with increase in checkpoint interval, enabling a tradeoff between execution time and number of checkpoint I/O operations in order to achieve better performance.
- For running time-critical applications for which hav-

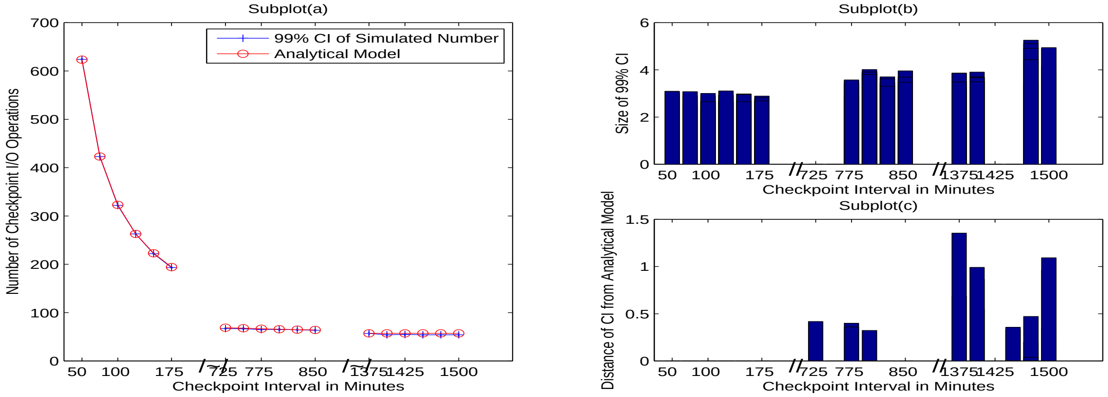

Figure 2: Subplot(a):Plot of Number of Checkpoint I/O as a function of checkpoint Intervals. Subplot(b): Size of the 99% confidence interval(CI) of the mean simulated number of checkpoint I/O operations. Subplot(c): Distance of 99% CI from the value indicated by Analytical model POCIMI. Both these bar graphs show the maximum values of the five trials.

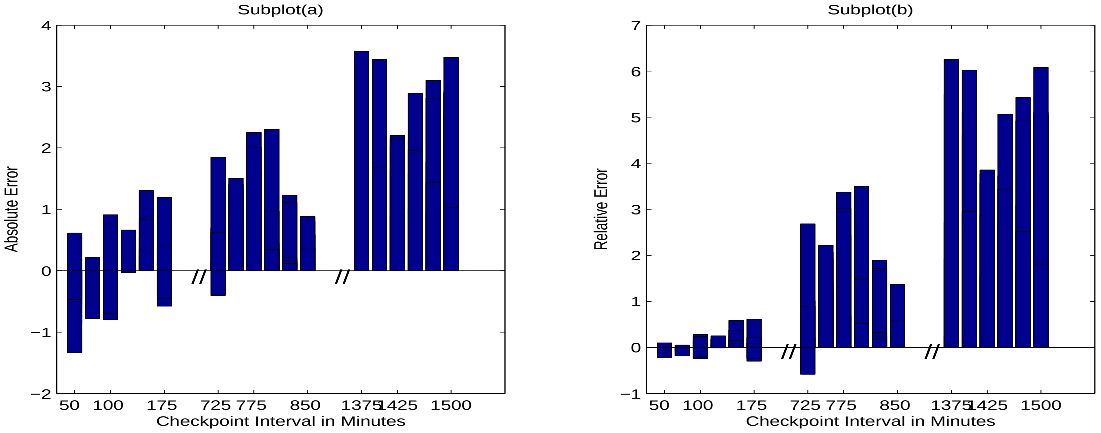

Figure 3: Subplot(a): Absolute error of POCIMI Subplot(b): Relative error. These cover maximum and minimum values of the five trials.

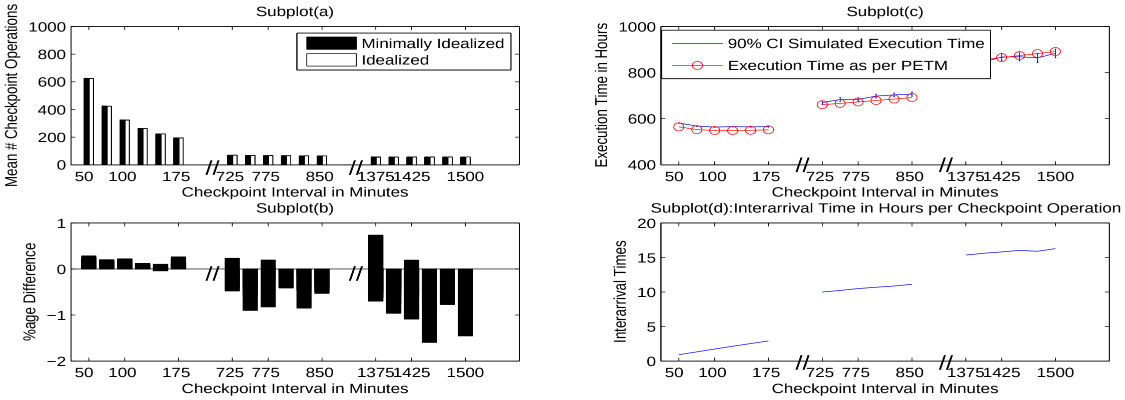

Figure 4: Subplot(a): Shows absolute errors of POCIMI wrt the idealized and the minimally idealized versions of simulations. Subplot(b): Difference between relative errors of POCIMI wrt the idealized and the minimally idealized version of simulation. Subplot(c): Execution time versus checkpoint intervals. Subplot(d): Mean Interarrival time of checkpoint operations in hours.

ing 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, as explained in the illustration in Section 1.1, it is possible to choose a value of checkpoint interval that is larger than the τ_{opt} such that the corresponding value of 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}.

- If we explore clues from visual inspection of Figure 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 number of checkpoint I/O operations corresponding to these are drastically smaller than values corresponding to τ_{opt} while the execution times are marginally larger than the minimum execution time. This, certainly, seems to be true for the example used in Section 1.1 in which increasing the checkpoint interval from τ_{opt}, a value of 9.8 minutes, to 67 minutes, lead to an increase of the execution time by a mere 5% of its minimum value, whereas, the number of aggregate checkpoint operations reduces steeply from a value of 3121 at τ_{opt} to a value of 507 at 67 minutes. This is an 83.78% reduction in the number of checkpoint I/O operations. Whether or not this observation holds good in general i.e., for all values of parameters, is not clear unless we have a rigorous mathematical analysis involving gradients of execution time function and the number of checkpoint I/O operation function in the region of interest. This seems to be a non-trivial mathematical exercise and it could be prospective future work. However, in the next section, we investigate this idea for specific cases using parameters from four MPP systems.

5. INVESTIGATION OF THE EXTENT OF PROMISE OF FEW CHECKPOINT INTERVALS

In this section, we present a summary of studies performed by us to evaluate the impact of increasing checkpoint intervals on the number of checkpoint I/O operations. Using POCIME and the model presented in this paper for the number of checkpoint I/O operations, 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, using parameters presented in Table 1. We investigate the extent of reduction of number of checkpoint I/O operations for a couple of checkpoint intervals as compared to the number of checkpoint I/O operations at τ_{opt}. We considered checkpoint intervals that result in an expected execution of 105% of E_{min} and 110% of T_s, denoted by $\tau_{1.05Emin}$ and $\tau_{1.1T_s}$, respectively. These are based on anecdotal evidence of what is considered a reasonable checkpoint overhead. Note that $\tau_{1.1T_s}$ is meaningful only if $E_{min} < 1.1 * T_s$. A representative application which has a solution time, $T_s = 500\ hours$ and a restart time, $R = 10\ minutes$ is considered for all experiments. For each MPP system, assume

- 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 during every checkpoint cycle.

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%. Then we assume that the MTTI of each node is 2.5 years instead of 5 years. Finally, we assume that the application runs on 1/8th of the nodes of each system instead of all the nodes. While computing checkpoint latency in this case, 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 presented in Table 2. For each of the four systems considered the case which 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%. However, for all cases considered in the table we found that the minimum execution time was larger than $1.1T_s$ so $\tau_{1.1T_s}$ was not relevant. Note that for the four machines when a partition comprising of 1/8th of the total number of nodes was considered, for calculating checkpoint latency we assumed that only 1/8th of the storage bandwidth was available to the partition. Another possibility is to assume that the entire storage bandwidth is available to the smaller partition and we consider such scenarios next. With the exception of Blue Gene/L, for all other machines, the minimum achievable execution time was larger than $1.1T_s$ and therefore we were able to explore the number of checkpoint operations at $\tau_{1.1T_s}$ Table 3 summarizes the results. In comparison to the number of checkpoint I/O operations at τ_{opt}, the corresponding numbers decrease and the percentage decrease is in the range 62% to 79% and 55% to 82%, for $\tau_{1.05E_{min}}$ and $\tau_{1.1T_s}$, respectively. These results provide pointers to potential targets when an application programmer deems it appropriate to use checkpoint intervals larger than τ_{opt}.

We envisage that there are several situations in high performance computing environments executing checkpointing applications where it could potentially be beneficial to use such checkpoint intervals. Some examples are -

1. When, as suggested by [11], an overlay network is used to increase effective I/O bandwidth via latency hiding mechanism, this increased effective bandwidth is not a sustainable one. Bursts of I/O can be handled at this increased rate only if they arrive at a rate that is less than a certain threshold value, depending on the amount of I/O data being handled. For a periodically checkpointing application running on a system with overlay networks, if using a checkpoint interval of τ_{opt} leads to checkpoint writes at a rate that is more than the threshold value for the overlay network then one needs to use a larger value of checkpoint interval.
2. When more than one checkpointing applications are executing concurrently on a large MPP system sharing I/O resources, if the performance of a high priority application is deteriorating due to the large volume of checkpoint I/O of a low priority application then it

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 Systems	Conditions	No of Checkpoint Operations corresponding to τ_{opt}	No of Checkpoint Operations corresponding to $\tau_{1.05E_{min}}$	% decrease in the number of checkpoint I/O operations
Red Storm	Standard	962	587	38.94
	25% of memory checkpointed	1248	669	46.35
	Size of partition is 1/8th of n_{max}	***280***	***109***	***61.04***
	MTTI of a node is 2.5years	1569	1091	30.48
Blue Gene/L	Standard	3407	2907	14.68
	25% of memory checkpointed	3660	2773	24.24
	Size of partition is 1/8th of n_{max}	631	380	39.42
	MTTI of a node is 2.5years	6212	5571	10.25
Jaguar	Standard	712	482	32.53
	25% of memory checkpointed	895	537	40.02
	Size of partition is 1/8th of n_{max}	***194***	***86***	***55.63***
	MTTI of a node is 2.5years	1215	924	23.94
Petaflop	Standard	2697	2100	22.35
	25% of memory checkpointed	3166	2195	32.22
	Size of partition is 1/8th of n_{max}	***615***	***324***	***47.22***
	MTTI of a node is 2.5years	5568	4852	12.86

might be a good idea to try and increase the checkpoint interval of the low priority application. This situation and the next one might need a supervisory view of the whole system which may not always be available.

3. When there are more than one checkpointing applications concurrently executing on a large MPP system, it is desirable to have the full I/O bandwidth of the system available to every application's checkpoint write data. This implies that we need to schedule the checkpoint write operations of these applications in such a way that no two of them collide. For periodic checkpointing applications, theoretically, there are algorithms that could be used to help achieve this. However, while using the algorithm, in case we encounter a situation wherein it is infeasible to checkpoint all the applications at their optimal checkpoint intervals, then it is useful to explore increased checkpoint intervals for some of these applications.
4. Combination of any of the above mentioned situations.

6. 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 [25]. However, POCIME, which is an extension of Young's model to a higher-order approximation, does [4]. In addition to considering checkpoint overhead and MTTI, the model discussed in [21] includes sustainable I/O bandwidth as a parameter and uses Markov processes to model the optimal checkpoint interval. The model described in [15] 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 [24]. To evaluate the performance and scalability of coordinated checkpointing in future large scale systems, [5] 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. [8] is yet another paper on optimal checkpoint interval.

Checkpointing for computer systems has been a major area

Table 3: In each of the four MPP systems, consider a partition which is an eighth of the maximum number of nodes in the system and assume that the full storage bandwidth of the system is available to this partition. Decrease in the number of checkpoint I/O operations at $\tau_{1.05E_{min}}$ and $\tau_{1.1T_s}$ are presented.

MPP Systems	Number of nodes	Number of Checkpoint I/O at τ_{opt}	Number of Checkpoint I/O at $\tau_{1.05E_{min}}$	% decrease in the number of checkpoint I/O operations at $\tau_{1.05E_{min}}$	Number of Checkpoint I/O at $\tau_{1.1T_s}$	% decrease in the number of checkpoint I/O operations at $\tau_{1.1T_s}$
Red Storm	1620	741	151	79	130	82
Blue Gene/L	8192	1498	557	62	NA	NA
Jaguar	1448	504	123	75	115	77
Petaflop	6250	1540	474	69	685	55

of research over the past few decades. There have been a number of studies on checkpointing based on certain failure characteristics [17], including Poisson distributions. Oldfield el.al., [12] present studies modeling the impact of checkpoints on next-generation systems. Tantawi and Ruschitzka [22] developed a theoretical framework for performance analysis of checkpointing schemes. In addition to considering arbitrary failure distributions, they present the concept of an equicost checkpointing strategy, which varies the checkpoint interval according to a balance between the checkpointing cost and the likelihood of failure. Application-initiated checkpointing is the dominant approach for most large-scale parallel systems. Agarwal [2] developed application-initiated checkpointing schemes for BG/L.

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 [10, 14]. [16] 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 [19] 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 [14]. Oliner et. al.,[13] 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 [14, 18].

Recently, there has been a lot of research towards finding alternatives for disk-based periodic checkpointing techniques [13, 9] 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 [20]. 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 it assumes 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 and 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 2050 single node unscheduled interrupts, gathered on two different platforms from LANL over a period of a year during January 2003 to December 2003 fits a Weibull distribution with shape factor 0.91/0.97. Since an exponential distribution is equivalent to a weibull distribution with a shape factor 1.0, Exponential distribution is a pretty close approximation to the real failure distribution. Due to space constraints, we do not present the plot in this paper.

7. CONCLUSION AND FUTURE WORK

We believe that the modeling work presented in this paper, POCIMI, is complementary to PETM and POCIME, the modeling work for execution time. 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, balancing the frequency at which the application performs checkpoint I/O operations as well as its expected execution time. To the best of our knowledge, at this time there is no quantitative guidance to facilitate such a trade off. 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 where the runtime has a system-wide view of all the applications and has control over the checkpoint parameters of concurrent applications, one could tune checkpoint intervals 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 and applications with lowest priority can be set to run with a checkpoint interval that is closer to the value of optimal 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 future work could be to provide specific guidelines about how to coordinate checkpoint operations of concurrent applications in order to achieve high system throughput.

8. 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] S. Agarwal, R. Garg, M. S. Gupta, and J. E. Moreira. Adaptive incremental checkpointing for massively parallel systems. In *Proceedings of the 18th Annual International Conference on Supercomputing*, pages 277–286, New York, NY, 2004. ACM Press.

[3] W. J. Camp and J. L. Tomkins. The red storm computer architecture and its implementation. In *The Conference on High-Speed Computing: LANL/LLNL/SNL*, Salishan Lodge, Glenedon Beach, Oregon, April 2003.

[4] J. Daly. A higher order estimate of the optimum checkpoint interval for restart dumps. *Future Generation Computer Systems*, 22:303–312, 2006.

[5] E. N. Elnozahy and J. S. Plank. 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, April–June 2004.

[6] G. Gibson, B. Schroeder, , and J. Digney. Failure tolerance in petascale computers. CTWatch Quarterly, November 2007.

[7] D. S. Greenberg, R. Brightwell, L. A. Fisk, A. Maccabe, and R. Riesen. A system software architecture for high-end computing. In *Supercomputing '97: Proceedings of the 1997 ACM/IEEE conference on Supercomputing (CDROM)*, pages 1–15, New York, NY, USA, 1997. ACM.

[8] G. P. Kavanaugh and W. H. S. Performance analysis of two time-based coordinated checkpointing protocols. In *In Pacific Rim International Symposium on Fault-Tolerant Systems*, 1997.

[9] Y. Kim, J. S. Plank, and J. J. Dongarra. Fault tolerant matrix operations for networks of workstations using multiple checkpointing. In *HPC-ASIA '97: Proceedings of the High-Performance Computing on the Information Superhighway, HPC-Asia '97*, page 460, Washington, DC, USA, 1997. IEEE Computer Society.

[10] Y. Liang, A. Sivasubramaniam, and J. Moreira. Filtering failure logs for a bluegene/l prototype. In *Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)*, pages 476–485, June 2005.

[11] R. A. Oldfield. Investigating lightweight storage and overlay networks for fault tolerence. In *Proceedings of the High Availability and Performance Computing Workshop*, Santa Fe, NM, Oct. 2006.

[12] R. A. Oldfield, S. Arunagiri, P. J. Teller, S. Seelam, R. Riesen, M. R. Varela, and P. C. Roth. Modeling the impact of checkpoints on next-generation systems. In *Proceedings of the 24th IEEE Conference on Mass Storage Systems and Technologies*, pages 30–43, San Diego, CA, September 2007.

[13] A. J. Oliner, L. Rudolph, and R. K. Sahoo. Cooperative checkpointing: a robust approach to large-scale systems reliability. In *ICS '06: Proceedings of the 20th annual international conference on Supercomputing*, pages 14–23, Cairns, Queensland, Australia, 2006. ACM Press.

[14] A. J. Oliner, L. Rudolph, and R. K. Sahoo. Cooperative checkpointing theory. In *Proceedings of IPDPS, Intl. Parallel and Distributed Processing Symposium*, 2006.

[15] K. Pattabiraman, C. Vick, and A. Wood. Modeling coordinated checkpointing for large-scale supercomputers. In *Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)*, pages 812–821, Washington, DC, 2005. IEEE Computer Society.

[16] J. S. Plank and W. R. Elwasif. 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*, pages 48 – 57, June 1998.

[17] J. S. Plank and M. G. Thomason. Processor allocation and checkpoint interval selection in cluster computing systems. *Journal of Parallel and Distributed Computing*, 61(11):1570–1590, 2001.

[18] R. K. Sahoo, M. Bae, R. Vilalta, J. Moreira, S. Ma, and M. Gupta. Providing persistent and consistent resources through event log analysis and predictions for large-scale computing systems. In *In SHAMAN, Workshop, ICSY02*, June 2002.

[19] R. K. Sahoo, A. Sivasubramaniam, M. S. Squillante, and Y. Zhang. Failure data analysis of a large-scale heterogeneous server environment. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN2004)*, pages 772–781, June 2004.

[20] R. Subramaniyan, E. Grobelny, S. Studham, and A. D. George. Optimization of checkpointing-related i/o for high-performance parallel and distributed computing. *J. Supercomput.*, 46(2):150–180, 2008.

[21] R. Subramaniyan, R. S. Studham, and E. Grobelny. 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*, pages 937–943, 2006.

[22] A. Tantawi and M. Ruschitzka. Performance analysis of checkpointing strategies. *ACM Transactions on Computer Systems*, 110:123–144, May 1984.

[23] T. B. Team. An overview of the BlueGene/L supercomputer. In *Proceedings of SC2002: High Performance Networking and Computing*, Baltimore, MD, November 2002.

[24] N. H. Vaidya. Impact of checkpoint latency on overhead ratio of a checkpointing scheme. *IEEE Transactions on Computers*, 46(8):942–947, 1997.

[25] J. W. Young. A first order approximation to the optimum checkpoint interval. *Communications of the ACM*, 17(9):530–531, 1974.