Opportunistic Checkpoint Intervals to Improve System Performance

Sarala Arunagiri* John T. Daly† Patricia J. Teller* Seetharami Seelam‡ Ron A. Oldfield§
Maria Ruiz Varela* Rolf Riesen§

Abstract

The massive scale of current and next-generation massively parallel processing (MPP) systems presents significant challenges related to fault tolerance. For applications that perform periodic checkpoints, the choice of the checkpoint interval, the period between checkpoints, can have a significant impact on the execution time of the application. Finding the optimal checkpoint interval that minimizes the wall clock execution time, has been a subject of research over the last decade. In an environment where there are concurrent applications competing for access to the network and storage resources, in addition to application execution times, contention at these shared resources need to be factored into the process of choosing checkpoint intervals. In this paper, we perform analytical modeling of a complementary performance metric - the aggregate number of checkpoint I/O operations. We then show the existence and characterize a range of checkpoint intervals which have a potential of improving application and system performance.

1. Introduction

Current Massively Parallel Processing (MPP) systems typically have tens of thousands of processors that are partitioned into specialized seta of components for computation, storage, and system services [6]. 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 [18]. Applications that run on these machines include, for example, simulations of environmental, biological, and seismic phenomena that address problems related to key scientific and social issues. Many of these applications require a large proportion of the MPP system for months at a time to achieve meaningful results.

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 operation* is the process of saving the computa-

*The University of Texas at El Paso
†Los Alamos National Laboratory
‡IBM TJ Watson Research Center
§Sandia National Laboratories

tion 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. *Checkpoint overhead* is the increase in the execution time of an application due to checkpointing.

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 after a failure can increase the time to solution substantially. Deciding upon the optimal checkpoint frequency is the *optimal checkpoint interval* problem, which is well-known. Most solutions attempt to minimize total execution time (i.e., the application time plus the checkpoint overhead) [20, 4]. 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's execution.

1.1. Motivation and Background

Given a set of parameters of the MPP system and the application, Daly's model [4] provides a way of computing the execution time and an approximation of the optimal checkpoint interval. For example, 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. According to Daly's model, the minimum expected execution time of this application is 519.76 hours, corresponding to a checkpoint interval of 9.8 minutes.

There are situations under which this minimum expected execution time cannot be achieved. Checkpoint operations consume several resources in addition to the processors that perform checkpoint operation, for e.g., network bandwidth, file system, storage bandwidth, etc. These resources are shared among checkpointing and other I/O operations of concurrently executing applications in an MPP. Contention at these shared resources can inhibit the applications from attaining the optimal execution time, even when they 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 the 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 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 execution time according to Daly's model. However, if any of these conditions are not satisfied then this 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 their performance 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.

The disparity between real execution time and the expected execution time is due to the fact that the execution time model assumes that the amount of progress made by the application during each checkpoint cycle is equal to the checkpoint interval. This is indeed the case when we consider a single large application running on an MPP system, for example, an application running in a capability computing environment. The model helps us predict the execution times accurately for such cases. However, in computing environments where resource contention between concurrently executing applications exists, application progress during a checkpoint cycle could be less than the checkpoint interval. As a result of this, execution time is larger than the expected optimal execution time even when checkpoint interval is the optimal checkpoint interval.

Questions that arise are -

1. Can we develop an analytical model for execution time of periodic checkpointing applications that factors contention at shared resources? This enables prediction of execution times of such applications.

2. Can we ameliorate resource contention caused by checkpoint I/O operations?

An analytical model as suggested by the first question would need to consider the system configuration and the characteristics of the set of concurrently executing applications to determine the nature and extent of the resource contention that

may arise. The complexity of this task makes it a daunting task if not infeasible. We do not attempt to do this. In this paper, we address the second question. Since we know that checkpoint write operations happen every checkpoint cycle, it seems reasonable to assume that the resource contention cause by checkpoint writes can be ameliorated by decreasing its frequency. The tunable checkpoint parameter that achieves this is the checkpoint interval, it can be increased.

Estimating the amount by which the checkpoint interval needs to increase in order to enable the network, file system, and storage system to be able to handle both checkpoint I/O and other application I/O efficiently, is yet another research problem that is outside the scope of this paper. However, information presented in this paper might help solve it. For the example applications running on 64 BG/L partitions, if we are willing to pay the price of a 5% increase in execution time over the minimum value of 519.76 hours, then the checkpoint interval can be increased by a factor of 6.85 which amounts to 585% increase. Using these new checkpoint intervals, these applications perform checkpoint operations once every 67 minutes as opposed to the original 9.8 minutes. The fraction of time that the storage system needs to write checkpoint data during every checkpoint cycle reduces to 9% as opposed to 62%. In this example, the total number of checkpoint I/O operations decreased from 3120 to 529, which is an 83.68% reduction in the number of checkpoints and the corresponding checkpoint data reduced to 130384 GB, which is 16.32% of the original value, 798927 GB. The expected execution time increases to 545.5 hours, i.e., a 5% increase. In summary, increasing the checkpoint interval by a factor of 6.85 increased the execution time by 5% but it increased the storage systems availability for I/O other than checkpoint I/O by 118%.

Although increasing checkpoint interval decreases the frequency of checkpoint write operations, it also increases the probability of failure during a checkpoint cycle. A failure results in a checkpoint read operation during the restart process. Therefore, at the outset, it is not clear whether increasing checkpoint interval beyond Daly's optimal checkpoint interval always decreases aggregate checkpoint I/O operations. In this paper,

- We study the nature of the variation of the total number of checkpoint I/O operations as a function of checkpoint intervals in the range $0 \leq \tau \leq M$ (Section 3). We show the existence and characterize the range of values of checkpoint intervals that are larger than Daly's optimal checkpoint interval such that the total number of checkpoint I/O operations decreases with increasing checkpoint intervals.

- We show that potentially we could increase the checkpoint interval by as much as 12% beyond Daly's optimal value without increasing the expected execution time (Section 5.1).

- We provide an expression for computing an increased checkpoint interval with a guarantee that the subsequent increase in the expected execution time is no more than 5% of the optimal expected execution time (Section 5.2).

- We use parameters of four MPP architectures, SNL's Red Storm, LLNL's Blue Gene/L (BG/L), ORNL's Jaguar, and

a theoretical Petaflop system to model the impact of the ideas suggested. The empirical evidence shows that there are checkpoint intervals that are larger than the optimal checkpoint interval that increase the expected execution time minimally while decreasing the number of checkpoint I/O operations substantially (Section 6). Thus, we identify and pave the way for further modeling work that might be of immense value in orchestrating concurrent checkpointing applications in order to enable good system performance.

As pointed out in Section 9, there are several situations under which increasing the checkpoint interval beyond Daly's optimal value could lead to performance benefits. The work presented in this paper is complementary to Daly's modeling work on checkpointing. Together, these two models are intended to provide pointers and insights for making an informed tradeoff between expected execution time and number of checkpoint I/O operations. To the best of our knowledge, at this time there is no quantitative guidance to facilitate such a trade off.

2. Choice of Checkpoint Interval

Minimizing application execution time is an important consideration in supercomputers. Therefore, our natural choice of checkpoint interval is the one that minimizes application execution time.

2.1. LANL's Model of Optimum Checkpoint Interval

The total wall clock time to complete the execution of an application and the approximate optimal checkpoint interval are given by the following equations [4]. We refer to them as Daly's execution time model and Daly's optimal checkpoint interval model.

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

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

Daly's approximation to $\tau_{opt}, \tau_{daly}$is given by

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

where

T_s = Time spent doing application computation,

τ = Checkpoint interval

δ = Checkpoint latency,

M = Mean time between interruptions (MTTI) of the application, and

R = restart time.

The approximate optimal checkpoint interval has a bounded relative error of 5.9% and relative error of total problem solution time less than 0.2%. In the rest of our discussion, by default, the reference value of checkpoint interval is τ_{daly}. We model the performance impact of increasing the checkpoint interval from the reference value to larger values.

3. Number of Checkpoint I/O Operations

The set of I/O operations performed by checkpoint/restart mechanisms contain both 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 co-ordinated 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.

The known consequences of increasing checkpoint interval -

- Increase in *checkpoint cycle* which is given by the sum of checkpoint interval and checkpoint latency, assuming that the checkpoint latency is not hidden.
- Increase in the expected execution time if the original checkpoint interval is greater than the optimal checkpoint interval [4].
- Decrease in the frequency of checkpoint write operations.
- Possible increase in checkpoint read operations which are performed as part of the process of recovery after a failure.
- A change in the total number of checkpoint reads and writes performed during the application run. It is not clear whether the sum increases or decreases.

We would like to determine the effect of increasing checkpoint interval on the total number of checkpoint I/O operations.

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

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

$$\tau_{I/O} = M\left(1 + ProductLog\left(-e^{-\frac{\delta+M}{M}} + e^{-\frac{R+\delta+M}{M}}\right)\right) \tag{4}$$

Proof. We now look for stationary points of $N_{I/O}$. These are values of τ at which the first derivative of $N_{I/O}$ w.r.t τ is zero.

:

$$
\begin{gathered}
\frac{dN_{I/O}}{d\tau} = \frac{T_s}{\tau^2}\left[\frac{\tau}{M}e^{\frac{R}{M}}e^{\frac{\delta+\tau}{M}} - \left(e^{\frac{R}{M}}(e^{\frac{\delta+\tau}{M}}-1)\right) - 1\right] \\
\frac{dN_{I/O}}{d\tau} = 0 \implies \\
e^{\frac{R}{M}}e^{\frac{\delta+\tau}{M}}\frac{\tau}{M} - e^{\frac{R}{M}}e^{\frac{\delta+\tau}{M}} + e^{\frac{R}{M}} - 1 = 0 \implies \\
e^{\frac{R}{M}}e^{\frac{\delta+\tau}{M}}\left(\frac{\tau}{M}-1\right) = 1 - e^{\frac{R}{M}} \implies \\
e^{\frac{\delta+\tau}{M}}\left(\frac{\tau}{M}-1\right) = -\left(1 - e^{-\frac{R}{M}}\right) \implies \\
e^{\frac{\tau}{M}}\left(\frac{\tau}{M}-1\right) = -e^{-\frac{\delta}{M}}\left(1 - e^{-\frac{R}{M}}\right) \implies \\
e^{\left(\frac{\tau}{M}-1\right)}\left(\frac{\tau}{M}-1\right) = -e^{-\frac{\delta+M}{M}}\left(1 - e^{-\frac{R}{M}}\right) \implies \\
\left(\frac{\tau}{M}-1\right) = ProductLog\left(-e^{-\frac{\delta+M}{M}}\left(1 - e^{-\frac{R}{M}}\right)\right) \implies \\
\frac{\tau}{M} = 1 + ProductLog\left(-e^{-\frac{\delta+M}{M}}\left(1 - e^{-\frac{R}{M}}\right)\right) \implies \\
\tau = M\left(1 + ProductLog\left(-e^{-\frac{\delta+M}{M}}\left(1 - e^{-\frac{R}{M}}\right)\right)\right) \implies \\
\tau = M\left(1 + ProductLog\left(-e^{-\frac{\delta+M}{M}} + e^{-\frac{R+\delta+M}{M}}\right)\right)
\end{gathered} \tag{5}
$$

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 8 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

$$\begin{aligned} N_{I/O} &= \frac{\text{Expected Execution Time}}{M} + \frac{T_s}{\tau} \\ &= \frac{T}{M} + \frac{T_s}{\tau} \\ \frac{dN_{I/O}}{d\tau} &= \frac{1}{M}\frac{dT}{d\tau} - \frac{T_s}{\tau^2} \\ \frac{d^2T}{d\tau^2} &= \frac{1}{M}\frac{d^2T}{d\tau^2} + 2\frac{T_s}{\tau^3} \end{aligned} \tag{6}$$

From [4] $\frac{d^2T}{d\tau^2}$ is known to be positive for all values of τ in the range $0 < \tau \leq M$. This makes the right hand side of Equation 6 and thus $\frac{d^2N_{I/O}}{d\tau^2}$positive for all τ in the range $0 < \tau \leq M$. Therefore, the stationary point $\tau_{I/O}$ is a minimum with respect to the number of I/O operations. □

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

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

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

$$\tau_{opt} = M\left(1 + ProductLog\left(-e^{-\frac{\delta+M}{M}}\right)\right) \tag{7}$$

$$\tau_{I/O} = M\left(1 + ProductLog\left(-e^{-\frac{\delta+M}{M}} + e^{-\frac{R+\delta+M}{M}}\right)\right) \tag{8}$$

Consider arguments to the $ProductLog$ function in Equations 7,8. They are both negative and the absolute value of the argument in Equation 7 is larger than that of Equation 8. Since $ProductLog(-1/e) = -1$ and since $ProductLog$ function is monotonically increasing in the range ($-\frac{1}{e}$ to 0).

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

□

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 3.

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. From [4], we know 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. □

3.1. Implications of Results in this Section

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. Another 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.

4. Stochastic Simulation to Validate Analytical Model

The goal of these simulations was to validate the analytical model for the number of checkpoint I/O operations. For a given set of parameters, our aim was to use stochastic simulation to investigate the variation of number of checkpoint I/O operations with varying checkpoint intervals.

We performed stochastic simulation of a 1000 node system. We considered two sets of parameters. For the first set of parameters we conducted simulations at 9 different checkpoint intervals and for the second set we considered 7 different checkpoint intervals. At each checkpoint interval we simulated 100 runs of an application with a solution time of 500 hrs and collected data on execution times and number of checkpoint I/O operations. The probability of node failures and the probability of repair of failed nodes were calculated using random numbers. Single component failures were assumed to have poisson distribution with a mean at Mean Time To Interrupt (MTTI) of the node. Repair times were assumed to have gamma distribution with a mean of 60 minutes. For each set of parameters, we conducted two kinds of simulations. The first one was stochastic simulation using a value of checkpoint latency of 5 minutes. The second kind of simulation was a Monte Carlo simulation where we assume that the checkpoint latency is a random variable that is uniformly distributed over the interval 4.75 to 5.25 minutes. This accounts for a 5% error in the estimation of checkpoint latency as 5 minutes.

We present results of the first set of parameters in Figures 1 and 2. The solid line in the figure shows the number of checkpoint I/O operations given by analytical modeling and the vertical bars give the spread of the values of aggregate

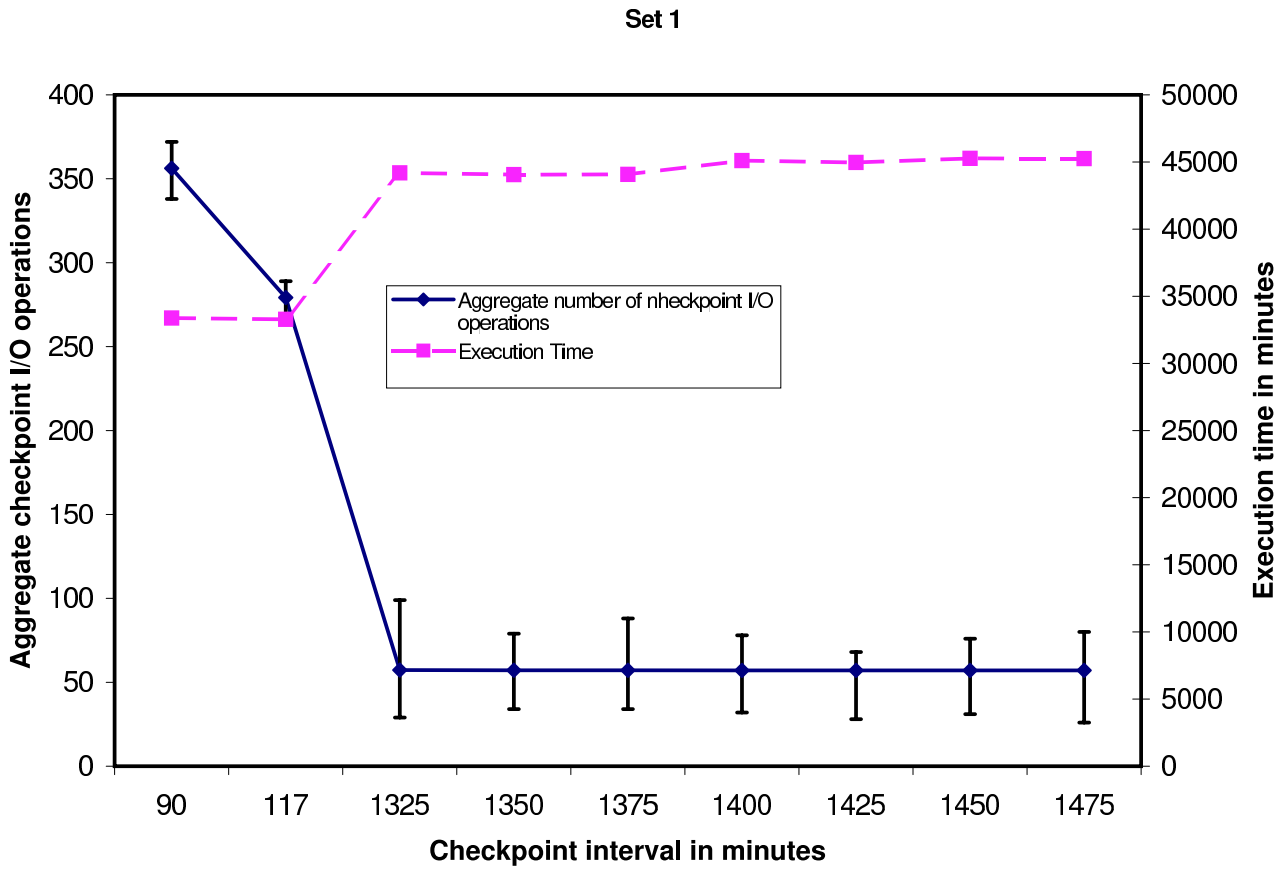

Figure 1. Simulation with M=24 hours, δ =5 minutes, T_s = 500 hours, and R= 10 minutes. For these parameters τ_{opt}=117 minutes and $\tau_{I/O}$= 1436 minutes.

checkpoint I/O operations for the 100 simulation runs corresponding to each checkpoint interval. The dashed line is the mean value of execution time. Note that, by design, the data points on the x-axis of these figures are not uniformly spaced, i.e., the first data point is 90 minutes, the second is 117 minutes and this is checkpoint interval that optimizes the execution time, the third data point is 1325. It is evenly spaced there onwards. The focus on the last 150 minutes starting from 1325 minutes was because the checkpoint interval that optimizes the number of checkpoint operations, which is 1436 minutes, lies in this range and we wanted to track changes here more closely. The simulated number of checkpoint I/O operations tracked the expected number of I/O operations given by the analytical model. The results of Monte Carlo simulation were not visually very different from the one with fixed checkpoint latency at the outset.

Thus far, using analytical modeling, we have established that using values of checkpoint interval that are in the range τ_{opt} to $\tau_{I/O}$ implies a reduction in the number of aggregate checkpoint I/O operations performed during an application execution. Next we explore candidate values of checkpoint intervals in this range that appear to be intuitive choices.

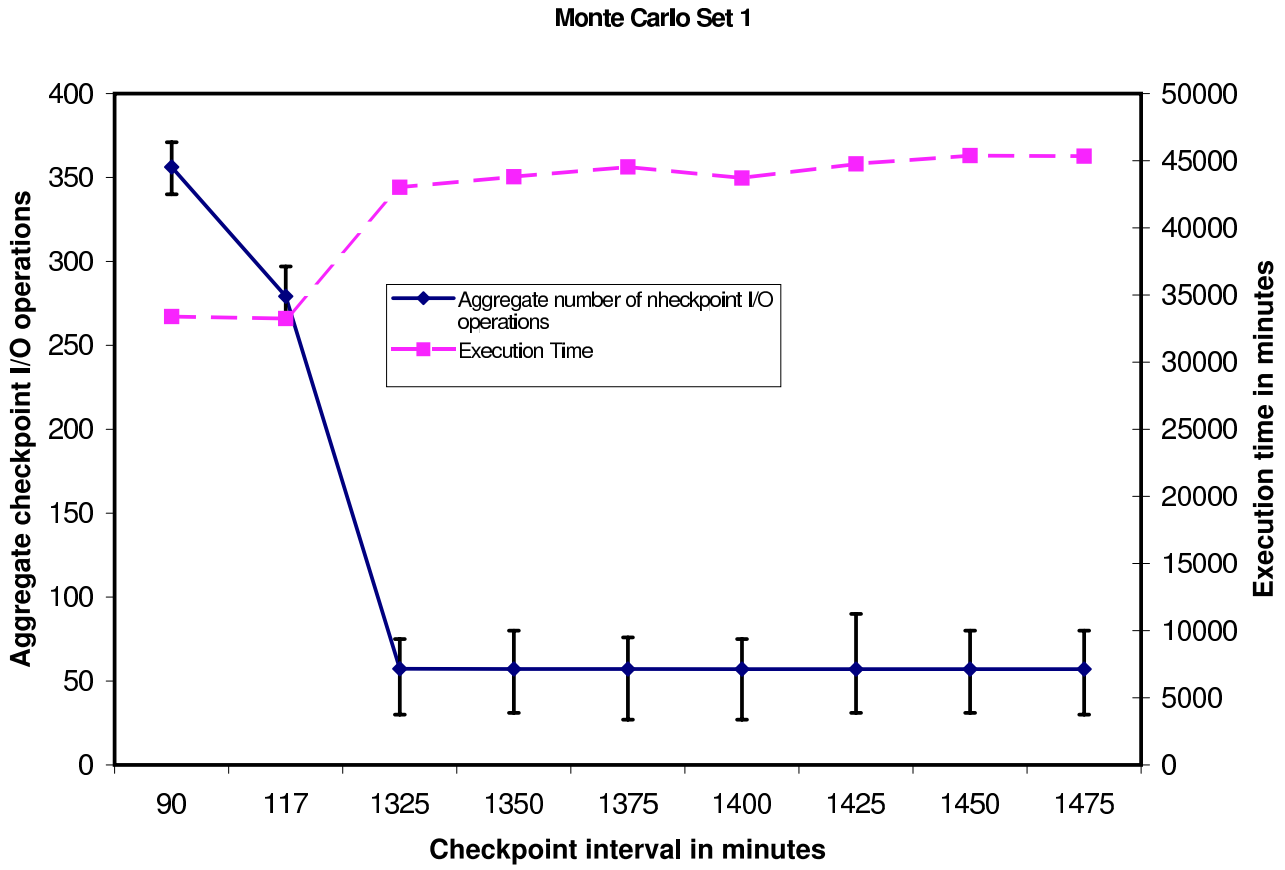

Figure 2. Monte Carlo simulation with the same parameters but with δ taking random values that are uniformly distributed in the range 4.75 minutes to 5.25 minutes

5. Potential Target Values for Increased Checkpoint Interval

As mentioned before, our point of reference is always Daly's optimal checkpoint interval, τ_{daly}. We first attempt to provide easily computable values of increased checkpoint interval with known bounded increase in expected execution times.

5.1. Exploiting Approximation Error of Daly's Optimal Checkpoint Interval

The first question we asked is whether τ_{daly} is always less than τ_{opt}. Suppose τ_{daly} is always less than τ_{opt}, then,

- Probably, there is no performance benefit in using a value of checkpoint interval less than τ_{daly}.
- There are checkpoint intervals greater than τ_{daly} which could be 0 to 12% larger than τ_{daly} and the corresponding values of both frequency of checkpoint write operations as well as execution times are smaller in comparison to the values at τ_{daly}. In Figure 4, this corresponds to values of τ in the range τ_{daly} to $\tau_{daly} + \alpha_{opt}$.
- For checkpoint intervals τ, such that $\tau \geq \tau_{daly} + \alpha_{opt}$, the execution time increases and the frequency of checkpoint write operations decreases.

This lead us to examine whether τ_{daly} is always less than τ_{opt}. We were able to prove that it is indeed so.

Theorem 3. *For any value of MTTI, M, and δ such that $0.01 \leq \frac{\delta}{2M} < 1$, the value of the optimal checkpoint interval computed using Equation 3, τ_{daly}, is less than the real optimal checkpoint interval, τ_{opt}.*

Proof. Proof of the theorem can be found in the appendix. □

Therefore, τ_{daly} and τ_{opt} relate to each other as depicted in Figure 4. For a given value of τ_{daly} and the remaining checkpointing parameters, the value of α_{opt} can be computed using Lambert W function. Although there are mathematical packages like MATLAB and Maple that provide the Lambert W function, not all math calculators provide it as an elementary function. Besides, it was suspected that computing a closed form expression for an approximation of α_{opt} in terms of elementary functions of the other parameters could possibly provide additional insights. The following theorem presents a closed form expression for an approximation of α_{opt}. The value thus obtained is always greater than the value of α_{opt}.

Theorem 4.

$$\alpha_{opt} = \left(\frac{2M}{\tau_{daly}} \left[(M - \tau_{daly}) - Me^{-(\tau_{daly}+\delta)/M} \right] \right)$$

Proof. Let

$$T_1 = Me^{R/M}(e^{(\tau_{daly}+\delta)/M} - 1)\frac{T_s}{\tau_{daly}}$$

and

$$T_2 = Me^{R/M}(e^{(\tau_{daly}+\alpha_{opt}+\delta_2)/M} - 1)\frac{T_s}{\tau_{daly}+\alpha_{opt}}$$

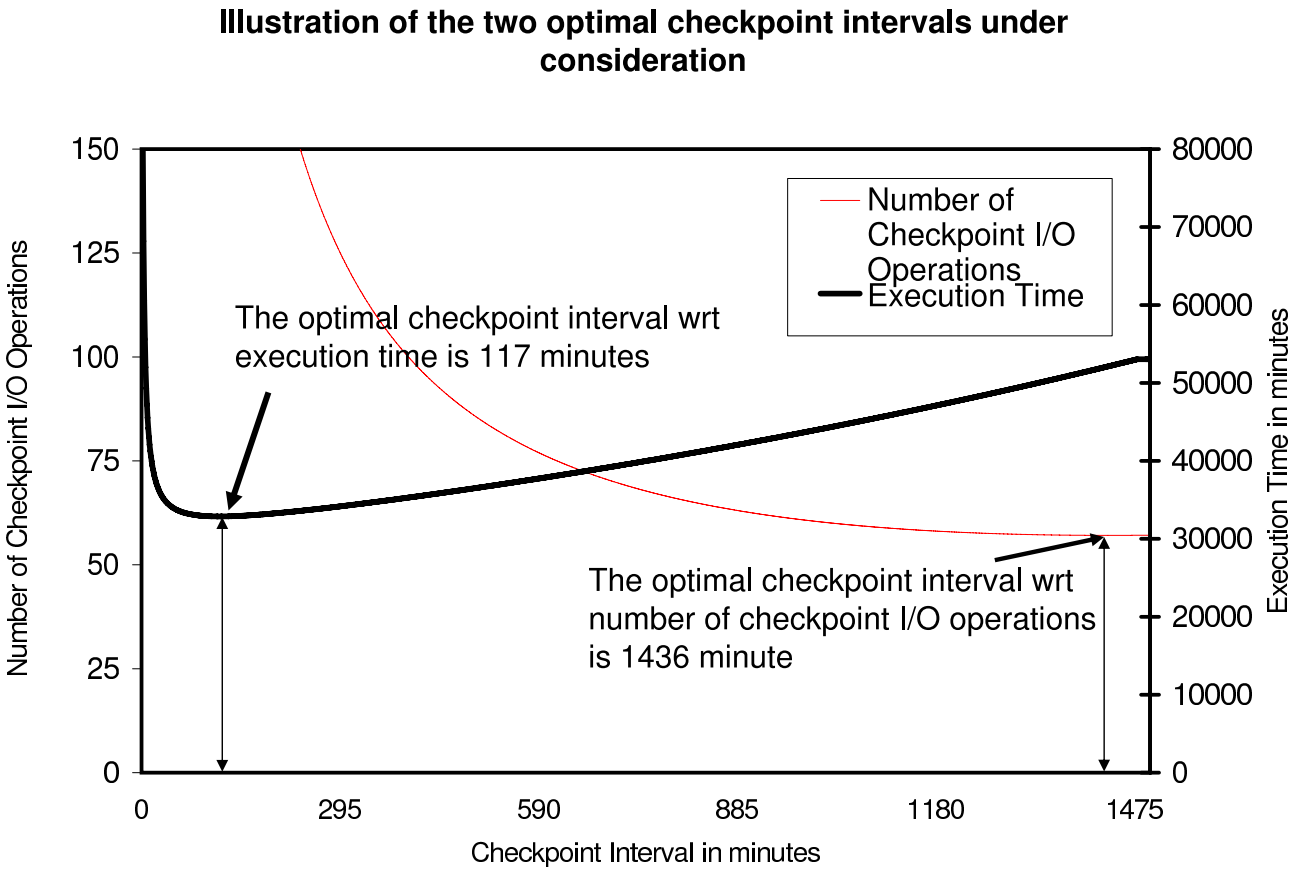

Figure 3. Illustration of variation of Execution Time and the Number of Checkpoint I/O Operations for a system with parameters M=24 hours, δ =5 minutes, T_s = 500 hours, and R= 10 minutes

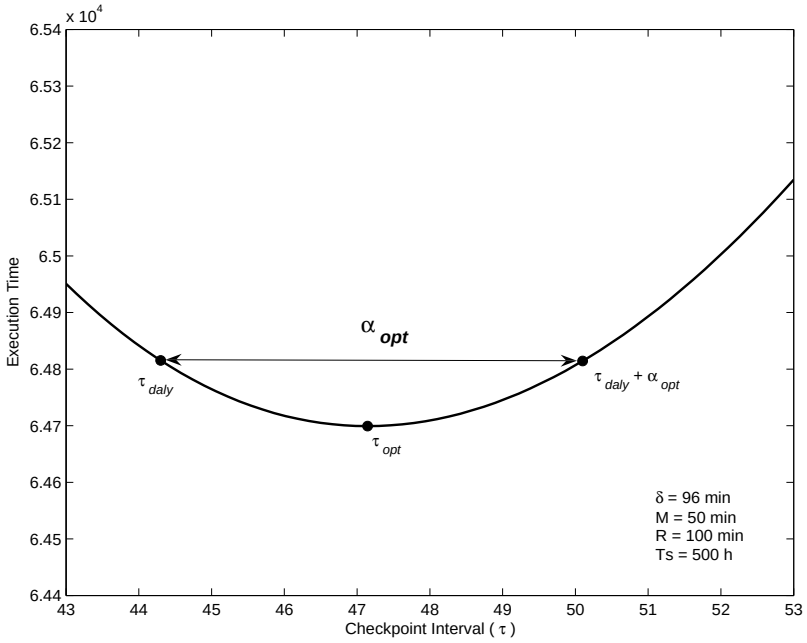

Figure 4. The optimal checkpoint interval τ_{daly} less than the real checkpoint interval τ_{opt}

Since $T_1 = T_2$,

$$\left(Me^{R/M}(e^{(\tau_{daly}+\delta)/M} - 1)\frac{T_s}{\tau_{daly}}\right) =$$

$$\left(Me^{R/M}(e^{(\tau_{daly}+\alpha_{opt}+\delta)/M} - 1)\frac{T_s}{\tau_{daly}+\alpha_{opt}}\right)$$

Simplifying this expression

$$\tau_{daly}e^{(\tau_{daly}+\delta)/M}(e^{\alpha_{opt}/M} - 1) = \alpha_{opt}(e^{(\tau_{daly}+\delta)/M} - 1) \tag{9}$$

The Taylor series expansion for $e^{\alpha_{opt}/M}$ is

$$e^{\alpha_{opt}/M} = 1 + \frac{1}{1!}\cdot\frac{\alpha_{opt}}{M} + \frac{1}{2!}\cdot\frac{\alpha_{opt}^2}{M^2} + \frac{1}{3!}\cdot\frac{\alpha_{opt}^3}{M^3} + \dots$$

Substituting this series for $e^{\alpha_{opt}/M}$ in Inequality 9 we get

$$\tau_{daly}e^{(\tau_{daly}+\delta)/M}\left(\frac{1}{1!}\cdot\frac{\alpha_{opt}}{M} + \frac{1}{2!}\cdot\frac{\alpha_{opt}^2}{M^2} + \frac{1}{3!}\cdot\frac{\alpha_{opt}^3}{M^3} + \dots\right) =$$

$$\alpha_{opt}\left(e^{(\tau_{daly}+\delta)/M} - 1\right)$$

Considering terms up to the quadratic term in the Taylor's series and ignoring higher order terms and simplifying, we obtain

$$\alpha_{opt} = \frac{2M}{\tau_{daly}} \left[(M - \tau_{daly}) - Me^{-(\tau_{daly}+\delta)/M} \right] \tag{10}$$

□

In obtaining the expression for α_{opt} in the above theorem, a truncation error is introduced because we consider the first three terms of the Taylor's series expansion and ignore the rest of the terms. We prove that the consequent error in the estimated value of α_{opt}, is bounded by $((0.0024 * \alpha_{opt}) + (0.0048 * M))$ [2].

Table 1 shows values of τ_{daly} and α_{opt} computed for a few pairs of values of δ and M. The expression for the approximation of α_{opt} and the data in the table show that the value of α_{opt} increases as δ approaches $2M$. For values of δ close enough to $2M$, α_{opt} attains the value of 12%. In today's MPP systems δ is still much smaller than M. However, if the size of MPP systems increase and values of MTTI/node stay the same, then, the value of M reduces. In these larger systems, if the storage bandwidth increases proportionally with the size of the systems, then, values of δ can be expected to stay the same. Considering these two factors, it is not unrealistic to assume that unless there are big strides to improve reliability, δ can become comparable to $2M$ in future large systems.

Table 1. Values of α_{opt} as a percentage of τ_{daly}

Checkpoint Latency in minutes	**M in minutes**	**Optimal Checkpoint Interval τ_{daly} in minutes**	**α_{opt} as a percentage of τ_{daly}**
5	10	6.94	1.15
6	3.5	3.10	10.32
10	25	16.19	0.74
20	15	12.98	6.39
45	25	22.18	11.41
70	40	35.44	10.79
96	50	44.43	12.94
120	65	57.71	11.97

Given a checkpoint latency, δ, an MTTI, M, an application with solution time, T_s, and restart time R, let the minimum achievable execution time of the application with checkpointing be denoted by E_{min} Thus, when we increase the checkpoint interval to $\tau_{daly} + \alpha_{opt}$, the percentage increase can potentially be as large as 12% of τ_{daly} and the new expected execution time is no larger than $1.002 * E_{min}$. This follows from the fact that the approximate optimal checkpoint interval, τ_{daly}, has a bounded relative error of 5.9% and relative error of total problem solution time less than 0.2%.

5.2. Using Daly's and Young's Model of Checkpointing

As stated earlier, in today's MPP systems, values of δ are much smaller than M and, therefore, α_{opt} is not expected to be large. Therefore we explore other possible target values of checkpoint intervals that yield bounded increase in execution times. Consider Young's model of optimal checkpoint interval given by $\tau_{FO} = \sqrt{2\delta M}$, [20]. Daly's model of optimal checkpoint interval is a better approximation and it is a refinement of Young's model. Figure 5 depicts these values for a BG/L. If we increase the checkpoint interval from τ_{daly} to $\tau_{FO} + \alpha_{FO}$, since Young's model has an upper bound on the relative error in problem solution time of 5%, the expected execution time is guaranteed to be no larger than $1.05 * E_{min}$. Since $\tau_{FO} + \alpha_{FO}$ is larger than $\tau_{daly} + \alpha_{opt}$, this gives us another increment of τ with a bounded increase in expected execution time.

Given a checkpoint latency, δ, an MTTI, M, and a τ_{FO}, α_{FO} can be computed similar to α_{opt}.

$$\alpha_{FO} = \left(\frac{2M}{\tau_{FO}} \left[(M - \tau_{FO}) - Me^{-(\tau_{FO}+\delta)/M} \right] \right)$$

Thus, $\tau_{daly} + \alpha_{opt}$ and $\tau_{FO} + \alpha_{FO}$ give us easily computable values to which the checkpoint interval can be increased with known bounded increase in expected execution times.

6. Empirical Studies Using Analytical Models

In this section, we present a summary of studies performed by us to evaluate the performance impact of increasing checkpoint intervals. Using Daly's execution time model and the model presented in this paper for the number of checkpoint I/O operations, 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 2.

Table 2. 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

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.

6.1. Increase Checkpoint Interval to $\tau_{daly} + \alpha_{opt}$

For all examples considered, τ_{daly} approximates the optimal checkpoint interval very closely and therefore α_{opt}s was small, less than 5-7%. Therefore the increase in checkpoint interval was insignificant. As mentioned earlier, α_{opt} is likely to be close to 12% when δ is close to $2M$. Considering current trends, in future machines this is a distinct possibility and the technique can potentially yield better results with such machines.

6.2. Increase checkpoint Interval to $\tau_{FO} + \alpha_{FO}$

Consider increasing the checkpointing interval from τ_{daly} to $\tau_{FO} + \alpha_{FO}$, as illustrated in Figure 5. The results are summarized in Table 3 where entries with greater than 10% increase in the value of τ are emphasized. We see that there are three of them and they range in value from 14.7% to 27.2%. However, it is important to note that the increase in τ is always at least an order of magnitude larger than the increase in expected execution time. We are not sure if this is a meaningful comparison. However, this was a compelling observation. When the checkpoint interval was increased to a value of $\tau_{FO} + \alpha_{FO}$, although the guarantee offered in terms of the increase in expected execution time was that it is bounded by 5% of E_{min}, our numbers show that, except the entry for BG/L with MTTI of 2.5 years, all others have less than 1 percent increase in expected execution time.

6.3. Further Increasing Checkpoint Interval - $\tau_{5\%E_{min}}$ and $\tau_{10\%T_s}$

Due to the observation that the increase in τ is always at least an order of magnitude larger than the increase in expected execution time it looks promising to increase values of checkpoint intervals further. We considered increasing it as much as we can while limiting the increase of expected execution time to 5% of the optimum expected execution time, E_{min}, and 10% of T_s. The 5% of E_{min} bound was chosen because when we decide to increase the checkpoint interval to $\tau_{FO} + \alpha_{FO}$ we implicitly expect to have to pay a price of increasing the expected execution to $1.05 * E_{min}$. We denote the value of the checkpoint interval corresponding to an expected execution time of $1.05 * E_{min}$ by $\tau_{5\%E_{min}}$.

From conversations with scientists, we gleaned that when an application is executed with checkpointing, 10% of T_s is considered an acceptable checkpoint overhead. We therefore define $\tau_{10\%T_s}$ as the checkpoint interval at which the expected

Table 3. For a representative application with $T_s = 500$ hours and $R = 10$ minutes, the table gives the potential increase in τ and the corresponding values of increase in execution time and decrease in number of checkpoints that can be achieved using Young's model of optimal checkpoint interval and Daly's model of optimal checkpoint interval.

MPP Systems	Conditions	Increase in τ in terms of $\%$ of τ_{daly}	Increase in execution time in terms of $\%$ of E_{min}	$\%$ decrease in the number of checkpoint I/O operations
Red Storm	Standard	4	0.016	3.12
	25% of memory checkpointed	2.7	0.0057	2.46
	Size of partition is 1/8th of n_{max}	1.28	0.0007	1.27
	MTTI of a node is 2.5 years	6.08	0.049	4.43
Blue Gene/L	Standard	***14.7***	***0.43***	***5.99***
	25% of memory checkpointed	8.8	0.033	5.02
	Size of partition is 1/8th of n_{max}	3.7	0.02	2.39
	MTTI of a node is 2.5 years	***27.2***	***1.81***	***6.26***
Jaguar	Standard	5.65	0.37	3.95
	25% of memory checkpointed	3.7	0.0117	2.93
	Size of partition is 1/8th of n_{max}	1.7	0.001	1.54
	MTTI of a node is 2.5 years	8.8	0.12	5.13
Petaflop	Standard	9.14	0.132	5.18
	25% of memory checkpointed	5.8	0.041	3.78
	Size of partition is 1/8th of n_{max}	2.6	0.0042	2.1
	MTTI of a node is 2.5 years	***15.29***	***0.047***	***5.43***

execution time is $1.1 * T_s$. This means that we can increase the value of the checkpoint interval up to $\tau_{10\%T_s}$ and still be sure that the checkpoint overhead is within an acceptable bound of 10% of T_s. The concepts of $\tau_{5\%E_{min}}$ and $\tau_{10\%T_s}$ are illustrated in Figure 6. Note that $\tau_{10\%T_s}$ is meaningful only if $E_{min} < 1.1 * T_s$.

The results of increasing the checkpoint interval to $\tau_{5\%E_{min}}$ and $\tau_{10\%T_s}$, for the sixteen cases discussed earlier, are presented in Table 4. The increase in checkpoint interval is in the range of 45% to 95% of τ_{daly}. In all of these cases, there was a decrease in the expected number of checkpoint I/O operations . The reduction was in the range of 10.25% to 61.07%. Note that when a partition comprising of 1/8th of the total number of nodes was considered, 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 this next. Table 5 summarizes the results.

Increase in values of checkpoint interval to $\tau_{5\%E_{min}}$ translates to a percentage increase in the range 210% to 450% of τ_{daly}. In all 16 cases considered, the total number of checkpoint I/O operations decreases and the percentage decrease is 67.7% to 81.8%. Similarly, $\tau_{10\%T_s}$ is 140% to 555% larger than τ_{daly} and the corresponding number of total checkpoint I/O

Table 4. Improvement in τ of a representative application with $T_s = 500$ hours and $R = 10$ minutes if an execution time of $1.05 * E_{min}$ is acceptable

MPP Systems	Conditions	$\tau_{5\%Emin} - \tau_{daly}$ expressed in terms of $\%$of τ_{daly}	$\%$ decrease in the number of checkpoint I/O operations
Red Storm	Standard	100	38.94
	25% of memory checkpointed	122	46.35
	Size of partition is 1/8th of n_{max}	***195***	***61.04***
	MTTI of a node is 2.5years	80	30.48
Blue Gene/L	Standard	55	14.68
	25% of memory checkpointed	67	24.24
	Size of partition is 1/8th of n_{max}	105	39.42
	MTTI of a node is 2.5years	47	10.25
Jaguar	Standard	82	32.53
	25% of memory checkpointed	102	40.02
	Size of partition is 1/8th of n_{max}	***163***	***55.63***
	MTTI of a node is 2.5years	67.5	23.94
Petaflop	Standard	66	22.35
	25% of memory checkpointed	83	32.22
	Size of partition is 1/8th of n_{max}	***125***	***47.22***
	MTTI of a node is 2.5years	55	12.86

operations are lesser than their counterparts at τ_{daly} by 55% to 82%.

7. Summary

The trends look encouraging. When we increase the checkpoint interval to $\tau_{FO} + \alpha_{FO}$, it amounts to an increase of 1.3% to 27.2% of τ_{daly} and the corresponding decrease in the number of checkpoint I/O operations is 1.3% to 7.9%. When we increase the checkpoint interval to $\tau_{5\%E_{min}}$ and $\tau_{10\%T_s}$, it amounts to an increase in the checkpoint interval by 10.25% to 555% of τ_{daly}. The decrease in the total number of checkpoint I/O operations was in the range of 10.25% to 82%. From the empirical evidence we have, an important observation is that - The gradient of the execution time versus checkpoint interval is very small for values of checkpoint interval close to and larger than the optimal checkpoint interval, τ_{opt}. From Section 3 we know that in this region there is a range of values of τ for which the number of checkpoint I/O operations are decreasing with increasing checkpoint intervals. This fact, in combination with the observation made above, leads us to believe that this is a good place to look for checkpoint intervals in order to reduce the total number of checkpoint I/O operations while increasing the execution time minimally as compared to their values at τ_{opt} or τ_{daly}.

Table 5. 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. Suppose expected execution times of $1.05 * E_{min}$ and $1.1 * T_s$ are acceptable, the table shows increase in values of τ for a representative application with $T_s = 500$ hours and $R = 10$ minutes

MPP Systems	Number of nodes	$(\tau_{5\%Emin} - \tau_{daly})$ expressed in terms of %age of τ_{daly}	% decrease in the number of checkpoint I/O operations	$(\tau_{10\%T_s} - \tau_{daly})$ expressed in terms of %age of τ_{daly}	% decrease in the number of checkpoint I/O operations
Red Storm	1620	450	79.59	555	82.39
Blue Gene/L	8192	210	62.95	NA	NA
Jaguar	1448	360	75.46	390	77.12
Petaflop	6250	270	69.27	140	55.54

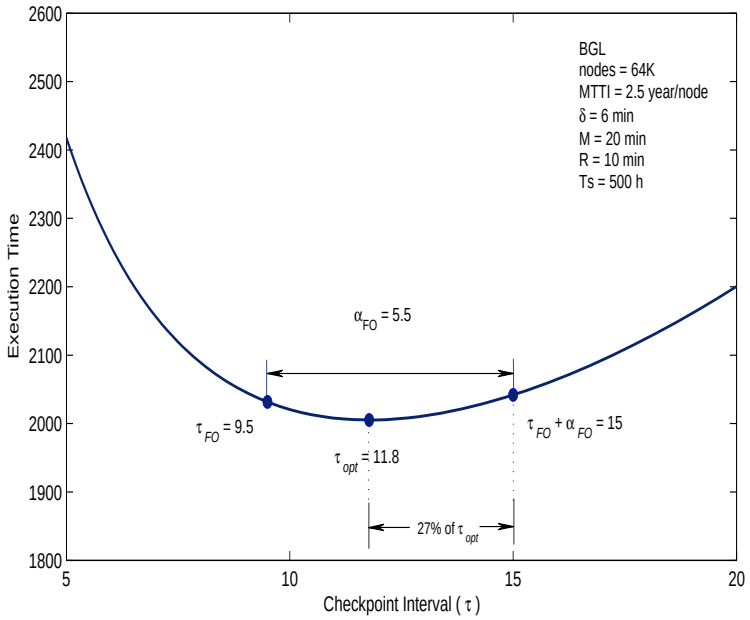

Figure 5. Illustration of the numbers computed in Table 3

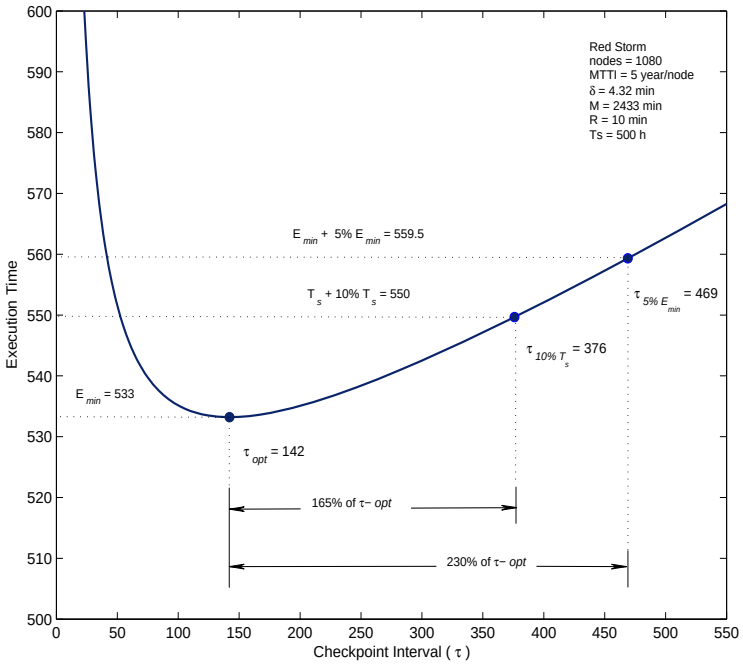

Figure 6. Illustration of the concept of $\tau_{5\%E_{min}}$ and $\tau_{10\%T_s}$

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. Certain high performance file systems have latency hiding mechanisms due to which the checkpoint latency has a small value. But as shown in [2] τ_{daly} is an increasing function of checkpoint latency and therefore the optimal checkpoint interval associated with the improved checkpoint latency is smaller than what would be the case if there was no latency hiding. This means that checkpoint write data arrives more frequently as an indirect consequence of latency hiding mechanism. However, since there is some background work that needs to be done by the filesystem in order to provide latency hiding, it has an I/O bandwidth limitation in terms of the number of I/O operations per unit of time. The filesystem may not be able to handle checkpoint I/O data that arrives once in $\tau_{opt} + \delta$ period. It might need a little more time between servicing checkpoint writes and if that is not provided the filesystem may become a bottleneck and the performance of the application goes down. This disparity occurs because in modeling optimal checkpoint interval, we assume that δ is the time taken to write checkpoint data. However, in filesystems with latency hiding mechanisms, at the end of δ time the filesystem still needs to do more processing, perhaps metadata processing, before the checkpoint write operation is taken care of completely.

2. When an overlay network is used to hide checkpoint latency, the above mentioned situation arises again but this time it is the overlay network that could cause the bottleneck.

3. When more than one checkpointing applications are executing concurrently on a large MPP system sharing I/O resources, if a high priority application's performance is deteriorating due to the large volume of checkpoint I/O of a low priority application then it 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, at this point of time.

4. When there are more than one checkpointing applications concurrently executing on a large MPP system, it might be desirable to ensure that the full I/O bandwidth of the system is 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 overlap. Conceptually, for periodic checkpointing applications there are algorithms that could help us do this. However, in case using the algorithm we determine that 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.

5. Combination of any of the above mentioned situations.

8. 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 [20]. However, Daly's extension of Young's model, a higher-order approximation, does [4]. 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 [19]. 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.

Checkpointing for computer systems has been a major area of research over the past few decades. There have been a number of studies on checkpointing based on certain failure characteristics [13], including Poisson distributions. Oldfield el.al., [8] present studies modeling the impact of checkpoints on next-generation systems. Tantawi and Ruschitzka [17] 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 [1] developed application-initiated checkpointing schemes for BG/L. There are also a number of studies reporting the effect of failures on checkpointing schemes and system performance. Most of these works assume Poisson failure distributions and fix a checkpointing interval at runtime.

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 [7, 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 [15] 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. 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 has also been explored [10, 14].

Recently, there has been a lot of research towards finding alternatives for periodic checkpointing techniques [9] and there have been some promising results. However, until these new techniques reach a level of maturity, periodic checkpointing techniques will continue to be popular methods of fault tolerance. Besides, a lot of important legacy scientific applications use periodic checkpointing and therefore issues related to periodic checkpointing still need to be addressed.

LANL's (Daly's) checkpoint model assumes an exponential failure distribution for the duration of the application run, which might be a few days, weeks, or months. In determining the value of MTTI, M, at the beginning of the application run, the failure distribution that is deemed right for the system, can be used. The model starts with that value of M and assumes that during the application run the failure distribution of the system is exponential. This makes the model mathematically amenable and elegant. Assumption of exponential failure distribution seems reasonable because we had the opportunity to see 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. It fits a Weibull distribution with shape factor 0.91/0.97. Since exponential is shape factor 1.0, it is a pretty close approximation to the real failure distribution. Due to space constraints, we do not present the plot in this paper.

9. Conclusion and Future Work

We are in the process of performing Monte Carlo simulations for varying values of both M and δ and performing statistical analysis of the outcomes, rather than visual inspection. Results from these runs could enlighten us about the impact of errors in estimation of values of M and δ.

We believe that the work presented in this paper is complementary to Daly's modeling work on execution time. 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. For example, if an application programmer needs to estimate the value of checkpoint latency and one is aware that the checkpoint latency in likely to be some value between $\delta 1$ and $\delta 2$, where $\delta 1 < \delta 2$. If this value is being used to find the optimal checkpoint interval (with respect to the execution time), should the application programmer pick a value of δ closer to $\delta 1$ or to $\delta 2$. Although the answer to this question depends on what performance metrics are being considered, in general, from insights obtained by analytical modeling we can see that it is safer to pick a value of δ closer to $\delta 2$. This is because; let us say the application programmer picks a value of δ close to $\delta 1$ then let the corresponding optimal checkpoint interval be $\tau 1_{opt}$. If due to resource contention the I/O bandwidth available to this application decreases, then its checkpoint latency might increase and the real value might be closer to $\delta 2$. The optimal checkpoint interval corresponding to this new value of checkpoint latency increases to $\tau 1_{opt}$. However the application is checkpointing at a checkpoint interval of $\tau 1_{opt}$ which is less than $\tau 2_{opt}$. From information gleaned from the analytical models, we know that we stand to lose in terms of both the execution time and the number of checkpoint operations when we use checkpoint intervals that are smaller than the optimal checkpoint interval(with respect to execution time). On the other hand if the application programmer chooses a value of δ closer to $\delta 2$, then if delta decreases and gets closer to $\delta 1$ then the optimal checkpoint interval becomes closer to $\tau 1_{opt}$. This means that the checkpoint interval being used is larger than the optimal value. As a consequence of this the wall clock execution time of the application increases as compared to its optimal value. However, the total number of checkpoint operations performed during the application execution decreases. There is a gain in at least one aspect.

In an MPP system where the runtime has a system-wide view of all the applications and has some 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 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. This is one of the crucial aspects of co-ordinating checkpointing applications running concurrently in order to achieve the goal of system performance.

Acknowledgments

- DoE, Office of Science (Grant Number DE-FG02-04ER25622)
- Sandia National Laboratories (Contract Number 579987 (PR 882412))
- AHPCRC (Grant Number W11NF-07-2-2007)

References

[1] 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.

[2] S. Arunagiri, S. Seelam, R. A. Oldfield, M. R. Varela, P. J. Teller, and R. Riesen. Impact of checkpoint latency on the optimal checkpoint interval and execution time. Technical Report UTEP-CS-07-55, University of Texas at El Paso, 2007.

[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] D. S. Greenberg, R. Brightwell, L. A. Fisk, A. B. Maccabe, and R. Riesen. A system software architecture for high-end computing. In *Proceedings of SC97: High Performance Networking and Computing*, pages 1–15, San Jose, California, November 1997. ACM Press.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

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

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

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

A. Appendix

Theorem 5. *For every value of MTTI such that $M > 0$ and every value of checkpoint latency, δ, such that ($0.01 \leq \frac{\delta}{2M} < 1$), the value of the optimal checkpoint interval, computed using Equation 3, is less than or equal to the real optimal checkpoint interval.*

Proof. For ease of representation in the context of this proof, let us define an ordered pair of values $< V1, V2 >$ as a *valid pair* if $0.02 * V1 \leq V2 < 2 * V1$ Given a valid pair of values of MTTI, M, and checkpoint latency, δ, $< M, \delta >$, let us denote the optimal checkpoint interval computed using Equation 3 by τ_{daly} Let τ_{opt} represent the real optimal checkpoint interval. Since $\tau_{daly} < M$, it belongs to one of the following partitions;

$$
\begin{aligned}
P1 &= \{\tau : 0 < \tau < \tau_{opt}\} \\
P2 &= \{\tau : \tau = \tau_{opt}\} \\
P3 &= \{\tau : \tau_{opt} < \tau \leq M\}.
\end{aligned}
$$

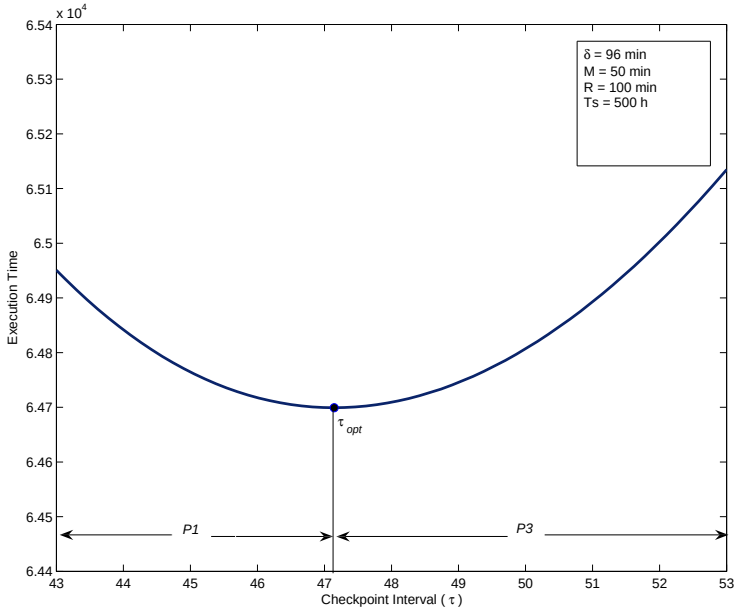

Figure 7. Illustration of a partition

The theorem asserts that $\tau_{daly} \in \{P1 \cup P2\}$

If $\tau_{daly} = \tau_{opt}$ then $\tau_{daly} \in \{P2\}$ and the theorem holds. Therefore, we only need to prove the theorem for $\tau_{daly} \neq \tau_{opt}$.

Suppose for a value of MTTI, M, and for a value of checkpoint latency, δ, we exhibit a $\triangle t > 0$ such that $[T(\tau_{daly} + \triangle t) < T(\tau_{daly})]$. The existence of such a $\triangle t$, proves that $\tau_{daly} \notin \{P3\}$, which implies that $\tau_{daly} \in \{P1 \cup P2\}$. This proves that for that pair of values of MTTI and checkpoint latency, $\tau_{daly} \in \{P1 \cup P2\}$. If we can provide a method of computing such a $\triangle t$ for any given valid pair of values of MTTI and checkpoint latency, $< M, \delta >$, then, that serves as a proof of Theorem 5. This is indeed the method we use.

In the following lemma, an alternate representation of τ_{daly} is obtained by a simple transformation of δ. It simplifies the proof of the theorem and improves readability.

Lemma 1. *If δ is represented as $2Ma$, where $0 \leq a < 1$, then, $\tau_{daly} = 2M(B - a)$, where $B = \left(\sqrt{a} + \frac{a}{3} + \frac{a^{\frac{3}{2}}}{9}\right)$.*

Proof. For this theorem, the range of values of δ of interest is $0 < \delta < 2M$. Consider the equation for τ_{daly} when $\delta < 2M$,

$$\tau_{daly} = \sqrt{2\delta M}\left[1 + \frac{1}{3}\left(\frac{\delta}{2M}\right)^{\frac{1}{2}} + \frac{1}{9}\left(\frac{\delta}{2M}\right)\right] - \delta.$$

Substituting $\delta = 2Ma$ in the above equation, we obtain

$$\begin{aligned}
\tau_{daly} &= \sqrt{4M^2a} + \frac{2Ma}{3} + \frac{2Ma^{\frac{3}{2}}}{9} - 2Ma \\
&= 2M\sqrt{a} + \frac{2Ma}{3} + \frac{2Ma^{\frac{3}{2}}}{9} - 2Ma \\
&= 2M\left(\sqrt{a} + \frac{a}{3} + \frac{a^{\frac{3}{2}}}{9}\right) - 2Ma \\
&= 2M(B - a) \; where \; B = \left(\sqrt{a} + \frac{a}{3} + \frac{a^{\frac{3}{2}}}{9}\right).
\end{aligned}$$

□

Since $\delta = 2Ma$, $(0.01 \leq \frac{\delta}{2M} < 1) \implies (0.01 \leq a < 1)$.

Example 1. *For values of parameters ($M = 1\,min$, $R = 20\,min$, $T_s = 500\,hrs$), and ($0.01 \leq a < 1$), Figures 8 and 9 are plots of $\left[T(\tau_{daly}) - T(\tau_{daly} + 10^{-5})\right]$ and $\left[log(T(\tau_{daly}) - T(\tau_{daly} + 10^{-5}))\right]$, respectively, as a function of a. In the range ($0.01 \leq a < 1$), note that the difference is always positive. This demonstrates that when $M = 1\,min$ and for all values*

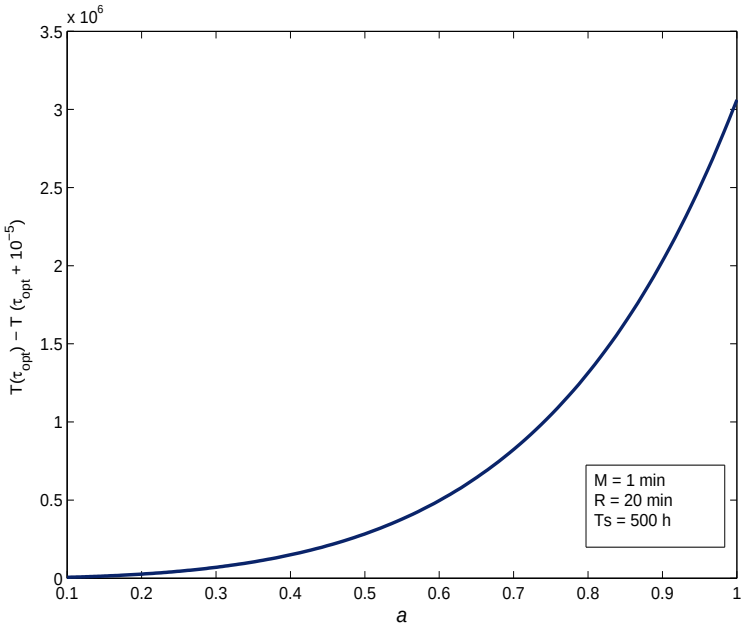

Figure 8. Exhibiting a $\triangle t$ for $M = 1\,min$

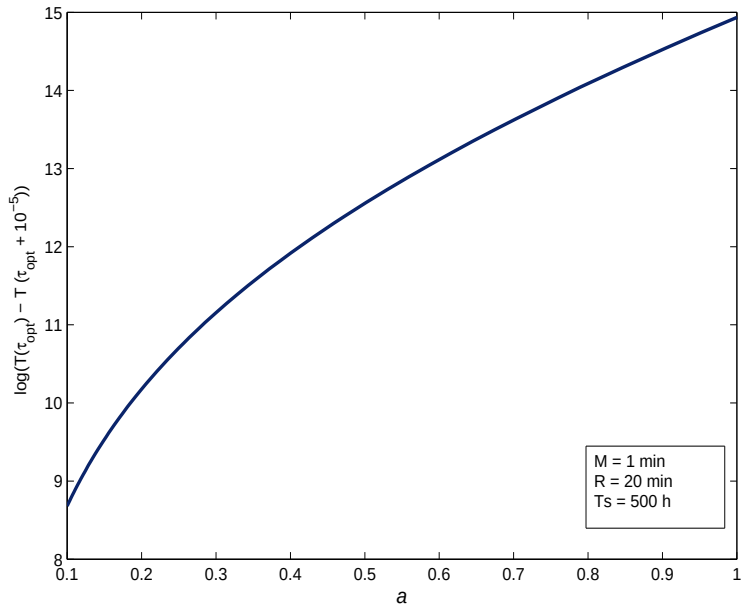

Figure 9. Difference in Execution times plotted on log-lin scale

of a in the range ($0.01 \leq a < 1$),

$$T(\tau_{daly}) > T(\tau_{daly} + 10^{-5}).$$

Thus, for $M = 1\,min$ and for all a such that $(0.01 \leq a < 1)$, we have exhibited a $\triangle t = 10^{-5}$ such that $[T(\tau_{daly}) > T(\tau_{daly} + \triangle t)]$. In order to prove the theorem, we still need to exhibit such a $\triangle t$ for every value of M and every a in the range $(0.01 \leq a < 1)$. The following lemma facilitates this.

Lemma 2. *For a given value of MTTI, M, let $\triangle t$ satisfy the property that for every δ such that ($0.01 \leq \frac{\delta}{2M} < 1$), the expected execution time corresponding to $[\tau_{daly} + \triangle t]$ is less than the expected execution time corresponding to τ_{daly}. For any other value of MTTI, $M' = f * M$, where $f > 0$, $\triangle t' = f * \triangle t$ satisfies the property that for every δ such that ($0.01 \leq \frac{\delta}{2M} < 1$), the expected execution time corresponding to $[\tau_{daly} + (\triangle t')]$ is less than the expected execution time corresponding to τ_{daly}.*

Proof.

$$Me^{R/M}T_s\left(\frac{e^{(\tau_{daly}+\delta)/M} - 1}{\tau_{daly}}\right) >$$
$$Me^{R/M}T_s\left(\frac{e^{((\tau_{daly}+\delta)/M+\triangle t/M)} - 1}{\tau_{daly} + \triangle t}\right)$$
since $Me^{R/M} > 0$ and $T_s > 0$
$$\left(\frac{e^{(\tau_{daly}+\delta)/M} - 1}{\tau_{daly}}\right) > \left(\frac{e^{((\tau_{daly}+\delta)/M+\triangle t/M)} - 1}{\tau_{daly} + \triangle t}\right)$$

From Lemma 1, τ_{daly} can be expressed as $2M(B - a)$ and $\frac{\tau_{daly}+\delta}{M} = 2B$, where a and B are defined as in Lemma 1.

$$\left(\frac{e^{(\tau_{daly}+\delta)/M} - 1}{\tau_{daly}}\right) > \left(\frac{e^{((\tau_{daly}+\delta)/M+\triangle t/M)} - 1}{\tau_{daly} + \triangle t}\right)$$

$$\Leftrightarrow \quad \left(\frac{e^{2B} - 1}{2M(B - a)}\right) > \left(\frac{e^{2B+\triangle t/M)} - 1}{2M(B - a) + \triangle t}\right)$$

$$\Leftrightarrow \quad \left(\frac{e^{2B} - 1}{2M(B - a)}\right) > \left(\frac{e^{(2B+t_1)} - 1}{2M(B - a) + M * t_1}\right)$$
where $t_1 = (\triangle t/M)$

(11)

$$\Leftrightarrow \quad \left(\frac{e^{2B} - 1}{2(B - a)}\right) > \left(\frac{e^{(2B+t_1)} - 1}{2(B - a) + t_1}\right) \tag{12}$$

Inequality 12 is equivalent to the condition that $(T(\tau_{daly}) - T(\tau_{daly} + (\triangle t))) > 0$ Note that, whether or not Inequality 12 is satisfied, is determined by the values of a and t_1. It can be verified that, if the value of MTTI changes from M to $f * M$, and if $\triangle t$ is substituted by $f * \triangle t$, then the condition $(T(\tau_{daly}) - T(\tau_{daly} + (f * \triangle t))) > 0$ turns out to be identical to Inequality 12.

□

Example 1 demonstrates that when $M = 1\,min$, a value of $\triangle t = 10^{-5}\,min$ satisfies Inequality 12 for all values of a in the range $0.01 \leq a < 1$. From Lemma 2, given any other value of MTTI, $M' = f * M$, where $f > 0$ a value of $\triangle t'$ that satisfies $[T(\tau_{daly}) > T(\tau_{daly} + \triangle t')]$ is given by $\triangle t' = \triangle t * f$. Since $M = 1\,min$, $f = M'/M = M'$, where M' is expressed in minutes.

Thus, we have presented a method of computing $\triangle t$ with the desired property for every $M > 0$ and for every δ in the corresponding relevant range. Therefore, as explained before, $\tau_{daly} \in \{P1 \cup P2\}$ is always true.

□