Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. IV (2009), No. 4, pp. 386-400

High Performance Computing Systems with Various Checkpointing Schemes

N. Naksinehaboon, M. Păun, R. Nassar, B. Leangsuksun, S. Scott

Nichamon Naksinehaboon
Louisiana Tech University
Computer Science Department
Louisiana, 71272, USA
E-mail: n.nichamon@gmail.com

Mihaela Păun
Louisiana Tech University
Mathematics and Statistics Department
Louisiana, 71272, USA
and
Finance and Banks Faculty
Spiru Haret University, Romania
E-mail: mpaun@latech.edu

Raja Nassar
Louisiana Tech University
Mathematics and Statistics Department
Louisiana, 71272, USA
E-mail: ran1@suddenlink.net

Chokchai Box Leangsuksun
Louisiana Tech University
Computer Science Department
Louisiana, 71272, USA
E-mail: box@latech.edu

Stephen Scott
Oak Ridge National Laboratory
Computer Science and Mathematics Division
TN 37831-6173
E-mail: scottsl@ornl.gov

Abstract: Finding the failure rate of a system is a crucial step in high performance computing systems analysis. To deal with this problem, a fault tolerant mechanism, called checkpoint/restart technique, was introduced. However, there are additional costs to perform this mechanism. Thus, we propose two models for different schemes (full and incremental checkpoint schemes). The models which are based on the reliability of the system are used to determine the checkpoint placements. Both proposed models consider a balance of between checkpoint overhead and the re-computing time. Due to the extra costs from each incremental checkpoint during the recovery period, a method to find the number of incremental checkpoints between two consecutive full checkpoints is given. Our simulation suggests that in most cases our incremental checkpoint model can reduce the waste time more than it is reduced by the full checkpoint model. The waste times produced by both models are in the range of 2% to 28% of the application completion time depending on the checkpoint overheads.
Keywords: Large-scale distributed system, reliability, fault-tolerance, checkpoint/restart model, HPC

1 Introduction

High performance computing (HPC) systems are used to address many challenging computational problems. However, an HPC system is not required only to solve these problems, but also to solve them in a short time. This

requirement drives the physical size of an HPC system larger. The reliability of a large HPC system is inversely proportional to its size. For example, the Blue Gene/L system hosted at Lawrence Livermore National Laboratory (LLNL), the world second fastest HPC system, has a failure rate of roughly one failure a week [1]. Unfortunately, handling the reliability of such large HPC systems is still problematic. A Checkpoint/Restart mechanism is widely used to deal with the high failure rate problem. If a failure occurs while an application is running, the portion of the application that has been already computed has to be re-computed again. The checkpoint mechanism is able to reduce this re-computing time of an application after a failure occurrence. However, there is an additional time that is spent to perform the mechanism. The detailed of the checkpoint mechanism will be provided in Section 2.1.

Since the Message Passing Interface (MPI) is one of the most popular parallel programming paradigms, we target our reliability study on the system using the MPI standard [20]. Normally, an MPI application is decomposed and executed among a group of nodes, where individual subtasks communicate through the MPI. Because of the static view of an MPI environment [17], a single node failure would cause the whole application to fail and requires an application restart. The checkpoint/restart scheme has been widely used in [2], [4], [7], [20] to address application outages. However, applications are practically checkpointed without considering the probability of failures. As such, some checkpoints are useless because, at the checkpoint time, the chance of a failure occurrence is small. Thus, it causes unnecessary checkpoint overhead. On the other hand, the re-computing time is large because applications are not checkpointed when the probability of failure is high. Ideally, a checkpoint is needed when a failure is going to occur within a certain period of time, so there are no useless checkpoints and expensive re-computing time. So we define the waste time of the checkpoint mechanism as the sum of checkpoint overhead, re-computing time and recovery time. Consequently, we aim to minimize the waste time by balancing the checkpoint overhead and the re-computing time.

In existing studies, the optimal checkpoint placement strategy is either based on the cost function models [2], [3], [20] or Markov availability models [5], [14], [19], [22]. In addition, it is typically assumed that the system failure is a Poisson process (with a fixed failure rate). However, in practice, the system failure may not always follow the Poisson model [16] and the overall system reliability is more complex than that of its individual components.

In this paper, we propose two stochastic models for improving the checkpoint/restart scheme in an HPC environment by reducing the waste time. The reliability function of the system is obtained by analyzing historical failure data from the system event log files.

2 Full Checkpoint/Restart Model

2.1 Behavior of Full Checkpoint/Restart Model

A full checkpoint/restart mechanism is a traditional checkpoint/restart mechanism which occasionally saves running application states to a local storage. After a failure occurs, the application can be recovered from the last saved state rather than from the starting point. This results in decreasing the time that is spent to re-compute the application. Conversely, there is an additional time to save the application states which is called the checkpoint overhead. To improve the checkpoint mechanism, checkpoints should not be performed too frequently, in such a way to balance of the checkpoint overhead and the application re-computing time. Thus, we focus on how to determine checkpoint placements or intervals that minimize the waste time.

In this section, we present a checkpoint/restart model with the re-computing time coefficient for fault-tolerant parallel applications. We assume that the application model is MPI and supports a coordinated checkpoint mechanism. The characteristic of the model is directly related to the system reliability function where one of the node outage will result in an application outage because of the MPI standard. For a parallel application, the coordinated checkpoint protocol guarantees that the checkpoint of an HPC system, resulted by the synchronization of the local checkpoint on individual process, is consistent. As a result, each process is checkpointed at almost the same time, so we assume that there is no time difference for each individual process checkpoint, and treat it as a single checkpoint.

We consider a failure model that allows more than one failure during the lifetime of a given application. Moreover, after each failure, the application will be restarted from the last checkpoint. Our checkpoint/restart model is shown in Figure 1. It follows a renewal reward process in which ω_i denotes the i^{th} time between failures in each repeated cycle. We assume that the waste time (checkpoint overhead, recovery time, and re-computing time) of each cycle is a random variable,W_1, W_2, W_3..., since it depends on when a failure occurs. Hence, the total waste time may be expressed as

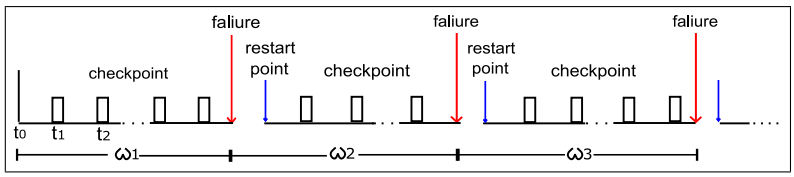

Figure 1: Checkpoint/restart as a stochastic renewal reward process

$$C_t = \sum_{i=1}^{m} W_i, \tag{1}$$

where $m = \max\{n \in \{1, 2, 3, ...\} | \ (\sum_{i=1}^{n} \omega_i) \leq t\}$, and C_t is called a renewal reward process.

From [15] the theorem of a renewal reward process is given as

$$\lim_{t \to \infty} \frac{E\left[\sum_{i=1}^{m} W_i\right]}{t} = \frac{E\left[W_1\right]}{E\left[\omega_1\right]} \tag{2}$$

In the checkpoint/restart model, Eq.(2) shows that the mean of the overall waste time (left hand side of the equation) can be expressed as a function of the mean waste time of the 1^{st} cycle. This means that minimizing the overall time lost is equivalent to minimizing the waste time in the 1^{st} cycle.

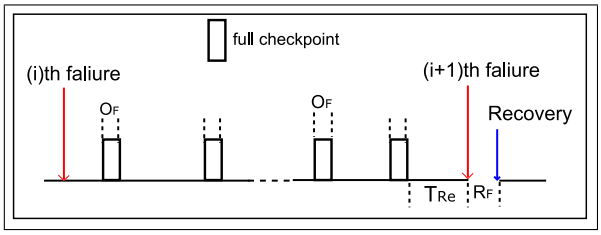

Figure 2: Checkpoint/restart as a stochastic renewal reward process

Table 1: Parameters of the Checkpoint/Restart Model

Parameters	Meaning
O_F	Checkpoint Overhead
R_F	Recovery Time
T_{Re}	Re-computing Time
$n_F(t)$	Checkpoint frequency function of the full checkpoint model
$f(t)$	Probability density function of TBF
ω_i	The cycle between failure i and failure (i+1)

Consider a checkpoint restart scheme in the first cycle ω_1 (time between two failures). Figure 2 illustrates the model with parameters such as checkpoint overhead O_F (time that is spent to save an application state), recovery time R_F (time that is spent to load the application saved state) and re-computing time T_{Re}. Through the rest of the paper, our failure model is based on the following assumptions:

1. A running application may be interrupted by a series of random transient failures where the time between failures has a certain probability density function (PDF), $f(t)$.
2. The system failure can be detected by a monitoring mechanism, and we assume that there is no failure during the re-computing and recovery time.
3. Each checkpoint overhead O_F is a constant. In practice, we can take this constant to be the average value of multiple checkpoint overheads.

4. The application can be recovered from the last checkpoint. This implies that the re-computing time T_{Re} is a period between the last checkpoint and the present failure.
5. The recovery time R_F is a constant.

Remark: Assumption 2 is satisfied since a well-managed system can be engineered with an efficient mechanism to immediately detect the failure. Assumption 5, R_F, is satisfied if there is a mechanism in place to replace the failed node with a spared node.

The objective of this model is the capability of giving the best checkpoint placement sequence that minimizes the total waste time

Definition 1. Let the sequence of discrete checkpoint placements be $0 = t_0 < t_1 < ... < t_n$, and let $n_F(t)$ be the checkpoint frequency function for the full checkpoint model defined by: $\int_a^b n_F(t)dt$ =the number of checkpoints from time a to time b. We then can imply that $\int_{t_i}^{t_{i+1}} n_F(t)\,dt = 1$.

In Figure 2, the waste time W_i (checkpoint overhead, re-computing time, and recovery time) in a given cycle ω_i can be expressed as

$$W_i = O_F \int_0^{\omega_i} n_F(\tau)\,d\tau + T_{Re} + R_F \tag{3}$$

From assumption 5, R_F is a constant, and, from assumption 5, we suppose that the system can be successfully recovered from the last checkpoint. The relationship between re-computing time T_{Re} and checkpoint interval is illustrated in Figure 3.

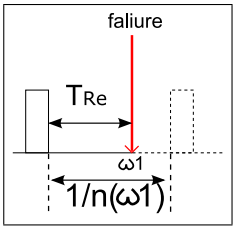

Figure 3: The relationship between rollback T_{Re} and checkpoint interval

Ling et al. [9] and Ozaki et al. [11] considered the recovery cost including both re-computing time and recovery time similar to those in our model. They represented their recovery cost by a function, called the recovery cost function. Moreover, they illustrated their model with respect to recovery cost by assuming the recovery function to be linear. Assuming linearity may be restrictive and may not lead to optimality. In our model, we consider a re-computing time coefficient k $(0 < k < 1)$ instead of a recovery cost function and propose an algorithm (end of Section 2.2) to estimate this re-computing time coefficient. The re-computing time coefficient is general and can be determined for any system failure distribution. This makes our approach useful for application purposes.

Since ω_i is the value between these checkpoint placements, by the Mean Value theorem, we can estimate the frequency of this interval by $n_F(\omega_i)$. Therefore, T_{Re} can be approximated by Eq.(4), where k is a re-computing time coefficient variable between (0,1), as seen in Figure 3.

$$T_{Re} \approx k/n_F(\omega_i), k \in (0,1) \tag{4}$$

Replacing T_{Re} in Eq.(3) by its value from Eq.(4) gives:

$$W_i = O_F \int_0^{\omega_i} n_F(t)\,dt + \frac{k}{n_F(\omega_i)} + R_F \tag{5}$$

According to the theorem of a renewal reward process in Eq.(2), the total waste time in the checkpoint and recovery process can be minimized by minimizing $E(W_1)$. Let $f(t)$ be the probability density function of time between failures. Then, the probability that the system fails in the interval $[t, t+\Delta t]$ is $f(t)\cdot\Delta t$. The expected waste time during a cycle in the checkpoint/restart process is

$$E[W_1] = \int_0^{\infty} \left(O_F \int_0^t n_F(\tau)\,d\tau + \frac{k}{n_F(t)} + R_F \right) \cdot f(t)\,dt. \tag{6}$$

Our interest is in determining the optimal checkpoint frequency $n(t)$ in order to minimize the expected waste time as defined by Eq.(6).

Solution: Letting $x(t) = \int_0^t n_F(\tau)\,d\tau$, then $x'(t) = n_F(t)$ and Eq.(6) becomes

$$E[W_1] = \int_0^\infty \left(O_F \cdot x(t) + \frac{k}{x'(t)} + R_F\right) \cdot f(t)\,dt. \tag{7}$$

Let $\Phi(x,x',t) = \left(O_F \cdot x(t) + \frac{k}{x'(t)} + R_F\right) \cdot f(t)$.

In order to have an extremum, the necessary condition for Eq.(7) requires that the variation of $E[W_1]$ (the first derivative regarding to the function x of $E[W_1]$) to vanish. Consequently, the function $\hat{\Phi}$ must satisfy the Euler's equation as the following [6].

$$\frac{\partial \Phi}{\partial x} - \frac{d}{dt} \cdot \frac{\partial \Phi}{\partial x'} = 0. \tag{8}$$

Since $\frac{\partial \Phi}{\partial x} = O_F \cdot f(t)$ and $\frac{\partial \Phi}{\partial x'} = -\frac{k \cdot f(t)}{(x'(t))^2}$, Eq.(8) becomes:

$$O_F \cdot f(t) + \frac{d}{dt} \cdot \frac{k \cdot f(t)}{(x'(t))^2} = 0 \tag{9}$$

By integrating from 0 to t Eq.(9) on both sides, the result is:

$$O_F \cdot F(t) + \frac{k \cdot f(t)}{(x'(t))^2} = C \tag{10}$$

Since $\lim_{t\to\infty} F(t) = 1$and $\lim_{t\to\infty} f(t) = 0$, $C = O_F$. Moreover, $x'(t) = n_F(t)$, then we obtain the optimal checkpoint frequency function in Eq.(11)

$$n_F(t) = \sqrt{\frac{k}{O_F}} \cdot \sqrt{\frac{f(t)}{1 - F(t)}}, \qquad k \in (0,1) \tag{11}$$

It is worth mentioning that the probability density function (PDF) and the cumulative distribution function (CDF) can be the joint PDF and joint CDF where each corresponding marginal is the failure distribution of each node in an HPC system.

2.2 Estimation of the Re-computing-time Coefficient k

In Figure 4, T_{Re} is the re-computing time of the application recovered after the failure. It is the time interval between the last checkpoint and the failure, which is a random variable depending on the time when the failure occurs from the checkpoint placement.

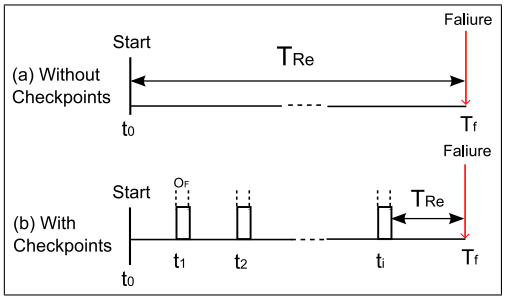

Figure 4: T_{Re} (a) without checkpoint, and (b) with checkpoint

In an application without checkpoints (Figure 4(a)), if a failure occurs at time T_f, then $T_{Re} = T_f - t_0$. With checkpoints (Figure 4(b)), it is obvious that T_{Re} is a random variable which depends on the time the failure occurs. Therefore, if we know the distribution of the time between failures, then T_{Re} can be estimated.

Definition 2. The re-computing time coefficient k is the ratio between the re-computing time and the checkpoint interval in which a failure occurs. As such,

$$k = \frac{T_{Re}}{t_{i+1} - t_i} = \frac{T_f - t_i}{t_{i+1} - t_i}$$

To estimate k, we first obtain the expected re-computing time for each checkpoint interval.

Definition 3. Excess life is a random variable, $S \geq 0$, which denotes system survival until time $t+S$ given that it survives until time t. We denote the CDF, the PDF, and the expected value of the excess life S as

$$\begin{aligned} F(t+s|t) &= P\left(T_f < t+S | T > t\right), \\ f(t+s|t) &= \tfrac{dF(t+s|t)}{ds}, \\ E[S] &= \textstyle\int_0^\infty s f(t+s|t)\, ds. \end{aligned} \tag{12}$$

In our checkpoint model, each checkpoint time t_i is the time that we expect a failure to occur. The re-computing time during the interval (t_i, t_{i+1}), T_{Rei} is a random variable such that its value is in the interval $(0, t_{i+1} - t_i)$. According to the excess life definition, the expected value of the re-computing time can be calculated as

$$E[T_{Rei}] = \int_0^{t_{i+1}-t_i} s f(t_i+s|t_i)\, ds \Big/ \int_0^{t_{i+1}-t_i} f(t_i+s|t_i)\, ds. \tag{13}$$

Therefore, for the expected k of the i^{th} checkpoint interval, $\bar{k}_i$, we obtain

$$\bar{k}_i = E[T_{Rei}] \,/\, (t_{i+1} - t_i). \tag{14}$$

Hence, the expected k, $\bar{k}$, can be express as

$$\bar{k} = \sum_{i=1}^{N} P_i \bar{k}_i \Big/ \sum_{i=1}^{N} P_i, \tag{15}$$

where $P_i = P\left(t_i < T_f < t_{i+1} | T_f > t_i\right)$ and N is the number of the checkpoints.
To estimate k iteratively, we assume an initial value $\hat{k}$ between 0 and 1. We then calculate the corresponding checkpoint sequence, $t_1, t_2, ..., t_N$, from Eq.(11). Next, we calculate $\bar{k}$ corresponding to the checkpoint sequence using Eqs. (13)(14)(15). We repeat the above procedure by varying $\hat{k}$ until we obtain a $\bar{k}$ value that is equal to $\hat{k}$.

Algorithm to estimate k
STEP 1: Assume $\hat{k} = a$, $a \in (0,1)$ and set $t_0 = 0$

STEP 2: Calculate the checkpoint sequence $t_1, t_2, ..., t_N$ corresponding to $\hat{k}$ from Step 1.

STEP 3: Calculate $\bar{k}$ from Eqs. (13)(14)(15) using the sequence in Step 2.

STEP 4: IF $\hat{k} = \bar{k}$, THEN set $k = \hat{k} = \bar{k}$ DONE
ELSE repeat Step 1.

2.3 Full Checkpoint Model Evaluation

Full Checkpoint Model for the Exponential Distribution

By substituting $f(t) = \lambda e^{-\lambda t}$, and $1 - F(t) = e^{-\lambda t}$ in Eq.(11), the checkpoint frequency function for the exponential distribution with the failure rate λ, according to Eq.(11), is given by

$$n_F(t) = \sqrt{\frac{k}{O_F}} \cdot \sqrt{\lambda}. \tag{16}$$

By the definition of the checkpoint frequency function, for i=0, 1, 2... we have that $\int_{t_i}^{i+1} n_F(t)\, dt = 1$, (19)

$$\int_{t_i}^{i+1} \sqrt{\frac{k}{O_F}} \cdot \sqrt{\lambda}\, dt = 1, \tag{17}$$

$$t_{i+1} = \sqrt{\frac{O_F}{k}} \cdot \sqrt{\frac{1}{\lambda}} + t_i. \tag{18}$$

Using induction when$t_0 = 0$, the sequence of the optimal checkpoint placements for the exponential distribution with failure rate λ is given by

$$t_i = \sqrt{\frac{O_F}{k}} \cdot \sqrt{\frac{1}{\lambda}}, \qquad k \in (0,1) \tag{19}$$

where t_i is the i^{th} checkpoint placement.

The checkpoint interval can be obtained by calculating the formula $t_i - t_{i-1}$, where $i \in \{1, 2, ...\}$ and $t_0 = 0$. Then the checkpoint interval for the exponential distribution is expressed as $\sqrt{\frac{k}{O_F}} \cdot \sqrt{\frac{1}{\lambda}}, k \in (0,1)$. Therefore, according to the proposed model for the exponential distribution, the checkpoint interval is a constant. This reflects the fact that if failures follow the exponential distribution, the failure rate is a constant.

Estimation of the Re-computing Time Coefficient k for the Exponential Distribution

According to our re-computing time coefficient estimation in Section 2.2, let $\hat{k} = a$, $0 < a < 1$ and let its corresponding checkpoint time sequence be $\{t_i\}_{i=0}^{N}$.

The CDF, PDF, and expected value of the excess life following an exponential distribution can be derived from Eq.(12) to give

$$F(t_i + s) = P(t_i + s | t_i) = 1 - e^{-\lambda t}, \tag{20}$$

$$f(t_i + s) = \lambda e^{-\lambda t}, \tag{21}$$

$$E[T_{Rei}] = 1 - \left(\sqrt{\frac{k}{O_F}} \sqrt{\lambda} \Big/ 1 - e^{-\lambda \sqrt{\lambda} \sqrt{\frac{O_F}{k}}} \right) \tag{22}$$

By substituting Eq.(22) into Eq.(14), we obtain the expected k of the i^{th} checkpoint interval ($\bar{k}_i$) and for the expected $\bar{k}$ we use Eq.(15).

3 Incremental Checkpoint/Restart Model

3.1 Behavior of the Incremental Checkpoint/Restart Model

Although full checkpoint/restart mechanism helps to reduce the overall waste time of an application in the case of a failure taking place, we have to spend some time to save the application states, so-called checkpoint overhead. If the checkpoint overhead can be reduced, we may have a cheaper waste time. Incremental checkpoint mechanism was introduced to reduce the checkpoint overhead by saving the pages that have been changed instead of saving the whole process [12], [8], [13],[18] .

In the incremental checkpoint scheme in Figure 5, the first checkpoint is typically a full checkpoint. After that, the mechanism determines which pages have changed since the last checkpoint and saves only those pages and repeats this process until another full checkpoint is performed. In order to recover the application, we will load a saved state from the last full checkpoint and load the changed pages from each incremental checkpoint following the last full checkpoint. This results in more expensive recovery cost than the recovery cost of the full checkpoint mechanism. Thus, finding the number of incremental checkpoints between two consecutive full checkpoints that balances the recovery cost and the total checkpoint overhead is crucial. This is because too many incremental checkpoints will lead to unnecessary recovery cost. A challenge in achieving minimum overhead using incremental checkpointing schemes is to find a maximum number of incremental checkpoints while maintaining

lower waste time than traditional checkpoint mechanism. The behavior of incremental checkpoint/restart model is illustrated in Figure 5.

The incremental checkpoint model consists of two types of checkpoints (full checkpoints and incremental checkpoints). The meaning of each parameter in the incremental checkpoint/restart model is listed in Table 2.

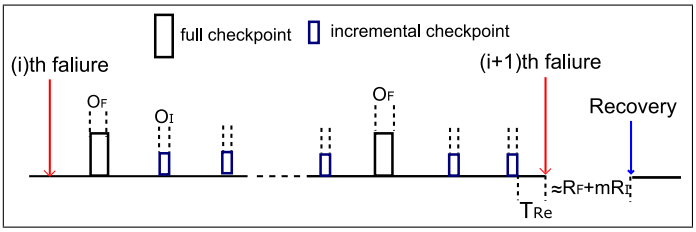

Figure 5: Behavior of Incremental checkpoint/restart model

Table 2: Parameters in Incremental Checkpoint/Restart Model.

Parameters	**Definitions**
O_F	Full Checkpoint Overhead.
O_I	Incremental Checkpoint Overhead.
T_b	Re-computing time
R_I	Recovery time per an incremental checkpoint.
R_F	Recovery time per a full checkpoint
m	Number of incremental checkpoints between two consecutive full checkpoints.
ω_i	The cycle between failure (i-1) and failure i where i = 1,2,3,...
$n_I(t)$	Checkpoint frequency function for the incremental checkpoint model

In our incremental checkpoint model, the recovery cost is decided by the number of incremental checkpoints. After m incremental checkpoints are performed, either another incremental checkpoint or a full checkpoint can be performed. A full checkpoint is chosen if the cost of performing a full checkpoint is cheaper than the recovery cost for an incremental checkpoint. This is what we call a breakeven point. The main idea is to balance a cost saving function with full and incremental checkpoint overheads and the complexity of the recovery that is introduced by the incremental model.

The incremental checkpoint model is an extension of the above full checkpoint model, so the assumptions of the full checkpoint model are applied to this incremental checkpoint model as well. However, there are additional assumptions regarding the factors in the incremental checkpoint scheme which are listed as follows:

1. The first checkpoint in an application is a full checkpoint. After an application is recovered from failure, the first checkpoint is a full checkpoint as well. After m consecutive incremental checkpoints, a full checkpoint may be performed if the overall cost reaches a breakeven point between incremental and full checkpoint. We will determine the value of m in the next section.

2. The incremental checkpoint overhead (O_I) may be viewed as an average of all incremental checkpoint overhead performed. Although, for an application, there are both small and large incremental checkpoint overheads, this assumption is reasonable because we aim to minimize the waste time caused by the incremental checkpoint mechanism of an application.

3. The recovery cost of the incremental checkpoint (R_I) and the number of incremental checkpoint between two consecutive full checkpoints m is a constant. The evaluation of m is given in Section 3.3.

In future work, we will extend the model to consider the incremental checkpoint overhead and the number of incremental checkpoints m as functions of time, in order to better represent a realistic scenario.

Definition 4. Let the sequence of discrete checkpoint placements be $0 = t_0 < t_1 < ... < t_n$, and the checkpoint frequency function for the incremental checkpoint model denoted by $n_I(t)$ is defined by $\int_a^b n_I(t)\,dt$ =the number of full and incremental checkpoints from time a to time b.

We still can imply that $\int_{t_i}^{t_{i+1}} n_I(t)\,dt = 1$. In Figure 5, the total number of checkpoints in cycle ω_i is

$$\int_0^{\omega_i} n_I(t)\,dt = N_F + N_I \tag{23}$$

where N_F is the number of full checkpoints in cycle ω_i and N_I is the number of incremental checkpoints in the same cycle ω_i.

We note that $N_I \approx mN_F$ and $N_F \approx \int_0^{\omega_i} n_I(t)\,dt$. Thus, $N_F = \frac{1}{m+1}\int_0^{\omega_i} n_I(t)\,dt$ where m is the number of incremental checkpoint between two consecutive full checkpoints.

We recall that the checkpoint procedure is a renewal process. Therefore, whenever a failure occurs, the new cycle starts. We follow the renewal reward theory to derive the optimal incremental checkpoint/restart model similarly to the full checkpoint model.

Therefore, to minimize the overall waste time of the incremental checkpoint model, it is sufficient to find the checkpoint frequency function $n_I(t)$ that minimizes only the waste time of the first cycle which consists of full checkpoint overhead, incremental checkpoint overhead, recovery time, and re-computing time in the first cycle. The waste time of the first cycle can be expressed as

$$W_1 = O_F \frac{1}{m+1}\int_0^{\omega_i} n_I(\tau)\,d\tau + O_I \int_0^{\omega_i} n_I(\tau)\,d\tau + (R_F + mR_I) + T_{Re}. \tag{24}$$

We suppose that the system can be successfully recovered from the last checkpoint, and the rollback cost T_{Re} can be estimated by $k/n_I(\omega_1), (0 < k < 1)$ as in the full checkpoint model, where $n_I(\omega_1)$is the checkpoint frequency at time ω_1, and k can be evaluated by the similar method as in the full checkpoint scheme. Therefore, we substitute T_{Re} in Eq. (25) and obtain:

$$W_1 = \left(\frac{O_F + mO_I}{m+1}\right)\int_0^{\omega_i} n_I(\tau)\,d\tau + (R_F + mR_I) + \frac{k}{n_I(\omega_1)}. \tag{25}$$

By following the stochastic renewal reward process theory, minimizing the overall waste time is equivalent to minimizing waste time in cycle ω_1. The expected waste time during a cycle in the checkpoint process $E[W_1]$ is

$$E[W_1] = \int_0^{\infty}\left[\left(\frac{O_F + mO_I}{m+1}\right)\int_0^{\omega_i} n_I(\tau)\,d\tau + (R_F + mR_I) + \frac{k}{n_I(\omega_1)}\right].f(t)\,dt \tag{26}$$

We are now looking for the solution of the overall checkpoint frequency $n_I(t)$ to minimize Eq.(26).

Solution: Let $x(t) = \int_0^t n_I(\tau)d\tau$, then $x'(t) = n_I(t)$. From Eq. (26), we obtain:

$$E[W_1] = \int_0^{\infty}\left[\frac{O_F + mO_I}{m+1}x(t) + (R_F + mR_I) + \frac{k}{x'(t)}\right]\cdot f(t)\,dt. \tag{27}$$

Let the function under the integral in right side of Eq.(27) be$\Phi(x,x',t)$. Then

$$\Phi(x,x',t) = \left[\frac{O_F + mO_I}{m+1}x(t) + (R_F + mR_I) + \frac{k}{x'(t)}\right] f(t). \tag{28}$$

By following the same argument as in the full checkpoint model, the checkpoint frequency function of the incremental checkpoint model can be expressed as:

$$n_I(t) = \sqrt{\frac{(m+1)k}{O_F + mO_I}}\sqrt{\frac{f(t)}{1-F(t)}},\ k \in (0,1). \tag{29}$$

Practically, the incremental checkpoint mechanism is an extension of the full (regular) checkpoint mechanism in the sense that incremental checkpoints are performed additionally in order to reduce the total checkpoint overhead of the full checkpoint mechanism. Alternately, we can see that the full checkpoint mechanism is the incremental checkpoint mechanism without any incremental checkpoints. According to Eq.(11) and Eq.(29), the derived models satisfy the connection between the full and incremental checkpoint mechanisms. That is, when m is equal to 0, the incremental checkpoint frequency function, Eq.(29), becomes the full checkpoint frequency function, Eq.(11).

3.2 Estimation of the Consecutive Incremental Checkpoint Number, m

We denote the number of incremental checkpoints between two consecutive full checkpoints as m. The value of depends on the next checkpoint type, either incremental or full checkpoint. As discussed earlier, the incremental checkpoint aims to reduce the checkpoint overhead. On the other hand, the recovery cost will increase as the number of subsequent incremental checkpoints (m) increases. This is because the application reconstruction phase requires information from each and every incremental checkpoint since the last full checkpoint. From the model description of the incremental checkpoint below, we assume that the first checkpoint is a full checkpoint, followed by a sequence of incremental checkpoints. Moreover, we will perform m incremental checkpoints after a full checkpoint if the expected waste time of having $m+1$ incremental checkpoint is more expensive than that of having m incremental checkpoints. We follow this idea to find m by comparing the expected waste time in two possible cases. In the first case, as shown in Figure 6(a), m continuous incremental checkpoints are followed by a full checkpoint. Alternatively, as shown in Figure 6(b), after placing m continuous incremental checkpoints, we continue to perform the $m+1^{th}$ incremental checkpoint. In each case, we consider the probability of failure. Details are discussed in what follows.

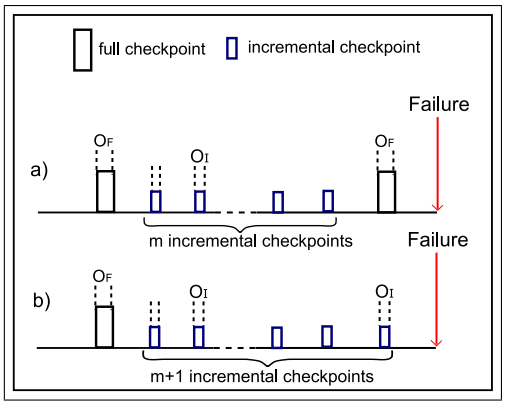

Figure 6: Sequential incremental checkpoint scenario

Case (a): After placing m continuous incremental checkpoints, a full checkpoint is performed next as shown in Figure 6(a).

Let P_I is the probability that a failure will occur after the second full checkpoint and before the next incremental checkpoint. Hence, $1-P_I$ is the probability that failure will not occur in that period.

If no failure occurs during this period, the overall cost is C_{a1}, $C_{a1} = (O_F + mO_I) + O_F$. Alternatively, if the failure occurs, the cost C_{a2} is $C_{a2} = (O_F + mO_I) + O_F + R_F$. Therefore, the expected cost is

$$C_a = (1-P_I)(2O_F + mO_I) + P_I(2O_F + mO_I + R_I). \tag{30}$$

Case (b): After reaching m consecutive incremental checkpoints, another incremental checkpoint is performed as shown in Figure 6(b). We consider that the probability of the failure events is approximately the same as in case (a), P_I.

When no failure occurs, the cost C_{b1} is $C_{b1} = (O_F + mO_I) + O_I$. Alternatively, if a failure happens, the cost C_{b2} is $C_{b2} = (O_F + mO_I) + O_I + (R_F + (m+1)R_I)$.

Therefore, the expected cost in case (b) is

$$C_b = (1-P_I)[O_F + (m+1)O_I) + P_I[O_F + (m+1)O_I + (R_F + (m+1)R_I)]. \tag{31}$$

We would like to have the number of incremental checkpoints as much as possible that yields the criteria that if another incremental checkpoint is added the expected waste time is larger than that of having another full checkpoint, and then the solution of m must be satisfied $C_b \geq C_a$. Thus, we will choose case (a) and perform a full checkpoint after m sequential incremental checkpoints.

Therefore, we obtain

$$m \geq \frac{O_F - O_I}{P_I R_I} - 1. \tag{32}$$

Inequality (32) suggests us that if $m \geq \frac{O_F - O_I}{P_I R_I} - 1$, the cost in case (b) will be greater than the cost in case (a). Thus, we take m as

$$m = \left\lceil \frac{O_F - O_I}{P_I R_I} - 1 \right\rceil, \tag{33}$$

where $\lceil \cdot \rceil$ is the ceiling function.

According to Eq.(33), m is proportional to the difference between the full and the incremental checkpoint overhead and inversely proportional to the incremental recovery cost and the probability of a failure occurrence P_I. The following points need to be raised at this point. Firstly, if the incremental checkpoint overhead O_I is nearly as large as the full checkpoint overhead O_F, performing incremental checkpoints instead of full checkpoints may not reduce the total overhead. In Eq.(33), when O_I approaches to O_F, the nominator approaches 0, then the number of incremental checkpoints m is small. Secondly, all incremental checkpoints following the last full checkpoints must be loaded after a failure occurs, causing extra cost in the application recovery period. Thus, if the incremental checkpoint recovery cost is expensive, then the number of incremental checkpoints should be small in order to maintain the low recovery cost. Thirdly, a full checkpoint should be performed when a chance of a failure occurrence is high because, when a failure happens, there are few incremental checkpoints to be loaded. Therefore, if the failure probability P_I is high, the number of incremental checkpoints m is small as in Eq.(33). Lastly, the number of incremental checkpoints m does not depend on the full checkpoint recovery cost because there is only one full checkpoint loaded during the application recovery time.

3.3 Incremental checkpoint Model Evaluation

We obtained the general solution for our incremental checkpoint model. Eq.(29) gives a checkpoint frequency function which is derived from a probability distribution function of the system time between failures (TBF). For the purpose of the incremental checkpoint/restart study and evaluation, we validate our model results only when the system failure follows the exponential distribution. This assumption will help simplify our validation. However, we plan to use these results as a guidance to further our study with other distributions such as time-varying one for the system failures.

Incremental checkpoint Model for the Exponential Distribution

For the time between failures (TBF) that follows an exponential distribution, we substitute $f(t) = \lambda e^{-\lambda t}$, and $1 - F(t) = e^{-\lambda t}$, $t \geq 0$, $\lambda > 0$ in Eq.(34).

The optimal model for an exponential Distribution can be written as

$$n_I(t) = \sqrt{\frac{(m+1)k}{O_F + mO_I}}\sqrt{\lambda}. \tag{34}$$

We can find the i^{th} checkpoint placement. For $i = 0, 1, 2, ...$ we have that:

$$\int_{t_i}^{t_{i+1}} \sqrt{\frac{(m+1)k}{O_F + mO_I}}\sqrt{\lambda}\, dt = 1. \tag{35}$$

Solving Eq.(35), we have that:

$$t_{i+1} = \sqrt{\frac{O_F + mO_I}{(m+1)k}}\sqrt{\lambda} + t_i. \tag{36}$$

Using induction and taking $t_0 = 0$, the sequence of the optimal checkpoint placements for the exponential distribution with failure rate λ is given by

$$t_i = i\sqrt{\frac{(m+1)k}{O_F + mO_I}}\sqrt{\frac{1}{\lambda}}, \quad k \in (0, 1), \tag{37}$$

where t_i is the i^{th} checkpoint placement either full checkpoint or incremental checkpoint. In Eq.(37), the number of incremental checkpoints can be obtained by Eq.(33), and we can obtain the failure probability P_I analytically. Please note that the checkpoint interval $(t_{i+1}-t_i)$ is a constant equal to $\sqrt{\frac{(m+1)k}{O_F+mO_I}}\sqrt{\frac{1}{\lambda}}$. From Eq.(33), the probability P_I of failure during the$(t_{i+1}-t_i)$ interval is

$$P_I = P\left(T_f < t_{i+1} | T_f > t_i\right) = \frac{F(t_{i+1}) - F(t_i)}{1 - F(t_i)} \tag{38}$$

For the exponential distribution, the CDF is $F(t) = 1 - e^{-\lambda t}$, and we have

$$P_I = \frac{\left(1 - e^{-\lambda t_{i+1}}\right) - \left(1 - e^{-\lambda t_i}\right)}{1 - \left(1 - e^{-\lambda t_i}\right)} \tag{39}$$

Since the checkpoint interval is constant and $t_{i+1} - t_i = t_1 - t_0 = t_1$, then we have

$$P_I = 1 - e^{-\lambda t_1}. \tag{40}$$

In practice, we have to find k (the re-computing time coefficient) and m at the same time.

Algorithm to find k and m:

Step 1: Initialize O_F, O_I, R_I and λ. Let $\hat{k} = 0.5$ and $t_0 = 0$.

//Find m corresponding to $\hat{k}$.

Step 2: Let $\hat{m} = 1$.
Step 3: Calculate t_1 from Eq. (35).
Step 4: Calculate P_I by Eq. (38).
Step 5: IF $\hat{m} < \frac{O_F - O_I}{P_I R_I} - 1$,
THEN $\hat{m} = \hat{m} + 1$, and go to Step 2.
ELSE go to Step 6.
Step 6: Set $m = \hat{m} - 1$

//Finished finding m

Step 7: Calculate the checkpoint sequence $t_1, t_2, ..., t_N$ corresponding to $\hat{k}$ and m from Eq. (35)
Step 8: Calculate $\bar{k}$ from Eqs. (11)(12)(13) using the sequence in Step 7.
Step 9: IF $\hat{k} = \bar{k}$,
THEN Set $k = \hat{k}$ DONE
ELSE Set $\hat{k} = \bar{k}$, and go to Step 1.

4 Comparisons between the Full Checkpoint Model and the Incremental Checkpoint Model

In this section, we assume that the waste time of the incremental checkpoint model would be less than that of the full checkpoint model. As we discussed before, the incremental checkpoint scheme aims to reduce the total checkpoint overhead. In contrast, it introduces more expensive recovery cost than the full checkpoint mechanism. Therefore, in order to provide evidences to the assumption, the comparisons between the waste times of both models are performed based on the actual failure data of White and Frost HPC systems.

White and Frost were IBM supercomputers at Lawrence Livermore National Laboratory. White consisted of 512 nodes with 16 processors per node and 8192 processors in total, and Frost consisted of 68 nodes with 16 processors per node and 1088 processors in total. Although White and Frost systems had retired already at July 26, 2006 and June 30, 2005, respectively, we choose both systems to analyze because we have the failure timestamps of both systems over the period of four years from 2000 to 2004.

Before we run the simulations, we process both failure data sets (White and Frost) as the following. First, we use the time window of one month to break the failure time stamps, and we then have 52 sets of failure times of White system and 50 sets of failure times of Frost system. Second, for each data set, we calculate the times between failures (TBFs) and the mean time between failures (MTBF). Next, the Kolmogorov-Smirnov test is used

to test the null hypothesis that the data sets follow the exponential distribution with the corresponding MTBF with the significance level of 0.1. After the test, there are 33 sets of TBFs of White and 42 sets of TBFs of Frost that follow the exponential distribution. These are the samples that we will use in the simulations for both systems. Furthermore, the MTBF of the White samples is around 23 hours, and the MTBF of the Frost samples is around 70 hours.

An objective of the simulations is to study the ratio between the waste time and the application completion time, denoted as RWC. The application completion time is the time from the beginning of the application computation to the completion including the checkpoint overhead, recovery time, and re-computing time. RWC can be used as a metric to evaluate the efficiency of the models. For example, if RWC value approaches 0, most of application execution time is used to compute the application, which indicates that the performances of the proposed models are likely to be good. Otherwise, most of application execution time is the waste time. However, there is no standard or threshold to indicate the goodness of a checkpoint model yet. To simulate the waste time of both models, for each TBF data set, the following procedure is applied. We determine the checkpoint sequences for the full and incremental checkpoint models, and then, for each sequence of TBFs, we compute the waste times for different completion time values, as the range of 0 to 30 days with the increment of 1 day. Next we calculate RWC of each completion time value and find the average of RWC over the completion time.

Four experiments are conducted with different values of full checkpoint overhead (0.5, 2, 10 and 30 minutes). For each experiment, we calculate the average of RWC for the full checkpoint model and the incremental checkpoint model with 3 different values of incremental checkpoint overheads as 10%, 50%, and 90% of the full checkpoint overhead. Therefore, we have 16 values of RWC in total for White and Frost systems as shown in Figure 7, Figure 8, respectively.

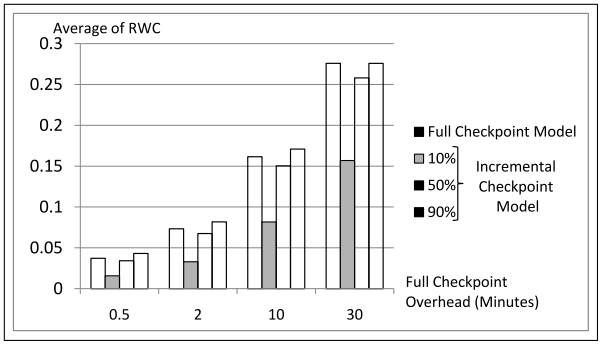

Figure 7: White Average of *RWCs* of the full checkpoint model and the incremental checkpoint model with different values of full checkpoint overhead of 0.5, 2, 10, and 30 minutes and different values of incremental checkpoint overhead of 10%, 50%, and 90% of the full checkpoint overhead values.

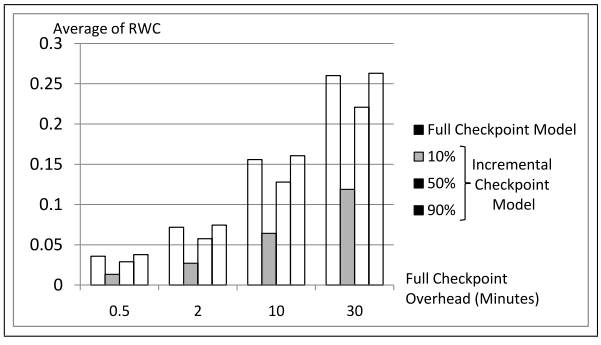

Figure 8: Frost Average of RWCs of the full checkpoint model and the incremental checkpoint model with different values of full checkpoint overhead of 0.5, 2, 10, and 30 minutes and different values of incremental checkpoint overhead of 10%, 50%, and 90% of the full checkpoint overhead values.

According to Figure 7 and Figure 8, in every case of full checkpoint overheads, we notice that the proportions of the RWCs of the incremental checkpoint model are not implied by the ratio between the incremental checkpoint

overhead and the full checkpoint overhead. For example, when the incremental checkpoint overhead is 10% of the full checkpoint overhead, the averages of RWCs of the incremental checkpoint model is approximately half of the averages of RWCs of the full checkpoint model. In practice, we expect that the waste time of the incremental checkpoint scheme should be less than the full checkpoint scheme, but the graphs show that, while the incremental checkpoint overhead is 90% of the full checkpoint overhead, the average of RWCs of the incremental checkpoint scheme is slightly greater than that of the full checkpoint scheme. Moreover, the behaviors of the RWC in White and Frost system are very similar although the MTBFs of both systems are notably different. We may imply that our model is able to give similar results in other systems.

Additionally, the full checkpoint model for the Weibul distribution has been compared with the Risk-Based model [10] in [21]. In the Risk-Based model, a checkpoint will be performed if the waste time when the checkpoint is performed is less than the waste time when the checkpoint is not performed. In [10], there are three other models, but the Risk-Based model was the only model chosen for comparison because it is the best one among all the models in [10]. According to [21], with different values of checkpoint overhead, the full checkpoint model produced less waste time than the Risk-Based model, except for the case when the full checkpoint overhead is equal to 1 hour. In this case, the waste times of both models were very similar.

5 Conclusions

In this paper, we have presented near optimal checkpoint/restart models in full and incremental checkpoint schemes in a large-scale HPC environment. In these models, the time sequence of checkpoint placements was derived using the theory of a stochastic renewal reward process. The models are general and can be applied to any distribution of time between failures. However, the given example is for the case of the exponential distribution. The re-computing time which directly relates to the failure time, is an important factor in a checkpoint/restart model. Instead of using the re-computing time, we introduced the re-computing time coefficient (k), its estimation approach, and an algorithm to estimate k. For the incremental checkpoint model, there is another significant factor which is the number of incremental checkpoints between consecutive two full checkpoints (m). The derived m yields the event that the expected waste time of performing a full checkpoint after m incremental checkpoints is less than that of performing the $(m+1)^{th}$ incremental checkpoint. The proposed algorithm does not provide only the evaluation of m but also the re-computing time coefficient. For the model analysis part, the failure data of White and Frost, the supercomputing system owned by LLNL were used to simulate the waste time of both proposed models. The comparisons between both models are provided. In most cases, the waste times of the incremental checkpoint model are less than those of the full checkpoint model, especially when the incremental checkpoint overhead is much less than the full checkpoint overhead. Furthermore, the proportion of the waste times of the incremental and full checkpoint models seems to not relate to the ratio of the incremental and full checkpoint overheads. Lastly, the results of White and Frost systems are very similar. This may indicate that the proposed models are able to give similar results for other systems.

6 Future Work

In the near future, we will extend the incremental checkpoint model for the case where the number of incremental checkpoints between two consecutive full checkpoints is not a constant. We expect that the extended model will give less waste time than the proposed one. Moreover, we will improve the method to evaluate the number of incremental checkpoints between two consecutive full checkpoints to yield optimality. In addition, we will work on improving the models. For example, in some applications, there are many communications between nodes. If one performs a checkpoint while there is a large amount of communications going on, the checkpoint overhead will be expensive. Therefore, the communication or I/O transfer rate may be another factor to consider when performing a checkpoint.

Acknowledgements. Work supported in part by the NSF grant no. 0834483, DOE grant no. DE-FG02-08ER25836 and the National Plan II of Romania RP5 grant.

Bibliography

[1] A.R. Adiga, G Almasi, and et al., An overview of the BlueGene/L supercomputer, *Proceedings of Supercomputing, IEEE/ACM Conference*, pp. 60-60, 2002.

[2] J.T. Daly, A Model for Predicting the Optimum Checkpoint Interval for Restart Dumps, ICCS 2003, *LNCS 2660*, Volume 4, pp. 3–12, 2003.

[3] J.T. Daly, A Higher Order Estimate of the Optimum Checkpoint Interval for Restart Dumps, *Future Generation Computer Systems*, Elsevier, Amsterdam, 2004.

[4] E. Elnozahy, J. Plank, Checkpointing for Peta-Scale Systems: A Look into the Future of Practical Rollback-Recovery, *IEEE Transactions on Dependable and Secure Computing,* vol.01,no.2, pp. 97-108, 2004.

[5] R. Geist, R. Reynolds, and J. Westall, Selection of a checkpoint interval in a critical-task environment, *IEEE Transactions on Reliability*, vol.37, no.4, pp. 395-400, 1988.

[6] I. M. Gelfand and S. V. Fomin, Calculus of Variations, *Dover Publications*, ISBN-10: 0486414485, 2000.

[7] L. Hancu, Data-Mining Techniques for Supporting Merging Decisions,*Int. J. of Computers, Communications and Control*,Vol. III (2008), pp. 322-326.

[8] D. I. Hunyadi, M. A. Musan, Modelling of the Distributed Databases. A Viewpoint Mechanism of the MVDB Model's Methodology, *Int. J. of Computers, Communications and Control*, Vol. III (2008),pp. 327-332.

[9] Y. Ling, J. Mi, and X. Lin, A Variational Calculus Approach to Optimal Checkpoint Placement, *IEEE Transactions on Computers*, vol. 50, no. 7, pp. 699-707, 2001.

[10] A.J. Oliner, L. Rudolph, and R.K. Sahoo, Cooperative Checkpointing: A Robust Approach to Large-scale Systems Reliability, P*roceedings of the 20th Annual International Conference on Supercomputing* (ICS), Australia, pp. 14-23, 2006.

[11] T. Ozaki, T. Dohi, and H. Okamura, Distribution-Free Checkpoint Placement Algorithms Based on Min-Max Principle, *IEEE Transactions on Dependable and Secure Computing*, Volume 3, Issue 2, pp. 130 – 140, 2006.

[12] A. C. Palaniswamy, and P. A. Wilsey, An analytical comparison of periodic checkpointing and incremental state saving, *Proc. of the Seventh Workshop on Parallel and Distributed Simulation*, California, pp. 127-134, 1993.

[13] J.S. Plank, J. Xu, and R.H. Netzer, 1995a. Compressed differences: an algorithm for fast incremental checkpointing, *Technical Report CS-95-302*, University of Tennessee at Knoxville, 1995.

[14] J.S. Plank, M.A. Thomason, The Average Availability of Parallel Checkpointing Systems and Its Importance in Selecting Runtime Parameters, T*he 29th International Symposium on Fault-Tolerant Computing*, Madison, WI, pp. 250-259, 1999.

[15] S.M. Ross, Stochastic Processes, *Wiley*; 2nd edition, ISBN-10: 0471120626, 1995.

[16] B. Schroeder, G.A. Gibson, A large-scale study of failures in high-performance computing systems, *Proceedings of International Symposium on Dependable Systems and Networks* (DSN). IEEE Computer Society, pp. 249–258, 2006.

[17] A. Tikotekar, C. Leangsuksun, S. Scott, On the survivability of standard MPI applications, *In Proceedings of 7th LCI International Conference on Linux Cluster*s: The HPC Revolution 2006.

[18] N. H. Vaidya, A case for two-level distributed recovery schemes, *in Proceedings of the 1995 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pp. 64–73, 1995.

[19] K. F. Wong, M.A. Franklin, Distributed Computing Systems and Checkpointing, *HPDC*, pp. 224-233, 1993.

[20] J.W. Young, A first-order approximation to the optimum checkpoint interval, *Communications of ACM* volume 17, Issue 9, pp. 530-531, 1974.

[21] Y. Liu, R. Nassar, C. B. Leangsuksun, N. Naksinehaboon, M. Paun, and S. L. Scott, An optimal checkpoint/restart model for a large scale high performance computing system, *in Proc. International Parallel and Distributed Processing Symposium*, (Miami, Florida, 2008) pp.1-9, 2008.

[22] D. Zmaranda, G. Gabor, Issues on Optimality Criteria Applied in Real-Time Scheduling,*Int. J. of Computers, Communications and Control*, Issue 3, pp.536-540, 2008.