Selection of a Checkpoint Interval in Coordinated Checkpointing Protocol for Fault Tolerant Open MPI

Mallikarjuna Shastry P.M.#[1] ,K. Venkatesh#[2]
[#1]Sapthagiri College of Engg., [#2]M. S. Ramaiah Institute of Technology,
[#]Affiliated to Vishweshwaraya Technological University,
Bangalore, Karnataka, India.,

Abstract— **The goal of this paper is to address the selection of efficient checkpoint interval which reduces the total overhead cost due to the checkpointing and restarting of the applications in a distributed system environment.**

Coordinated checkpointing rollback recovery protocol is used for making the application programs fault tolerant on a stand-alone system under no load conditions using BLCR and OPEN MPI at system level.

We have presented an experimental study in which we have used the optimum checkpoint interval determined by an existing model to compare the performance of coordinated checkpointing protocol using two types of checkpointing intervals namely fixed and incremental checkpoint intervals. We measured the checkpoint cost, rollback cost and total cost of overheads caused by the above two methods of checkpointing intervals

Failures are simulated using the Poisson distribution with one failure per hour and the inter arrival time between the failures follow exponential distribution.

We have observed from the results that, rollback overhead and total cost of overheads due to checkpointing the application are very high in incremental checkpoint interval method than in fixed checkpoint interval method.

Hence, we conclude that fixed checkpointing interval method is more efficient as it reduces the rollback overhead and also total cost of overheads considerably.

Keywords - ***Checkpoint; Checkpoint Interval; Fault tolerance, Marker, Checkpoint Overheads***.

I. INTRODUCTION

Since, the recent trends in HPC and even stand alone systems employ a very large number of processors to execute the large size application programs in a distributed system environment, it is required to provide the fault tolerance to such applications. As the complexity of the program increases, the number of processors to be added to the cluster / HPC / Super Computer also increases which in turn decreases the MTBF (mean time between failures) of the processors or the machines. It means that the probability of failure of one or more processors will be very high before the completion of the execution of the long running application being executed parallely on several processors. When a processor fails, we need to restart the entire application on all the processors from the beginning. Hence, it is required to address the issues like the scalability and fault tolerance.

Fault tolerance provides the reliability and availability to the large size applications programs executed in a distributed system environment. Fault tolerance is achieved using coordinated checkpointing rollback recovery protocol in which an initiator takes a checkpoint by synchronizing with all other processes of MPI application [1]. For MPI applications, a cluster consisting of a group of processes interacting with each other is formed and each individual process in the cluster is checkpointed and a global state is formed out of it.

The global state contains the "set of checkpoints exactly one from each processor". The global state is consistent if and only if for each message received by a processor (receiver), there is a corresponding sender. The latest consistent global state is known as the recovery line [2]. The checkpoint / restart scheme has been widely used in [3]-[9] to address the failure of execution of an application.

Checkpoints can be taken using either fixed checkpoint interval or variable checkpoint interval [10]. In case of fixed checkpoint interval, checkpoint interval size remains same between any two successive checkpoints. But, in case of the incremental checkpoint interval method discussed in this paper, the second checkpoint interval size is 2 times the 1st one and third checkpoint interval is 3 times the 1st one and so on in each cycle.

A cycle is the execution time interval of the application between two successive failures. Since, these two methods of checkpoint intervals are not compared in the literature as we understand; we have carried out an experiment to determine the behavior of the coordinated checkpointing protocol using the fixed and incremental checkpoint interval methods.

The rest of the paper is organized as follows. Section 2 introduces the related works carried out in checkpoint and restart schemes using different checkpoint intervals. Section 3

presents the different notations used in this paper. The implementation of the coordinated checkpointing protocol, description of fixed and incremental checkpoint intervals are discussed in section 4. Computation of cost of overheads is discussed in section 5. Section 6 presents the experimental setup and results. Section 7 presents the conclusion.

II. Related Works

Young [5] has presented an optimum checkpoint and restart model and has shown that the total waste time due to checkpointing can be reduced using fixed checkpoint interval. But, this model [5] does not consider the restart time required to resume the application from the most recent checkpoint after a failure.

An optimal checkpoint / restart model presented by Yudun liu [11] uses varying checkpoint interval with different failure distributions. But, varying checkpoint interval does not yield optimal rollback and checkpoint cost. R. Geist et. Al [12] discusses the selection of checkpoint interval in a critical task environment, but it does not present any optimal solution for selecting the checkpoint interval.

J.T. Daly [6], [9] presents a method for determining the optimum checkpoint interval but they do not discuss the comparison of the performance of the coordinated checkpointing protocol with respect to fixed and incremental checkpointing interval methods.

III. Notations Used

1. R_{bi} - the cost of rollback in ith cycle.
2. R_b - total rollback cost.
3. R - restart time required to resume the execution of an application from the most recent checkpoint.
4. F - the number of failures during the execution of the application.
5. T_S - time required to save each checkpoint on to a local disk.
6. N_i - the number of checkpoints taken in i^{th} cycle.
7. C_i - starting time of ith checkpoint.
8. T_C - optimum checkpoint interval size and is used as fixed checkpoint interval.
9. T_{Ci} - i^{th} checkpoint interval which is incremental.
10. CC_i - the cost of checkpoints in i^{th} cycle.
11. CC – total cost of checkpoints
12. P - the number of processes / processors used for parallelism.
13. λ – Number of failures per hour.
14. T_F – time to failure.
15. T_i - the time at which the i^{th} failure occurs.

IV. Implementation of Coordinated Checkpointing Protocol

A. Protocol

Master MPI process with rank i=0 takes the tentative checkpoint and then sends the marker to MPI process with rank (i+1) % N. When MPI process i > 0 receives the marker from (i + N-1) % N, takes its tentative checkpoint and sends the marker to MPI process with rank (i + 1) % N.

When the MPI process with rank 0 receives the marker from MPI process N-1, a global consistent checkpoint is formed out of all the local checkpoints and then sends the checkpoint file to the local disk and then initiates the next checkpoint cycle after the checkpoint interval as specified by the user. The checkpoint period can be either fixed or incremental which will be discussed in the following subsections

B. Optimal Checkpoint Interval

Knowing the number of processors (P) used for computation and the failure rate (λ) of the processors, we can compute the time to failure [14] of the application during run time as follows.

$$T_F = 1 / (P \lambda) \quad (1)$$

Once, we determine the time to failure T_F and the checkpoint overhead T_S (time required to save each checkpoint onto local disk), the checkpoint interval T_C can be computed [5] as follows.

$$T_C = \sqrt{2 T_S T_F} \quad (2)$$

The equation (2) is obtained by using second order approximation for exponential distribution [5]. We have used the equation (2) to compare the two different checkpoint interval methods as discussed in the subsequent sections.

C. Fixed Checkpoint Interval

The total execution time of the application program is divided into n checkpoint intervals of length T_C as computed in the previous section. The first checkpoint is initiated by the master MPI process after completion of Tc minutes of execution of the application program and second checkpoint is initiated after completion of $2 T_C + T_S$ minutes and so on as shown in figure 1.

In general, the starting time of i^{th} checkpoint C_i is computed as follows.

$$C_i = i T_C + (i-1) T_S \quad (3)$$

But, the length of each checkpoint interval is fixed.

In general $T_{Ci} = T_C$

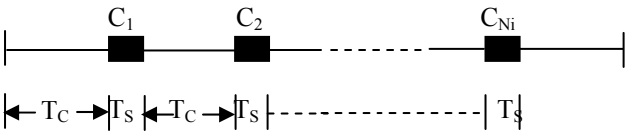

Fig. 1 Fixed Checkpoint Interval

D. Incremental Checkpoint Interval

Figure 2 shows the checkpointing of the application with an incremental checkpoint interval. In this case, in each cycle, the first checkpoint is initiated after (Tc_1) or T_C minutes of execution of the application and second checkpoint is initiated after $T_{C1} + T_{C2} + T_S$ minutes and the third checkpoint is initiated after $T_{C1} + T_{C2} + T_{C3} + 2TS$ and so on as shown in figure 2.

In general the starting time of i[th] checkpoint C_i is computed as follows.

$$C_i = \sum_{k=1}^{i} T_{CK} + (i-1)\, T_S \qquad (4)$$

And the i[th] checkpoint interval is computed as follows.

$$T_{C\,i} = i\, T_C \qquad (5)$$

V. Cost of Overheads Due To Checkpointing and Restarting

During recovery from failure, the master MPI process coordinates with all the other MPI processes and restarts by rolling back the application to the most recent consistent global checkpoint. The cost of rollback, cost of checkpointing and the restart cost are the 3 components which are used to determine the waste time in each cycle of the application. If the application undergoes F failures, the execution of the application will have F cycles. We present in the following sections the determination of these costs using two different methods of checkpoint intervals such as fixed and incremental checkpoint intervals.

A. Cost of Overheads in Fixed Checkpoint Interval

Failure of a fault tolerant application using fixed checkpoint interval is shown in figure 3. Since, all the checkpoint intervals have same length; the number of checkpoints to be taken in i[th] cycle (before i[th] failure occurs) is computed as follows.

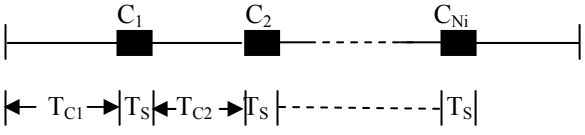

Fig. 2 Incremental Checkpoint Interval

$$N_i = \lfloor T_i / (T_C + T_S) \rfloor \qquad (6)$$

Then, the cost of checkpoint in i[th] cycle is computed as follows.

$$CC_i = N_i\, T_S \qquad (7)$$

The cost of rollback in i[th] cycle is then computed as follows

$$Rb_i = (T_i - N_i\, (T_C + T_S)) \qquad (8)$$

The time lost in i[th] cycle TL_i due to a failure can be obtained by adding checkpoint cost, rollback cost, restart cost together as follows.

$$TL_i = CC_i + Rb_i + R \qquad (9)$$

The total waste time due to F failures is then computed as follows.

$$TL = \sum_{i=1}^{F} TL_i \qquad (10)$$

Suppose, if the first failure occurs at 40 minutes of execution and checkpoint interval size T_C is 4 minutes, time to save a checkpoint is 35 seconds and number of checkpoints taken before failure occurs is 8, The different overhead costs are determined as follows.

i) Rollback cost: By applying equation (8), we can determine the rollback cost as follows.
$Rb_1 = (2400 – 8 * (240 +35)) = 200$ seconds

ii) Checkpoint cost (CC_1) = 8 checkpoints * cost of each checkpoint = 8 * 35 = 280 seconds

iii) It was found from the experimental setup that the time required to restart an application after a failure is just about 24 seconds. So, the total time lost due to a failure of application in fixed checkpoint interval case after first failure is

TL_1 =cost of rollback (200 seconds) + cost of checkpoints (280 seconds) + restart cost (24 seconds) = 504 seconds (about 21 % of execution time of 1[st] cycle is wasted due to checkpointing, rollback and restart). It was observed that, the cost of rollback is dependent on the amount of time elapsed since the last checkpoint.

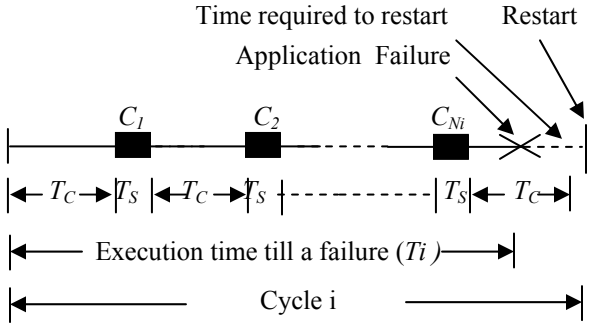

Fig. 3 Cost of Overheads in Fixed Checkpoint Interval

B. Cost of Overheads in Incremental Checkpoint Interval

Failure of a fault tolerant application using incremental checkpoint interval is shown in figure 4. In this, method, size of first checkpoint interval is T_C minutes and size of other checkpoint intervals is T_C minutes more than its previous checkpoint interval in each cycle. Hence, the number of checkpoints (N_i) to be taken in ith cycle will vary in this method and it is computed as follows.

$$a.\ if\ T_1 < (T_{C1} + T_S), \qquad N_1 = 0 \tag{11}$$

$$b.\ if\ (\sum_{k=1}^{n} K\ T_C + n\ T_S) < T_i\ and$$

$$T_i < (\sum_{k=1}^{n+1} K\ T_C + (n+1)\ T_s) \tag{12}$$

$$for\ \ n = 1,2,3,..\ \ \ and\ \ \ for\ \ \ i = 1,2,3,..F$$

The number of checkpoints to be taken in ith cycle can be computed as follows.

$$N_i = n \tag{13}$$

Though, the checkpoint interval length keeps increasing, from one checkpoint to another checkpoint, the per checkpoint cost remains same as per the experimental results that we have obtained. So, the total checkpoint cost in ith cycle can be computed as follows.

$$CC_i = N_i\ T_S \tag{14}$$

The cost of rollback in ith cycle is then computed as follows.

$$Rb_i = (T_i - (N_i\ T_S + \sum_{k=1}^{n} K\ T_C\ )) \tag{15}$$

The time lost in ith cycle TL_i due to a failure can be obtained by adding checkpoint cost, rollback cost, and restart cost together using equation (9). Total waste time due to F failures is computed using equation (10). It was observed that, the cost of rollback depends on two factors like the time of failure and the checkpoint interval size. This is because, in this method, the checkpoint interval size varies from one checkpoint to another checkpoint. Suppose, if the first failure occurs at 40 minutes of execution and initial checkpoint interval T_C is 4 minutes,2nd checkpoint interval is 8 minutes and 3rd checkpoint interval is 12 minutes and the 4th checkpoint interval is 16 minutes (during which the failure occurs), time to save a checkpoint is 35 seconds and number of checkpoints taken before failure occurs is 3 after applying the equation (12) and (13). Then, the different overhead costs are determined as follows.

i) rollback cost : after applying the equation (15), we get
Rb_1 =(2400 – (105 + 1440)) = 855 seconds
ii) checkpoint cost is (CC_1) = 3 checkpoints * cost of each checkpoint=3*35 = 105 seconds

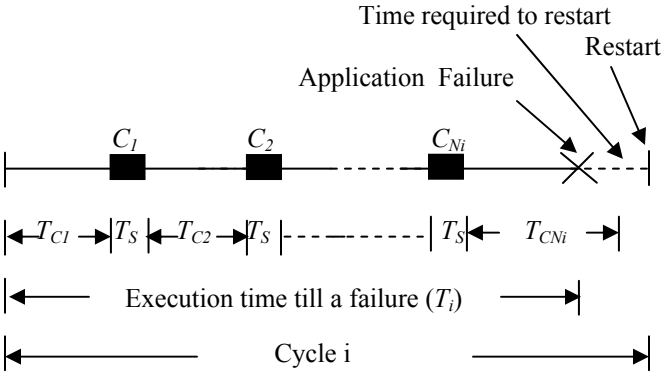

FIG. 4 COST OF OVERHEADS IN INCREMENTAL CHECKPOINT INTERVAL

iii) and it was found from the experimental setup that the time required to restart an application after a failure is just about 24 seconds.

So, the total time lost due to a failure of application in incremental checkpoint interval case is

TL_1 =cost of rollback + cost of checkpoints + restart cost
= 984 seconds

(41 % of execution time of 1st cycle is wasted due to checkpointing, rollback and restart).

VI. EXPERIMENTAL SET-UP AND RESULTS

We have taken the application program that multiplies 2 integer matrices of size 7000 * 7000. The above application is written in C language and run on a standalone system using scattered method of MPI under no load conditions.

In scattered method, one of the matrices, say the matrix B is broadcasted across all the processors using MPI_Bcast(). The matrix A is divided equally among the number of processors used for parallelism and each of the processors gets only a portion of matrix A allocated for it for the computation using MPI_Scatter().

The above application was run on a system with 6 GB of RAM, Intel ® Core ™ 2 Duo CPU,E7200 @ 2.53 GHz and 110 GB of HDD and the execution time of the application considered in our experimental setup on this system is 67 minutes without checkpointing.

The monitor program is written in a shell script which runs at the background and keeps monitoring whether the MPI processes grouped under mpirun are running or not. Once, monitor program learns that an MPI process has failed, it calls the restart() routine of BLCR to restart the application.

During the restart or recovery state, the MPI application rolls back to the most recent checkpoint as discussed and resumes the execution of the application from that point.

BLCR checkpoint and restart library [13] is used to implement the blocking coordinated checkpointing protocol to checkpoint the application. The application was run 10 times for different number of processors varying from 1 to 10. We observed that the checkpoint cost and the restart cost increase linearly with the increase in the number of processors.

The results obtained for 10 processors are presented in this paper. The number of arrivals N (t) in a finite interval of length t obeys the Poisson (λt) distribution.

$$P\{N(t) = n\} = (\lambda t)^n e^{-\lambda t} / n!$$

The inter arrival times of the failures are independent and obey the exponential distribution.

$$f(x) = \lambda e^{-\lambda x} \quad for \quad x >= 0$$
$$= 0, \quad otherwise$$

We have used Poisson distribution with 1 failure per hour (λ = 1) for 10 processors and generated the probability distributions for the inter arrival times of failures. Failures are simulated using these probability distributions and the results are presented in the form of graphs.

Figures 5a and 5b present the comparison of total cost of all the overheads due to checkpointing and restarting of the application with different checkpoint interval sizes when the application considered in our experimental setup fails at different timings.

We have tested for 10 different failures (whose failure timings are generated using Poisson arrival process) and determined the total cost of overheads incurred due to checkpointing and restarting using 5 different checkpoint intervals of size 2,3,4,5 and 10 minutes as shown in figures 5a) and 5b). In 7 different cases out of 10 failures at different timings, checkpoint interval size with 4 minutes was found to be optimal as it yields minimum total cost of all the overheads as shown in figures 5a and 5b when the application fails at 10,15,20,42,47,50 and 60 minutes of execution of the application.

From our experiment, we determined that, per checkpoint cost T_S is 35 seconds for 10 processors and it remains same for all the other checkpoints taken at different timings for the same number of processors. Per checkpoint cost is determined by taking the average value of checkpointing the application at 20 different timings on 10 processors.

Restart cost is found to be only 24 seconds to resume the execution of the application on 10 processors after a failure occurs. This restart cost is determined by taking the average of 20 different restart costs measured when the application failed at different timings.

We have used the equations (1) and (2) to determine the optimum checkpoint interval size when the time to failure T_F and checkpoint cost T_S are known. The value of T_C obtained from equation (2) shows that the optimum checkpoint interval size is 2.64 minutes as shown below.

$T_F = 1 / (P * \lambda) = 1 / (10 * (1/60))$ minutes = 6 minutes
and T_C = sqrt (2 T_S T_F)
= sqrt(2 * (35 /60) * 6) = 2.64 minutes

This checkpoint interval value is almost matching with the optimum checkpoint interval of 4 minutes obtained from the figures 5a) and 5b) based on our experimental results. The value of T_C calculated should yield almost the exact result, if the value of T_F is quite large in which case R << T_F. Hence, the equations (1), (2) and (3) are validated based on our experimental results and discussion.

As, we have obtained 4 minutes as the optimum checkpoint interval size from our experimental analysis, in our further analysis and discussion (figures 6 to 9), fixed checkpoint interval size taken is 4 minutes and in incremental checkpoint interval method, the first checkpoint interval size taken is 4 minutes, second checkpoint interval size taken is 8 minutes and third checkpoint interval size is 12 minutes and so on.

We have presented the results in the form of graphs for one failure in an hour with λ = 1 using Poisson distribution for arrival of failures. Figure 6 presents the comparison of number of checkpoints taken in fixed and incremental checkpoint interval methods. Figures 7, 8, and 9 present the comparison of i) cost of checkpoints, ii) cost of rollback and iii) total cost of overheads caused by fixed and incremental checkpoint interval methods respectively.

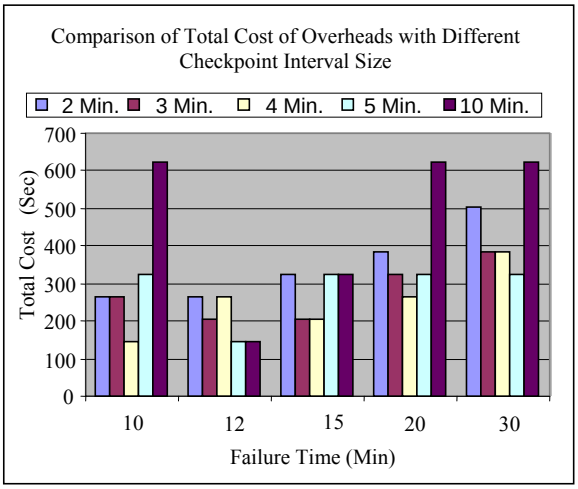

Fig 5a. Comparison of Total Cost of Overheads with Different Checkpoint Interval Size.

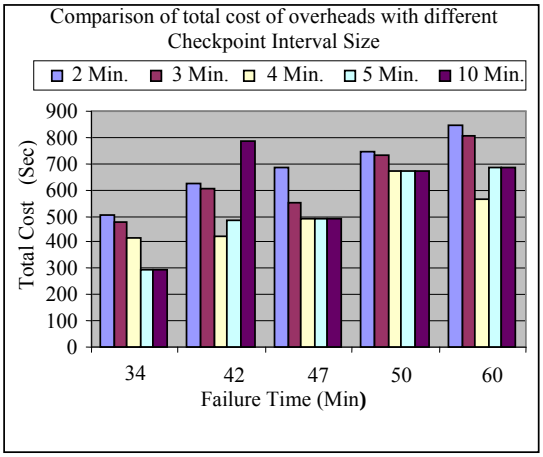

Fig 5b. Comparison of Total Cost of Overheads with Different Checkpoint Interval Size.

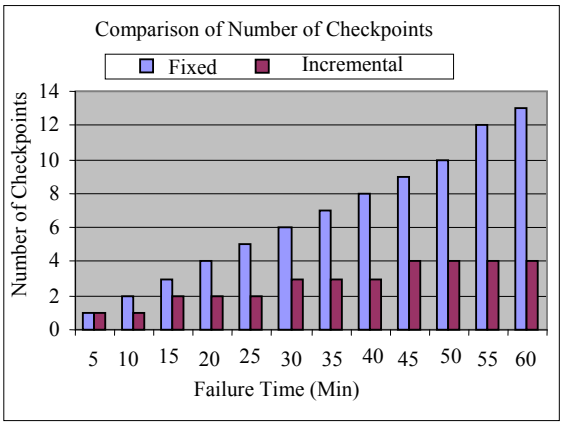

Fig 6. Comparison of Number of Checkpoints of Fixed and Incremental Checkpoint Intervals

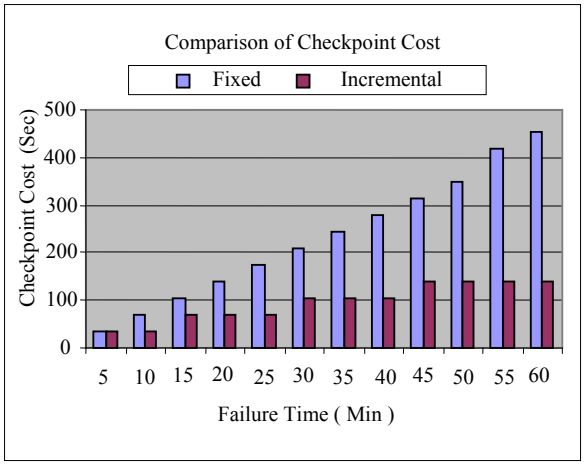

Fig 7. Comparison of Checkpoint Cost of Fixed and Incremental Checkpoint Intervals

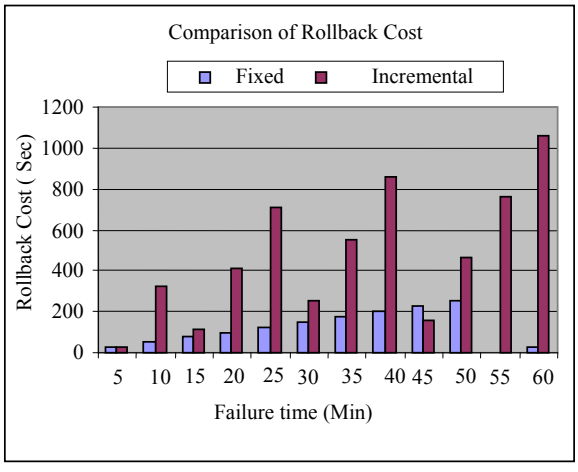

Fig 8. Comparison of Rollback Cost of Fixed and Incremental Checkpoint Intervals

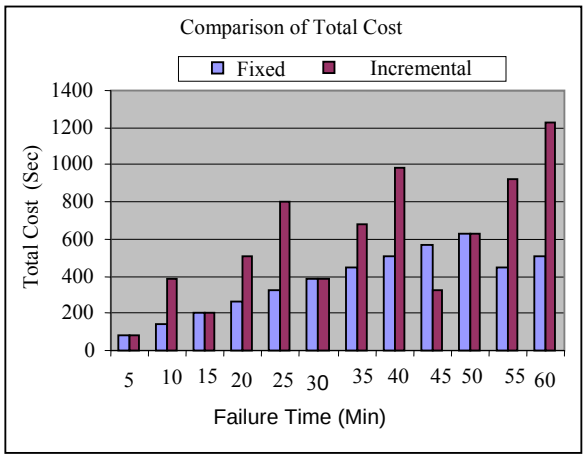

Fig 9. Comparison of Total Cost of overheads caused by Fixed and Incremental Checkpoint Intervals

VII. CONCLUSIONS

Figures 5a and 5b show the comparison of total cost of overheads with different checkpoint interval size. From figures 5a and 5b, it is clear that the total cost of overheads is quite minimum when checkpoint interval size is 4 minutes. We have even validated the model developed by Young [5] to determine the optimum checkpoint interval.

An approximate estimate of the checkpoint interval can be calculated from equation (2). From figure 6, we see that the fixed checkpoint interval method causes more number of checkpoints than incremental checkpoint interval method. So, the checkpoint cost is also quite high in fixed checkpoint interval method as compared to the incremental checkpoint interval method when the application fails after first checkpoint as shown in figure 7.

But, the rollback cost and the total cost of overheads produced by fixed checkpoint interval are quite low as compared to the incremental checkpoint interval when the application fails after first checkpoint as shown in figure 8 and figure 9 respectively. Fixed checkpoint interval reduces more than 50% of total overhead cost as compared to the incremental checkpoint interval.

Hence, we conclude that using fixed checkpoint interval for checkpointing an application would be more advantageous than using incremental checkpoint interval because fixed checkpoint interval reduces both rollback cost and the total cost of overheads significantly.

ACKNOWLEDGMENT

The authors thank the Head of Research Centre, CSE dept. and head of ISE dept M.S Ramaiah Institute of Technology, Bangalore for their constant encouragement. The Computer system acquired under the project sanctioned by BRNS, INDIA, bearing sanction No. 2008/37/15/BRNS, has been used to carry out our experimental work.

REFERENCES

[1] Luis Moura Silva and Joao Gabriel Silva, "The Performance of Coordinated and Independent Checkpointing", IEEE Trans, 1999.

[2] G.E. Fagg, A. Bukovsky and J.J. Dongarra, "Harness and Fault Tolerant MPI", Parallel Computing, 27(11):1479-1495, 2001.

[3. K.M. Chandy, "A survey of analytic models of roll-back and recovery strategies," Computer 8, 5 (May 1975), 40-47.

[4]. K.M. Chandy, J.C. Browne, C. W. Dissly, and W. R. Uhrig, "Analytic models for rollback and recovery stratagems in data base systems," IEEE Trans Software Engg. SE-1, (March 1975), 100-110

[5] J.W. Young, "A first order approximation to the optimum checkpoint interval," Communications of ACM 17, 9(Sept 1974), 530-531.

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

[7] E. Elnozahy, J. Plank, "Checkpointing for Peta Scale Systems: A Look into the Future of Practical Rollback-Recovery," IEEE Trans. Dependable Sec. Comput.1 (2):97-108(2004).

[8] M.Treaster, "A survey of fault-tolerance and fault-recovery techniques in parallel systems, "Technical Report cs.DC / 0501002, ACM computing Research Repository (CoRR), January 2005.

[9] J. T. Daly, "A Model for Predicting the Optimum Checkpoint Interval for Restart Dumps," ICCS 2003, LNCS 2660, Proceedings 4 (2003) 3-12.

[10] Yudun Liu, Raja Nassar, Chokchai (box) Leangsuksum, Nichamon Naksinehaboon, Mihaels Paun, Stephen L. Scott, "An Optimal Checkpoint /Restart Model for a Large Scale High Performance Computing System," IEEE Trans. 2008.

[11] Y. Liu, "Reliability Aware Optimal Checkpoint / Restart Model in High Performance Computing, PhD Thesis," Louisiana Tech university, Ruston, LA, USA, May-2007.

[12] R. Geist, R. Reynolds, and J. Westall," Selection of a checkpoint interval in a critical-task environment," IEEE Trans. Reliability, 37, (4), 395-400 (1988).

[13] H. Paul Hargrove and C. Jason Duell, "Berkeley lab checkpoint / restart (BLCR) for Linux clusters", Journal of Physics, Conference series 46 (2006), 494-499, SciDAC 2006.

[14] James S. Plank and MichG.Thomason, "The Average Availability of Parallel Checkpointing Systems and Its Importance in Selecting Runtime Parameters",29th Internatioonal symposium on Fault Tolerant Computing , Madison WI, Jun-1999, pg 250-259.

AUTHORS PROFILE

MR. Mallikarjuna Shastry P.M has received his B.E. and M.Tech in Computer Science and Engineering from Karnataka University Dharwar, and Vishweshwaraiah Technological University, Belgaum, Karnataka, India respectively. He is currently pursuing Ph.D on "Analysis of Fault Tolerant Methods and Performance Evaluation in Distributed Systems " under the guidance of Prof. Dr. K.Venkatesh at M.S.Ramaiah Institute of Technology, Bangalore-54, Karnataka, India. He is working as a Professor in the department of Computer Science and Engg. at Sapthagiri College of Engg, Bangalore, Karnataka, India. He has totally 18 years of teaching experience in Computer Science and Engg.

Prof. Dr. K. Venkatesh has received his M.Sc. in Physics from Mysore University in 1973 and MS from BITS Pilani in 2001 respectively. He has received Ph.D in 1980 from Mysore University. He is currently working as a professor at M.S.Ramaiah Institute of Technology, Bangalore-54 and has totally 30 years of teaching experience.