Analysis of Dependencies of Checkpoint Cost and Checkpoint Interval of Fault Tolerant MPI Applications

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

Abstract – In this paper, we have analysed i) the relationship between the checkpoint cost and the optimal checkpoint interval and ii) the relationship between the checkpoint cost and the number of processors (processes) and we have also determined the optimal number of processors (processes) required for executing the fault tolerant MPI applications.

We have presented an experimental study in which, we have used an optimal checkpoint restart model [1] with Weibull's and Exponential distributions to determine the optimal checkpoint interval.

We have observed that, the optimal checkpoint intervals obtained using Weibull's distribution, produce minimal average completion time; as compared with the optimal checkpoint intervals obtained using Exponential distribution.

The optimal checkpoint interval is approximately directly proportional to the checkpoint cost and inversely proportional to shape parameter.

The study indicates that, the checkpoint cost of MPI applications increases with the number of processors (processes) used for execution.

We have determined the optimal number of processes (processors) required to execute the MPI applications considered in this paper, as 4.

Keywords: OPEN MPI, Fault Tolerance, Optimal Checkpoint Interval, Checkpoint Cost, Speedup, Efficiency.

I. Introduction

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 [2]. When a processor fails or aborts before the completion of execution of the MPI application, the MPI application is restarted from the beginning. Hence, it is required to provide the fault tolerance to all the MPI applications which run on multiple processors or processes [1][3]-[7].

Fault tolerant MPI applications are checkpointed periodically and the checkpoints are stored either locally or sent to a remote server. When a fault tolerant MPI application fails/aborts, it is restarted from the most recently saved checkpoint on a local disk. Hence, it is not required to restart the application from the beginning [1][3]-[7].

The rollback recovery protocols such as uncoordinated, coordinated, communication induced checkpointing and message logging protocols [5] can be used to achieve the fault tolerance. We have used the coordinated checkpointing protocol [5][7] to provide the fault tolerance to MPI applications considered in this paper.

MPI collective communications are used to develop the MPI applications [7]. The optimal checkpoint restart model (OCRM) developed by M. Shastry et al. [1] has been used in this paper, to determine the optimal checkpoint interval, when the scale parameter and shape parameter of Weibull's distribution are known along with checkpoint cost.

Fixed checkpoint interval is used to checkpoint the MPI applications considered in this paper [8]. The checkpointing of MPI applications includes different costs like, checkpoint cost, rollback cost, and restart cost [1][8][9]. These costs are collectively called the checkpoint overheads. The average completion time of an MPI application is obtained by adding the checkpoint overheads to the execution time of MPI application [1][8][9].

We would like to determine i) the optimal number of processes (processors) required to execute the MPI applications of different size, so that the average completion time of these MPI applications with checkpointing would be minimal, ii) the relationship between the checkpoint cost and the optimal checkpoint interval, and iii) the relationship between checkpoint cost and number of processes used to execute the MPI applications considered in this paper.

Rest of the paper is organized as follows. In section 2, the related works carried out on the study of relationship between checkpoint cost and checkpoint interval, checkpoint / restart models used to determine checkpoint intervals are discussed.

In section 3, the different parameters used and the assumptions made in this paper are presented. In section 4, determination of the optimal checkpoint interval using Weibull's and Exponential distributions is discussed briefly.

In section 5, the equations used for determining the average completion time of the MPI applications are presented. In section 6, the results of the case studies considered in this paper are presented. In section 7, the experimental setup used in the analysis is discussed and in section 8, conclusions are presented.

II. Related Works

An excellent survey on rollback recovery protocols has been carried out by Elnozahy et al.[5]. Chandy[10][11] and Treaster[12] have presented a survey on rollback and recovery strategies used in fault tolerant MPI applications.

M. Shastry et al. [8] have shown that the fixed checkpoint interval reduces the checkpoint overheads of fault tolerant MPI applications as compared with the incremental or varying checkpoint interval. Hence, we have used the fixed checkpoint interval in our experiment to checkpoint the MPI applications.

Geff et al. [9] have used a simulator to determine the relationship between the checkpoint interval and the various checkpoint overheads.

The OCRM discussed by M.Shastry et al. [1] have shown the significant improvement with regard to the total wastage time as compared with the checkpoint restart models developed by Yudun liu et al. [13][14] and Bouguerra Mohammed Slim et al. [15],

The checkpoint interval obtained by OCRM [1] using Weibull's and Exponential distributions, produces minimal checkpoint overheads as compared with Yudun liu et al. [13][14] and Bouguerra Mohammed Slim et al. [15].

Hoda El-Sayed et al. [16] and Xin Wang [17] have discussed about the speedup and the efficiency of MPI applications.

III. Parameters/ Notations Used and Assumptions Made.

A. Parameters and Notations Used.

The parameters/notations used in this paper are presented in the table 1.

Table 1. Parameters/Notations used.

Parameter /Notation	Meaning
T_C	Optimal Checkpoint interval.
T_S	Time required to save the checkpoint on a local disk (Checkpoint Cost).
β	Shape Parameter of Weibull's distribution.
α	Scale Parameter of Weibull's distribution.
T_i	Execution time till a failure occurs in i^{th} cycle.
N_i	Number of checkpoints taken till a failure occurs in i^{th} cycle.
RB_i	Rollback cost due to a failure in i^{th} cycle
CC_i	Checkpoint cost in i^{th} cycle
R	Restart cost(time required to resume the execution of the application after a failure).
TL_i	Total time lost in i^{th} cycle.
F	Number of failures.
TL	Total time lost due to F failures during the execution of MPI application.
ET	Execution Time of MPI application without checkpointing.
ACT	Average Completion Time of MPI Application.
T_1	Time required to execute the application sequentially (using a single processor) without checkpointing.
T_P	Time required for parallel execution of the same application using p processors without checkpointing.
Mat4k	MPI Application, which multiplies two matrices of size 4000 * 4000 integers.
Mat5k	MPI Application, which multiplies two matrices of size 5000 * 5000 integers.
Prime4L	MPI Application, which generates 4 lakh prime numbers on each process(processor).
Prime5L	MPI Application, which generates 5 lakh prime numbers on each process (processor).

B. Assumptions Made

We have made the following assumptions, which are similar to the assumptions made in [1] to checkpoint the MPI applications considered in this paper.

1. A series of random failures Fi (i = 1, 2, 3,..N) may interrupt the execution of MPI application.

2. A separate monitoring software system is used to monitor continuously the failure of a fault tolerant MPI application.

3. Checkpoint interval (T_C) is fixed and a checkpoint is taken periodically after the time T_C.

4. When a failure occurs during the execution of MPI application, MPI application is rolled back to the most recent checkpoint using locally stored checkpoint file.

5. Time required for writing a checkpoint (T_S) onto a local disk is a constant and only one copy of the most recent checkpoint is stored on a local disk.

6. Time required for resuming/restarting (restart cost, R) the MPI application from the most recent checkpoint is a constant.

IV. Determination of Optimal Checkpoint Interval

A. Using Weibull's Distribution

The optimal checkpoint interval T_C is obtained by using the optimal checkpoint restart model (OCRM) developed by [1] using Weibull's distribution. The algorithm Estimate-T_C ( ) of [1] is used with the shape parameter β and the scale parameter α for the given checkpoint cost, T_S. The shape parameter β is set to 0.5.

B. Using Exponential Distribution.

The optimal checkpoint interval T_C is obtained by using the optimal checkpoint restart model (OCRM) developed by [1] using Exponential distribution. The algorithm Estimate-T_C ( ) of [1] is used with the shape parameter β and the scale parameter α for the given checkpoint cost, T_S. The shape parameter β is set to 1.0.

V. Determination of Total Time Lost

The number of checkpoints to be taken in a cycle before the occurrence of a failure can be determined by [1][8]

$$N_i = \lfloor T_i / (T_C + T_S) \rfloor \qquad (1)$$

Then, the cost of checkpoint in i^{th} cycle is computed as follows [1][8].

$$CC_i = N_i \, T_S \qquad (2)$$

The cost of rollback in i^{th} cycle is then computed as follows [1][8].

$$Rb_i = (T_i - N_i \, (T_C + T_S)) \qquad (3)$$

The time lost in i^{th} cycle TL_i due to a failure can be obtained by adding checkpoint cost, rollback cost and restart cost together as follows[1][8].

$$TL_i = CC_i + Rb_i + R \qquad (4)$$

The time lost in ith cycle can also be computed [1] as follows by substituting (2) and (3) in (4).

$$TL_i = T_i - N_i \, T_C + R \qquad (5)$$

The total time lost during the execution of a fault tolerant MPI application is determined using the following equation [1][8].

$$TL = \sum_{i=1}^{F} TL_i \qquad (6)$$

The average completion time of a MPI application is computed as follows

$$ACT = ET + TL \qquad (7)$$

VI. Case Studies

A. Variation in Execution Time (*ET*) with Number of Processors (Processes).

Variation in execution time of Mat4k and Mat5k with number of processes is represented in figure 1 a). Figure 1a) shows that, the execution time of Mat4k and Mat5k is very high, when number of processes is 1. When number of processes is increased to 2, the execution time of both applications is reduced to almost 50%. When the number of processes is increased to 4 and 8, the execution time of Mat5k increases linearly, but, the execution time of Mat4k remains almost constant. When the number of processes is increased to 16 and 32, the execution time remains almost constant for both applications as shown in figure 1a).

Variation in execution time of Prime4L and Prime5L with number of processes is represented in figure 1 b). Figure 1b) shows that, the execution time of Prime4L and Prime5L is very high when number of processes is 1. When number of processes is increased to 2, 4 and 8 the execution time of both applications is reduced by 26% to 40%.

The execution time of both applications remains almost constant, when the number of processes is increased to 16 and 32 as shown in figure 1b).

When P processors (processes) are used to execute the MPI applications, the speed up and the efficiency [16][17] are computed as follows.

Speed up = T_1 / T_P.

Efficiency = Speed up / P.

Variation in speed up with the number of processes used for executing the MPI applications considered in this paper, is represented in the figure 1c).

The figure 1c) shows that, when the number of processes is between 1 and 4, the speed up is between 1 and 2 in all the 4 cases. Speedup remains same in case of Mat4k and Mat5k, when the number of processes is between 1 and 32. But the speedup increases gradually in case of Prime4L and Prime5L, when the number of processes is between 1 and 16 and remains constant, when the number of processes is between 16 and 32.

It is clear from the figure 1c) that, the speedup remains same in all the four cases, when the number of processes is 4.

Variation in efficiency with the number of processes used for executing the MPI applications considered in this paper, is represented in the figure 1d).

The figure 1d) shows that, when the number of processes is between 1 and 4, efficiency is between 0.5 and 1 in all the 4 cases. Efficiency reduces gradually in all the 4 cases, as the number of processes is increased from 4 to 16 and reaches almost zero, when number of processes is 32.

When the number of processors (processes) used for executing the MPI applications is P, speedup is above 2 and the efficiency is between 0.5 and 1, the optimal number of processors (processes), P_O, to be used for executing MPI applications is, $P_O = P$.

Since the speedup is 2 and the efficiency is above 0.5 as shown in figures 1c) and 1d) respectively, in all the 4 cases, we conclude that, the optimal number of processors (processes) to be used for executing all the 4 MPI applications considered in this paper, is, 4 ($P_O = 4$).

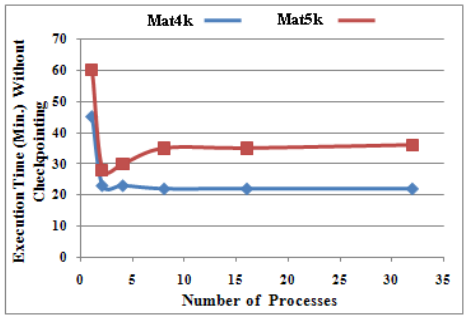

Fig 1 a). Variation in Execution Time of Mat4k and Mat5k.

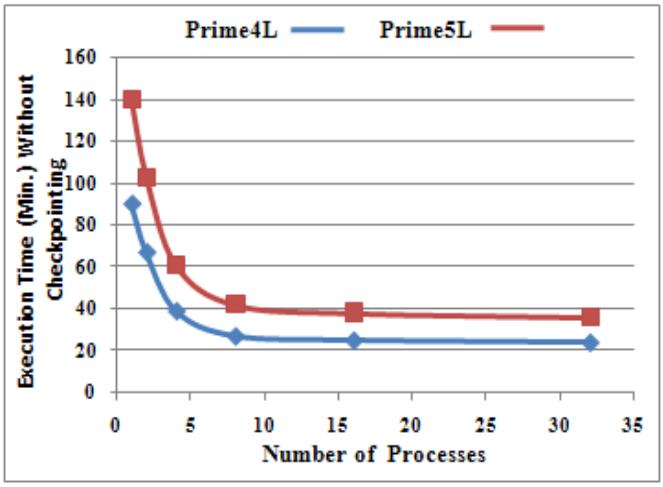

Fig 1 b). Variation in Execution Time of Prime4L and Prime5L.

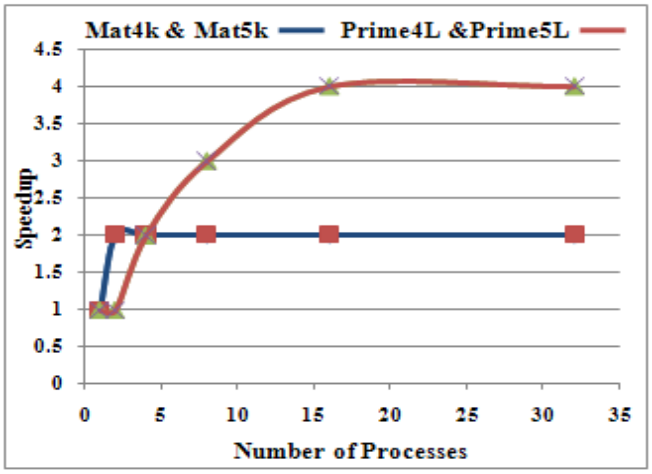

Fig 1c). Variation in Speedup.

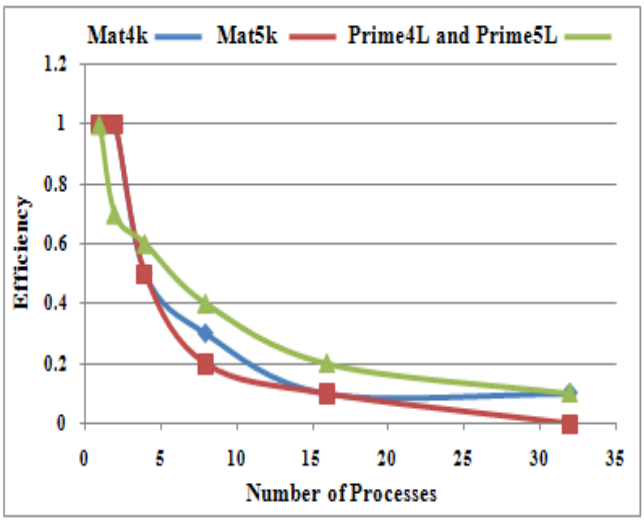

Fig 1d). Variation in Efficiency.

B. Variation in Checkpoint Cost with Number of Processes (Processors).

Figures 2a) and 2b) present the variation in checkpoint cost of Mat4K, Mat5K and Prime4L, Prime5L MPI applications with the number of processes respectively.

It is clear from the figures 2a) and 2b) that, the checkpoint cost of MPI applications increases linearly as the number of processes increases from 1 to 32.

Thus, the checkpoint cost is directly proportional to the number of processes used for executing the MPI applications.

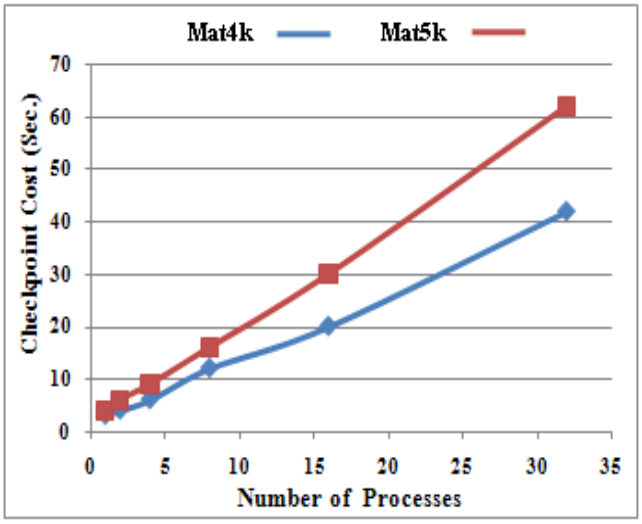

Fig 2 a). Variation in Checkpoint Cost of Mat4k and Mat5k.

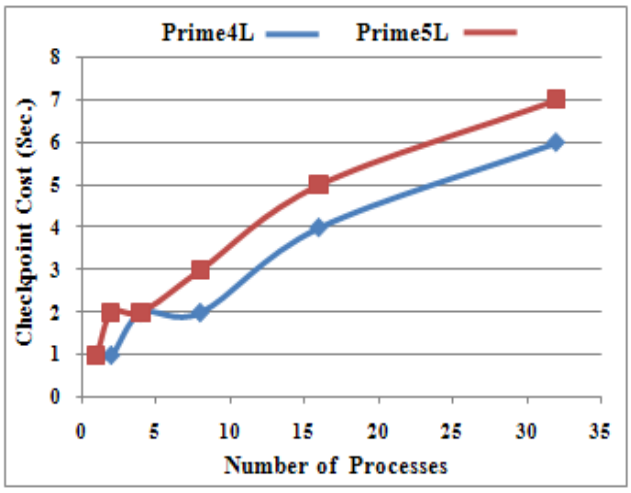

Fig 2 b). Variation in Checkpoint Cost of Prime4L and Prime5L.

C. Variation in Optimal Checkpoint Interval with Checkpoint Cost.

Figure 3 represents the optimal checkpoint intervals obtained from OCRM [1] using Weibull's distribution with shape parameter, $\beta = 0.5$ and Exponential distribution with shape parameter, $\beta = 1.0$ for the different values of checkpoint cost, T_S of MPI applications considered in this paper. The scale parameter α, is set to 4.8 minutes ($\lambda = 300$ failures per day).

In figure 3, when $\beta = 0.5$, the T_C remains 1.5 minutes, when T_S is less than 8 seconds and T_C increases to 2 minutes, when T_S is between 9 and 16 seconds. The T_C value increases gradually when T_S is between 16 and 30 seconds. The value of T_C decreases gradually further when T_S is above 30 seconds.

When $\beta = 1.0$, the optimal T_C remains 0.5 minutes, when T_S is less than 5 seconds and T_C increases to 1 minute when T_S is between 5 and 6 seconds. The T_C value increases gradually when T_S is between 10 and 42 seconds. The value of T_C remains almost constant, when T_S is above 42 seconds.

Thus, it is clear from the figure 3 that, the optimal checkpoint intervals obtained from OCRM [1] using Exponential distribution with $\beta = 1.0$ are low as compared to the optimal checkpoint intervals obtained from OCRM [1] using Weibull's distribution with $\beta = 0.5$. This clearly indicates that, the number of checkpoints to be taken for an MPI application is more in the case of Exponential distribution ($\beta = 1.0$) as compared with the Weibull's distribution($\beta = 0.5$). It is also clear from the figure 3 that, the optimal checkpoint interval is approximately directly proportional to the checkpoint cost, T_S of MPI application and inversely proportional to the shape parameter, β of Weibull's distribution.

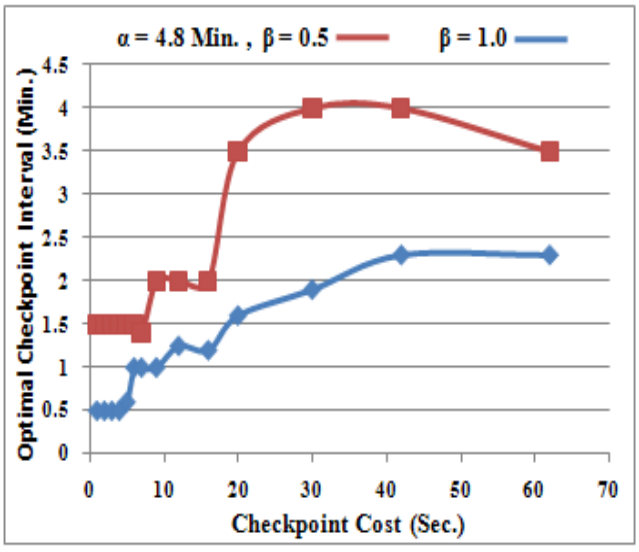

Fig 3. Variation in Optimal Checkpoint Interval

D. Variation in Average Completion time with Number of Processors (Processes).

Figures 4 and 5 represent the variation in the average completion time of Mat4k and Mat5k with number of processes respectively. Figures 6 and 7 represent the variation in average completion time of Prime4L and Prime5L with number of processes, respectively. The average completion time of all 4 MPI applications is computed using the equation (7).

The average completion time of Mat4k, as shown in the figure 4, is almost same for $\beta = 0.5$ and $\beta = 1.0$, when the number of processes is between 1 and 16. Similarly, the average completion time of Mat5k, as shown in figure 5, is almost same for $\beta = 0.5$ and $\beta = 1.0$, when the number of processes is between 1 and 16

Since, the optimal checkpoint intervals obtained from OCRM [1] using exponential distribution with $\beta = 1.0$ are low as compared with the optimal checkpoint intervals obtained from OCRM [1] using Weibull's distribution with $\beta = 0.5$ as shown in figure 3, the average completion time of Mat4k is 9% less and Mat5k is 7% less, when $\beta = 0.5$, as shown in the figures 4 and 5 respectively, when the number of processes is above 16. This result is as expected from OCRM[1], because, the number of checkpoints to be taken will be more in the case of Exponential distribution with $\beta = 1.0$ as compared to the Weibull's distribution with $\beta = 0.5$.

The average completion time of the Prime4L, as shown in figure 6, is almost same for both $\beta = 0.5$ and $\beta = 1.0$, when the number of processes is between 1 and 32. Similarly, the average completion time of the Prime5L, as shown in figure 7, is almost same for both $\beta = 0.5$ and $\beta = 1.0$, when the number of processes is between 1 and 32. But, still the average completion time of these applications is 3 % less, when $\beta = 0.5$, as shown in figures 6 and 7.

Thus, it is clear from the figures 4, 5, 6 and 7 that, the optimal checkpoint intervals obtained from OCRM [1] using Weibull's distribution with $\beta = 0.5$, yield minimal average completion time as compared to the optimal checkpoint intervals obtained from OCRM [1] using Exponential distribution with $\beta = 1.0$, irrespective of the number of processes used to execute the MPI applications considered in this paper.

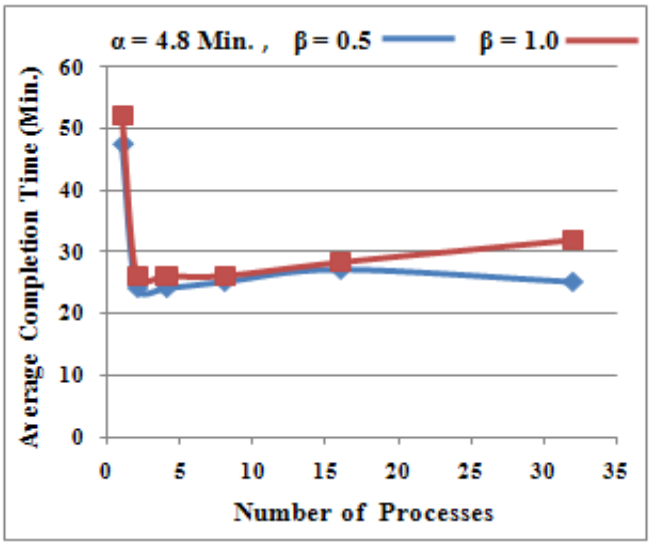

Fig 4. Average Completion Time of Mat4k

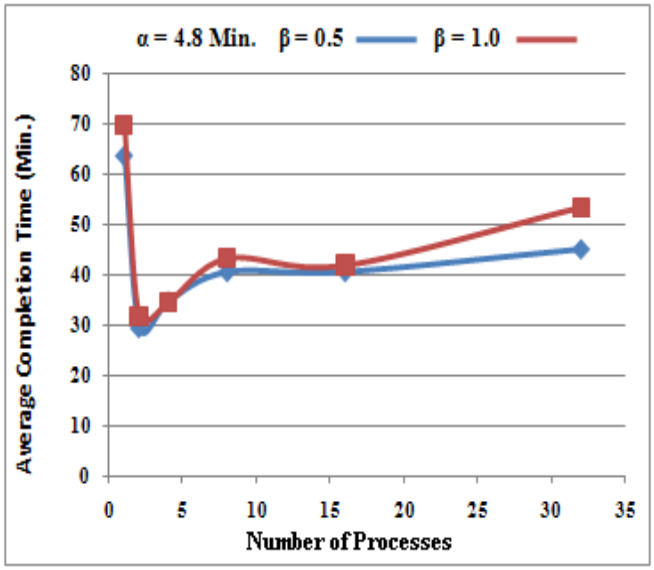

Fig 5. Average Completion Time of Mat5k

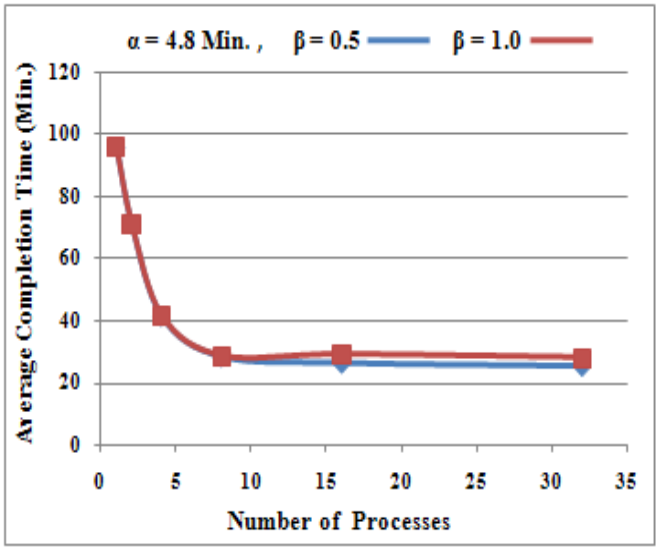

Fig 6. Average Completion Time of Prime4L

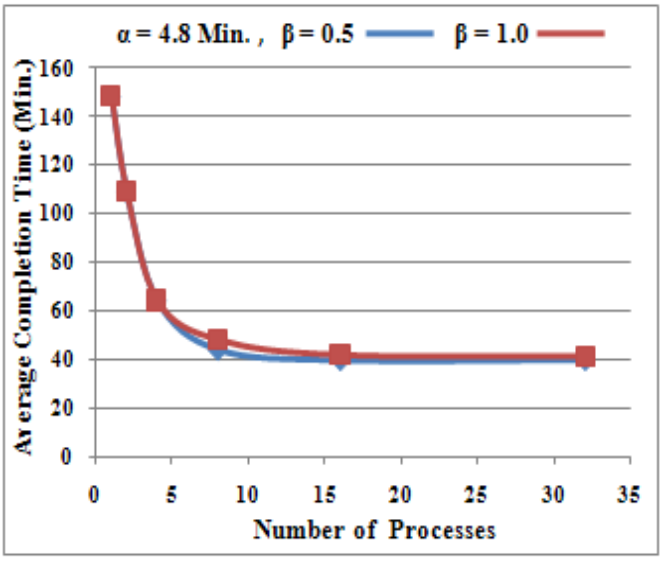

Fig 7. Average Completion Time of Prime5L

VII. Experimental Setup

An AMD Athlon ™ II X_4 630 (Quad Core) processor with 4 GB of RAM is used to execute the MPI applications.

We have implemented 4 MPI applications, Mat4k, Mat5k, Prime4L and Prime5L in "C" programming by using OPEN MPI [18] on a standalone system. OPEN MPI [18] allows the execution and checkpointing of MPI applications on standalone systems also. MPI collective communication methods like MPI_Scatter() and MPI_Bcast() are used to distribute the data to different processes involved in the cluster[7].

Berkeley Lab's Checkpoint restart (BLCR) technique along with OPEN MPI is used to checkpoint the above MPI applications. BLCR is a LINUX based coordinated checkpointing protocol [19]. We have used fixed checkpoint interval to checkpoint the MPI applications [8], as it reduces the total waste time due to the checkpointing of MPI applications.

The above mentioned MPI applications are executed and checkpointed periodically after the time T_C. We have generated $\lambda = 300$ failures per day ($\alpha = 4.8$ minutes) during the execution of above MPI applications.

Weibull's distribution with $\beta = 0.5$ and Exponential distribution with $\beta = 1.0$ are used to determine the optimal checkpoint intervals using OCRM [1], when $\alpha = 4.8$ minutes.

The monitor program is developed in a shell script. This program runs at the background and monitors continuously the execution of the MPI applications. Once, the monitor program learns that the MPI application has failed due to an interruption; it uses the most recent checkpoint stored locally and resumes the execution of the MPI application from the most recent checkpoint.

VIII. Conclusions.

Execution time of the MPI applications as shown in figures 1a) and 1b), reduces to more than 30% to 50% when the number of processes is increased from 1 to 8 and the execution time almost remains constant when the number of processes is increased from 8 to 32.

Since, when the number of processes is 4, for all the 4 applications, the speedup is 2 and the efficiency is above 0.5 as shown in figures 1c) and 1d) respectively, we conclude that, the optimal number of processes to be used for executing all the 4 MPI applications considered in this paper, is 4.

Figures 2a) and 2b) show that, the checkpoint cost of MPI applications increases gradually as the number of processes increases from 1 to 32. Thus, the checkpoint cost is directly proportional to the number of processes used to execute an MPI application.

Figure 3 shows that, the optimal checkpoint intervals obtained using Exponential distribution with $\beta = 1.0$ are low as compared to the optimal checkpoint intervals obtained using Weibull's distribution with $\beta = 0.5$. This clearly indicates that, the number of checkpoints to be taken for an MPI application is more in the case of Exponential distribution ($\beta = 1.0$) as compared with the Weibull's distribution ($\beta = 0.5$).

It is also clear from the figure 3 that, the optimal checkpoint interval is approximately directly proportional to the checkpoint cost, T_S of MPI application and inversely proportional the shape parameter, β of Weibull's distribution.

It is quite clear from the figures 4, 5, 6 and 7 that, the optimal checkpoint intervals obtained from OCRM [1] using Weibull's distribution with $\beta = 0.5$, yield minimal average completion time as compared with the optimal checkpoint intervals obtained from [1] using Exponential distribution with $\beta = 1.0$. This is because; the number of checkpoints to be taken is more in case of Exponential distribution as compared with Weibull's distribution.

Our results discussed in this paper again emphasis that, when the scale parameter α and the checkpoint cost, T_S of MPI applications are known, the Weibull's distribution can be used to determine optimal checkpoint intervals from OCRM [1] for any given value of shape parameter, β.

REFERENCES

[1] Mallikarjuna Shastry P.M and K.Venkatesh," An Optimal Checkpoint Restart Model using Weibull's and Exponential distributions for Fault Tolerant Large Scale MPI applications", Being processed by JPDC, Elsevier, Un Published.

[2] A.R. Adiga, G.almasi, and et al., "An Overview of the BlueGene/L Supercomputer," In Proceedings of Supercomputing, IEEE/ACM Conference, pp60-60,2002.

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

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

[5] E. N. Elnozahy, Lorenzo Alvisi, Yi-Min Wang, and B. David Johnson, " A Survey of Rollback-Recovery Protocols in Message-Passing Systems", ACM Computing Surveys, Vol. 34, No. 3 , pp. 375-408, Sep-2002.

[6] 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.

[7] Mallikarjuna Shastry P.M, K.Venkatesh," Performance Evaluation of Coordinated Checkpointing Protocol using MPI point to point and collective communications", IJACC, Accepted, in Press.

[8] Mallikarjuna Shastry P.M. and K.Venkatesh, "Selection of a Checkpoint Interval in Coordinated Checkpointing Protocol for Fault Tolerant Open MPI," IJCSE, Vol. 02, No. 06, pp 2064-2070, Sep - 2010.

[9] Geff Vallee, Anand Tikotekar, Stephen L Scott, "Impact of Fault Tolerant Policies: Feasibility Study", Oak National Laboratory, Apr-2008.

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

[11] 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

[12] 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.

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

[14] 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.

[15] Bouguerra Mohammed Slim, Thierry Gautier, Denis Trystram, Jean ,Marc Vincent, "A New flexible Checkpoint/Restart Model," INRIA, Dec-2008.

[16] Hoda El-Sayed and Eric Wright," Sparse Matrix Multiplication Using UPC," Proceedings of the World Congress on Engineering 2007, Vol. I, WCE 2007, July 2 - 4, 2007, London, U.K.

[17] Xin Wang," Scalable Parallel Matrix MultiplicationAlgorithms with Application to a Maximum Entropy Problem, Sep – 1997.

[18] The Open MPI Team,"Open MPI Checkpoint/Restart Fault Tolerance User's Guide," OPEN MPI Team, Oct-2008.

[19] 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.

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 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 a professor at M.S.Ramaiah Institute of Technology, Bangalore-54 and has totally 30 years of teaching experience.