An Optimal Checkpoint Automation Mechanism for Fault Tolerance in Computational Grid

Ch. Ramesh Babu

Abstract-- The vast dynamic virtual computing systems are more often vulnerable to failure due to heterogeneous and autonomic nature, so that grid application may loss several hours/days of computation. Checkpointing is the widely used approach to provide fault tolerance in computational grid environment. In this work we have developed a novel approach for optimal checkpoint automation by forecasting failure patterns in computational resources using Hidden Makov Model (HMM) so as to select optimal failure resources to reduce checkpoints in grid TORQUE resource manager. The failure patterns can be predicted by using HMM to assign checkpoint interval for each grid resource and to provide automatic replica (context file) when a failure occurs. The experimental results has shown that the proposed methodology considerably reduces the checkpoint overhead, storage space and turn-around time when compared to adaptive checkpointing approaches.

Index Terms-- Automatic checkpointing, Computational grid, Data logs, Fault-tolerance, optimal checkpointing, Replication.

1. INTRODUCTION

Computational grid is an independent multiple-owner heterogeneous architecture to accomplish huge computationally intensive tasks. The computational grids are the current trend advanced technology to execute the large scale application in assorted heterogeneous resources. Advanced computing technologies are changing to virtual resources sharing, resource-pooling in dynamic environment. Most frequently computational grid faces problems with delay and job failures due to heterogeneity and autonomic nature of grid's resources. Many times computational grids are prone to failure due to lack of proper resource scheduling and fault tolerance mechanism. So the failure rate of the nodes is relative to the number of processors. The computational grid workload-logs study says that system failure occurred 2 to 5 times per day on a large computing system which has about on average of 5000 processors [1]. Even though the number of failures or failure rate of each processor is somewhat low, it might affect the node's reliability. As a result it is need to make a fault tolerant system. Most of the computational problems, the complexity is habitually exacted using the number of failure nodes with respect to the number of calculations necessary to solve them and to find the complexity. So the large computational problems take much time to solve it.

In many algorithms computational problems which have more calculation and computational capability such as climatologic forecast, analysis of weather forecasting, scientific simulations, seismic analysis and the genome sequencing like many computational applications needs to investigate more number of parameters which requires large scale parallel processing systems. High performance computing (HPC) can be used effectively to solve and address these types of issues. In HPC the jobs are allocated to complex heterogeneous group of dissimilar nodes, in order that jobs might be executed concurrently in independent nodes or processors [2, 3].

Typically these sophisticated computational resources require fault tolerance mechanism. According to the large scale, dynamic virtual computational resources such as NASA iPSC, LANL CM5 work-load-logs traces shows that job delay or job failure is the major challenging task to provide fault tolerance [11, 12, 13 and 14]. Since very robust fault tolerant checkpointing and scheduling algorithms are used to handle various resource allocations in computational grids. However there are some performance issues needs to enhance the parameters such as number of checkpoints overhead, checkpoint storage space and low throughput. Consequently the major issue in fault-tolerance is to effectively tolerate failures using job checkpointing and fault-tolerant job scheduling to make efficient failure handling mechanism in presence of faults. Currently using techniques for fault tolerance in the widely held computational applications are adaptive checkpointing and replication. Since most of the job

- *Ch Ramesh Babu is currently working an Associate Professor in the Dept of CSE in Malla Reddy Engineering College(Autonomous),Hyderabad, ,TS. He completed his B.Tech(CSE) and M.Tech(CS) from JNT U, Hyderabad. He submitted his PhD(CSE) thesis in JNTUK, Kakinada ,AP, India., and is waiting for viva voce.. E-mail:chramesh522@gmail.com.*

checkpointing techniques are not merely based on scheduling algorithm [5, 17].

Since very robust scheduling algorithms are used to handle various resource allocation in computational grids. However, it is necessary in earlier studies to remedy the failures and delay of executing jobs with respect to resource availability, which can handle scheduling and failures in any large scale high performance computational applications. The major issue is fault-tolerance with regard to job scheduling and failure handling mechanism. Currently using techniques for fault tolerance in the widely held computational applications are adaptive checkpointing [15] or without checkpoint automation and replication [5, 7].

The main objective of computational grids is to execute large computational tasks more effectively to improve throughput and to save processing time. Therefore the user submits jobs to the Grid Scheduler (GS) along with their Quality of Service (QoS) requirements. These requirements may include the deadline in which users want jobs to be executed, the type of the resources required to execute the job and the type of the platform needed. The GS of the present scheduling systems allocates each job to the most suitable resource. In case of fault free resource, the results of executing the job are returned to the user after completion of the job. If a grid resource fails during the execution of a job, then that job is rescheduled on another resource which starts executing the job from scratch. This leads to more time consumption for the job than expected time so that the user's QoS requirements may not satisfy [4]. In this paper we have presented heuristics that resolves number of checkpoint overhead issue so as to provide high job throughput in presence of failure.

The major drawback in grid middleware is lack of synchronization among the nodes and the grid middleware. Whenever a node fails it has to recover from last saved checkpoint to overcome job delay in execution. But many times there is an overhead due to many checkpoints in without checkpoint automation approaches. In order to tackle this we have proposed an optimal checkpoint automation strategy which effectively changes the dynamic behavior of the grid middleware working mechanism according to the grid resource failure information.

2.RELATED WORK

Most of the time in computational grid more number of checkpoints causes checkpoint overhead, so as to reduce the overheads, different approaches have been developed. One of the conventional techniques of checkpointing is incremental checkpointing which stores only modified data during checkpointing. At the initial checkpointing operation all pages of the program address space is saved. After each checkpointing operation all the modified pages will be updated in checkpoint server [8]. Here the checkpoint library needs to be system initiated rather than application-initiated in-order to truly adaptive. Real measurements on an actual file system are still required to validate. Moreover a large file system bandwidth might be the hashing overhead.

In min-max checkpoint placement algorithm which determines the uncertain circumstances in case of the system failure. Here checkpoint interval is considered without the complete knowledge on system failure distribution. Even if optimal checkpoint interval is found before, the checkpoint may not be possible to change over time [9, 18]. When system failure time distribution is known then the optimal checkpoint interval may change and less number of checkpoints may be possible.

According to the adaptive task checkpointing and replication scheme, propose two principles, Last Failure Dependent Checkpointing (LastFailureCP) and Mean Failure Dependent Checkpointing (MeanFailureCP). In LastFailureCP algorithm it omits unnecessary checkpoint placement with reference to the total execution time and failure frequency of the resource. This algorithm keeps a time stamp LF_r that gives the time when the last failure had occurred. Initially checkpointing request will be given at time interval I and then request will be executed by Grid Scheduler by comparing whether $t_c - LF_r <= E_{rj}$, where t_c is the current time and E_{rj} is the execution time of job j on resource r, and LF_r is the last failure time of resource r. If the condition is true then checkpointing is allowed otherwise checkpointing is omitted. In case of MeanFailureCP, the checkpointing interval changes according to the remaining execution time and mean failure interval. Initially the algorithm gives checkpoint request within fixed and preferably short time period t_i [5, 20]. In the above last failure and mean failure algorithms there may not be any scheduling algorithm exists for scheduling jobs to grid nodes. In this Adaptive checkpointing approach, scheduling methods needs to be considered that adapt to dynamically changing and estimations of job execution time to make efficient optimal checkpointing.

In Fault Tolerant Scheduling System Based on Checkpointing method proposes an average failure time and failure rate of resources combined with response time when taking scheduling decisions. The checkpoint interval is calculated using resource failure rate. This shows the effectiveness of in view of resource failure rate and resource failure time over considering the resource fault index [4]. In this methodology if failure history of the resource is unknown then the performance may be decreased.

In Implementing and Evaluating Automatic Checkpointing, an automatic checkpointing was proposed for distributed computing environment extending the LAM/MPI using a basic infrastructure provided by BLCR [6, 19]. Here this method uses the local disks to record the process contexts, which will be better investigated to reduce the traffic in the interconnection network and the overhead in the file server, thus increasing the checkpointing performance [10, 11].

In recent studies on automatic checkpoint based fault tolerance in computational grid proposes the need for checkpoint automation in computational grid and mechanism to combine grid job scheduling and optimal checkpointing process [16, 17 and 19].

3.CHECKPOINT AUTOMATION MODEL FOR COMPUTATIONAL GRID

The aim of this work is to optimize the performance of the grid in the presence of faults and to improve throughput value. When a fault occur a grid resource may not complete its job within the given QoS. The main strategy of the proposed OCA mechanism is to minimize the effect of grid faults and to reduce the fault recovery time using optimal automation of checkpointing so as to minimize the amount of checkpoint overheads. To evaluate the above prototype we have considered faults history so that each resource scheduled based on the next sequence of pattern failures. The failure patterns can be predicted by using HMM to assign checkpoint interval and also to provide automatic failure replica (context file of checkpoint) to the grid resource.

3.1. Optimal Checkpoint Automation architecture (OCA)

The interaction among different components of the OCA is shown in the following Figure.1. The OCA optimizes the checkpoints to improve the efficiency over the execution of the failed job from the last saved checkpoint. Thus it reduces the response time of the job by reducing the time wasted in additional checkpoints storage.

A grid contains multiple grid resources that provide computing services to users. The main component of the OCA is the pbs_scheduler. It receives the jobs with their information from users. Job information includes job number, job type, and job size. Also the user submits QoS requirements for grid application, such as the deadline to complete its execution, the number of required resources and the type of these resources.

The main function of pbs_scheduler is to find and sort the most suitable resources that can execute the job and satisfy user QoS requirements. In order to perform this function, the pbs_scheduler connects to the pbs_server to get information of all available grid resources to execute the job. The pbs_scheduler uses response time, resource failure rate and resource failure time to construct the list of suitable resources that can execute the job.

In Figure.1 OCA architecture contains the components such as pbs_scheduler, pbs_server, Ckpt_server, and autonomous computational resources as portable batch system message oriented middleware (pbs_mom). Each component accomplishes set of tasks, explained in 3.2 and 3.3.

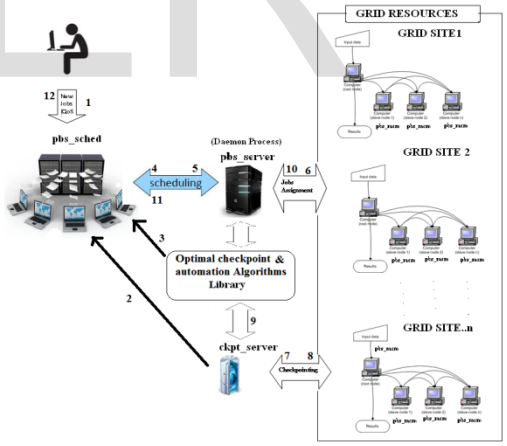

Figure 1: Checkpoint automation in TORQUE resource manager.

3.2. Considerations for optimal checkpoint automation in TORQUE Resource Manager.

- To solve grid application, the grid user submits the jobs along with QoS which requires large computational power for computation.

- Ckpt_server gets the list of failure resources and checkpoint file information using computational resource work-load-logs.
- The pbs_scheduler gets the failure patterns, checkpoint intervals and available resources from Ckpt_server, optimal checkpoint automation library and pbs_server respectively.
- The pbs_server gets the all the information about the resources like response time, failure rate and sorts the list of resources according to resource response time and make new list of available resources with QoS requirements to find checkpoint interval.
- The pbs_scheduler finds the optimal checkpoints for each resource and sends the list to pbs_server to allocate the jobs to resources.
- The pbs_server allocates the jobs to the selected resources.
- Based on the checkpoint interval the Ckpt_server takes the snapshots (context file) from computing nodes. If any failure occurs in a node then latest context job file can be retransmitted automatically.
- If more number of failures occurs in the same grid site then the computational resource can be rescheduled to another site.
- Optimal checkpoint automation library reschedules the jobs to new grid resource through pbs_server.
- Pbs_server collects aggregated results and ships it to grid user.

3.3 Components of OCM in TORQUE

3.3.1. Grid Application:
This component takes grid application from the user which typically requires large scale heterogeneous resources, processing nodes with QoS of an application.

3.3.2. Pbs_ Server:
Pbs_server contains information of all available resources in the grid required by the pbs_scheduler. The information includes resource speed, current load, resource failure rate and total failure time of each resource. For each job j dispatched by the ckpt_server, if ckpt_server receives a job completion message then it sends a message to the pbs_server to increment S (Success) or if ckpt_server receives a response as failure message then send it a message to pbs_server to increment F (Failure), if there is a checkpoint stored then ckpt_server dispatches the not completed part of the job along the checkpoint status to the second resource in the resources list, else ckpt_server dispatches the whole job along to the second resource in the resource list (see Figure 2).
The Pbs_server receives and stores partially executed results of a job from children of pbs_mom. These intermediate results are called checkpoint status. For each job there is only one record of checkpoint status. When ckpt_server receives a new checkpoint status it overwrites the old one. If ckpt_server receives a job completion message from the resource it removes the record of such job.

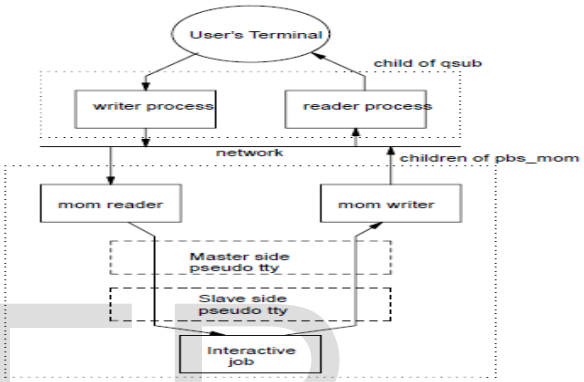

Figure 2: Interactive Job Communication Flow.

3.3.3. Checkpointing Server:
Ckpt_server is an important component of OCA. The main functions of ckpt_server are determining the number of checkpoints for each pbs_mom and the checkpoint interval for each job. ckpt_server receives a job with its assigned list of resources from pbs_scheduler. It connects to ckpt_server to get information about the failure history of grid resources assigned to the job. Based on failure rate of the resource, the ckpt_server determines the number of checkpoints and the checkpoint intervals for each job. Then, it submits the job to the first grid resource in the resources list. The ckpt_server calculates the number of checkpoints and the checkpoint interval for each resource using work-load log files which are collected from server_logs and mom_logs (see Figure 3).

3.3.4. Computational Resource:
This component consists of various heterogeneous virtual computational resources which are managed autonomously at different grid sites.

3.3.5. Pbs_scheduler:

The Pbs_scheduler is the major component in grid architecture. The pbs_scheduler initiates the application with n-number of jobs to allocate grid resource. Now for resources from pbs_server The pbs_scheduler sorts the ach set of job the pbs_scheduler gets the list of available

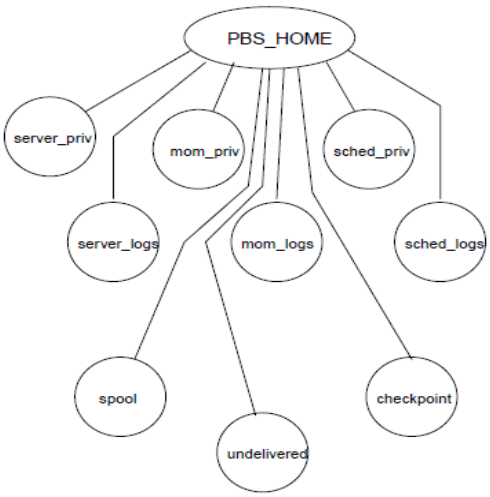

Figure 3: Checkpointing at the Pbs_checkpointing server.

suitable list of resources according to resource response time -RRT, failure rate –FR and average failure time -AFT. Afterwards pbs_scheduler dispatches the list to checkpoint header. The pbs_scheduler schedules the resources using optimal checkpointing approach-OCA with respect to job list acquired from ckpt_server.

3.4 Finding Optimal checkpoint interval

The goal optimizing checkpoint intervals is to minimize the number of checkpoints for each resource. This work presents the design and implementation of the automatic checkpoint mechanism using OpenMPI toolkit, including high level complex applications context file recording, failure detection and application recovery, using BLCR framework in TORQUE.

3.5.1. The objective function:

$$MIN\ TT_{(X)} = \Sigma^{N}_{i=0}(T_{ij}{}^{l} + T_{ij}{}^{l} * F_i{}^{l}) \quad \ldots.(1)$$

Where $T^{l}_{ij} = t_{js} + t_{je} + t_{jr}$ ….(2)

$F_i = t_{jc} + t_{jr}$ ….(3)

Subject To:

(i). $T_{ij}{}^{l} < RT^{i}_{ij}$

(ii). $0 < F_i < 1$

Where, $TT_{(X)}$ is the estimated minimum turnaround time for j resources to carry out application X of i jobs to checkpoint in optimal manner. $T_{ij}{}^{l}$ is the estimated minimum turnaround time for j resources to carry out job i, $F_i{}^{l}$ is the failure resource, t_{js} scheduling time for i^{th} job on resource j, $t^{ij}{}_{e}$ execution time for i^{th} job on resource j, t_{jr} restart time for i^{th} job on resource j.

Find the failure probability density function of resource using exponential distribution as

$$P(k,j\,|\,) = \begin{cases} kje^{-kjx}, & x \geq 1 \\ 0 & , \text{Otherwise} \end{cases} \quad \ldots. (4)$$

Each resource x checkpoint interval I can be generated as using Bayes formula

$$CI_i = P(x_i \mid k, j) = P(k, j \mid x_i) * P(x_i) \quad \ldots.(5)$$

Differentiate interval I(x) with respect to mean and last failure of resources.

$$l1 = \frac{d\,CIi}{dj} \quad \ldots. (6)$$

$$l2 = \frac{d\,CIi}{dk} \quad \ldots. (7)$$

Optimal checkpoint interval can be calculated as

$CI_{i\_new} = (l1+l2)/2$ …. (8)

Where, CI_{i_new} is the new optimal checkpoint interval. The above equations 4-8 are used to simulate grid failure resource with optimal number of checkpoints by using the parameters LastFailure (k) and MeanFailure (j). Here CI_{i_new} is the optimal checkpoint interval which reduces the number of checkpoints, thus the checkpoint overheads are reduced.

3.5.2. Checkpoint automation

In our mechanism, fabrication of snapshot points are generated automatically (checkpoints) in a frequency predicted by the job scheduling strategies, based on OpenMPI collective function calls. The implementation of the automatic checkpoint operations is based on threads created in runtime, which execute the new functionalities inserted into OpenMPI using TORQUE. We have implemented different threads for different applications. The following are the scheduling strategy and thread workflow mechanism. The cycle of steps among the

components of planned tactic of checkpoint plotting and replication are explained in equations 9 to 14.

3.5 Failure forecasting of resources using HMM.

The Hidden Markov Model (HMM) is a state machine. Here the failure states of the model are represented as nodes and the failure transition are represented as edges. The HMM have become the well-known and widely used statistical approach to characterizing the spectral properties failure prediction approaches. HMM is a stochastic modeling tool having an advantage of providing the high reliable and natural way of failure analysis for resources. HMM integrates into the systems involving information related to last and mean failure approaches, currently it is predominant approach for the optimal checkpointing in computational grid.

HMM provides a method which directly estimates the conditional probability of index of failures in resource given a hypothesized identity for index of failures in resource. HMMs consists of two processes namely Hidden and the Observed process. The Hidden process consists of a collection of failure states connected by the resources. And each of these transactions is described by two sets of probabilities:

Steps involved in making HMM's work:

- Estimating conditional probabilities to last and mean failure sequence to given a model OCA methodology.
- Finds the best checkpoint interval which is closely matches the input sequence. This enables to assign optimal checkpoint interval from last failures pattern.
- Prepare a model to using the mean and last failure parameters of checkpoints and its corresponding transition probabilities to best account for a checkpoint library.

The failure patterns of the resources can find using the formula

$$w_j = P(x_i \mid r_j) = P(r_j \mid x_i) * P(x_i) / P(r_j). \quad ...(9)$$

The grid Resource are sorted by using

$$(r_j, w_j) \quad r_j = r_j * w_j \quad ...(10)$$

Finding the checkpoint interval

$$CI = \sum_{i,j=1}^{n,m} (TT_{(X)})/N_{if}; \quad ....(11)$$

Finding number of checkpoints

$$CN = T_j^i/CI_i; \quad ....(12)$$

According to number of checkpoints the resources can be rescheduled using

$$CT_r^i \ \% \ CI_i == 0 \ \&\& RE_r^i > CI_i \quad ....(13)$$

Finding the turnaround time of the application $X_{i=1}^n$.

$$TT_j = T_r^i * (1 + CN_r^i) \quad ....(14)$$

Throughput of the application is

$$n = N / \sum TT^{tp_j} \quad ....(15)$$

Where, $X_{i=1}^n$ Computational jobs ($x_1, x_2, \ldots x_n$), CI- j^{th} resource failure weight, n- Throughput of an application X, CN_j -Number of Checkpoints for resource j, $TT_{(X)}$- Total turnaround time of application X.

Computational grid User can submit jobs through the grid user interface. The application interface receives user jobs and transforms to scheduler. Typically the job information consists of job number, job type, and job size also receives Quality of Service requirements of each job such as the deadline to complete its execution, the number of required resources and the type of these resources. The scheduler assigns each job to the most reliable, suitable, and available resource to execute the job. The most reliable resource is the resource that has a lower fault rate.

4.PERFORMANCE EVALUATION

In this experiment, applications with 100 to 2000 jobs with 10-200 faults are modelled. The size of each job is randomly selected from 1 KB up to 10 MB. The number of resources in the grid can reach up to 10. Different simulation experiments have been conducted with variation in the total number of jobs submitted to the grid and measuring the throughput, turnaround time and the tendency of resources to fail. The proposed OCA approach is compared with the Adaptive algorithms the details of that experiment can be found in Table-1. Based on the experiments the following graphs Figure 2 and Figure 3 are plotted.

4.1 Optimal Checkpoint Automation setup

Most of the existing grid simulators like GridSim, SimGrid do not support fault-tolerant scheduling and checkpointing. To assess the optimal checkpoint

automation operation, we have arranged heterogeneous nodes called NUMA Non-uniform memory architecture environment in TORQUE resource manager which composed of 50 nodes, running Open SUSE Linux 12.2. Each node has 1 GB RAM, 1 GB swap and a Gigabit network card. We have taken the first five nodes to operate Open SUSE Linux with TORQUE resource manager and the remaining nodes to execute Windows-XP to access and operate the jobs through Putty software. The TORQUE resource manager assigns the jobs to the idle resources. We have used the Networking File System (NFS) to store checkpoints into the checkpoint server so that if any one node fails then chronologically the last saved checkpoint can be accessed automatically. At present our claim methodology ropes optimal checkpoint automation for large scale grid application. We have installed TORQUE resource manager service on all nodes and applied auto checkpoint generation based on optimal checkpoint interval. Since it improves the performance of the grid application in presence of a failure when compare to Adaptive checkpointing approaches.

Nodes	Job	Adaptive Algorithms			Optimal checkpoint automation Algorithm		
		CH	M	T	CH	M	T
50	100	10	.7	0.8	3	0.3	.3
50	500	50	.9	3	15	0.5	.8
50	100	100	1	4	32	0.7	1.3
50	150	150	2	6	50	1.0	1.8
50	200	200	2	7	65	1.3	2.6

Table-1: Comparison of adaptive checkpointing with OCA Optimal Checkpointing. In the table the notations CH – checkpoints, M-memory, T-time.

The Table-1 shows the comparison between adaptive checkpointing and optimal checkpoint automation for job processing times.

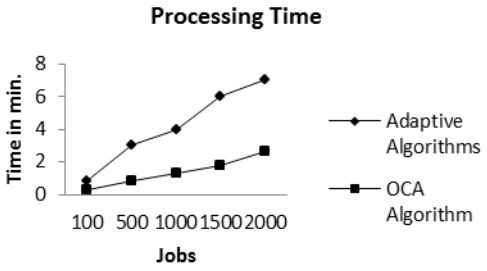

Figure 2: Processing time comparison by varying number of jobs

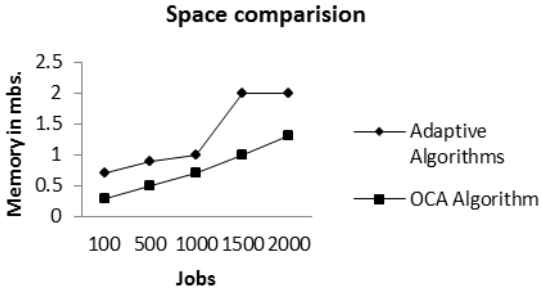

Figure 3: Space comparison by varying number of jobs

The Figure 2, Figure. 3 depict processing time and space comparison of checkpoint algorithms respectively. The proposed system depends on average failure time and last failure rate to make optimal checkpointing decisions using HMM model. The checkpoint interval is calculated merely using resource failure rate prior information. The performance of Optimal Checkpoint Automation algorithm is compared with Checkpointing based Fault-Tolerant Grid System (or adaptive checkpointing Algorithm-CFTG), in which these algorithms depends on the response time and the fault index of resources to make checkpoint interval. But the OCA method uses resource fault index to make checkpoint using Hidden Markov Model states (HMM) to make checkpoint scheduling for calculating effective prediction based checkpoint interval.

Experimental results show that OCA approach effectively schedules jobs in the presence of failures. It improves the turnaround time and throughput when compared with the adaptive checkpointing algorithms. Moreover the failure tendency for the proposed OCA methodology is far better than the adaptive checkpointing algorithm. Thus, it can be concluded that the proposed scheduling system provides better performance when compared to adaptive Algorithm. In this approach the

failure history of resources must be known in prior to make the optimal checkpointing approach using fault tolerant scheduling approach. If the computational resources are new then checkpointing interval might be unpredictable.

5.CONCLUSIONS

In this work we have proposed an optimal checkpoint automation mechanism for computational grids. Here the optimal checkpoints are calculated using the predicted failure patterns of resources using HMM. The unique optimal checkpoint interval is generated for each resource, which yields reduction in number of checkpoints. Hence this methodology considerably reduces the checkpoint overheads by reducing number of checkpoints. The developed model, optimal checkpoint automation (OCA method) is implemented in TORQUE resource manager frame-work, which shows better turnaround time, storage space and throughput of an application compare to adaptive algorithms. Experimental results show that OCA methodology effectively tolerates the resource failures. Therefore we conclude that the developed checkpointing system provides better performance when compare to adaptive checkpointing algorithms.

REFERENCES

[1]. Schroeder, B. Gibson, G.A, "A Large-Scale Study of Failures in High-Performance Computing Systems", Dependable and Secure Computing, IEEE Transactions on (Volume:7 , Issue: 4), 2010.

[2]. G. Kandaswamy, A. Mandal, and D. A. Reed, "Fault tolerance and recovery of scientific workflows on computational grids", in 8th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'08), 2008, pp. 777–782.

[3]. Sangho Yi, Derrick Kondo, Bongjae Kim, Geunyoung Park, "Using Replication and Checkpointing for Reliable Task Management in Computational Grids", IEEE Transactions -2010.

[4]. Mohammed Amoon, "A Fault Tolerant Scheduling System Based on Checkpointing for Computational Grids", IJAST, Vol. 48, November, 2012.

[5]. Maria Chtepen, Filip H.A. Claeys, Bart Dhoedt, "Adaptive Task Checkpointing and Replication: Toward Efficient Fault-Tolerant Grids", ieee-transations on parallel and distributed systems, vol.20, no.2, February 2009.

[6]. Antonio S. Martins Jr., Ronaldo A. L. Gonçalves, "Implementing and Evaluating Automatic Checkpointing", IEEE Transactions -2007.

[7]. Chang, R.-S., National Dong Hwa University, Hualien, Chang, H.-P., Wang, and Y. T., "A dynamic weighted data replication strategy in data grids", In: Computer Systems and Applications, AICCSA 2008, IEEE/ACS 2008.

[8]. Saurabh Agarwal, Rahul Garg, Joes Moreira, "Adaptive Incremental checkpointing for Massively Parallel Systems", ACM, 2004.

[9]. T. Ozaki, T. Dohi, H. Okamura and N. Kaio, "Min-Max Checkpoint Placement under Incomplete Failure Information", Proc. Int'l Conf. Dependable Systems and Networks (DSN '04),June-July 2004.

[10]. Http://www.open-mpi.org/

[11]. Http://crd.lbl.gov/groups-depts/ftg/projects/current-projects/BLCR

[12]. Dilli babu, S., Ramesh Babu, Ch., Subba Rao, Ch.D.V., "An efficient fault-tolerance technique using check-pointing and replication in grids using data logs.", In: publications of problems and application in engineering research—ijpaper.com, vol 04. Special issue 01, 2013.

[13]. Http://gwa.ewi.tudelft.nl/

[14]. Http://www.cs.huji.ac.il/labs/parallel/workload/l_lcg/index.html

[15]. Http://www.adaptivecomputing.com/

[16]. Ch. Ramesh Babu, Ch. D. V. Subba Rao, "Automatic checkpoint based fault tolerance in computational grid", IEEE Conference COMMANTEL-2014.

[17]. G. Molto, V. Hernández, J.M. Alonso, "Automatic replication of WSRF-based Grid services via operation providers" J. of FGCS-2009.

[18]. Greg Bronevetsky, Daniel Marques, Keshav Pingali, Paul Stodghill, "C3: A System for Automating Application-level Checkpointing of MPI Programs", Department of Computer Science, Cornell University.

[19]. Gengbin Zheng, Chao Huang and Laxmikant V. Kal´e, "Performance Evaluation of Automatic Checkpointbased Fault Tolerance for AMPI and Charm++", Department of Computer Science, University of Illinois at UrbanaChampaign.

[20]. Najme Mansouri, "An Effective Weighted Data Replication Strategy for Data Grid", Australian Journal of Basic and Applied Sciences- 2012.