# Fault Tolerance and Recovery for Grid Application Reliability using Check Pointing Mechanism

S.Baghavathi Priya
Research Scholar
Jawarharlal Nehru Technological University
Hyderabad, Andhra Pradesh,India.

Dr.T.Ravichandran
Principal
Hindustan Institute of Technology
Coimbatore, India.

## ABSTRACT
The check pointing mechanism and rollback recovery is a well-known method to achieve fault tolerance in grid computing systems. If any resource or process is tending to be faulty in run time that will be detected by check pointing mechanism through the Task Dependency Graph (TDG) and their respective worst case execution time and deadline parameters are used to decide the schedulability. The common approach is to use rollback-dependent graph or check point graph. The scheduling of concurrent tasks can be done using the proposed Concurrent Task Scheduling Algorithm (CTSA) algorithm to recover from the faulty states using replication or rollback techniques. The earlier fault detection methods are not scalable with the diversity of user applications and the frequency of faults varies dynamically making the faults hard to detect and recover. The check pointing and replication mechanisms have been used in high performance grid computing where the synchronization between communicating processes is needed to enhance the efficiency of check pointing mechanism. The performance improvements over the faulty conditions can be obtained with or without data and process replication. The experimental results show that the CTSA can lead to significant performance gain for a variety of scenarios.

## Keywords
Check pointing, Reliability, Rollback, Replication, Fault tolerance.

## 1. INTRODUCTION
The application reliability in a grid environment depends on the synchronized behavior of many heterogeneous hardware and software resources deployed in a distributed manner. The efficient scheduling of such resources can be achieved in a multi-tier component or service mapping between the processes and the replicated applications and possible rollbacks. A large scale infra structure services and fault report services are needed to claim the maximum reliability for the application deployed or demanded over the grid. Irrespective of the service, the grid components ranging from grid node to grid broker must communicate in asynchronous or synchronous manner to complete the submitted tasks from the grid users. Minimum Execution Time(MET) [9] assigns each task to the resource with the best expected execution time for that task. The motivation of MET is to give each task its best machine. This can cause a severe load imbalance among machines. Minimum Completion Time(MCT) assigns each task, in an arbitrary order to the resource with the minimum expected completion time for that task. Many scheduling algorithms have been designed for grid environments, to solve the problem of mapping a set of tasks to a set of machines. Min-min algorithm is based on the minimum completion time, as is MCT. The Max-min algorithm begins with the set U of all unmapped tasks. Then, the set of minimum completion time M is found. Next, the task with the overall maximum from M is selected as assigned to the corresponding machine. Last, the newly mapped task is removed from U, and the process repeats until all tasks are mapped. The main focus of the paper is to formally specify the behavior needed for the various fault tolerant grid services through check pointing strategy. The proposed check pointing techniques invoke the necessary replicas or rollbacks in order to meet the user application reliability requirements in the case of faults occurred either in the processes and or in the resources.

The paper is organized as follows: Section 3 discusses a generic Grid architecture needed for enhancing the application reliability through check pointing strategy and the essential component services. Section 4 explores the need for a feasible mapping between available application replicas and the reliable roll backs of the erroneous tasks within the grid environment. Section 5 describes the computational model of concurrent activation of processes with the help of their corresponding stacks and the collaboration or various grids component services and the application reliability analysis through process failure probabilities. Section 6 focuses on the validity of the model based on the cost and complexity of the replicas and rollbacks respectively and concludes the work.

## 2. RELATED WORK
The Replica Resource Selection Algorithm (RRSA) proposed in 2010 uses a check point setter through the mean time to release (MTTR) the service or resource is optimized [1].But the mechanism for the faster recovery from the faulty status of the particular process is not considered. A fault tolerant load

balancing model was proposed using a fault detector and manager services through which the check pointing technique was compared [2]. Even with standard interfaces and communications protocols in place, resource heterogeneity and dynamism will likely lead to component interactions that result in faults and failures which when executing grid user applications. Another well-known failure detection problem occurs in asynchronous distributed systems where management functions are decentralized and they may be subject to failure. One important issue is finding efficient methods for check pointing many concurrent, intercommunicating processes, so that in the event of failure, they can resume processing from a common saved state. Early efforts in using checkpoint and process migration in large-scale grids were reported in the Legion system and Cactus. A work flow approach is used in the Linked Environment for Atmospheric Discovery (LEAD) infrastructure to establish a fault tolerant computational grid and the success probability was determined [3]. A Stochastic modeling was introduced to analyze the performance of different check pointing schemes where the time complexity of synchronous and asynchronous check pointing was focused. In Synchronous Check pointing if all the processes take check points at the same time instant, the set of check points would be consistent. But since globally synchronized clocks are very difficult to implement, processes may take check points within an interval. Processes synchronize through system messages before taking checkpoints. These synchronization messages contribute to extra overhead [4]. This model of analysis compared both the pessimistic and optimistic approaches to find out the minimum message logging [5]. A decentralized fault tolerant framework for grid computing was proposed in which node-bound proactive maintenance strategies are identified to reduce the service failures [6].A flexible failure handling framework for grid services was proposed in which fault notification generation was utilized to determine the state of the task in progress as Grid-WFS[7].The Grid Resource Management System GRID–RMS addresses the need for adaptability and scalability of the grid resources to meet the specific needs demanded by the grid users[8]. Saeed Parsa et al. have proposed a new task scheduling algorithm called RASA[10]. K.Etminani et al. have proposed a new algorithm which uses Max-min and Min-min algorithms [11]. The algorithm determines to select one of these two algorithms, dependent on the standard deviation of the expected completion times of the tasks on each of the resources. Therefore grid computing environments should have fault-tolerance mechanisms to increase their reliability.

## 3. GENERIC GRID ARCHITECTURE FOR FALUT TOLERANT GRID APPLICATION

Grid resources include processor clusters, supercomputers, storage devices, and related hardware, together with operating system and other software for managing these resources. Grid resources can also include dedicated software components that may be engaged by users to perform various functions, such as data mining and other analysis. A third category of grid resources are data stores used in data and multimedia grids. The most common technique of check pointing can be applied to improve the fault tolerant behavior of the application by having suitable number of replicas for the critical tasks in the

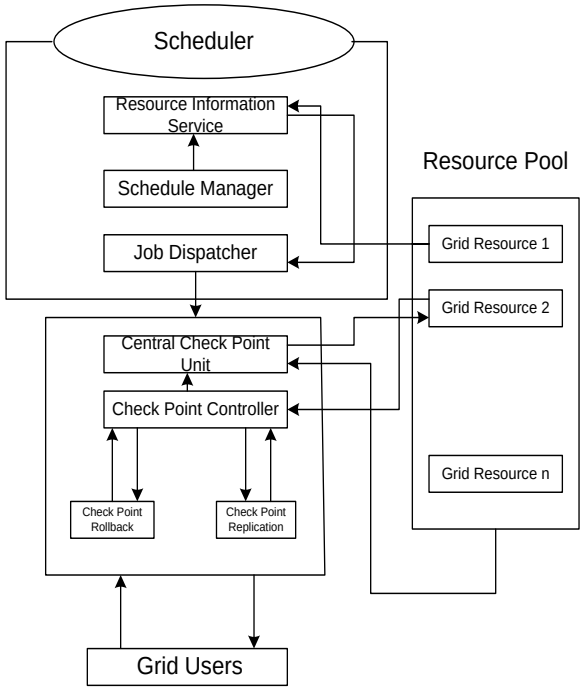


**Figure 1. Checkpoint architecture with Replication and Rollback**

computational scheme In the case of data grid services, there is a need for roll back mechanism since the data from the previous node may be faulty or to recover from the faulty state of the forwarding nodes in a cluster as shown in Figure1.The Grid jobs are executed by the computational grid as follows: (i) The Grid users submit their jobs to the scheduler Manager their requirements i.e., deadline in which the users want their jobs to be executed.[13].(ii)The Scheduler Manager component submits user jobs to the Resource Information Service for getting optimized resources. (iii) The Resource Information Service submits job with matched resource to the Job Dispatcher to a Resource Pool which is a member of a grid and it offers computing services to grid users.

### 3.1 Schedule Manager

The Scheduler Manager plays a vital role in the grid in receiving the jobs from grid users. It selects feasible resources by submitting jobs to Resource Information Service. Then it generates job-to-resource mapping and the entities of scheduler are the Schedule Manager providing the Resource Information

Service, Job Dispatcher service. A job dispatcher dispatches the jobs one by one to the Central Checkpoint Unit.

## 3.2 Checkpoint Unit

It receives the scheduled jobs from the schedule manager and sets checkpoint based on the failure rate of the resource on which it is scheduled. Then it submits the job to the resource. Central Checkpoint Unit receives job completion message or job failure message from the resource pool and responds to that accordingly. During execution, if job failure occurs, the job is reached.

## 3.3 Checkpoint Controller

On each checkpoint set by the Central checkpoint Unit, job status is reported to the checkpoint Controller. Checkpoint Controller save the job status and return it on demand i.e., during job/resource failure.

## 3.4 Checkpoint Rollback or Check Point Replication

When a new checkpoint is created, the Checkpoint Controller initiates Checkpoint Rollback mechanism. It will perform the possible rollback operations for a faulty process and find the list of possible states from recovery is possible. If rollback is not possible, the Checkpoint Controller initiates Checkpoint Replication services from where once the application replicas may be called. If not available the on demand replication is to be carried out and the critical services are replicated. The information service logs on the details like the time of initiation of the replicas and their identity are stored in the buffer inside the Checkpoint Controller.

# 4. PROCESS MAPPING ACROSS REPLICAS AND ROLLBACKS

The check pointing can be carried out in an asynchronous manner either on demand or on supervision by the grid manager service. Based on the nature of the tasks periodicity and possibility of rollbacks, the processes may request to find a mapping from replica to rollbacks. The start time and deadline parameters of the submitted tasks, the check point controller identifies by running the feasibility and schedule analysis of all the tasks. Once a set of solutions are identified, the controller checks the availability of the needed resources from the respective service and dispatches the jobs to them for early completion with or without replication of the application.

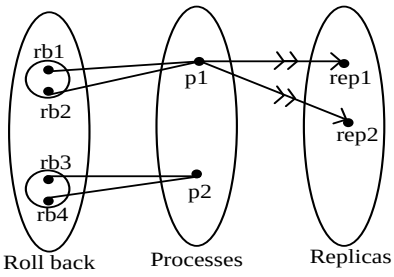


**Figure 2. Mapping of Replicas and Rollbacks**

The mapping of the process across the rollbacks and replicas are shown in Figure 2 and if there is no mapping available, the application may be replicated on demand or the grid manager starts negotiating with the user regarding the slippage in the scheduling for further actions as in the proposed concurrent task scheduling algorithm (CTSA) mentioned below.

Concurrent Task Schedulability Algorithm::
Tasks: n
Independent tasks:m
Concurrent : n-m
No of replica : $n_{re}$
nN of rollback : $n_{ro}$
P(selection replica or rollback)= 0.5
select one of $n_{re}$ or one of $n_{ro}$
probability of failure of one task $P^{y}_{x}$
if x € n-m // failed task is concurrent

$$P_{failure} = P^{y}_{x} * \sum_{i=1}^{n-m} P^{y}_{x+a}$$

// probability that concurrent process fail one after another//
else // failed task is independent
$P_{failure} = P^{y}_{x}$
Communication failure= P(CF)
// failure in synchronous or asynchronous
mode of communication//

$$P^{tot}_{nre\ nro} = P_{failure} * P(CF)$$

$$\{\sum_{j=1}^{m+1} \sum_{k=1}^{} 1/n\text{-}m * 0.5\ [P(r_k / r) + P(re_j / re)$$

$$* \sum_{q=1}^{nre\ nro} 1/(n\text{-}m\text{-}q)\ \} * P(CF)$$ // Concurrent

$$\{\sum_{j=1}^{} \sum_{k=1}^{} 1/m * 0.5\ [P(r_k / r) + P(re_j / re)$$

* P(CF) // Independent

**Figure3. Concurrent Task Schedulability Algorithm**

The mapping of the processes and their permitted rollbacks and possible replicas are shown which are under the control of the scheduler. [12]. The replication of a process is executed in parallel with the identical state but on different resources so that one replica is guaranteed to finish the process correctly.

## 5. COMPUTATIONAL MODEL OF CONCURRENT ACTIVATION OF PROCESSES

The concurrent process and the independent processes are to be identified by the information service and the all the components must collaborate to complete the submitted tasks. For the execution and allocation components, grid stacks are introduced in the selected points identified by the grid manager. The job attributes and their restoration states are compared to initiate the recovery processes. [14]. The various grid stacks like S stack that represents start time stack, W stack that represents worst case execution time stack, D stack represents deadline stack and C stack represents cost stack are shown in the Figure4. For enhancing the grid application reliability the three check pointing mechanisms such as check pointing on supervision, check pointing on demand and check pointing on requestor may be done by Grid supervisor, Grid monitor and Grid requestor. After execution the tasks, the checkpoint checker will determine whether the task is an independent task or dependent task. If it is an independent task then it will start replication.

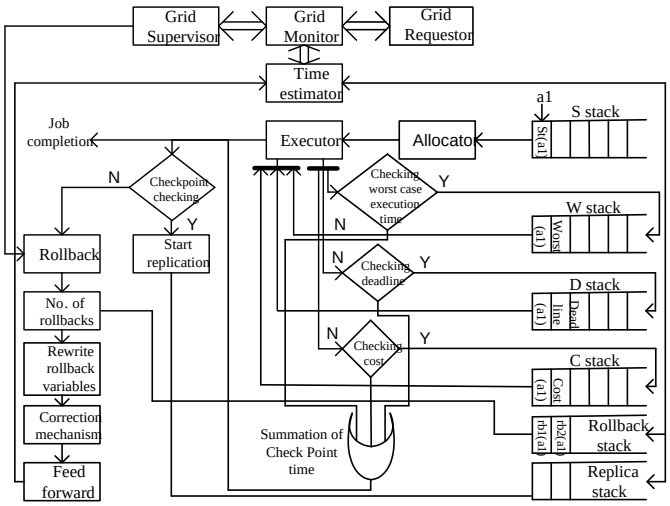


**Figure4. Computational Model for Replication and Rollback with task stacks**

The task is a dependent one then it will start performing the rollback mechanism by calling the FALL methods in the appropriate session in the protected mode. Then the recovery or the correction mechanism can be initiated so as to forward the status to the time estimator service component through the global grid interface. The time estimator determines the most probable time for calling the replicas using CALL methods through the private interface allotted for that process to minimize the replication time. In the coordinated check pointing mechanism, the processes will attempt to synchronize the call or demand for check points to ensure individual saved states are consistent with respect to each other. In contrast, in uncoordinated check pointing, processes schedule check points independently at different times and do not account for messages. The recovery process may be initiated by considering the timed activities and the values for rewrite and rollback variables. It is the challenge to activate the right correction process that is based on the task dependency and the cost – complexity of the individual tasks as negotiated with the grid users. The firing of the correction or recovery is modeled as a stochastic activity network using Mobius as shown in the Figure 5. The Checkpoint time can be calculated as the summation of checkpoint start time, status collection time, decision time, rollback time, rewrite rollback time, correction time, feed forward time and restart time.

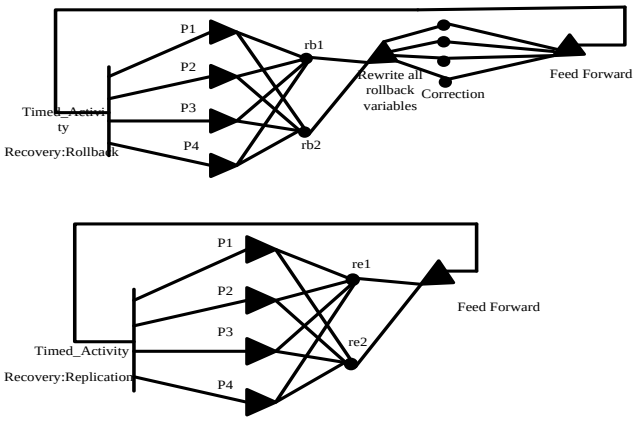


**Figure5. SAN model of Recovery**

The checkpoint time can be calculated as the summation of checkpoint start time, status collection time, decision time, rollback time, rewrite rollback time, correction time, feed forward time and restart time of individual processes for the assigned tasks. The grid application reliability can be calculated based on the successful negotiation and the synchronization between various component services, The reliability calculation also considers the cost factor on which the service agreement was drawn with grid users, The on demand check pointing will identify the failed processes and activate the application replicas at the detected node or from the other node identified by the global detection service component. The reliability of the grid application was determined by injecting random faults in the task .

Application Reliability = $R^{g}_{app}$ (time, cost)

$R^{g}_{app}$ (time) = 1- $P^{time}_{total}$(failure)

= 1-$P_{total}$ (process_faulty)

$$= 1 - \sum_{i=1}^{n} P_f \text{ (task)}$$

$$= 1 - [\sum_{i=1}^{n} P(\text{concurrent tasks}) + \sum_{j=1}^{m} P(\text{independent tasks})] \quad \text{--(1)}$$

$R^{g}_{app}$ (cost) = 1- $P^{cost}_{total}$(task)

$$= 1 - [\sum_{k=1}^{n+m} P(\text{negotiation}) * P(\text{synchronization})] \quad \text{----------(2)}$$

$R^{g}_{app}$ (t, c) = $R^{g}_{app}$ (t) + $R^{g}_{app}$ (cost) ----------(3)

## 6. SIMULATION RESULTS AND EVALUATION

Based on the values in Table, various simulations were made in many aspects by changing values of several parameters.

**Table 1. Values of parameters used in performance evaluation**

| Parameters | Value | Description |
|---|---|---|
| Pse | $1\text{-}1.0.10^{-12}$ | 99.999…% propability for successful execution. |
| m | 2000 | Total number of computing nodes |
| tr | 100-500 | Each task has one of between 100 and 500s requirement |
| Ci | 3s | Check pointing cost |
| r | 2s | Recovery cost |
| Sr | 1s | Task scheduling request arrival interval |

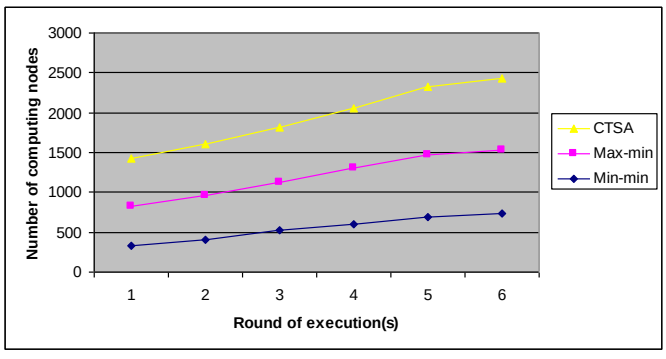


**Figure6. Number of computing nodes according to the changed probability for successful execution**

## 7. CONCLUSION

The check pointing strategy is used to detect the failed process in the grid environment using the on demand approach and the recovery action is carried with the help of concurrent task scheduling algorithm. The computational model of collaboration of grid component services using attribute stacks is introduced. The task dependency is considered to identify the best scheduling and the corresponding replicas are activated to improve the reliability of grid application through fault tolerant techniques. The technique is tested as a stochastic activity network and the schedulability is shown as a timing graph. The work is having a limitation on the number of concurrent tasks that should always be maintained as lower than that of the independent tasks. The size of the attribute stacks should be increasingly large if the numbers of grid requests are more than that of the available resources at any point of run time. The future work focuses on the enhancement of the proposed algorithm in the case of resource conflicts when two processes are demanding the same resource at the same time when there is no application replicas exist in the grid environment.

## 8. REFERENCES

[1] Malarvizhi Nandagopal, V.Rhymend Uthariaraj, "Fault Tolerant Scheduling Strategy for Computational Grid Environment", International Journal of Engineering Science and Technology, Vol.2(9), 4361-4372, 2010.

[2] J.Jaybharathy and Ayeshaa Parveen.A,"A Fault Tolerant Load Balancing Model for Grid Environment", Internatinal Journal of Recent trends in Engineering, Vol 2, No.2, 162-164,2009.

[3] Gopi Kandaswamy, Anirban Mandal, and Daniel A.Reed, "Fault Tolerance and Recovery of Scientific Workflows on Computational Grids", IEEE Computer Society, 2008.

[4] Partha Sarathi Mandal, Krishnendu Mukhopadhyaya, "Performance analysis of different checkpointing and recovery schemes using stochastic model", Journal of Parallel and Distributed Computing, 66, 99-107,2006.

[5] Youcef Derbal,"A new fault-tolerance framework for grid computing", An International Journal on Multiagent and Grid System, 2,115-133, 2006.

[6] Jia Yu and Rajkumar Buyya,"A Taxonomy of Scientific Workflow Systems for Grid Computing", SIGMOD Record, Vol.34, No.3, 2005.

[7] Soonwook Hwang and Carl Kesselman, "A Flexible Framework for Fault Tolerance in the Grid", Journal of Grid Computing 1,251-272,2003.

[8] Klaus Krauter, Rajkumar Buyya and Muthucumaru Maheswaran, " A taxonomy and survey of grid resource management systems for distributed computing ", Software – Practice and Experience 32, 135-164,2002.

[9] Fangpeng Dong and Selim G.Akl, " Scheduling Algorithms for Grid Computing State of the Art and Open Problems", Technical Report No. 2006-504.

[10] Saeed Parsa, Reza Entezari-Maleki, "RASA : A New Grid Task Scheduling Algorithm", International Journal of Digital Content Technology and its Applications Volume 3, Number 4, December 2009.

[11] Kobra Etminani, Prof.M.Naghibzadeh,"A Min-min Max-min Selective Algorithm for Grid Task Scheduling", IEEE, 2007.

[12] Baghavathi Priya.S, Chandrasekaran Subramaniam, Ravichandran.T,"On Demand Check Pointing for Grid Application Reliability using Communicating Process Model", IEEE, 2011.

[13] S.Baghavathi Priya.S,Dr.T.Ravichandran, "Fault Recovery Mechanisms using Check Point in Grid Environment", ICFET, 2010.

S.Baghavathi Priya, Dr.K.K.Dhawan,"Fault-Tolerance Genetic Algorithm for Grid Task Scheduling using Check Point", IEEE, 2007