ACADEMIC
World Academic Union

**ISSN** 1746-7659, England, UK
**Journal of Information and Computing Science**
Vol. 12, No. 4, 2017, *pp*.243-247

# Reducing Checkpoint Overhead in Grid Environment

A. S. Faki*, R. G. Jimoh
*Computer Science Department, Faculty of Communication and Information Science,*
*University of Ilorin, Ilorin-Nigeria*
*E-mail: faki.silas@binghamuni.edu.ng.*
(Received June 19, 2017, accepted September 01, 2017)

**Abstract.** Grid Computing has become major player in super-computing community. But due to the diversity and disruptive nature of its resources, failure of jobs is not an exception. However, many researchers have come up with models that enhance jobs survivability. Popular among this model is checkpoint model which have the ability of saving already computed jobs on a stable secured storage. This model avoids re-computing of already computed jobs from the scratch in case of resources failure. But the time a job takes in checkpoinitng also becomes another task which adds overheads to computing resources thereby reducing the resources performance. In order not to add too many overheads to computing resources, the number of checkpoints must be minimized. This study proposed checkpoint interval models which is implemented based on fault index history of computing resources. Failed jobs are re-allocated from their last saved checkpoint using an exception handler. The study observed that arithmetic checkpoint model is better used when fault index of computing resources is high while geometric checkpoint model is better when fault index of resources is low.

**Keywords:** Arithmetic Checkpoint, Exception Handler, Fault Tolerance. Geometric Checkpoint.

## 1. Introduction

Grid has proven to be a great computational resource in computing circle. But due to its unstable and unpredictable nature of its resources, resources and job failure is a norm rather than exception. Systems and networks can fail, resources can be turned on and off and the introduction of more users can result in resource starvations which significantly affect quality of service (QoS). To maintain a tolerable QoS level for users, fault tolerance mechanisms are incorporated. Providing fault tolerance in a grid environment while on the other hand optimizing resources utilization and job execution time is a challenging task [1]. This is because fault tolerance measures increase computing overheads to grid resources. This study adopts checkpoint and recovery system as a measure to increase job life line thereby optimizing job completion time. The goal of this method is aimed at increasing the survivability of interrupted jobs by re-scheduling them at last save checkpoint and minimizing the number of checkpoints in order not to incur too many overheads. This model therefore, reduces the number of checkpoints and increase the survivability of jobs thereby optimization resources performance [2]

## 2. Literature Review

The desire of grid checkpoint service is to meet basic requirements which are: ability of software to exchange information among resources, ability of grid middleware and infrastructures to exchange and maintain vital information, and availability of checkpoint data to all resources [3]. But achieving interoperability in grid is difficult due complexity and unstable nature grid resources. The function of fault tolerance as opined by [4] is to preserve the delivery of expected service despite the presence of faulty processors and declining resources capability within the system. This means errors within grid system should be detected and corrected promptly. Permanent fault that could cause great havocs should be located and remove quickly. All these are to allow grid resources deliver a tangible and acceptable QoS. Due to technological advancement and increase in autonomous systems, many researchers are exploring ways to design and implement fault detective models that can predict and perform recovery from crash processors of computing resources. These models aim at improving resources performance and job survivability in the presence of failed processors, their effectiveness largely depends on tuning runtime parameters such as the checkpoint interval and number of replicas [5]. Fault tolerance schemes in grid can either be pro-active or post-active [6, 7]. In pro-active mechanism, the resources make a failure prediction process before jobs are

scheduled with the hope that it will follow the prediction process. Whereas post-active mechanism handles the job failure after it has occurred. Most approaches applied to fault tolerance in grid environment are on post-active rather than the pro-active approaches [8].

According to [9], the most popular fault-tolerance model in use is checkpointing which involved periodic saving of snapshot of job progress on a stable storage device which can survive failures (hard disk). The information stored on the stable storage is called a checkpoint. Any time a processor or job crashes, the last saved checkpoint is used to restart the job or resource rather than from beginning. There exist varieties of checkpoint model but the popular ones could either be categorized as coordinated, uncoordinated and communication induced checkpointing [10].

The main advantage of checkpointing is that, it is a general technique which can be applied to any type of parallel applications. The time to take a snapshot or make a checkpoint is also a task or job which adds execution time overheads to a resource even when crashes have not occurred. This overhead is dependent on the frequency at which checkpoints are taken and in turn depends on the programmer or middleware designed.

To improve fault tolerance of grid system, [11] proposed hierarchical performance of checkpoint protocols grid computing which discussed protocols based on rollback recovery that was classified into two categories: checkpoint based rollback recovery and message logging protocols. However, the performance of protocols was observed to depend on the characteristics of the system, network and applications running. In situations where the computational intensity is low, the Algorithm Based on System checkpoint (ABSC) model is applicable. Meanwhile, if the computational intensity is high, Algorithm based on application Checkpoint (ABAC) model is more suitable though with production of slight overheads in fault free situations and very reliable in faulty situations [3]. Adaptive fault tolerant scheduling utilizes an adaptive number of job replicas according to the grid failure history. This technique composed of Adaptive Job Replication (AJR) and Backup Resource Selection (BRS) where AJR determines number of replicas according to selected resources.

To reduce checkpoint overheads, [12] proposed Optical Checkpoint Automation (OCA). The goal of this model is to make the most effective use of grid resources and also to improve throughput value in the mist of fault. This model focus on minimizing the effect of grid faults and reduces fault recovery time using optimal automation of checkpoint. To evaluate the performance of the model, fault index of resources were kept and analyzed and jobs were assigned to resources based on next sequence of pattern of failure. The failure patterns were predicted by using Hidden Markov Model (HMM) to assign checkpoint interval and also to provide automatic failure replica (context file of checkpoint) to the grid resource. The proposed OCA model provide a better performance as compare to adaptive algorithm though the model depends on previous history of computing resources which implies that checkpoint interval of new resources are unpredictable.

Checkpointing is the most common method to achieve fault tolerance. Though, there exist research issues on how improve its efficiency and overheads can be, [13] proposed a novel solution on how checkpoint can operate on parallel application in grid. The model allows checkpoint on applications at regions where there no inter-process communication thereby reducing the checkpoint overheads and checkpoint size. In [14], a novel technique to analyze the performance of checkpointing algorithms is proposed. The model was implemented in a suitable cloud environment with six service node with analysis made in terms of parallel jobs execution.

## 3. Arithmetic and Geometric Checkpoint Model (Agcm)

This model allows users submit jobs to the grid scheduler. Each submitted job is then assigned to a computing resource that matches the requirement by the scheduler as specified by user. As computing resources continue execution of job, statistics about each job and their resources are sent to the Grid Information Service (GIS) for storage. Resource capacity, checkpoint activities and current load are some of the information's that are stored in the GIS.

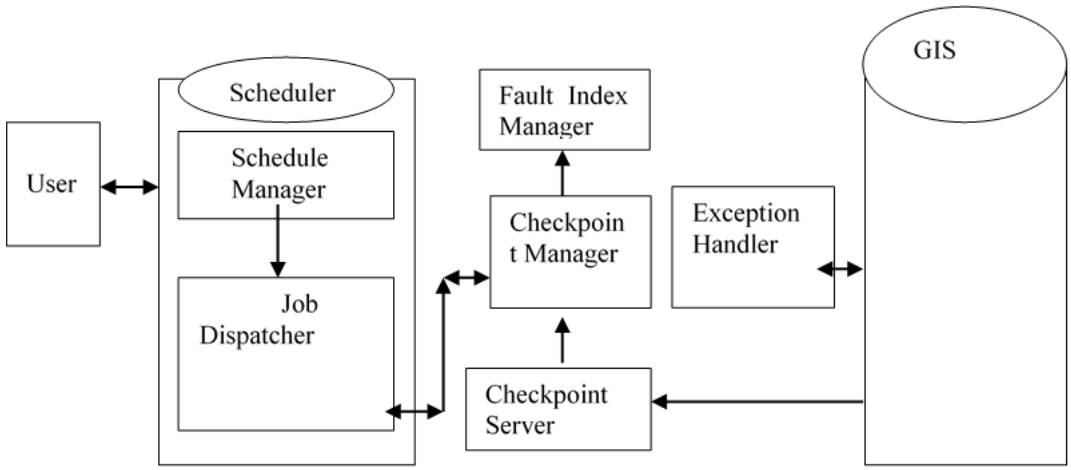


Fig-3.1: Model with Checkpoint and exception handler

Resources in grid environment are not of the same specifications. Specifications like processing speed, internal scheduling policy and load factor varies for resources to resources. In the same way, each job differs from other jobs by execution time, deadline, and so forth. Fault indexes here indicate the frequencies of failure of a particular resource. This fault indexes are stored in the GIS as performing history of every resources. The fault indexes are used by this model to determine how frequently a checkpoint is made on a resource. This is to make checkpoint in a most reasonable way bearing in mind that the time to make a checkpoint also adds to the execution time of computing resources. In other to make checkpoint at appropriate and economical time, checkpoint interval must be minimize. All computing resources with high fault index history use arithmetic model (eqn I) to set their checkpoint while those resources with low fault index uses geometric checkpoint model (eqn II) as shown in eqn I and II respectively.

$$J_{\text{endT}} = J_{\text{startT}} + (n - 1)\, Ci \tag{3.1}$$

$$J_{endT} = J_{startT} * Ci^{(n-1)} \tag{3.2}$$

Where $J_{\text{endT}}$ is the end time of execution of job j

$J_{\text{startT}}$ is the start time of execution of job j,

$n$ is the number of checkpoint of job j with the execution time,

Ci is the checkpoint interval at which checkpoint is to be set.

The last saved checkpoint ($LScp$) for any job as distributed to all computing resources can be recovered by the model in eqn 3.

$$LScp = \sum_{i=1}^{n} J_{endT} \tag{3.3}$$

This shows that module(s) of a job can be recovered form one or more computing resources at the time of crashes or processor failure.

Recovery and reassigning of failed jobs to new computing resources is achieved by an exception handler.

**Exception Handler Algorithm:**

Step 1: Start

Step 2: if every processor $P_i$ is recovering from failure

Checkpoint is last storage saved massage for processor $P_i$

Step 3: for k = 1 to N (N is the number of processor holding the same jobs)

Do

For every processor $p_j$ that holds the same job

Step 4: invoke exception handler

Step 5: Stop

## 4. Checkpoint Model Simulation

A submitted job has an estimated start and end time with a checkpoint set a fixed interval in between them. To test the performance of checkpoint overhead in the system, the study simulates checkpoints based on our proposed model. Chosen any of the proposed models is based on fault index of a computing resource. The performances of the two models are shown in figure 1. From the simulation, job start time at first minute, and checkpoint interval is set every two minutes, eight checkpoints were taken for each model.

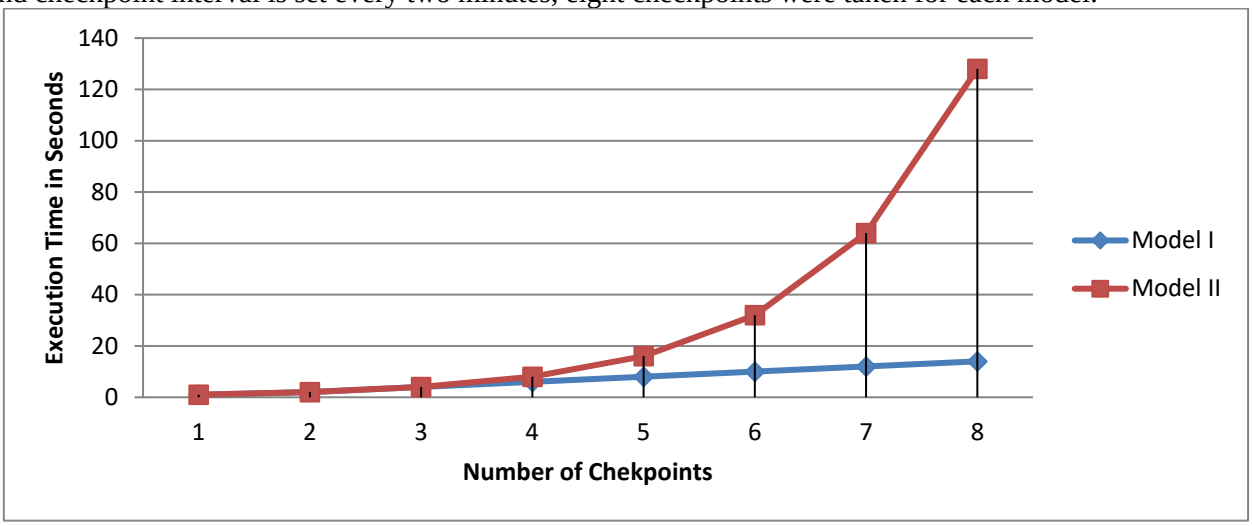


Fig-4.1:- Graph of checkpoint interval vs. time of execution

It can be seen that though, same numbers of checkpoints were set, execution time of jobs differs as checkpoint progresses. At 4th checkpoint, model I (eqn I) has done 6 minutes while model II (eqn 2) has done 8 minutes of execution. At 6th checkpoint, model I (eqn I) has executed for 10 minutes while model II (eqn 2) has executed for 32 minutes.

It can be observes that both models exhibit close related behavior up to the fourth checkpoint interval, the execution time and the number of checkpoint made is almost the same. Though, the numbers of checkpoints made are the same, the time used in execution of the job varies. Model II (eqn 2) execution time is more than model I (eqn I). The 8th checkpoint was when job has executed for 14 minutes for model I (eqn I ) and 128 minutes for model II (eqn 2). This implies that Model I will make more checkpoint than model II thereby incurring more checkpoint overhead. In terms of recovery, model II will do more re-computing of failed job because the checkpoints are far apart in the terms of time. Conclusively, model I is more suitable for computing resources that have high fault index (mission critical jobs) and computing resources with poor uptime. Meanwhile Model II is suitable for computing resources with more up time and less processor crashes.

The model recovery uses exception handler which is done from last checkpoint saved on stable storage. The exception handling model handles fault and reallocate jobs automatically to other resources without jobs beginning from the scratch because jobs start from the last served checkpoint. The exception handler seamlessly relocates jobs to the scheduler making it less costly and incurring little overhead in job.

## 5. Conclusion

In order to minimize checkpoint overheads incur by resources, the study proposed two models. One was arithmetic (model I) and the other geometric (model II). Based on computing resource history, a geometric checkpoint is good for a resource that has a good uptime time with less withdrawer and crashes while arithmetic checkpoint is good for a mission critical job or resources that is known to have a high faulty index. Exception handler was use to re-assigned jobs that were unable to finishes due to processor crashes to new computing resources that meet their requirement. Being a simple design, exception handler incurs little over heads and recovers jobs in a seamless manner.

## 6. References

[1] M. Nakkeeran. A survey on task checkpointing and replication based fault tolerance in grid computing. *International Research Journal of Engineering and Technology (IRJET)*, 3(9), 832-838, 2015.

[2] Y. Li, and M. Msacagni. Improving performance via computational replica on a large scale computational grid. Proceedings of third International Symposium on Cluster Computing and the Grid, Tokyo-Japan, 2003.

[3] B. Mangesh, and S. Urmila. Checkpointing based fault tolerant job scheduling system for computational grid. *International Journal of Advancements in Technology*, 5(2), 2014.

[4] T. Paul and X. Jie. *Fault tolerance within a grid environment*, University of Durham DHI 3LE, United Kingdom, 2003.

[5] G. Pankaj. Grid computing and checkpoint approach. *International Journal of Computer and management Studies*. 11(1), 2231-5268, 2011.

[6] R. Garg and A. K. Singh. Fault tolerance in grid computing: state of the art and open issues. International Journal Computer Science Engineering Survey.2(1), 88., 2011, doi: 10.5121/ijcses.2011.2107.

[7] K. Ganga and S. Karthik. A survey on fault tolerance in workflow management and scheduling. IJARCET, 1( 8) , 176, 2012.

[8] H. Sajjad, and N. Babar. Fault tolerance in computational grids: perspectives, challenges, and issues, *SpringerPlus*, 5(1), Nov 18, 2016. doi: 10.1186/s40064-016-3669-0.

[9] A. S. Tanenbaun, and M. Van Steen. *Distributed Systems, Principles and Paradigms*, Prentice Hall, 2002.

[10] E.A.Elnozahy, L. Alvisi, Y. M. Wang and D. B. Johnson. A survey of Roll-Back recovery protocols in massage - passing system, *ACM Computing Survey*, 34(3), 375-408, 2002.

[11] M.N. Ndeye, S. Pierre and T. Ousmane. Performance comparison of hierarchical checkpoint protocols grid computing. *International Journal of Interactive Multimedia and Artificial Intelligence*, 1(5), 46-53, 2003.

[12] B. Ramesh. An optimal checkpoint automation mechanism for fault tolerance in computational grid. *International Journal of Scientific & Engineering Research*, 7(2), 1012-1019, 2016.

[13] K. Sajadah. Checkpointing of parallel applications in a grid environment, A dissertation submitted to the University of Westminster for the degree of Master of Philosophy, Centre for Parallel Computing, School of Electronics and Computer Science, University of Westminster, London, United Kingdom, 2011.

[14] S. Dilbag, S. Jaswinder and C. Amit. Evaluating overheads of integrated multilevel checkpointing algorithms in cloud computing environment, *International. Journal of. Computer Network and Information Security*, 4(5), 29-38, 2012.