International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 10 (2015) pp. 25957-25965

http://www.ripublication.com

# An Analytical Model of Consistent Checkpointing In Computational Grid

**Ch. Ramesh Babu[a] [1], Dr. Ch. D. V. Subba Rao[b]**
[a] *Research Scholar, JNTU Kakinada, AP, India. E-mail:crb.challagundla@gmail.com*
[b] *Professor, Dept of CSE, SV University Tirupati, AP, India.*

## Abstract

The TORQUE resource manager tool provides an environment for open-source batch processing jobs and distributed computing nodes, which simulates grid based operating system based on Open SUSE Linux. To generate consistent checkpointing for grid applications we used a distributed grid check pointing service called Torque GCP. This service is considered to support various checkpointing protocols and different checkpointing packages such as BLCR, Open MPI in an efficient manner. In this paper, we combined both the features of last failure and mean failure checkpointing protocol with a FTCS scheduling strategy to generate consistent check pointing using Torque GCP service. To assess this prototype we run it within a heterogeneous environment composed of autonomous computing nodes in different LAN cluster. The experimental results prove that the capability of the Torque GCP service to integrate different optimal checkpointing protocol in a distributed application within a heterogeneous computational grid. Furthermore, the performance valuations also prove that our solution better performs than the existing checkpointing approaches in terms of space and throughput.

## Introduction

A Grid is a type of distributed computational environment which supports resource sharing, coordinated use of geographically dispersed nodes and autonomous management of physical and dynamic virtual organizations that share the common goal of solving large-scale applications. The increasing demands of computational power of large scale applications, grids become popular managing and running large computing resources during the last fifteen years. Computational Grids provide a large-scale distributed environment for the execution of various forms of computational and data relevant applications such as particle system simulation studies, weather prediction etc., across many varied resources [5]. These large applications executing on Grid or cluster architectures consisting of computational nodes often creates problems with reliability. The source of the problems is node

failures and job delay which leads to dynamic configuration over extensive runtime [2].

Computational Grids and Cluster Computing architectures have gained popularity for computationally intensive parallel applications. However, the complexity of the organization, consisting of autonomous computational nodes, large storage, and interconnection networks, poses great difficulties with respect to overall system reliability. Simple tools of reliability analysis show that as the difficulty of the system increases, its reliability, and thus, Mean Time to Failure (MTF), decreases [8]. The reliability of the computational grid systems is computed as the product of the reliabilities of all system components or nodes. For applications executing on large clusters or a Grid, the long-time execution tasks may exceed the MTF of the infrastructure and, thus, reduce the execution infeasible. The high failure probabilities are due to the fact that, in the absence of appropriate fault tolerance mechanisms, the failure of a single node will cause the entire execution to fail or delay. Hence this simple example does not even reflect network failures, which are typically more likely than computer failure. Fault tolerance is thus necessity factor to consider avoid failure in large applications, such as found in scientific computing, executing on a Grid, or large cluster [6]. Efficient technique of fault tolerance for distributed applications is periodic checkpointing. But this checkpointing approach further leads to more number of checkpoints overhead due to appropriate checkpointing algorithm. The following sections discuss the various checkpointing studies, limitations and proposed solution with results [10].

## Torque Grid Check pointing Service

In this section we briefly describe the architecture of the Torque Grid Check pointing service (Torque GCP) and its components related to optimal checkpointing [23]. Fault tolerance technique based on checkpointing can be used to save the process state of the application on secondary storages so as to recover checkpoint in the time of failure or job delay. Moreover, checkpointing and replications are crucial in the process of virtual grid resource in order to distribute the jobs across multiple resources the Open MPI is used [11,12, 15, 17].

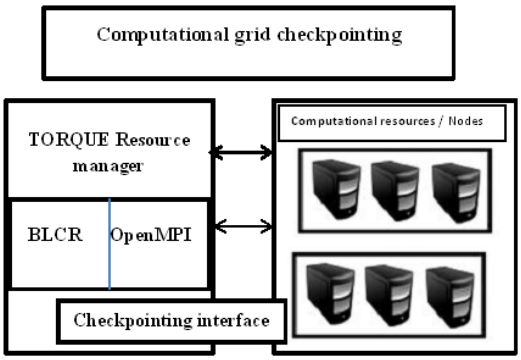


**Figure 1:** Torque Grid Check pointing architecture

In figure 1 it shows that the computational jobs large application chunks can be distributed across many resources, most of the times it is not sufficient to exclusively rely on the discrete checkpoints of the processes in order to compute consistent global state. The messages needed to be handled otherwise it can lead to inconsistent checkpoints [5]. The checkpoint context files are process states whose receive events are part of the destination process checkpoint but the corresponding send events are lost. In case of a recovery the destination process would receive that checkpoint context twice, which could result in unpredictable application behaviour. On the other hand, checkpoint context occur when the send events are part of the sender-side checkpoint, however the receiving events are lost. As a consequence, checkpointing protocols and adaptive checkpointing systems are needed in order to address the problem of inappropriate checkpointing. In the two common checkpointing protocols, the processes uncoordinated their checkpoints or take them independently. Some examples for adaptive cluster-level checkpointing include BLCR and Open MPI [21, 22], which supports both protocols within MPI which integrates the checkpointing protocol [18, 19 ].

Computational Grids are large-scale distributed and heterogeneous resources which can be composed out of hundreds or thousands of nodes. It is clear that accuracy of each grid node to use one checkpointing approach. Therefore, in order to achieve fault tolerance for large computational applications in grids, TORQUE deploys a heterogeneous grid check pointing service called Torque GCP. This service is designed to support different checkpointing protocols and address the fundamental checkpointing packages in a obvious manner through a uniform interface. Even with the general design of this service, until recently it only implemented the coordinated checkpointing protocol [9, 10]. But with computational grid environment where data and computing resources of one application can be dispersed across hundreds or thousands of grid nodes, coordinated checkpointing can be expensive in terms of scalability [10]. Also, the entire grid application needs to be replicated in the event of a single node failure. Thus, in additional more number of checkpoints overhead and consistency of checkpoints is added to the replication and storage infrastructure [20].

In this paper, we present the design and implementation of a solution for Optimal check pointing within Torque GCP. Therefore, we describe all the necessary steps towards this goal. This involves consistent dependency tracking among processes, transparent selection of the underlying checkpoint interval, monitoring the process failures, computing a consistent global state of the information and retrieval of an application. Our solution is not bound to a specific checkpointing approach and can be obviously used on any existing checkpointing package.

The result of this work is an extended Torque GCP service with the support of two checkpointing protocols, which is able to checkpoint an application distributed across different checkpointing packages. This is the first work dealing with integration aspects of Optimal checkpointing in grids [7]. We have confidence in our results can help future works to identify the issues related to implementing consistent checkpointing protocols in large heterogeneous environments such as computational grids or cloud computing etc.

The next sections of paper are organized as follows. Section 3 outlines the integration of Optimal check pointing within Torque GCP. Section 4 describes the results from the performance evaluation of our prototype. Finally, Section 5 reports the paper with conclusions and future work.

## Implementation of Optimal Check pointing Within Torque GCP

Check pointing is the commonly used technique for fault tolerance and improving system availability and reliability. It stores the status of current application state, and then it can be used for restarting the execution in case of failure avoiding the job to start from beginning.

The main drawback of unconditional classical checkpointing approach called periodic checkpointing, which performs checkpoint operation periodically whether the job is executed on a stable resource. This approach generates lot of checkpoints, sometimes the checkpoint operation time may exceed the normal processing times. According to the adaptive checkpointing scheme, it uses the information about the remaining job execution time, time left before the deadline and the expected remaining needed, to decide whether checkpointing is to be done or not [13, 14]. It also gives information on when the next checkpoint request need to be given. There are two methods in adaptive checkpointing scheme Last Failure Dependent Check pointing (Last Failure CP) and Mean Failure Dependent Check pointing (Mean Failure CP). In Last Failure CP 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 LFr 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<=$E_r^j$, where tc is the current time and $E_r^j$ is the execution time of job j on resource ,if the condition is true then checkpointing is allowed otherwise checkpointing is omitted [1]. In fault tolerant checkpoint scheduling system (FTCS) [24], this effectively schedules the jobs in the presence of failures [3]. This approach depends on mean failure time and the failure amount of resources combined with resource response time to make checkpoint scheduling decisions. In this approach, if any one of mean failure rate or resource response time or response time of resource fails then this approach fails to get better checkpointing scheduling decision [15, 16].

Now our approach integrates the adaptive checkpointing schemes [1,3 and 24] to get better consistent checkpointing approach. To make this the last failure rate and mean failure rate are considered from the resources, and then finds the appropriate checkpoint interval for each node. Let resource failure rate RF [24], last failure rate LF, mean failure rate MF, and fix k and j values derived from the grid workload logs [4], then find probability density function(pdf) of an exponential distribution as.

$$f(k;j) = \begin{cases} kje^{-kjx}, x \geq 0 \\ \\ 0, x < 0 \end{cases} \quad \ldots.(1)$$

Now, differentiate the probability density functions with respect to last failure rate and mean failure rate combine to get predictive checkpoints with optimum checkpoint intervals.

$$LFR = \frac{d\,(f(k,j))}{dk} = 0 \qquad \ldots..(2)$$

$$MFR = \frac{d\,(f(k,j))}{dj} = 0 \qquad \ldots..(3)$$

Then we have found consistent checkpoint interval using the following equation.

$$CFR = \frac{(RF)}{(LF + MF)} \qquad \ldots\ldots(4)$$

By using the above formulae [24], we have found consistent checkpoint interval for job failures in computational grid. In such a case, the Torque GCP service computes consistent global states based on the last failure and mean failure with integration of overall resource failure to integrate each job unit and restores the job units afterwards. The consistent checkpointing scheme identifies the job units belonging to the job based on the workload log meta information. Then, the last and mean failure information is loaded and the consistent checkpoint of the application is generated. We use the Torque GCP mechanism to stores a restart replication context file and restart them within this replication before starting the job checkpoint execution.

## Performance Evaluation of Torque GCP

According to the presumed settings we have inspected different applications with adaptive method such as periodic, Last Failure CP, Mean Failure CP, and Fault tolerant checkpoint scheduling. Here we have experimented mathematical, computers and mechanical engineering applications in 50 nodes with varying number of processes (i.e., threads or subtasks using forks) and checkpoints.

**Table 1:** Check pointing approaches comparisons

| **Checkpointing Approaches** | **Jobs Submitted** | **No. of Jobs Executed in different intervals** | | | | **Avg no. of Checkpoints in different intervals** | | | | **Avg. Job Execution time (min) in different intervals** | | | | **Avg CKPT Space (kb.) in different intervals** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0.01** | **0.5** | **1** | **5** | **0.01** | **0.5** | **1** | **5** | **0.01** | **0.5** | **1** | **5** | **0.01** | **0.5** | **1** | **5** |
| Periodic Checkpointing | 500 | 280 | 420 | 380 | 310 | 980 | 400 | 50 | 20 | 9 | 7.8 | 8 | 8.2 | 122.5 | 50 | 6.25 | 2.5 |
| Last Failure Dependent CP | 500 | 320 | 390 | 330 | 420 | 280 | 120 | 60 | 20 | 7.8 | 7.6 | 7.5 | 7.8 | 35 | 15 | 7.5 | 2.5 |
| Mean Failure Dependent CP | 500 | 450 | 420 | 410 | 420 | 75 | 160 | 190 | 80 | 6.5 | 7 | 7 | 7.2 | 9.3 | 21 | 24 | 9.8 |
| FTCS | 500 | 340 | 380 | 410 | 460 | 250 | 200 | 100 | 80 | 7.4 | 7.4 | 7.2 | 7.3 | 32 | 25 | 12.4 | 9.6 |
| Torque GCP | 500 | 390 | 410 | 440 | 470 | 150 | 90 | 100 | 60 | 7.3 | 7.3 | 7.2 | 7.1 | 18 | 12 | 11 | 7.8 |

We have evaluated the Torque GCP check pointing protocol using the different approaches such as shown in the Table I with some application within a heterogeneous environment consisting of single PC-nodes and Linux cluster. Therefore, we have measured the time, it takes to checkpoint this application when it opens up to 50 communication channels and sends jobs takes the checkpoints at intervals of 0.01,0.5,1 and 5 minutes. The result from this evaluation is that it takes approximately 4.25 hours to synchronize the processes. However, the results show that the adaptive checkpointing approaches have inconsistent over number of jobs executed, average number of checkpoints generated, the average time execution of application and the average storage space of a checkpoint context file etc as shown in the Figure 2,3,4and 5. Our approach Torque GCP got consistency over all checkpoint duration.

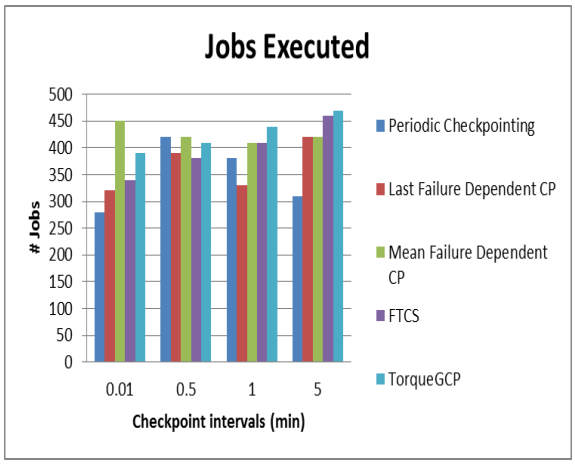


**Figure 2:** Jobs successfully executed

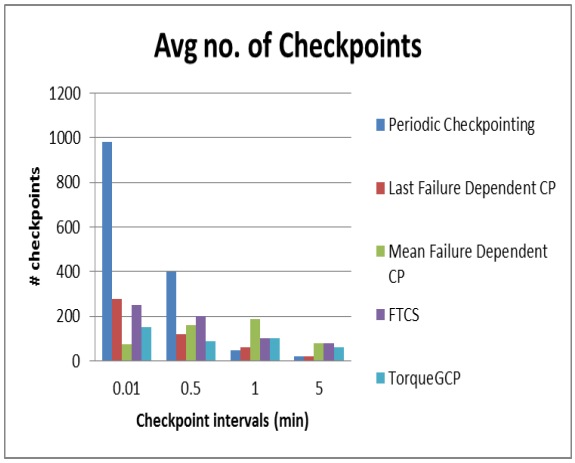


**Figure 3:** Average number of Checkpoints

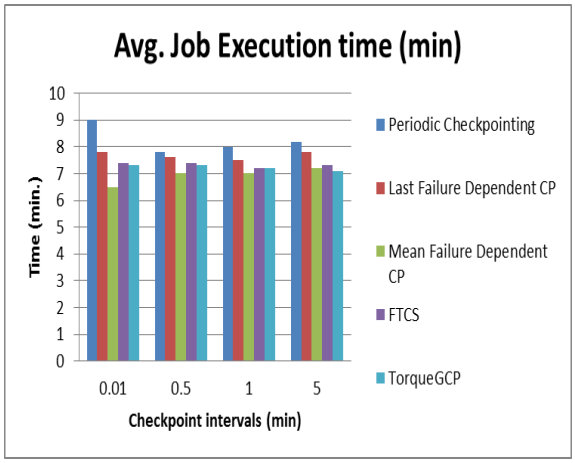


**Figure 4:** Average Job Execution time (min)

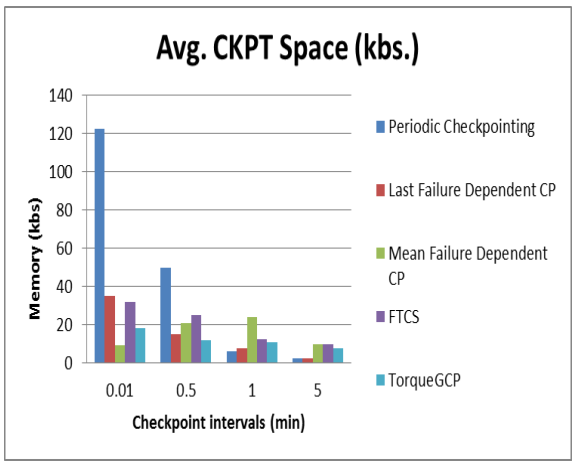


**Figure 5:** Average CKPT Space (kb.)

## Conclusions

In this paper, a Torque GCP check pointing system for computational grids is proposed and evaluated. The performance evaluation of consistent check pointing approach Torque GCP got better consistency when compared with the periodic CP, Last FCP, Mean FCP and FTCS schemes. The metrics used for evaluation are processing time, throughput and checkpoint storage space. The experimental results show that Torque GCP have consistency over adaptive approaches. It is observed that the throughput and over all processing time for the proposed system got better results over Mean FCP. Thus, it is concluded that the proposed Torque GCP provides performance consistency over the adaptive approaches.

## References


[1] 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.

[2] Brent Rood, John Paul Walters, Vipin Chaudhary, and Michael J. Lewis,"Failure Prediction and Scalable Checkpointing for Reliable Large-Scale Grid Computing".,The 16th IEEE transactions, June 2007.

[3] M. Chtepen, F. Claeys, B. Dhoedt, F. De Turck, P. Vanrolleghem, and P. Demeester, "Dynamic Scheduling of Computationally Intensive Applications on Unreliable Infrastructures," Proc. Second European Modeling and Simulation Symp. (EMSS '06), Oct. 2006.

[4] D.Feitelson, Parallel Workloads Archive, http://www.cs.huji.ac.il/ labs / parallel / workload /, 2008.

[5] Rajkumar Buyya and Srikumar venugopal, "A Gentle Introduction to Grid Computing and Technologies",http://www.buyya.com/papers/gridintro-CSI2005.pdf.

[6] Gokuldev S, Valarmathi M, "A New Checkpoint Approach for Fault Checkpoint Approach for Fault Checkpoint Approach for Fault Tolerance in Computing" IJCNS Volume 2, Issue 3, June 2013

[7] J.Young, "A First Order Approximation to the Optimum Checkpoint Interval",Comm. ACM, vol. 17, no.9,pp.530-531,Sept 1974.

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

[9] Suri P.K., Singh, M. "An Efficient Decentralized Load Balancing Algorithm for Grid," IEEE 2nd International, Advance Computing Conference, 201, pp. 10-130.Feb 2010.

[10] Oliner, A.; Rudolph, L.; Sahoo, R.; "Cooperative checkpointing theory", Parallel and Distributed Processing Symposium, 2006. IPDPS 2006.

[11] Menno Dobber, Rob van der Mei, and Ger Koole, "Dynamic Load Balancing and Job Replication in a Global-Scale Grid Environment: A Comparison" IEEE transaction on parallel and distributed systems, vol. 20, no. 2, February 2009.

[12] 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 Ap-plications, AICCSA 2008, IEEE/ACS 2008.

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

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

[15] Dilli babu, S., Ramesh Babu, Ch., Subba Rao, Ch.D.V., "An efficient fault tolerance technique using checkpointing and replication in grids using data logs.", In: publications of problems and application in engineering research—ijpaper.com, vol 04. Special issue 01, 2013.
[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 Urban a Champaign.
[20] Najme Mansouri, "An Effective Weighted Data Replication Strategy for Data Grid", Australian Journal of Basic and Applied Sciences- 2012.
[21] Http://www.open-mpi.org/
[22] Http://crd.lbl.gov/groups-depts/ftg/projects/current-projects/BLCR
[23] Http://www.adaptivecomputing.com/
[24] Mohammed Amoon, "A job checkpointing system for computational grids", CEJES Springer, 2013.