



An Automatic File Replication Mechanism in Computational Grid

Ch. Ramesh Babu^a, Dr. Ch. D. V. Subba Rao^b

^a Research Scholar, JNTU Kakinada, AP, India.

^b Professor, Dept of CSE, SV University Tirupati, AP, India.

Abstract: The massive dynamic virtual computing systems often generate large number of files as checkpoints for fault tolerance. In computational grid the vulnerability of node is high, so nodes more prone to frequent failures due to heterogeneous and autonomic nature of grid computational resource. To facilitate this, grid resources takes checkpoints to recover loss several hours/days of computation using replication. Replication is one of the widely used methods to provide fault tolerance in computational grid environment. In this work we have developed an automation of checkpoint replication by using Two Level Scheduling Strategy model, so as to select optimal failure resources to reduce checkpoint replication in computational grid. The failure can be systematically restarted with minimal interval of time for each grid resource using automatic replica (context file). The experimental results has shown that the proposed Automatic File Replication- AFR methodology considerably reduces the replication overhead, delay, storage space and turn-around time when compared to adaptive replication approaches.

Keywords: automatic replication, fault tolerance, computational grid.

I. 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 [4, 6]. 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 [18]. 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 [1, 27].

Typically these sophisticated computational resources require fault tolerance mechanism. According to the large scale, dynamic virtual computational resources such as NASA iPSC, LANL CM5 replica-logs traces shows that job delay or job failure is the major challenging task to provide fault tolerance [2, 3, 7]. Since very robust fault tolerant checkpointing, replication 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 replication overhead, replication process context storage space and low throughput. Consequently the major issue in fault-tolerance is to effectively tolerate failures using job replication 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 replication techniques are not merely based on scheduling algorithm [3, 14]. 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 [15, 16]. 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 replication or without checkpoint automation and replication [21].

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, 24]. In this paper we have presented heuristics that resolves number of replica overhead issue so as to provide high job throughput in presence of failure [25].

II Automatic File Replication Service

The major drawbacks in most of grid simulators such as SimJava, Bricks, MicroGrid, SimGrid, MicroGrid, SimGrid, GangSim, MONARC, OptorSim, EcoGrid, Opportunistic Grid, SchMng are based on scheduling of jobs to grid resources and lack of synchronization among the simulation components and the grid middleware. These are all based on to simulate scheduling of jobs, job migration, replication analysis etc. Most of these simulators don't analyses overhead of grids completely and not support for fault tolerant analysis such as checkpointing and replication, whenever a node fails it has to recover from last saved checkpoint to overcome job delay. But many times there is an overhead due to many replicas files in without appropriate file replication automation approaches. The simulators GridSim for scheduling of jobs and OptorSim data replication analysis toolkit are java based simulation toolkits. In order to tackle this we have proposed an optimal checkpoint replication automation strategy which effectively changes the dynamic behavior of the grid middleware working mechanism according to the grid resource failure information [9, 20, and 25].

The problem of transferring a checkpoint from one site to another in real time takes bandwidth consumption and access delay. Data replication technique is a frequently reduce bandwidth consumption and access latency by taking one or more checkpoints to other sites. Two kinds of replication methods are possible. The static replication creates and manages the replicas manually. The dynamic replication changes the location of replicas and creates new replicas to other sites automatically. Judging from the fact that resources or data checkpoints are always changing, it is clear that dynamic replication is more appropriate for the Data Grid [5, 8]. In this we have observed several dynamic replication strategies proposed in earlier [20, 26]. The multi-tier hierarchical Data Grid architecture supports an efficient method for sharing of data, computational and other resources. In [11, 12], two dynamic replication algorithms had been proposed for the multi-tier Data Grid, which are Simple Bottom-Up (SBU) and Aggregate Bottom-Up (ABU). The basic concept of SBU is to create the replicas as close as possible to the clients that request the data checkpoints with a rate exceeding a predefined threshold. If the number of requests for checkpoint f exceeds the threshold and f has existed in the parent node of the client which has the highest request rate, then there is might not to replicate [13, 17 and 23]. The SBU replication algorithm has the disadvantage that it does not consider the relations among these historical records but processes the records individually. After aggregation, the checkpoint with the highest request rate will be replicated if the rate is above a threshold. With the exception of aggregation steps, all concepts in ABU are similar to the SBU replication algorithm [1, 3, and 14].

III Automatic Replication File 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 AFR mechanism is to minimize the effect of grid faults and to reduce the fault recovery time using optimal automation of replication of context 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 scheduled by Two Level Scheduling Strategy to assign checkpoint interval and also to provide automatic failure replica (context file of checkpoint) to the grid resource. The interaction among different components of the AFR is shown in the following Figure.1. The AFR optimizes the checkpoints to improve the efficiency over the execution of the failed job from the last saved checkpoint [25]. 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 AFR is the cluster information to reduce the data transfer time by reducing number of intercommunications [19, 22]. 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 Work_Manager 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 Work_Manager connects to the External_Schedulers to get information of all available grid resources to execute the job. The Work_Manager uses response time, resource failure rate and resource failure time to construct the list of suitable resources that can execute the job.

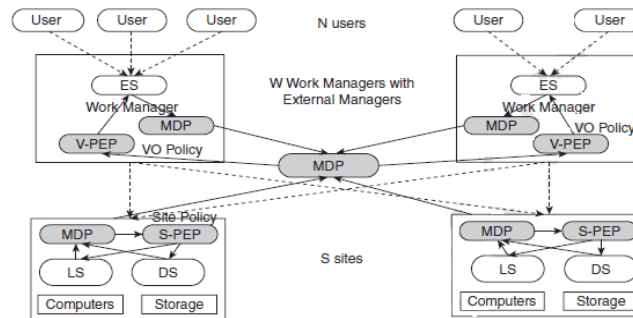


Figure 1: GridSim Components.

In Figure.1 GridSim architecture contains the components such as Work_Manager, External_Schedulers, Virtual_Organization Points, and autonomous computational resources as portable batch system message oriented middleware (Local_Schedulers and Data_Schedulers). Each component accomplishes set of tasks, explained in 3.2 and 3.3.

To solve grid application, the grid user submits the jobs along with QoS which requires large computational power for computation. Virtual_Organization Points gets the list of failure resources and checkpoint file information using computational resource replica-logs. The Work_Manager gets the failure patterns, checkpoint intervals and available resources from Virtual_Organization Points, optimal checkpoint automation library and External_Schedulers respectively. The External_Schedulers 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 [10]. The Work_Manager finds the optimal checkpoints for each resource and sends the list to External_Schedulers to allocate the jobs to resources. The External_Schedulers allocates the jobs to the selected resources. Based on the checkpoint interval the Virtual_Organization Points 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 External_Schedulers. External_Schedulers collects aggregated results and ships it to grid user.

This component takes grid application from the user which typically requires large scale heterogeneous resources, processing nodes with QoS of an application. External_Schedulers contains information of all available resources in the grid required by the Work_Manager. The information includes resource speed, current load, resource failure rate and total failure time of each resource. For each job j dispatched by the Virtual_Organization Points, if Virtual_Organization Points receives a job completion message then it sends a message to the External_Schedulers to increment S (Success) or if Virtual_Organization Points receives a response as failure message then send it a message to External_Schedulers to increment F (Failure), if there is a checkpoint stored then Virtual_Organization Points dispatches the not completed part of the job along the checkpoint status to the second resource in the resources list, else Virtual_Organization Points dispatches the whole job along to the second resource in the resource list (see Figure 2).

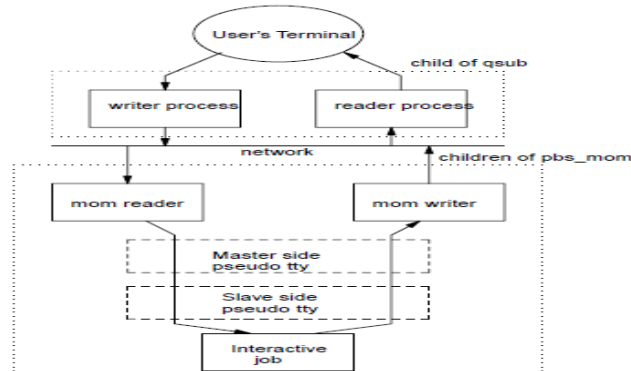


Figure 2: Interactive Job Communication Flow.

The *Internal_Schedulers* receives and stores partially executed results of a job from children of *Local_Schedulers* and *Data_Schedulers*. These intermediate results are called checkpoint status. For each job there is only one record of checkpoint status. When *Virtual_Organization Points* receives a new checkpoint status it overwrites the old one and if it receives a job completion message from the resource it removes the record of such job.

Virtual_Organization Points is an important component of AFR. The main functions of *Virtual_Organization Points* are determining the number of checkpoints for each *Local_Schedulers* and *Data_Schedulers* and the checkpoint interval for each job. *Virtual_Organization Points* receives a job with its assigned list of resources from *Work_Manager*. It connects to *Virtual_Organization Points* to get information about the failure history of grid resources assigned to the job. Based on failure rate of the resource, the *Virtual_Organization Points* 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 *Virtual_Organization Points* calculates the number of checkpoints and the checkpoint interval for each resource using replica log files which are collected.

The *Work_Manager* is the major component in grid architecture. The *Work_Manager* initiates the application with n-number of jobs to allocate grid resource. Now for each set of job the *Work_Manager* gets the list of available resources from *Internal_Schedulers*. The *Work_Manager* sorts the suitable list of resources according to resource response time -RRT, failure rate -FR and average failure time -AFT. Afterwards *Work_Manager* dispatches the list to checkpoint header. The *Work_Manager* schedules the resources using optimal checkpointing approach-AFR with respect to job list acquired from *Virtual_Organization Points*.

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

$$\text{MIN } TT_{(X)} = \sum_{i=0}^N (T_{ij}^1 + T_{ij}^1 * F_i^1) \quad (1)$$

$$\text{Where } T_{ij}^1 = t_{js} + t_{je} + t_{jr} \quad (2)$$

$$F_i = t_{jc} + t_{jr} \quad (3)$$

Subject To:

$$(i). T_{ij}^1 < RT_{ij}^i$$

$$(ii). 0 < F_i < 1$$

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}^1 is the estimated minimum turnaround time for j resources to carry out job i, F_i^1 is the failure resource, t_{js} scheduling time for i^{th} job on resource j, t_{je}^{ij} 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

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

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

$$CI_i = P(x_i | k, j) = P(k, j | x_i) * P(x_i) \quad (5)$$

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

$$l1 = \frac{d^*CI_i}{dj} \quad (6)$$

$$l2 = \frac{d^*CI_i}{dk} \quad (7)$$

Optimal checkpoint interval can be calculated as

$$CI_{i_new} = (l1+l2)/2 \quad (8)$$

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.

IV Replication 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. 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 AFR 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

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

The grid Resource are sorted by using

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

Finding the checkpoint interval

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

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

$$n = N / \sum TT_j^{tp} \quad (15)$$

Where, $X_{i=1}^n$ Computational jobs $(x_1, x_2, \dots, x_n)$, CI_j - 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.

V. Performance Evaluation

In this experiment, applications with 10 to 200 jobs with 10-200 faults are modelled. The size of each job replica context is approximately selected from 1 KB up to 250 MB. The number of resources in the grid can reach up to 4 clusters with cluster have 30 nodes. 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 AFR approach is compared with the combination of Two level replication strategy (TLRS) with Two level scheduling strategy (TLSS), Structure based on internet hierarchy (BHR) with TLSS, Least Recently Used (LRU) with TLSS the details of that experiment can be found in Table-1. Based on the experiments the following graphs Figure 3 and Figure 4 are plotted. In this experiment, applications with 100 to 2000 jobs with 10-200 faults are modeled. 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 AFR approach is compared with the other replications methods, the details of the experiments are be found in Table-1&s. Based on the experiments the following graphs Figure 3 and Figure 4 are plotted.

Replication Algorithm	Number of Jobs				
	10	50	100	150	200
AFR	350	1750	3600	4700	7200
TLRS+TLSS	360	1800	3800	4800	7500
BHR+TLSS	365	1850	3900	4940	7650
LRU+TLSS	370	1880	4100	5100	7900

Table 1: Average replication time (sec).

Replication Algorithm	Number of Jobs				
	10	50	100	150	200
AFR	14	64.81481	124.1379	156.6667	218.1818
TLRS+TLSS	14.4	66.66667	131.0345	160	227.2727
BHR+TLSS	14.6	68.51852	134.4828	164.6667	231.8182
LRU+TLSS	14.8	69.62963	141.3793	170	239.3939

Table 2: Average storage usage (mbs).

The Table-1& 2 shows the comparison between AFR approach with TLRS, TLSS, BHR and LRU with TLSS strategy [7, 28]. Most of the existing grid simulators like GridSim, SimGrid do not support fault-tolerant scheduling and checkpointing. To assess the optimal replica file 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 replica context into the checkpoint server so that if any one node fails then chronologically the last saved context file can be accessed automatically. At present our claim methodology ropes optimal replica automation for large scale grid application small statistical samples. We have installed TORQUE resource manager service on all nodes and applied auto replica generation based on optimal replication interval. Since it improves the performance of the grid application in presence of a failure when compare to TLRS&TLSS approaches.

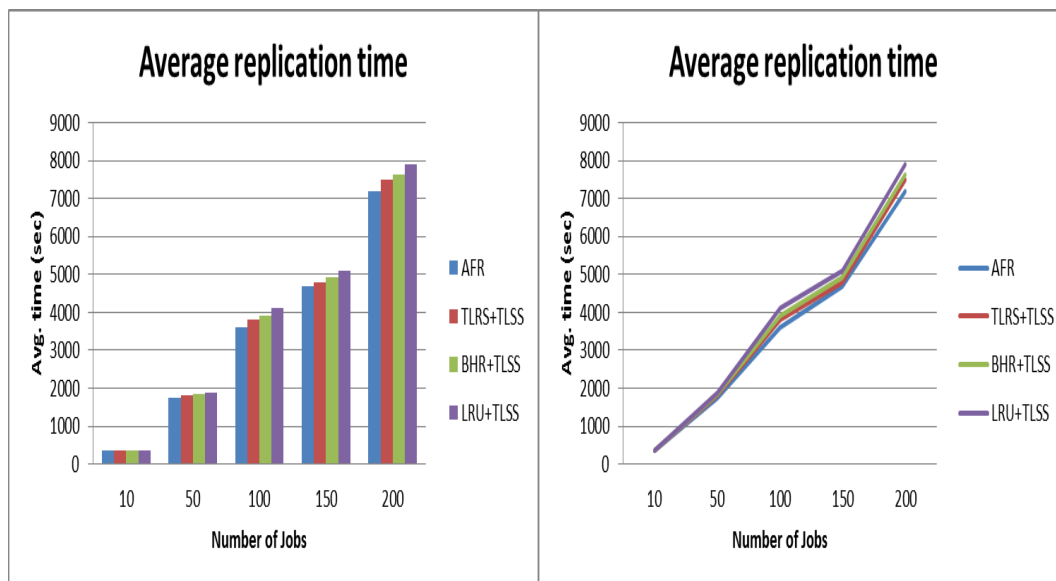


Figure 3: Processing time comparison by varying number of jobs.

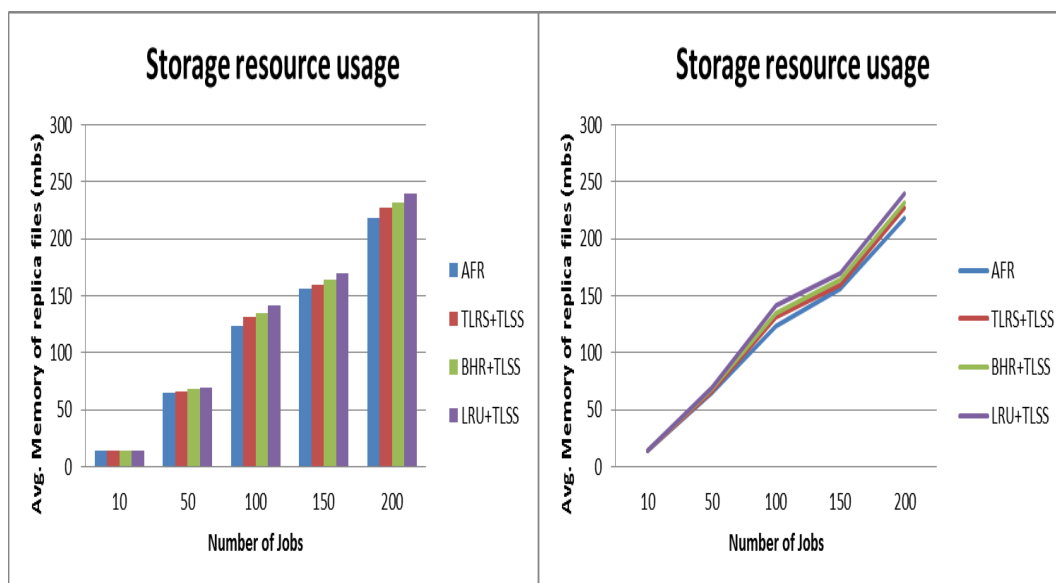


Figure 4: Space comparison by varying number of jobs

The Figure 3, Figure. 4 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 AFR algorithm is compared with LRU, BHR and TLRS in which these algorithms depend on TLSS scheduling to make replication. But the AFR method uses resource fault index to make replication using Hidden Markov Model states (HMM) to make replica scheduling for calculating effective prediction based replication files. Experimental results show that AFR approach effectively replicates 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 AFR methodology is far better than the TLRS&TLSS 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 replication approach using fault tolerant scheduling approach. If the computational resources are new then checkpointing interval might be unpredictable.

VI. Conclusions

In this work we have proposed an Automatic File Replication mechanism for computational grids. Here the optimal replica files are calculated using the predicted failure patterns of resources using HMM. The unique optimal replica context files are generated for each resource, which yields reduction in number of context replicas. Hence this methodology considerably reduces the replication overheads by reducing number of context files. The developed AFR model which shows better turnaround time, storage space and throughput of different

jobs compare to LFU and TLSS algorithms. Experimental results show that AFR methodology effectively tolerates the resource failures. Therefore we conclude that the developed checkpointing system provides better performance when compare to adaptive replication approaches.

References

- [1]. Jianhua Jiang, Huifang Ji, "scheduling algorithm with potential behaviors", Journal of Computers, VOL. 3, NO. 12 , December 2008.
- [2]. Ming Tang, Bu-Sung Lee, Xueyan Tang, Chai-Kiat Yeo. "The Impact of Data Replication on Job Scheduling Performance in the Data Grid", Future Generation Computer Systems, Volume 22, Issue 3, February 2006, Pages 254-268
- [3]. Ali Elghirani, Riky Subrata, Albert Y. Zomaya, and Ali Al Mazari., "Performance Enhancement through Hybrid Replication and Genetic Algorithm Co-Scheduling in Data Grids", Advanced Networks Research Group, School of Information Technologies, University of Sydney, NSW 2006 Australia.
- [4]. Sang-Min Park, Jai-Hoon Kim, Young-Bae Ko: "Dynamic Grid Replication Strategy based on Internet Hierarchy", Book Series Lecture Notes in Computer Science, Grid and Cooperative computing book, Publisher Springer, August 2005, Volume 3033/2004, Pages 838-846
- [5]. Ruay-Shiung Chang, Jih-Sheng Chang, Shin-Yi Lin, "Job scheduling and data replication on data grids", Future Generation Computer Systems, Volume 23, Issue 7, August 2007, Pages 846
- [6]. Ashish Kumar Singh, Shashank Srivastava and Udai Shanker, "A Survey on Dynamic Replication Strategies for Improving Response Time in Data Grids", JBSTR, JULY 2013.
- [7]. Ruay-Shiung Chang ; Nat. Dong Hwa Univ., Hualien ; Hui-Ping Chang ; Yun-Ting Wang, "A dynamic weighted data replication strategy in data grids ", Computer Systems and Applications, AICCSA 2008, IEEE/ACS 2008.
- [8]. A. Chervenak, I. Foster, C. Kesselman, C. Salisburly, and S. Tuecke, "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets," Journal of Network and Computer Application, vol. 23, pages 187-200, 2000.
- [9]. D. G. Cameron, R. C. Schiaffino, P. Millar, C. Nicholson, K. Stockinger, and F. Zini, "OptorSim: A Grid Simulator for Replica Optimization," UK e-Science All Hands Conference 31 August - 3 September 2004.
- [10]. I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems," IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2005.
- [11]. W. Hoschek, F. J. Jaen-Martinez, A. Samar, H. Stockinger, K. Stockinger, "Data management in an International data grid project", Proceedings of the First IEEE/ACM International Workshop on Grid Computing (GRID '00), Lecture Notes in Computer Science, vol. 1971, pages 77-90, Bangalore, India, December 2000.
- [12]. The European Data Grid Project, <http://eudatagrid.web.cern.ch/eu-datagrid/Winter>.
- [13]. Nhan Nguyen Dang, Sang Boem Lim: "Combination of Replication and Scheduling in Data Grids", IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.3, March 2007.
- [14]. T. Phan, K. Ranganathan, and R. Sion, "Evolving toward the perfect schedule: Co-scheduling job assignments and data replication in wide-area systems using a genetic algorithm", Job scheduling strategies for parallel processing (11th international workshop), JSSPP 2005, Cambridge MA, 2005.
- [15]. Klaus Krauter, Rajkuar Buyya and Muthucumaru Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing", 2002; 32:135-164 (DOI: 10.1002/spe.432).
- [16]. Klaus Krauter and Manzur Murshed, GridSim a toolkit for the modelling and simulation of distributed resource management and scheduling for grid computing, CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE Concurrency Computat.: Pract. Exper. 2002; 14:1175-1220 (DOI: 10.1002/cpe.710).
- [17]. The Data Grid Project. <http://www.eu-datagrid.org>
- [18]. Parallel workload Project. <http://www.parallelworkload.org>.
- [19]. The European data grid project. <http://eudatagrid.web.cern.ch/eu-datagrid/Winter>.
- [20]. W.H. Bell, D.G. Cameron, L. Capozza, P. Millar, K. Stockinger, F. Zini, Simulation of dynamic grid replication strategies in OptorSim, in: Proceedings of the Third ACM/IEEE International Workshop on Grid Computing, Grid2002, Baltimore, USA, in: Lecture Notes in Computer Science, vol. 2536, 2002, pp. 46-57.
- [21]. E. Deelman, H. Lamahemedi, B. Szymanski, S. Zujun, Data replication strategies in grid environments, in: Proceedings of 5th International Conference on Algorithms and Architecture for Parallel Processing, ICA3PP'2002, IEEE Computer Science Press, Beijing, China, 2002, pp.378-383.
- [22]. H.H. Mohamed, D.H.J. Epema, An evaluation of the close-to-files processor and data coallocation policy in multiclustes, in: 2004 IEEE International Conference on Cluster Computing, IEEE Society Press, San Diego, California, USA, 2004, pp. 287-298.
- [23]. M. Carman, F. Zini, L. Serafini, K. Stockinger, Towards an economybased optimisation of fileaccess and replication on a data grid, in: Proceedings of 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGrid 2002, IEEE-CS Press, Berlin, Germany, 2002, pp. 340-345.
- [24]. P. Kunszt, E. Laure, H. Stockinger, K. Stockinger, Advanced replica management with rector, in: Proceedings of 5th International Conference on Parallel Processing and Applied Mathematics, PPAM 2003, Czestochowa, Poland, September 2003, pp. 848-855.
- [25]. D.G. Cameron, A.P. Millar, C. Nicholson, OptorSim: A simulation tool for scheduling and replica optimisation in data grids, in: Proceedings of Computing in High Energy Physics, CHEP 2004, Interlaken, Switzerland, September 2004.
- [26]. K. Ranganathan and I. Foster, "Identifying Dynamic Replication Strategies for a High Performance Data Grid". In Proceedings of the International Grid Computing Workshop, Denver, Colorado, USA, 2001. International Journal of Grid Computing & Applications (IJGCA) Vol.1, No.1, September 201037
- [27]. I. Foster, K. Ranganathan, Decoupling computation and data scheduling in distributed data intensive applications, in: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing, HPDC-11, IEEE, CS Press, Edinburgh, UK, 2002, pp. 352-3
- [28]. Somayeh Abdi, Somayeh Mohamadi, "Two Level Job Scheduling and Data Replication in Data Grid", IJGCA-Sep-2010.