Global Journal of Computer Science and Technology
Cloud and Distributed
Volume 13 Issue 2 Version 1.0 Year 2013
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Fault Tolerant Scheduling of Partitioned and Grouped Jobs in Grid Computing (FTPG)

By K. Srikala & S. Ramachandram
Osmania University, India

Abstract - Computational grids have the potential for solving scientific and large - scale problems using heterogeneous and geographically distributed resources. In addition to the challenges of managing and scheduling resources reliable challenges arise because the grid infrastructure is unreliable. There are two major problems in Scheduling the Grid 1) Efficient Scheduling of jobs, 2) Providing fault tolerance in a reliable manner. Most of the existing strategies do not provide fault tolerance. There are some algorithms which provide fault tolerance but, they do a large amount of redundant computation to provide fault tolerance. This paper addresses this issue and minimizes redundant work by using a group level table of data. This technique is suitable for partitioning and group scheduling of jobs.

Keywords : *grid computing, grid scheduling, fault tolerance, redundant computation and group result table.*

GJCST-B Classification : *C.2.1*

Strictly as per the compliance and regulations of:

Fault Tolerant Scheduling of Partitioned and Grouped Jobs in Grid Computing (FTPG)

K. Srikala [α] & S. Ramachandram [σ]

Abstract - Computational grids have the potential for solving scientific and large - scale problems using heterogeneous and geographically distributed resources. In addition to the challenges of managing and scheduling resources reliable challenges arise because the grid infrastructure is unreliable. There are two major problems in Scheduling the Grid 1) Efficient Scheduling of jobs, 2) Providing fault tolerance in a reliable manner. Most of the existing strategies do not provide fault tolerance. There are some algorithms which provide fault tolerance but, they do a large amount of redundant computation to provide fault tolerance. This paper addresses this issue and minimizes redundant work by using a group level table of data. This technique is suitable for partitioning and group scheduling of jobs.

Keywords : grid computing, grid scheduling, fault tolerance, redundant computation and group result table.

I. Introduction

Internet computing has become popular and its popularity is growing with the emerged infrastructure such as web services and computational grids. It is possible to develop applications that support various Internet wide collaborations through seamlessly harnessing appropriate Internet resources. Grid computing is different from distributed computing as grid computing is concerned with large-scale resource sharing, innovative applications and the orientation is high – performance. Therefore, it is important to provide fault tolerant scheduling of jobs.

In this paper we propose a fault tolerant scheduling algorithm for partitioning and group scheduling of jobs, i.e., partitioning a Heavy Weight job into a group of Light Weight Jobs and Scheduling Light Weight Jobs as a group.

There are three categories of fault tolerant approaches proposed for Grids. They are Job replication, Check pointing and rollback and Adaptive fault tolerance.

a) Job Replication

In Job replication the same job is replicated to be executed on multiple, not dependable resources to guard the job against a resource failure. Number of replicas could be generated statically or dynamically using the history of failure percentage of a node.

b) Check Pointing and Rollback

In this the state of the running job saved at some fixed points called check points. This state used for recovery of the job in case of a resource failure.

c) Adaptive Fault Tolerance

This is the approach where Job replication combines with Check pointing. Here, only one replica will be executing and all other replicas updated periodically, so that if active replica is not available during the recovery of a resource failure we use other replicas.

In First two Categories of Fault Tolerance mechanisms there is so much redundant computation involved which leads to high overhead. Though the redundant computation minimized in adaptive fault tolerance communication overhead is still there in this.

FTPG provides Fault tolerant scheduling of jobs by maintaining a group table of values for Orphan jobs (Jobs not able to reach Parent Job because of failure of processor at Parent Job). M. Amoon et al [1] developed Design of a Fault – Tolerant Scheduling for Grid Computing using Job replication. This uses a fixed number of replicas. Ming Tao et al [2] developed a New Replication Strategy for Grid Workflow Applications. They conclude that their strategy optimizes the replication phase in terms of the characteristics of the workflow applications. J.H. Abawahy et al[3] developed Fault Tolerant Scheduling Policy for Grid Computing Systems. This uses job replicas. Malarvizhi et al [4] developed Fault Tolerant Strategy for Computational Grid Environment.

This Scheme uses history of fault occurrence of nodes. August´_n Caminero et al [5] developed Extending GridSim with architecture for Failure Detection. In all these Schemes the main drawback is redundant computation and some techniques are not able to complete the jobs within the deadlines as a result there is no Quality of Service provided. These are the observations that bring new challenges in designing Fault tolerant Scheduling of jobs, That allows application to continue execution in case of a resource failure. The FTPG addresses all the issues mentioned above.

Author α : Offshore Faculty, TutorVista Global Pvt. Ltd., Hyderabad, Andhra Pradesh, India. E-mail : kesreekala@gmail.com
Author σ : PROFESSOR, UCE, Computer Science Department, Osmania University, Hyderabad, Andhra Pradesh, India.
E-mail : schandram@gmail.com

II. Related Work

Review of literature reveals that grid environments are more failure prone as the resources may come and go at any time. The most popular fault tolerant mechanism is check pointing. Check pointing is used in Grid computing by systems such as Condor and Cactus. Writing the state of process to a stable storage is the main overhead of this technique. As discussed in the previous section creating greater number of replicas and starting them altogether causes Overhead because of redundant computation.

FTPG is different from other fault-tolerance algorithms. This technique is suitable for partitioned and grouped gridlets. That is Heavy weight gridlets are partitioned into a group of Light weight gridlets and Light weight gridlets are scheduled as a group.

FTPG based on redoing work lost in crashes. FTPG eliminates the common problem of most fault tolerant algorithms that is redundant computation done by a live processor as a result of crash of its parent processor. This happens in case of Orphan gridlets. Therefore, the Overhead in FTPG is very less. In most of the existing fault tolerant algorithms the processor which has finished working of an Orphan gridlet must discard the result of this gridlet because it does not know where to return that result.

The main objectives of FTPG technique are

- Reduce the amount of redundant computations.
- Providing Fault tolerant service within deadlines imposed by the user.

III. Fault Tolerant Scheduling Algorithm

In this section the fault tolerant scheduling algorithm for a group of light weight jobs presented and shown in Figure 1.

```
1. Start
2. forall (gridlets lost because of a crashed processor)
     put job back into the work queue.
3. forall (descendant gridlets of crashed processor)
     if (descendant is finished)
     store the result in group table.
     else
     abort the descendant.
```

Figure 1 : The crash recovery procedure for a live processor

IV. Process

FTPG is suitable for Scheduling a group of light weight jobs. That is when user submits a Heavy Weight Job that is partitioned into Light Weight Jobs and these Light Weight Jobs are scheduled as a group. Or if the user generates Light Weight Jobs they are scheduled as a group. FTPG is based on storing a table of values for the Orphan Jobs. Suppose that a grouped job is sent for execution. All the members of a group has access to group result table designed for this group. This table is used during the crash recovery procedure for storing/reusing the Partial results of Orphan job. (A job/gridlet becomes Orphan when it is not able to send its result back to the parent job.) . Each job can be identified by using its group id, job id and other parameters. When the light weight job's processor is not able to locate its grouped job processor each of the light weight gridlet becomes an orphan and its results are stored in group result table. This table is replicated on all the processors of the group members. The replicas of the table do not have to be strongly consistent because if the processor is not able to find a job it can be recomputed quickly since we are using Light weight jobs here. So updates of the table are propagated asynchronously (for ex: after a fixed amount of time). Since the table stores only Orphan jobs, number of jobs stored is very less. Therefore, the overhead by using the table is very less.

With the use of a group table we are storing only the results of finished jobs. They are very easy to store at the same time redundant computation exists only for partially executed light weight jobs. Though this computation is redundant as the gridlet is light weight gridlet it can be computed very quickly and it is possible to finish the jobs within the deadlines imposed by the user.

Figure 2 and Figure 3 demonstrates an example of before the crash and after the crash of group level processor GJ2.

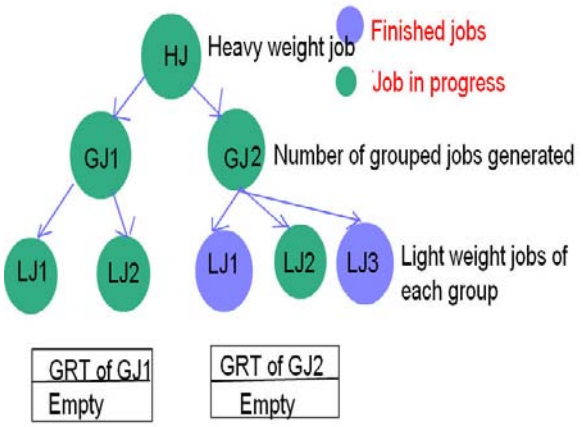

Figure 2 : Before the crash of GJ2's Processor

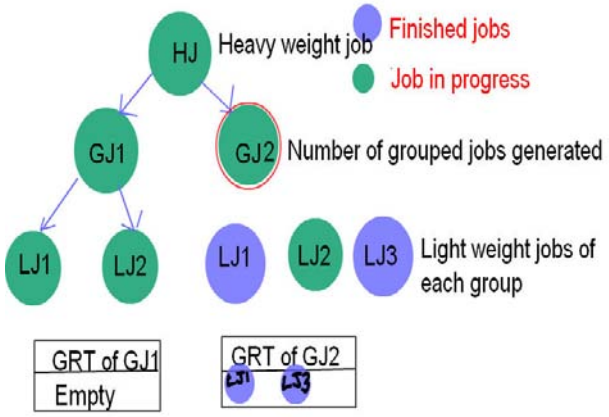

Figure 3 : After the crash of GJ2's Processor

As shown above when the Parent processor is crashed Orphan jobs store the output of finished job in Group result table (GRT). If the processor of Light weight job that is under computation is crashed it can be recomputed as the job is a Light weight job and it takes less time for executing it.

Replicas of the table are maintained at each of the processor of Light weight jobs in a group. Update propagation is Asynchronous, The replica's need not be strongly consistent because all the jobs are light weight jobs , So if the finished job's entry is not found in a table it can be computed again in a short period.

In this technique very less amount of redundant computation is there after a crash and adds very little overhead. This Technique is suitable where the jobs are partitioned and a grouped job is sent for execution. In this way redundant computation is minimized that is present in other schemes like check pointing and job replication.

V. Simulation Environment and Results

The Algorithms for comparison (Check pointing (CP),Fixed number of job replicas (4 replicas) FJR (4),Adopted job replication AJR, and fault tolerant Scheduling of Partitioned and grouped jobs FTPG) are implemented on a machine based on Pentium 1.6GHz with 120GB HDD and RAM of 1GB on Microsoft Windows XP. This experiment is carried out on a simulated grid environment provided by Gridsim.

The Simulation environment consists of 6 resources. These 6 resources have 25 PE's in total having different processing capabilities in terms of Millions of Instructions per Second (MIPS).Further it is assumed that there are 8 users with 12 to 30 jobs.

FTPG is compared with CP, JR (4) and AJR and the results are recorded on graphs.

The parameters used for performance comparison are explained below.

a) Redundant Computations

Calculations that are carried out by more than one processor.

b) Processing time of gridlet

The amount of time taken by the processor to finish execution of gridlet.

c) Fault tolerant Service

Fault-tolerant service describes a technique that is used in case of a processor failure, a backup component or procedure can immediately take its place with no loss of data.

The following are the graphs drawn for comparison.

Figure4 shows the number of gridlets completed within the deadlines when 20% faults are injected. We can see that a good percentage of gridlets are completed by using FTPG when compared to AJR, FJR (4) and CP. As the deadline is increased number of jobs completed is also increased. Where as it is not true in case of CP and FJR (4). When the Fixed number of replicas is used as the number of replicas increases the number of jobs executed will be less because of the less availability of resources. FTPG reduces resource wastage and increases user benefit.

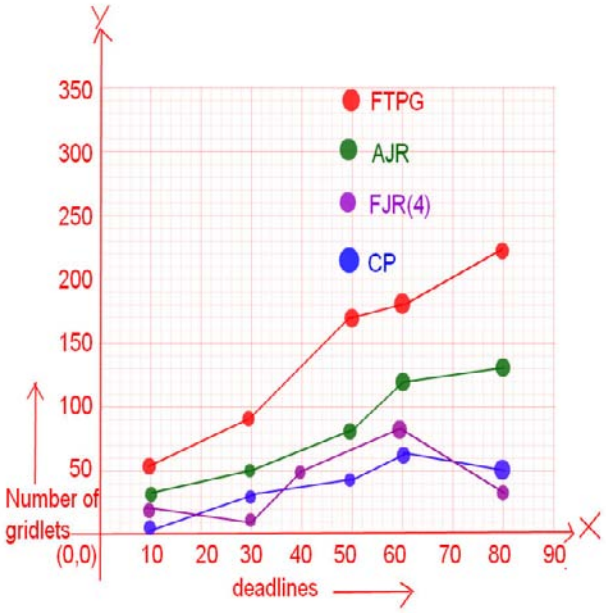

Figure 4 : Graph between the number of gridlets completed within deadlines when 20% faults are injected

The number of gridlets and their execution time is plotted in the graph as shown in Figure 5. These values are measured when 30% of the processors/resources are failed. It can be observed that as the number of gridlets is increased there is a very little increase in the execution time by using FTPG whereas in the other techniques AJR, FJR (4) and CP as the gridlets are increased their execution time increases drastically.

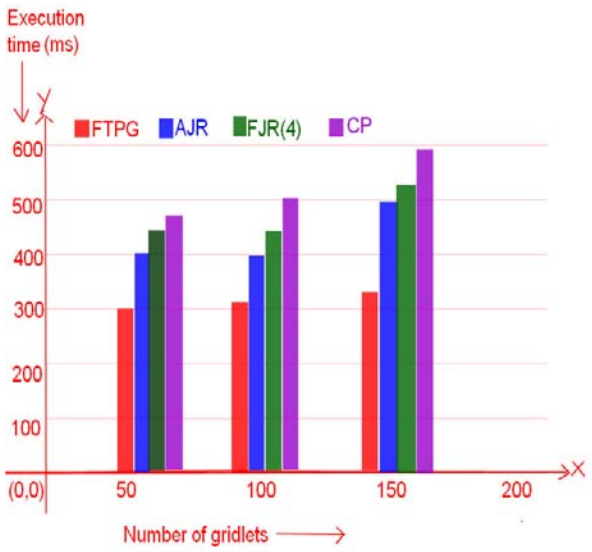

Figure 5 : Graph between number of gridlets and the time taken for execution when 30% faults are injected

The number of redundant computations when 30% of the resources failed is measured using CP, AJR, FJR (4) and FTPG and recorded on a pie chart. The results are recorded in figure 6. Figure 6 demonstrates that by using FTPG number of redundant computations were very less in percentage. This improves resource owner benefit by utilizing the resources properly. Therefore, it is good to use FTPG for fault tolerant Scheduling.

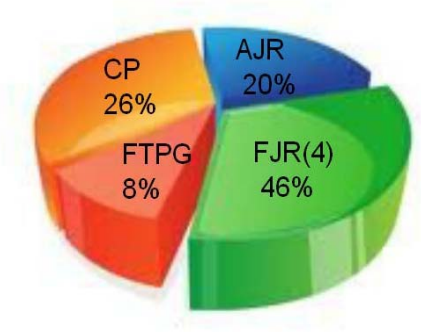

Figure 6 : Redundant computations because of 30% of the resources are failed

VI. Conclusions and Future Work

The FTPG algorithm provides QoS to the user at a good level of acceptance. It uses very less storage for Group Result Table and redundant computations are greatly reduced. The Overhead in this technique is very less.

This technique improves user benefit by providing Quality of Service and improves resource owner benefit by minimizing the wastage of resources. This technique can be extended to include network failure, data and service resource failures.

References Références Referencias

1. M. Amoon, "Design of a fault tolerant Scheduling system for Grid Computing" IEEE Computer Society 2011 Second International conference on Networking and Distributed Computing © 2011 IEEE DOI 10.1109/ICNDC.2011.29.
2. Ming Tao, Shoubin Dong and Kejing He, "A New Replication Scheduling Strategy for Grid workflow Applications," IEEE@Computer Society 2011 Sixth Annual China Grid Conference.
3. J. H. Abawajy, "Fault-Tolerant Scheduling Policy for Grid Computing Systems" Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04) (C) 2004 IEEE.
4. MALARVIZHI NANDAGOPAL, "FAULT TOLERANT SCHEDULING STRATEGY FOR COMPUTATIONAL GRID ENVIRONMENT" Malarvizhi Nandagopal et. al. / International Journal of Engineering Science and Technology Vol. 2(9), 2010, 4361-4372.
5. August´_n Caminero Anthony Sulistio, Blanca Caminero, Carmen Carri´on and Rajkumar Buyya, "Extending GridSim with an Architecture for Failure Detection".

Global Journals Inc. (US) Guidelines Handbook 2013

www.GlobalJournals.org