www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 03 Issue 07 July, 2014 Page No. 6904-6908

Detection of Crash Transient Failure during Job Scheduling using Replication Technique

Inderpreet Kaur[1], Sarpreet Singh[2]

[1]Department of Computer Science Engg. ,Sri Guru Granth Sahib World University,
Fatehgarh Sahib, Punjab, India
inderpreetrao@gmail.com
2Department of Computer Science Engg. ,Sri Guru Granth Sahib World University,
Fatehgarh Sahib, Punjab, India
ersarpreetvirk@gmail.com

Abstract: ***Grid computing or computational grid is always a vast research field in academic. Computational grid provides resource sharing through multi-institutional virtual organizations for dynamic problem solving. Various heterogeneous resources of different administrative domain are virtually distributed through different network in computational grids. Thus any type of failure can occur at any point of time and node running in grid environment might fail. Hence fault tolerance is an important and challenging issue in grid computing as the dependability of individual grid resources may not be guaranteed. In order to make computational grids more effective and reliable fault tolerant system is necessary. The objective of this paper is to test the crash and omission transient failure in resource scheduling. This paper presents an overview of fault tolerance and its techniques, task replication and most fitting resource allocation algorithm.***

KEYWORDS:- Computational grid, Fault tolerance, Failure, Task Replication

1. Introduction

Grid computing is a form of distributed computing that involves coordinating and sharing computational power, data, and storage and network resources across dynamic and geographically dispersed organizations [1]. A grid [2], known to be a large-scale virtual organization, is enabling to solve complex scientific and compute-intensive problems. The virtual organization is formed with geographically distributed hardware and software infrastructure of flexible, secure and coordinated shared vast amounts of heterogeneous resources from multiple administrative domains. Computational grid environment is shown in Figure 1.

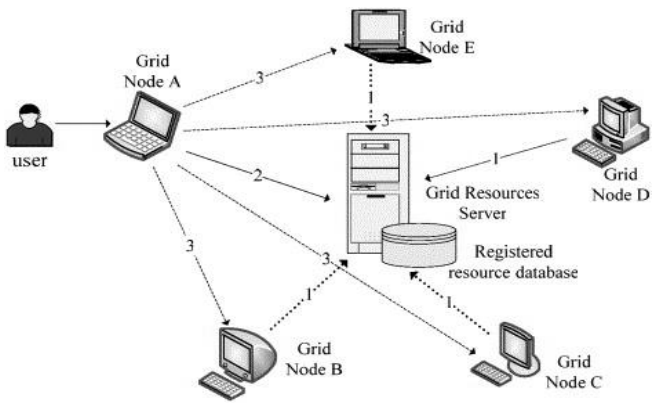

Figure 1: Computational Grid Environment [3]

Heterogeneous computational nodes have connected to form a Grid test-bed. In this test-bed registered resource database and Grid resources server is also shown. Server or database might be accessed during computation of large job. User can submit job through any node among Node A, Node B, Node C, Node D or Node E in Grid. Job might necessitate adapting the changed resource scenario in Grid environment. Hence, fault tolerance of resources is major challenging issue in dynamic virtual computational Grids.

Fault tolerance is an important property in grid computing as the dependability of individual grid resources may not be guaranteed. In many cases, an organization may send out jobs for remote execution on resources upon which no trust can be placed; for example, the resources may be outside of its organizational boundaries, or may be shared by different users at the same time.

A fault tolerant approach may therefore be useful in order to potentially prevent a malicious node affecting the overall performance of the application. As applications scale to take advantage of Grid resources, their size and complexity will increase dramatically.

A major challenge in a dynamic grid with thousands of nodes connected to each other is fault tolerance. The more resources and components involved the more complicated and error-prone becomes the system. To comprehend fault tolerance mechanisms, it is important to point out the difference between faults, errors and failures [4].

- *Fault*: A fault is a violation of a system's underlying assumptions.
- *Error:* An error is an internal data state that reflects a fault.
- *Failure:* A failure is an externally visible deviation from specifications.
- *Transient Failure:* Staying for a short time period.
- *Crash:* When nothing happens.
- *Omission:* Error occurs when one or more responses fail.

In reality, a fault need not result in an error, or an error in a failure.

Different types of faults, classified based on several factors, are mentioned in the following:

- *Physical faults:* faulty storage, faulty CPUs, faulty memory.
- *Unconditional termination:* Mostly, user pressed Ctrl+C.
- *Network faults:* packet corruption, faults due to network partition, packet loss.
- *Lifecycle faults:* Legacy or versioning faults.
- *Processor faults:* Machine or operating system crashes.
- *Media faults:* Disk head crashes.
- *Service expiry fault:* The service time of a resource may expire while application is using the resources in grid.
- *Process faults:* software bug, resource shortage.
- *Interaction faults:* timing overhead, protocol incompatibilities, security incompatibilities, policy problems.

2. Review of Fault Tolerance Techniques

The main objective of grid computing is to maintain the workflows or services in presence of faults so that no failure stage is reached. This section presents a few fault tolerant techniques in the following:

2.1. Replication Technique

The term replication implies making copies or replicas of an existing entity. In grid environment, Job or task or data are replicated to tackle the faults. Replication technique to improve the fault tolerance of the fittest resource scheduling algorithm[5]. The fittest resource scheduling algorithm searches for the appropriate resource based on the job requirements, in contrary to the general scheduling algorithms where jobs are scheduled to the resources with best performance factor.

2.2. Checkpointing

The check pointing is one of the most popular technique to provide fault-tolerance on unreliable systems. It is a record of the snapshot of the entire system state in order to restart the application after the occurrence of some failure. The checkpoint can be stored on temporary as well as stable storage [6]. However, the efficiency of the mechanism is strongly dependent on the length of the check pointing interval. Frequent check pointing may enhance the overhead, while lazy check pointing may lead to loss of significant computation. Hence, the decision about the size of the check pointing interval and the check pointing technique is a complicated task and should be based upon the knowledge about the application as well as the system.

Therefore, various types of check pointing optimization have been considered by the researchers, e.g., (i) Full check pointing or Incremental check pointing (ii) Unconditional periodic check pointing or Optimal (Dynamic) check pointing (iii) Synchronous (Coordinated) or asynchronous (Uncoordinated) check pointing, and (iv) Kernel, Application or User level check pointing.

A. In ***pro-active mechanisms***, the failure consideration for the grid is made before the scheduling of a job, and dispatched with hopes that the job does not fail. Whereas, ***post-active mechanisms*** handles the job failures after it has occurred. However, in the dynamic systems only post-active mechanism is relevant [7].

B. In order to detect occurrence of fault in any grid resource two approaches can be used: the push or the pull model. In ***the push model***, grid components periodically send heartbeat messages to a failure detector, announcing that they are alive. In the absence of any such message from any grid component, the fault detector recognizes that failure has occurred at that grid component. It then implements appropriate measures dictated by the predefined fault tolerance mechanism. In contrast, in ***the pull model*** the failure detector sends liveliness requests ("Are you alive?" messages) periodically to grid components[8].

2.3 Failure Detection Service (FDS)

Hwang et al. [9] presented a failure detection service (FDS) and a flexible failure handling framework (Grid-WFS) as a fault tolerance mechanism on the grid. The FDS enables the detection of both task crashes and user defined exceptions. Like any middleware, a grid middleware is also responsible to hide, from the application developer, the technical details related to different syntax and access methods and to provide a consistent and homogeneous access to resources managed locally.

2.4 Proactive Fault Tolerance

Mohammad et al. [10] use agents to inject proactive fault tolerance in grids .Here autonomous, light-weight, intelligent agents monitor with individual faults. Agents maintain log of various information related to hardware conditions, memory utilization, resource constraints, network status and component failure. Based on this information and critical states, agent can enhance the reliability and efficiency of grid services.

2.5. Rescheduling Approach

F. Berman, H. Casanova, and A. Chien presented a rescheduling approach using stop and restart. The motivation for rescheduling is similar to the one, rescheduling a job to a better resource. However, a similar strategy can be adopted for fault tolerance. In[11] when a running application is signaled to migrate, it checkpoints user-specified data. For fault-tolerance, instead of user-directed check pointing such as the one in [11], an automatic check pointing approach has to be followed. In automatic check pointing, the state of the program is saved periodically to a persistent storage. When the machine running the program crashes, the program can be rescheduled to run on a different machine, continuing from the last check pointed state.

2.6. Distributed Fault-Tolerant Scheduling (DFTS)

Distributed Fault-Tolerant Scheduling (DFTS) [12] policy is based on the job replication mechanism. It does not reschedule the job when a fault occurs. Instead it schedules more than one copy of the job (called replicas) to a different set of resources. The job is considered complete if one of the set of resources successfully executed the complete job. The underlying assumption for job replication is that in a grid many resources may be underutilized.

2.7. FPLT ALGORITHM

D.Saha et.al[13] describes the FPLT algorithm in which the scheduler sorts the node and task information by each node's CPU speed and task's workload in order to reduce complexity for searching the fastest node and the largest task all the time. Then the scheduler assigns the largest task of the waiting tasks to the available fastest node. FPLTF will reduce to the same as Work queue when the tasks arrive one by one.

2.8. MFTF ALGORITHM

Wang et.al [14] in their MFTF algorithm declares the fitness between the task and the node based of the expected execution time and execution time of the node. The node which has much less difference is expected to be the fittest node and the task is allocated to it. The fitness definition seems to be arbitrary.

2.9. TASK DUPLICATION BASED SCHEDULING ALGORITHM (TANH)

Bajaj and Aggarwal [15] proposed an algorithm TANH (Task duplication-based scheduling Algorithm for Network of Heterogeneous systems) in which a new parameter is introduced for each task: the favorite processor (fp), which can complete the task earliest. Other parameters of a task are computed based on the value of fp. In the clustering step, the initial task of a cluster is assigned to its first fp, and if the first fp has already been assigned, then to the second and so on. Duplication based algorithms are very useful in Grid environments. The computational Grid usually has abundant computational resources (recall that the number of resource is unbounded in some duplication algorithms), but high communication cost. This will make task duplication very cost effective.

3 Proposed Work

As the most fitting resource algorithm (MFRS) [1] proposed the closeness factor which describes the appropriateness of the resource with the job requirements. The scheduling algorithm schedules the job to the appropriate resource rather the best performing resource. In the MFRS [1] algorithm the resources are categorized into L discrete levels. The literature entire resource is divided into ten levels and the fittest resources are allocated to the nodes from these levels.

In this paper two parameters energy consumption and CPU utilization. When work load increases then the chances for the node failure increases. So once the node fails the jobs allocated to that node fails. If a node fails then we can't be sure that the Gridlet submitted to that particular node will execute successfully. This problem will lead to the wastage of time as well as the resource and due to the average waiting time for the Gridlet.

The tasks that are handled by the crashed node will be replicated and new Gridlets ensures that the Job will be executed successfully. This will reduce the risk of job failure also.

When node will take job beyond its limit or energy consumption increases, work load will increasing rapidly that will create the chances of node crash. In any case, if any nodes break down occurs, then analyse the node. Every node will be having job assigned and shall be working independently in a connected grid network. The fault tolerance will behave as a non disturbing element. These factors will not affect the whole computational grid environment of the system.

The fault tolerance testing for crash and omission transient failure can be defined through the data flow diagram as below.

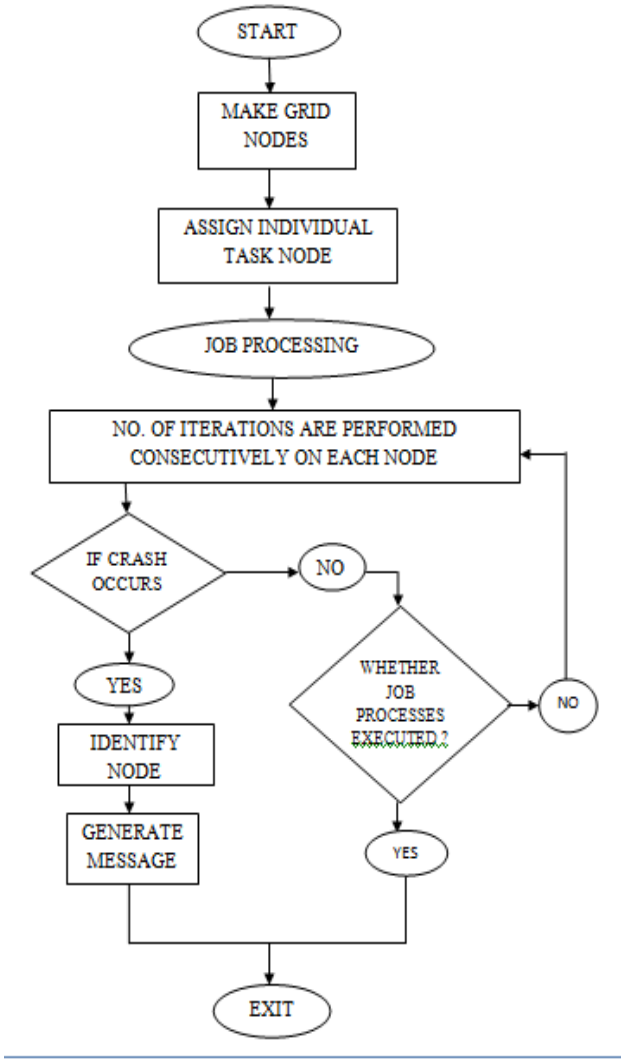

Figure 2: Data Flow Diagram for Fault Tolerance Testing of Crash Failure

3.1 Algorithm

Step 1: The Task Ti is submitted for scheduling and assigned task to the node having minimum job.

Step 2: Then task will replicate to the next node at the same level and data will store in the database of all the nodes assigned.

Step 3: Number of iterations are performed consecutively on each node.

Step 4: If energy consumption of node increases.

Step 5: Crash occurs and identify the crashed node.

Step 6: Generate the message that task will replicate to the next node at the same level.

Step 7: Else execute job processes

Step 8: Exit

4 CONCLUSION

In the grid environment the resources are heterogeneous and highly distributed. Hence they are prone to failures. Any scheduling algorithm will be more effective if fault tolerance is taken into account. The fault tolerant most fitting resource scheduling was implemented in a java based grid simulator and the results are evaluated and concluded that with fault tolerance feature added to the scheduling algorithm. Crash and omission transient failure is detected and using replication technique job assigned to the next node. At the end of proposed work, our scope would be to test the time taken to resolve crash and omission transient failure.

References

[1] Ian Foster and Carl Kesselman, S.T., (2001) "The Anatomy of the Grid: Enabling Scalable
Virtual Organizations", Intl J. Supercomputer Applications, 15: 200-222.

[2] Felipe Pontes Guimaraes and Alba Cristina Magalhaes Alves de Melo, "User-Defined Adaptive Fault-Tolerant Execution of Workflows in the Grid," 11th IEEE International Conference on Computer and Information Technology, 2011, IEEE Computer Society, pp.356-362.

[3] Shen et al, "System design and implementation of digital-image processing using computational grids", Computers & Geosciences, volume 31, Issue 5, pp: 619-630

[4] Arindam Da, Ajanta De Sarkar, "On fault tolerance of resources in computational grids", International Journal of Grid Computing & Applications (IJGCA) Vol.3, No.3, September 2012

[5] Ruay-Shiung Chang, Chun-Fu Lin, Jen-Jom Chen,"Selecting the most fitting resource for task execution" *Future Generation Computer Systems*, Volume 27, Issue 2, February 2011, Pages 227-231

[6] Oliner, A.J., Sahoo, R.K., Moreira, J.E., Gupta, M.: "Performance Implications of Periodic Check pointing on Large-Scale Cluster Systems", In Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Washington, 2005.

[7] R. Medeiros, W. Cirne, F. Brasileiro, J. Sauve, "Faults in grids: why are they so bad and what can be done about it?" In proceedings of the 4th international workshop, November 2003, pp 18–24.

[8] Y. Li, Z. Lan, "Exploit failure prediction for adaptive fault-tolerance in cluster".In: Proceedings of the sixth IEEE International symposium on cluster computing and the grid, Vol 1, May 2006, pp.531-538.

[9] S. Hwang and C. Kesselman, "A Generic Failure Detection Service for the Grid", Technical Report ISI-TR-568, USC Information Sciences Institute, 2003.

[10] Mohammad Tanvir Huda, Heinz W. Schmidt, Ian D. Peake, "An Agent Oriented Proactive Fault-Tolerant Framework for Grid Computing", In Proceedings of the First International Conference on e-Science and Grid Computing, 2005, pp.304-311.

[11] F. Berman, H. Casanova, and A. Chien. New grid scheduling and rescheduling methods in the GrADS project. *International Journal of Parallel Programming*, 33(2–3):209–229, June 2005

[12] Jemal H. Abawajy. Fault-tolerant scheduling policy for grid computing systems. In *Proceedings of 18th International Parallel and Distributed Processing Symposium*, volume 14, page 238, April 2004.

[13] D. Saha, D. Menasce, S. Porto, et al., "Static and dynamic processor scheduling disciplines in heterogeneous parallel architectures", *Journal of Parallel and Distributed Computing* 28 (1)(1995) 1–18.

[14] S. Wang, I. Hsu, Z. Huang , "Dynamic scheduling method for computational grid Environments" , *in: Proceedings of the International Conference on Parallel and Distributed Systems*, July 2005, pp. 22–28.

[15] S. Ranaweera and D. P. Agrawal, "A Task Duplication Based Scheduling Algorithm for Heterogeneous Systems", *in Proc. of 14th International Parallel and Distributed Processing Symposium (IPDPS'00*), pp. 445-450, Cancun, Mexico, May 2000.

Author profile

Inderpreet Kaur is a M.Tech. student in the Department of computer science and engineering at Sri Guru Granth Sahib World University of Fatehgarh Sahib, Punjab, India. She received the B.Tech. degree in information technology from Banda Singh Bahadur Engineering College of Fatehgarh Sahib in 2012. Her research interests include grid computing.

Sarpreet Singh received his B.Tech degree in Computer Science from Sant Longowal Institute of Engineering and Technology(SLIET) of Punjab in 2006, M.Tech degree in Computer Science from Punjabi University Patiala of Punjab in 2009. He is a Assistant Professor of Computer Engineering at Sri Guru Granth Sahib World University, Fatehgarh Sahib now. His research interests include Grid computing, Distributed databases design concepts and knowledge management.