Global Journal of Computer Science and Technology: B
Cloud and Distributed
Volume 14 Issue 1 Version 1.0 Year 2014
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Load Balancing and Job Migration Algorithms for Autonomic Grid Environment

By Paritosh Kumar, Pankaj Kumar & Monika Singh
Thapar University, India

Abstract- Resource management and load balancing are the main areas of concern in a distributed, heterogeneous and dynamic environment like Grid. Load balancing may further cause Job migration or in some cases resubmission of Job. In this paper a number of job migration algorithms have been surveyed and studied which have resulted because of the Load balancing problem. A comparative analysis of these algorithms has also been presented which summarizes the utility and applicability of different algorithms in different environment and circumstances.

GJCST-B Classification: C.1.4

Strictly as per the compliance and regulations of:

Load Balancing and Job Migration Algorithms for Autonomic Grid Environment

Paritosh Kumar [α], Pankaj Kumar [σ] & Monika Singh [ρ]

Abstract- Resource management and load balancing are the main areas of concern in a distributed, heterogeneous and dynamic environment like Grid. Load balancing may further cause Job migration or in some cases resubmission of Job. In this paper a number of job migration algorithms have been surveyed and studied which have resulted because of the Load balancing problem. A comparative analysis of these algorithms has also been presented which summarizes the utility and applicability of different algorithms in different environment and circumstances.

I. Introduction

Grid has a number of resources working independently with different processing capability and processes different workloads accordingly. Grid computing joins all the scattered resources into a large problem solving heterogeneous environment for different types of applications, which can run in parallel. Considering the whole distributed system as one unit, workload should be evenly distributed over all the resources as per the configuration of the system, to minimize the job execution time. Therefore, Load balancing and resource management are major areas of concern for a Grid environment.

Main objective of load balancing is to optimize the response time of the application by which workload would be maintained according to resources. There are broadly three reasons which are the major causes of load balancing, resubmission of jobs and job migration; heterogeneity of resources, dynamic nature of resource's performance and diversity of applications in case of Grids [3]. This is even more crucial in computational Grid where the main concern is to fairly assign jobs to resources and to minimize the difference between the heaviest and the lightest resource load [4].

This paper presents a survey of job migration algorithms and techniques, which is done to balance the load in a Grid environment. It also compares and construes the applicability of each technique as per the requirement. The paper is organized as: Section 2 contains background of load balancing, and job migration. In Section 3, existing job migration algorithms are discussed. In the section 4, describe the proposed load balancing algorithm. Finally Section 5 concludes the paper and provides the future scope of work.

Author α: Computer Science and Engineering Department Thapar University, India. e-mail: paritosh200623@gmail.com
Author σ: Ramjas College, University of Delhi.
e-mail: pkumar240183@gmailcom
Author ρ: MITS Lakshmangarh, Rajasthan.
e-mail: dhariwal.monika@gmail.com

II. Load Balancing and Job Migration

Load balancing is main area of concern in distributed environment whereas job migration is one of the best solutions to handle load balancing problems.

a) Load Balancing

An important issue of distributed and heterogeneous environment is the efficient assignment of tasks and utilization of resources, commonly referred to as load balancing problem [13].

Load balancing is required to disperse the resource's load evenly so that maximum resource utilization and minimum task execution time could be possible. This is very crucial concern in distributed environment to fairly assign jobs to resources. Generally, load balancing mechanisms can be broadly categorized as centralized or decentralized, dynamic or static, and periodic or non periodic [5]. All load balancing methods are designed such as, to spread the load on resources equally and maximize their utilization while minimizing the total task execution time. Selecting the optimal set of jobs for transferring has a significant role on the efficiency of the load balancing method as well as Grid resource utilization. This problem has been neglected by researchers in most of previous contributions on load balancing, either in distributed systems or in the Grid environment [7].

Job migration is the only efficient way to guarantee that submitted jobs are completed reliably and efficiently in case of process failure, processer failure, node crash, network failure, system performance degradation, communication delay; addition of new machines dynamically even though a resource failure occurs which changes the distributed environment [12].

Load balancing strategies aim to adapt the load optimally to the environment. However, they mainly consider the application running on a parallel, homogeneous system.

b) Job Migration

Grid is inherently a dynamic system where environmental conditions are subjected to unpredictable changes like system or network failures, system performance degradation, addition of new machines, variations in the cost of resources etc. Job migration is

the next step when there is no proper scheduling or resubmission of jobs. Whenever any resources encounter problem, then job migration to the next eligible system is suggested. Migration behavior of jobs lead to the assumption that small sites tend to migrate resourcedemanding jobs, while large sites confine to pass only small jobs to the central job pool. Job migration is the only efficient way to guarantee that the submitted jobs are completed and that the user restrictions are met [10].

Job migration mechanisms, which take the nondedicated and dynamic natures of Grids into consideration, become important for optimizing the application performance [13]. Job monitoring, rescheduling and check pointing are some steps involved in job migration. Job monitoring contains all performance related data of all the resources so that it could initiate the migration. Further this information is reported to the rescheduler, which evaluates if it is worth Migrating the job, and in that case, decides a new allocation for the job. Check pointing is capturing a snapshot of the state of a running job, in such a way that the job can be restarted from that state in a later time in case of migration.

III. Survey of Existing Job Migration Algorithms

There are many mechanism but only five mechanism is surveyed here which is surveyed here. Which are Virtual machine migration, node reconfiguration method, check pointing, Robin-hood algorithms and load based graph method.

a) Virtual Machine Migration (Live Migration)

In Virtual machine migration snapshots of machine are sent to other machine that's why it is called the virtual machine migration. There are two methods for virtual machine migration. First one is live migration and second one is regular migration [1]. In live migration, running domain between the different host machines is migrated without stopping the job. In between it stops job and gathers all required data then resumes. But this happens only in same layer –layer network and IP subnet. In regular migration generally stop the job then migrated.

An important aspect of this mechanism is to make the run-time job migration with non-dedicated shared resources in dynamic Grid environment. Virtual machine migration provides high isolation, security and customization environment in which administrator privileges the user to execute the work. Ether IP and IP tunneling are required while migrating in this mechanism. This algorithm redistributes the load coming to any particular node, which may be the old connected node or newly added node for that load.

b) Node reconfiguration by User Level Thread Migration

This mechanism makes application workload migrate from source node to destination node, and then let source node depart from original computing environment .There are two mechanism for this, first one is node reconfiguration by user-level thread migration and another one is node reconfiguration by kernel level thread migration. Node reconfiguration by user level thread migration has been discussed in this survey.

There is two-implementation fashion of node reconfiguration. One is synchronous method and the other is asynchronous method. In synchronous method, all nodes are paused during reconfiguration. On the other hand, in asynchronous method all nodes continue to work simultaneously with reconfiguration. Synchronous method may make performance down even though it is easier to design. Alternatively, better performance can be obtained by asynchronous method as long as more attention paid to correctly maintain the order of node reconfiguration messages [1].

Information regarding redistribution of workload and how to add/delete nodes is present in the implementation of node reconfiguration mechanism. With the help of user level thread migration, which is already supported by the thread package workload, is redistributed here. Same as virtual machine migration, node reconfiguration mechanism also needs to transfer in memory states from source node to destination node.

c) Check-Pointing Approach

Checkpoint is defined as a designated place in a program at which normal processing is interrupted specifically to preserve the status information necessary to allow resumption of processing at a later time. By periodically invoking the check pointing process, one can save the status of a program at regular intervals. If there is a failure one may restart computation from the last check point thereby avoiding repeating the computation from the beginning. The process of resuming computation by rolling back to a saved state is called rollback recovery [2].

There are three types of check pointing implementations: kernel-level, user-level and application level. These implementations differ in level of transparency, efficiency and mechanism used to initiate checkpoint and restart. In kernel level check pointing user does not have to change the application at all so least efficient, because system does not have the knowledge about the application. Developer achieves user level check pointing, and he puts or implements some set of procedures that handle check pointing and restart. Developer knows all about the application that's why this approach is more efficient. The developer itself achieves application-level check pointing. This appr-oach is the most efficient, because developer has detailed knowledge about application.

This is very useful in case of preemption and migration and is used in making fault tolerant systems. Most common benefit of the check pointing technology is the high level of fault tolerance offered by the applications that can be check pointed. Besides it used to recover from failures, check pointing is also used in playback debugging distributed programs, migrating processes in a multiprocessor system, software rejuvenation and optimistic simulation.

Check pointing balances the load of processors in a distributed system; processes are moved from heavily loaded processors to lightly loaded ones. Check pointing process periodically provides the information necessary to move it from one processor to another.

d) Robin Hood: An Active Objects Load Balancing Mechanism for Intranet

Robin Hood algorithms present a new totally non centralized solution, multicast channel to communicate, and synchronize the processors and proactive tools to migrate jobs between them. Proactive techniques are very useful and provide the mobility and security in uniform framework. This work focuses on dynamic load balancing. Main objective of this algorithm is to improve the decision time in non-centralized environment.

In this mechanism two basic things have been considered, first one to know about the local load and second one transfer the load from high dense node to the less loaded node. This uses the non-centralized architecture and non-broadcasting of the balance of each node to reduce the overload in network. This is totally non-centralized load balancing mechanism, using the proactive library for the migration of jobs, and a multicast channel for node coordination.

e) Load Graph Based Transfer Method

Load based graph method is based on network graph where each node is represented with its load, whereas load can be the number of users, average queue length or the memory utilization. It uses analytic model and single load determination policy throughout the system and load is determined on the basis of memory utilization and average queue length. This algorithm is based on three layered structure. Top layer is load balancing layer which takes care of token generation, taking decision about task transfer; middle one is called monitoring layer and acts as an interface between top and middle and monitors load changes and third one called communication layer which take care of actual task transfer.

Here token is generated on the basis of outgoing and incoming edges and initialized on the basis of some specific value HWM & LWM (Highest Water Mark, Lowest Water Mark). Specific values are decided on the basis of load value of neighbors. Nodes having load value greater than HWM and are local maxima or nodes having load value less than LWM and are the local minima, can initiate token [9].

Maximum message transfer per node, if N is number of nodes and X is maximum message transfer per node

Total message transfer =NX

And transfer of task will occur only if there would be proper load difference between the nodes as

1. La - Lb > M where M is the required

Load difference for the task transfer.

Token will be generated if following conditions will be satisfied

1. For nth node (Load) n > L where L is maximum Load where load balancing is not required.
2. (Load) n > Σ sum of load of all nodes

If both conditions are satisfied then the token is generated in more than sixty percent of the cases where load imbalance exists token finds out the proper node for the task transfer which improves the system performance[9]. In this algorithm along with the task transfer among the neighboring nodes with the token transfer method care is taken to avoid the starvation of those nodes for which neighbors are not suitable for the task transfer.

The major parameter, network-partitioning issues along with inter-cluster and intra-cluster transfers for decision making of load balancing for the transfer is considered here.

IV. Proposed Load Balancing Algorithm

Proposed load balancing algorithm is developed considering main characteristics like performance, throughput, and resource utilization.

a) Architecture of Load Balancing System

This section discusses about the architecture of load balancing algorithm-imposed system. Figure 4.2 presents a pictorial view of the system. Monitor server uses monitoring tool to gather information about all the connected nodes. This resource information is managed, processed and updated to a database. This information is accessed through web pages and is presented to the users. The web pages can be accessed from any nodes at the same network.

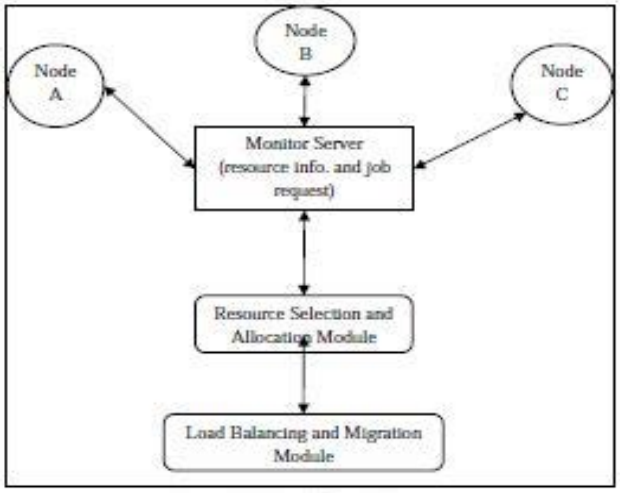

Figure 4.1 : System Overall Architecture

Server node gets all information from the nodes via monitoring tool and updates the database. Web pages are created and hosted on local network using Apache HTTP server to provide the resources information to users, any system on local network running Apache HTTP server can access these web pages. This is shown in figure:

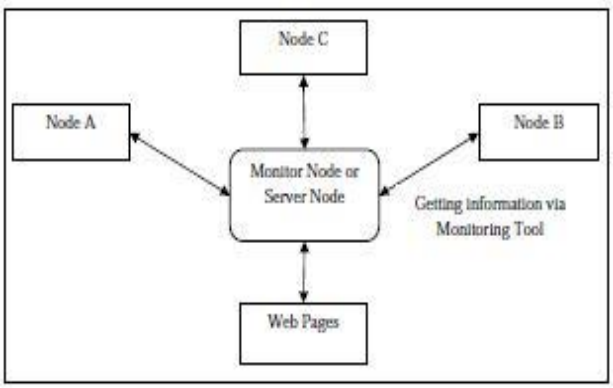

Figure 4.2

b) Design of Proposed Load Balancing Algorithm

Pseudo code and flow chart of the proposed load balancing algorithm. Proposed algorithm for load balancing is given below:

Begin
(Initiating activity happens)
(Load balancing start)
Monitoring info. And job request (n resources & n jobs)
Create job queue
If (job req. not matched with resource.)
Job goes to main queue
Else
Assigned job to resource
If (Machine failure)
Job goes to main queue with check-pointed data
Else
Job completed
End

c) Complexity of Proposed Load Balancing Algorithm

Complexity is a measure of the performance of an algorithm in term of CPU time and memory usage. In this case computational complexity has been considered as this algorithm is for grid environment.

Computational Complexity: To measure computational complexity computation number should be known.

By Big-O notation

Above proposed algorithm have equation like this

$N3+N2+N+C$, where C is constant

According to Big-O notation

$f(n) = O(N3) + O(N3) + O(N) + O(C)$,C is constant

$f(n) = O(N3)$

Another formula to find out complexity of algorithm

Complexity = No. Of closed loop = 3;

Another formula to find out Complexity of algorithm

Complexity = No. of decision making statements + 1;

Complexity = 2 +1 =3;

V. Conclusion

Load balancing is a key issue in grid resource management and results in job migration or re-submission of job. In this paper load balancing and job migration algorithms have been surveyed and studied which have been designed for different scenarios. Based on all these algorithms new algorithm have been introduced, considering main characteristics like performance, throughput, and resource utilization.

Different algorithms do well in their respective contexts like multiple token policy results in optimal resource selection and minimum migration time where as Robin-hood provides better security and check-pointing provides good results for fault tolerant systems.

References Références Referencias

1. Po-Cheng Chen, Cheng-I Lin, Sheng-Wei Huang, Jyh-Biau Chang, Ce-Kuen Shieh, Tyng-Yeu Liang" A Performance Study of Virtual Machine Migration vs Thread Migration for Grid Systems" 22nd International Conference on Advanced Information Networking and Applications – Workshops.
2. S Kalaiselvi, V Rajaraman, "A Survey of Check-Pointing Algorithms for Parallel and Distributed Computers" v ol. 25, Part 5, October 2000, pp.489±510.
3. Belabbas Yagoubi, Hadj Tayab Lillia and Halima Si Mo ussa, "Load Balancing in Grid Computing".
4. Mohsen Amini Salehi, Hossein Deldari "Balancing Load in a Computational Grid Applying Adaptive, Intelligent Colonies of Ants".
5. Y. Lan, T. Yu (1995) "A Dynamic Central Scheduler load-Balancing Mechanism", Proc. 14th IEEE Conf.

on Computers and Communication, Tokyo, Japan, pp. 734-740.
6. Mohsen Amini Salehi Hamid Tabatabaee Yazdi, Mohammad Reza Akbarzade Toutoonchi," An Optimal Job Selection method in Load Balancing Algorithms of Economical Grids".
7. S. Rips "Load Balancing Support for Grid-enabled Applications" NIC Series, Vol. 33, ISBN 3-00-017352-8, pp. 97-104, 2006.
8. Gracjan Jankowski, Radoslaw Januszewski, Rafal Mikolajczak Poznan, " Improving the fault-tolerance level within the GRID computing environment - integration with the low-level check pointing packages" Core Grid TR-0158 June 16, 2008.
9. Parag Kulkarni & Indranil Sengupta," Load Balancing with Multiple Token policy".
10. Rafael A.Moreno, Ana B. Alonso-conde" Job Scheduling and Resource Management Techniques in Dynamic Grid Environments" Volume 2970 of LNCS (2004).
11. Luis Ferreira, Viktors Berstis, Jonathan Armstrong, Mike Kendzierski, Andreas Neukoetter, asanobu Takagi, Richard Bing-Wo, Adeeb Amir, Ryo Murakawa, Olegario. Hernandez, James Magowan, Norbert Bieberstein "Red Books" ibm.com/Redbooks.
12. Belabbas Yagoubi and Yahya Slimani" Dynamic Load Balancing Strategy for Grid Computing" Proceedings of World Academy of Science, Engineering and Technology Volume 13 May 2006 ISSN 1307-6884.
13. Neeraj Nehra, R.B. Patel, and V.K. Bhat" Load Balancing with fault tolerance and Optimal Resource Utilization in grid Computing" Information Technology Journal vol. 6:784-797, 2007.

This page is intentionally left blank