ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

Resource Discovery and Load Balancing in Grid Environment

Bhairavnath Jadhav[1], Riyaz Shaikh[2], Irshad Sagar[3]

Department of Computer Engineering, SP Pune University, India[1,2,3]

Abstract: Grid technology has emerged as a new way of large scale distributed computing with high-performance orientation. Grid computing is being adopted in various areas from academic, industry research to government use. Grids are becoming platforms for high performance and distributed computing. Grid computing is the next generation IT infrastructure that promises to transform the way organizations and individuals compute, communicate and collaborate. The goal of Grid computing is to create the illusion of a simple but large and powerful self-managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources. The main goal of load balancing is to provide a distributed, low cost, scheme that balances the load across all the processors. To improve the global throughput of Grid resources, effective and efficient load balancing algorithms are fundamentally important. Focus of this project is on analyzing Load balancing requirements in a Grid environment and proposing a centralized and sender initiated load balancing algorithm. In this work we have proposed an efficient load balancing algorithm which optimizes the response time and latency time with respect to the server.

Keywords: Load balancing, grid computing.

I INTRODUCTION

The rapid development in computing resources has enhanced the performance of computers and reduced their costs. This availability of low cost powerful computers coupled with the popularity of the Internet and high-speed networks has led the computing environment to be mapped from distributed to Grid environments. In fact, recent researches on computing architectures are allowed the emergence of a new computing paradigm known as Grid computing. Grid is a type of distributed system which Supports the sharing and coordinated use of geographically distributed and multiword resources, independently from their physical type and location, in dynamic virtual organizations that share the same goal of solving large-scale applications. In order to fulfill the user expectations in terms of performance and efficiency, the Grid system needs efficient load balancing algorithms for the distribution of tasks. A load balancing algorithm attempts to improve the response time of user's submitted applications by ensuring maximal utilization of available resources. The main goal is to prevent, if possible, the condition where some processors are overloaded with a set of tasks while others are lightly loaded or even idle. Although load balancing problem in conventional distributed systems has been intensively studied, new challenges in Grid computing still make it an interesting topic and many research projects are under way. This is due to the characteristics of Grid computing and the complex nature of the problem itself. Load balancing algorithms in classical distributed systems, which usually run on homogeneous and dedicated resources, cannot work well in the Grid architectures. Grid Resource Management is defined as the process of identifying requirements, matching resources to applications, allocating those resources, and scheduling and monitoring Grid resources over time in order to run Grid applications as efficiently as possible. Resource discovery is the first phase of resource management. Scheduling and monitoring is the next step. Scheduling process directs the job to appropriate resource and monitoring process monitors the resources. The resources which will be heavily loaded will act as server of task and the resources which are Lightly Loaded will act as receiver of task. Task will be migrated from heavily loaded node to lightly loaded node. Resources are dynamic in nature so the load of resources varies with change in configuration of Grid so the Load Balancing of the tasks in a Grid environment can significantly influence Grid's performance.

II. LOAD BALANCING CATEGORIES

Load balancing problem has been discussed in traditional distributed systems literature for more than two decades and various algorithms, strategies and policies have been proposed, classified and implemented. Load balancing algorithms can be classified into two categories, static and dynamic.

A. Static load balancing Algorithms

Static load balancing algorithms allocate tasks of a parallel program to workstations based on either the load at the time nodes are allocated to some task, or based on average load of workstation cluster.

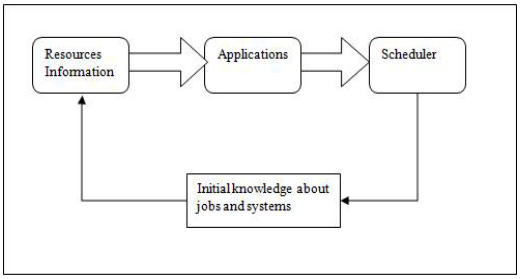

Fig. 1 Static Load Balancing

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

The decisions related to load balance are made at compile time when resource requirements are estimated. The advantage in this sort of algorithm is the simplicity in terms of both implementation as well as overhead, since there is no need to constantly monitor the workstations for performance statistics. The decisions related to load balance are made at compile time when resource requirements are estimated. The advantage in this sort of algorithm is the simplicity in terms of both implementation as well as overhead, since there is no need to constantly monitor the workstations for performance statistics. However, static algorithms only work well, when there is not much variation in the load on the workstations. Clearly, static load balancing algorithms aren't well suited to a grid environment, where loads may vary significantly at various times.

A few static load balancing techniques are:

- Round-Robin Algorithm: tasks are passed to processes in a sequential order, when the last process has received a task the schedule continues with the first process (a new round).
- Randomized Algorithm: allocation of tasks to processes is random.
- Simulated Annealing or Genetic Algorithms: mixture allocation procedure including optimization techniques.

Drawbacks of Static Load Balancing Algorithms

- It is very difficult to estimate a-priori (in an accurate way) the execution time of various parts of a program.
- Sometimes there are communication delays that vary in an uncontrollable way.
- For some problems the number of steps to reach a solution is not known in advance.

B. Dynamic load balancing Algorithms

According to the name dynamic load balancing algorithms takes decision at run time, and use current or recent load information when making distribution decisions. In grid environment with dynamic load balancing allocate/reallocate resources at runtime based on no a priori task information, which determine when and which task has to be migrated.

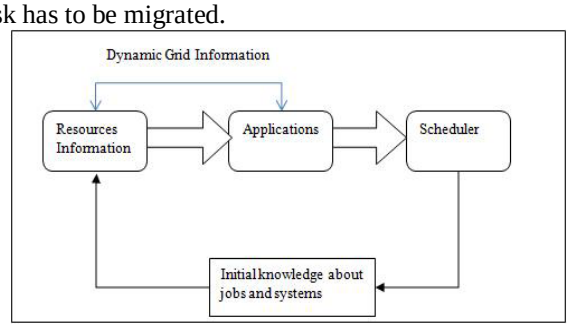

Fig. 2 Dynamic Load Balancing

After using effectively dynamic load balancing algorithms can provide a significant improvement in performance over static algorithms. But this comes at the additional cost of collecting and maintaining load information, so it is important to keep these overheads within reasonable limits.

III. LOAD BALANCING STRATEGIES

There are three major parameters which usually define the strategy of a specific load balancing algorithm. Some load balancing strategies are being discussed in the following section.

A. Sender-Initiated v/s. Receiver-Initiated Strategies

In sender-initiated policies, congested nodes attempt to move work to lightly-loaded nodes. In receiver-initiated policies, lightly-loaded nodes look for heavily-loaded nodes from which work may be received. The sender-initiated policy performing better than the receiver-initiated policy at low to moderate system loads. Reasons are that at these loads, the probability of finding a lightly-loaded node is higher than that of finding a heavily-loaded node. Similarly, at high system loads, the receiver initiated policy performs better since it is much easier to find a heavily-loaded node. As a result, adaptive policies have been proposed which behave like sender-initiated policies at low to moderate system loads, while at high system loads they behave like receiver-initiated policies.

B. Global v/s. Local Strategies

Global or local policies answer the question of what information will be used to make a load balancing decision in global policies. The load balancer uses the performance profiles of all available workstations. In local policies, workstations are partitioned into different groups.

The benefit in a local scheme is that performance profile information is only exchanged within the group. The choice of a global or local policy depends upon the behavior of an application, which will exhibit. For global schemes, balanced load convergence is faster compared to a local scheme since all workstations are considered at the same time.

C. Centralized v/s. De-centralized Strategies

A load balancing strategy is categorized as either centralized or distributed, both these define where load balancing decisions are made. In a centralized scheme, algorithm is located on one master workstation node and all decisions are made there. In a de-centralized scheme, the load balancer is replicated on all workstations. There are different algorithms used in de-centralized scheme for job selection. These algorithms are round-robin algorithm, random polling algorithm etc.

IV. LOAD BALANCING POLICIES

Load balancing algorithms can be based on many policies; some important policies are defined below.

- Information policy: This policy specifies what workload information should be collected, when it is to be collected and from where.
- Triggering policy: This policy determines the appropriate period to start a load balancing operation.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

- Resource type policy: This policy classifies a resource as server or receiver of tasks according to its availability status.
- Location policy: This policy uses the results of the resource type policy to find a suitable partner for a server or receiver.
- Selection policy: This policy defines the tasks that should be migrated from overloaded resources (source) to most idle resources (receiver).

The main objective of load balancing methods is to speed up the execution of applications on resources whose workload varies at run time in unpredictable way. Hence it is significant to define metrics to measure the resource workload. Every dynamic load balancing method must estimate the timely workload information of each resource. Success of a load balancing algorithm depends upon stability of the number of messages (small overhead), support environment, low cost update of the workload, and short mean response time which is a significant measurement for a user. It is also essential to measure the communication cost induced by a load balancing operation, but to achieve all these, anyone would have to face great challenge in grid environment V. Load Balancing Mechanism.

There are some load balancing algorithms like virtual machine migration, node reconfiguration by user level thread migration, robin-hood an active objects migration mechanism for intranet, load based graph method and data consolidation.

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. 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-2 network and IP subnet. In regular migration generally stop the job then migrated.

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 are two implementation methods 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.

C. 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 to 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.

D. 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.

V. PROPOSED LOAD BALANCING ALGORITHM

Load balancing is defined as the allocation of the work of a single application to processors at run-time so that the execution time of the application is minimized. Load balancing is defined as the allocation of the work of a single application to processors at run-time so that the execution time of the application is minimized. This chapter is going to discuss the design of proposed Load Balancing algorithm. The choice of a load balancing algorithm for a Grid environment is not always an easy task. Various algorithms have been proposed in the literature, and each of them varies based on some specific application domain. Some load balancing strategies work well for applications with large parallel jobs, while others work well for short, quick jobs. Some strategies are focused towards handling data heavy tasks, while others are more suited to parallel tasks that are computation heavy. While many different load balancing algorithms have been proposed, there are basic steps that nearly all algorithms have in common:

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

- Monitoring workstation performance (load monitoring)
- Exchanging this information between workstations (synchronization)

Efficient Load Balancing algorithm makes Grid Middleware efficient and which will ultimately leads to fast execution of application in Grid environment. In this work, an attempt has been made to formulate a decentralized, sender-initiated load balancing algorithm for Grid environments which is based on different parameters. One of the important characteristics of this algorithm is to estimate system parameters such as CPU utilization of each participating nodes.

VI. DESIGN OF LOAD BALANCING ALGORITHM

Load balancing should take place when the load situation has changed. There are some particular activities which change the load configuration in Grid environment. The activities can be categorized as following:

- Selection of static or dynamic load balancing category.
- Defining the various parameters.
- Connection with the server.
- Sending threads to the server and executing results.

For static load balancing first of all Collect Host information from user (i.e. ip address, port, request URL etc), when we execute this will try to connect to the host. If it get connected to host it will perform Simulating the number requests to the Host using no. of threads, then it will bring Result From the Server and populate into the view area. For dynamic load balancing, Schedule the host information for Dynamic Execution, Check the Date and Time for every second comparing the scheduled date and time, Try to connect the Host, Simulating the number requests to the Host using no. of threads, Bring Result From the Server and populate into the view area.

6.1 Efficient Load Balancing Algorithm

1. Start
2. For static load balancing, Collect Host information from user (i.e. ip address, port, request URL etc.) {Host name/IP, Port, Protocol type, Requested URL, No of request, Execution time, Request to}
3. When we execute this will try to connect to the host. If does not get connected again go to step 2.
4. If it get connected to host it will perform Simulating the number requests to the Host using no. of threads, then it will bring Result From the Server and populate into the view area.
5. For dynamic load balancing, Schedule the host information for Dynamic Execution.
6. Check the Date and Time for every second comparing the scheduled date and time.
7. Try to connect the Host, Simulating the number requests to the Host using no. of threads. If not connected back to step 5.
8. Bring Result from the Server and populate into the view area.
9. End

The following flow diagram shows the overall system architecture and flow of the system for static load balancing and dynamic load balancing.

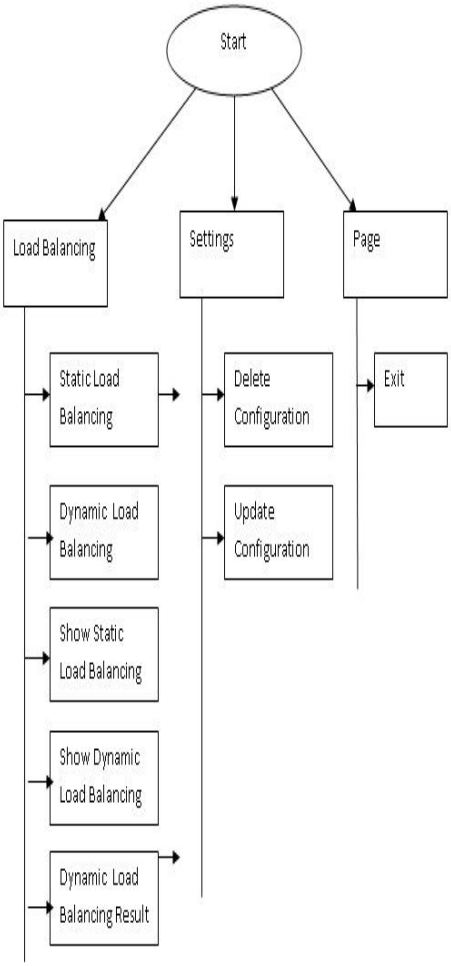

Fig. 3: Flow Chart of Overview of Algorithm

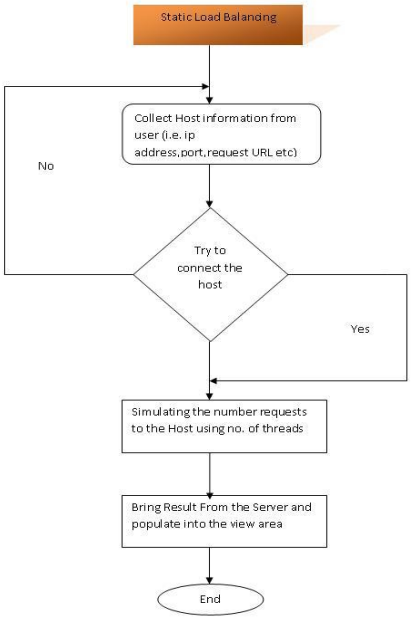

Fig. 4: Flow Chart of Static load balancing Algorithm

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

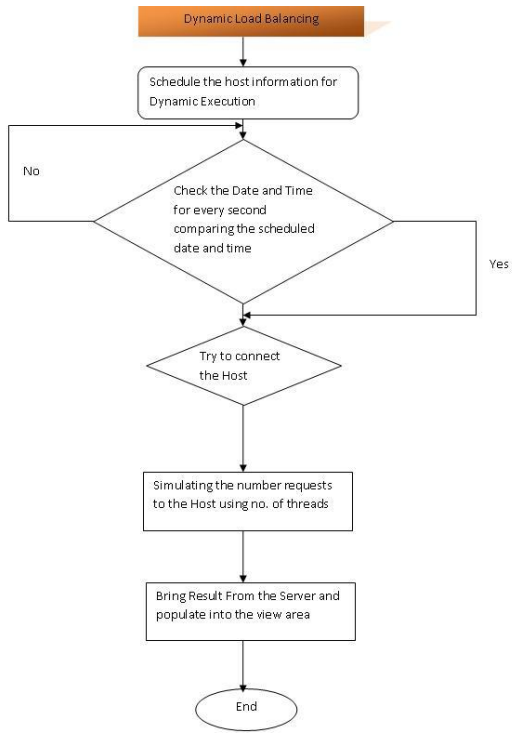

Fig. 5: Flow Chart of Dynamic load balancing Algorithm

Latency is a measure of time delay experienced in a system, the precise definition of which depends on the system and the time being measured. Latencies may have different meaning in different contexts. In simulation applications, 'latency' refers to the time delay, normally measured in milliseconds (1/1,000 sec), between initial input and an output clearly discernible to the simulator trainee or simulator subject. Latency is sometimes also called transport delay. Some authorities distinguish between latency and transport delay by using the term 'latency' in the sense of the extra time delay of a system over and above the reaction time of the vehicle being simulated, but this requires a detailed knowledge of the vehicle dynamics and can be controversial.

Response time may refer to: Response time (technology), the time a generic system or functional unit takes to react to a given input Responsiveness, how quickly an interactive system responds to user input.

Request methods- HTTP defines nine methods (sometimes referred to as "verbs") indicating the desired action to be performed on the identified resource. What this resource represents, whether pre-existing data or data that is generated dynamically, depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server.

HEAD- Asks for the response identical to the one that would correspond to a GET request, but without the response body. This is useful for retrieving meta information written in response headers, without having to transport the entire content.

GET- Requests a representation of the specified resource. Requests using GET (and a few other HTTP methods) "SHOULD NOT have the significance of taking an action other than retrieval". The W3C has published guidance principles on this distinction, saying, "Web application design should be informed by the above principles, but also by the relevant limitations."

POST- Submits data to be processed (e.g., from an HTML form) to the identified resource. The data is included in the body of the request. This may result in the creation of a new resource or the updates of existing resources or both.
Status Code Definitions- Each Status-Code is described below, including a description of which method(s) it can follow and any misinformation required in the response.

200 OK- The request has succeeded. The information returned with the response is dependent on the method used in the request, for example:

GET an entity corresponding to the requested resource is sent in the response; HEAD the entity-header fields corresponding to the requested resource are sent in the response without any message-body; POST an entity describing or containing the result of the action; TRACE an entity containing the request message as received by the end server.

404 Not Found - The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent. The 410 (Gone) status code SHOULD be used if the server knows, through some internally configurable mechanism, that an old resource is permanently unavailable and has no forwarding address. This status code is commonly used when the server does not wish to reveal exactly why the request has been refused, or when no other response is applicable.

VII. CONCLUSION AND FUTURE WORK

In this paper, we have described multiple aspects of Grid Computing and introduced numerous concepts which illustrate its broad capabilities. Grid Computing is definitely a promising tendency to solve high demanding applications and all kinds of problems. Objective of the grid environment is to achieve high performance computing by optimal usage of geographically distributed and heterogeneous resources. But grid application performance remains a challenge in dynamic grid environment. Resources can be submitted to Grid and can be withdrawn from Grid at any moment. This characteristic of Grid makes Load Balancing one of the critical features of Grid infrastructure. There are a number of factors, which can affect the grid application performance like load balancing, heterogeneity of resources and resource sharing in the Grid environment. In this project we have focused on Load Balancing and tried to present the impacts of Load Balancing on grid application performance and finally proposed an efficient Load Balancing algorithm for Grid environment. Every

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

Load Balancing algorithm implements five policies. The efficient implementation of these policies decides overall performance of Load Balancing algorithm. In this work we analyzed existing Load Balancing algorithm, proposed and implemented an enhanced algorithm which more efficiently implements three out of five policies implemented in existing Load Balancing algorithm. These three policies are: Information Policy, Triggering Policy and Selection Policy.

The future scope of this work includes implementing the application using tools such as Globus or gridsim for showing the simulation, Migration of jobs, Design and Testing of load balancing algorithm in a Multi-middleware scenario. A further extension to this work would be in making this Load balancing Module a middleware independent module.

REFERENCES

[1]. Yajun Li, Yuhang Yang Rongbo Zhu "A Hybrid Load balancing Strategy of Sequential Tasks for Computational Grids" 2009 International Conference on Networking and Digital Society
[2]. B. Yagoubi and Y. Slimani "Task Load Balancing Strategy for Grid Computing" Journal of Computer Science 3 (3): 186-194, 2007 ISSN 1546-9239, 2007 Science Publications
[3]. Wang Qiong, He Xinhua, Zhao Yingkun "Load Balancing Algorithm Based on Dual-Load Prediction" 2009 Second International Conference on Intelligent Networks and Intelligent Systems
[4]. Belabbas Yagoubi and Meriem Medebber "A Load Balancing Model for Grid Environment" 1-4244-1364-8/07/$25.00 ©2007 IEEE
[5]. Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma "Performance Analysis of Load Balancing Algorithms" World Academy of Science, Engineering and Technology 38 2008
[6]. A.K. Aggarwal, Robert Kent and Jun Wei "PATH Algorithm for Adaptive Load Balancing on a Grid"
[7]. Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma "Performance Analysis of Load Balancing Algorithms" World Academy of Science, Engineering and Technology 38 2008
[8]. Robert Elsässer Burkhard Monien Stefan Schamberger "Load Balancing in Dynamic Networks" Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'04) 2004 IEEE

BIOGRAPHIES

Bhairavnath Jadhav is pursuing Bachelors of Engineering in SKN-SITS, Lonavla, Pune.

Riyaz Shaikh is pursuing Bachelors of Engineering in SKN-SITS, Lonavla, Pune.

Irshad Sagar is pursuing Bachelors of Engineering in SKN-SITS, Lonavla, Pune.