Research Article | Impact Factor: 0.621 | ISSN: 2319-507X
Tanvi Mayekar, IJPRET, 2014; Volume 2 (9): 843-849 | IJPRET

INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

EFFECTIVE LOAD BALANCING SCHEME IN GRID

TANVI MAYEKAR[1], DR. AMEETA G. SINAI AMONKAR[2]

1. Department of Electronic and Telecommunication Engineering.
2. Goa College of Engineering – Goa University Farmagudi, Ponda, Goa.

Accepted Date: 27/02/2014 ; Published Date: 01/05/2014

Abstract: In grid computing workload management is a key grid service of the grid infrastructure, where issues of load balancing represent a common concern for most grid infrastructure developers. Load balancing is a great challenge in dynamic and heterogeneous grid environment. Load balancing is a technique to enhance resources, improve throughput, and to cut response time through appropriate distribution of the applications. An effective and efficient load balancing algorithm is required to balance the load in grid environment. In this paper, a new load balancing scheme is proposed which will enhance the performance of the system and reduce the communication overhead.

Keywords: Grid Computing, Load Balancing, Communication Overhead

PAPER-QR CODE

Corresponding Author: MS. TANVI MAYEKAR

Access Online On:

www.ijpret.com

How to Cite This Article:

Tanvi Mayekar, IJPRET, 2014; Volume 2 (9): 843-849

INTRODUCTION

A grid is a type of distributed system that enables coordinated sharing and aggregation of distributed, autonomous, heterogeneous resources based on users' QoS (Quality of Service) requirements. Grids are commonly used to support applications emerging in the areas of e-Science and e-Business, which commonly involve geographically distributed communities of people who engage in collaborative activities to solve large scale problems and require sharing of various resources such as computers, data, applications and scientific instruments.

Load balancing is a computer networking method for distributing workloads across multiple computing resources, such as computers, a computer cluster, network links, and central processing units. Load balancing aims to optimize resource use, maximize throughput, minimize response time, and avoid overload of any one of the resources. Using multiple components with load balancing instead of a single component may increase reliability through redundancy. Load balancing is usually provided by dedicated software or hardware or both.

Load Balancing Algorithms can be classified in two different ways [3]:

1. Static load balancing algorithms allocate the tasks of a parallel program to workstations based on either the load at the time nodes are allocated to some task, or based on an average load of our workstation cluster. 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 - the 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: the allocation of tasks to processes is random.

2. Dynamic load balancing algorithms make changes to the distribution of work among workstations at run-time; they use current or recent load information when making distribution decisions. Multicomputers with dynamic load balancing allocate/reallocate resources at runtime based on no a priori task information, which may determine when and whose tasks can be migrated.

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

load balancing Strategies

As in [3], the various load balancing strategies are:

Sender-Initiated vs. 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.

Global vs. Local Strategies

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 on the behavior an application will exhibit. For global schemes, balanced load convergence is faster compared to a local scheme since all workstations are considered at the same time. However, this requires additional communication and synchronization between the various workstations; the local schemes minimize this extra overhead. But the reduced synchronization between workstations is also a downfall of the local schemes if the various groups exhibit major differences in performance. If one group has processors with poor performance (high load), and another group has very fast processors (little or no load), the latter will finish quite early while the former group is overloaded.

Centralized vs. De-centralized Strategies

A load balancer is categorized as either centralized or distributed, both of which define where load balancing decisions are made. In a centralized scheme, the load balancer is located on one master workstation node and all decisions are made there.

Basic features of centralized approach are:

• a master node holds the collection of tasks to be performed,

• tasks are sent to the execution node

• when a execution process completes one task, it requests another task from the master node

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.

For centralized schemes, the reliance on one central point of balancing control could limit future scalability. The distributed scheme helps solve the scalability problems, but at the expense of an "all-to-all" broadcast of profile information between workstations.

III. Load Balancing Policies

Load balancing algorithms can be defined by their implementation of the following policies [4]:

• Information policy: specifies what workload information to be collected, when it is to be collected and from where.

• Triggering policy: determines the appropriate period to start a load balancing operation.

• Resource type policy: classifies a resource as server or receiver of tasks according to its availability status.

• Location policy: uses the results of the resource type policy to find a suitable partner for a server or receiver.

• Selection 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.

Several load indices have been proposed, like CPU queue length, average CPU queue length, CPU utilization, etc. The success of a load balancing algorithm depends from 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.

New proposed load balancing algorithm

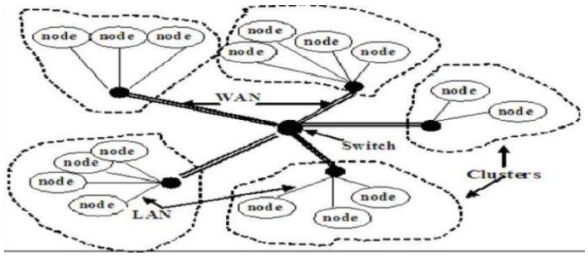

Figure 1: grid topology[5]

The new distributed load balancing strategy has the following objectives:

(i) reduction in mean response time of tasks

(ii) reduction of the communication costs/overhead

(iii) meet the tasks deadline by ensuring maximal utilization of available resources.

The proposed load balancing algorithm will partition the task into subtasks to distribute load evenly across the nodes. Further these subtasks will be grouped using job grouping strategy. This is done in order to balance load more efficiently so as to reduce its processing time. And grouping the subtasks might reduce the communication overhead time and processing overhead time of each user job.

The tasks are partitioned into subtasks and are placed in a global task set. The scheduler receives the job list created and sorts it according to their MI in decreasing order. It then receives the resource list and sorts it in decreasing order according to their MIPS. It counts the number of jobs and resources and store them in Jsize and Rsize respectively. The group size is determined by the formula:

Gsize = Jsize / Rsize

Jobs are added into group based on group size by alternatively taking jobs from front end i.e. job with highest length and then rear end of the job list i.e. job with smallest length in job list

G (i)size<=G_size

When above condition fails, it stops grouping and job group send to the corresponding resource for computation. Job grouping process is repeated as long as jobs exist.

CONCLUSION

The main 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. This paper addresses a new dynamic load balancing algorithm for computational grid environment to produce load balancing in grid computing systems, to decrease the wait time of tasks and to minimize the delay of task execution, resulting in increased performance and reduced communication overhead.

REFERENCES

1. Suri P.K., Singh, M. An Efficient Decentralized Load Balancing Algorithm for Grid, IEEE 2nd International, Advance Computing Conference, 2010.

2. Sandeep Kaur, Sukhpreet Kaur, Load Balancing Dynamic Job Grouping Based Scheduling Algorithm in Grid Computing, International Journal of Scientific & Engineering Research, July-2013.

3. M. Kamarunisha, S. Ranichandra, T. K. P. Rajagopal, Recitation of Load Balancing Algorithms In Grid Computing Environment Using Policies And Strategies - An Approach, International Journal of Scientific & Engineering Research Volume 2, Issue 3, March-2011.

4. Sandip S. Patil, Preeti Singh, Design and Implementation of Efficient Load Balancing Algorithm in Grid Environment, International Journal of Computer Science and Information Technologies, Vol. 2 (5) , 2011.

5. Belabbas Yagoubi, Meriem Meddeber, Distributed Load Balancing Model for Grid Computing, ARIMA Journal, 2010.