IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 1, March, 2013
ISSN: 2320 - 8791
www.ijreat.org

Improving Grid Resource Allocation to monitor the task Scheduling and Binding

P.Raga Priya[1], S.Prathiba[2]

[1,2] Department of Computer Applications, B.S.Abdur Rahman University
Chennai, Tamil Nadu, India.

Abstract

This paper aims to improve Resource provisioning is one of the challenges in the Grid environments. Selecting a computational resource in a grid environment depends on computational power, completion time; cost of utilizing the resources is a complex task. As the number of resources in Grids increases rapidly, selecting an appropriate resource for jobs has become a crucial issue. During the selection of the resources grid should not be overloaded. The problem arises when there are insufficient resources for users to be served. To overcome this improved Resource allocation Algorithm for scheduling jobs has been proposed. The resource estimator estimates and chooses only the feasible resources depending upon the resource cost and memory cost.

Keywords: *Grid, Resource allocation.*

1. Introduction

The proliferation of the Internet and the availability of powerful computers and high-speed networks as low-cost commodity components are changing the way we do large-scale parallel and distributed computing. The interest in coupling geographically distributed (computational) resources is also growing for solving large-scale problems, leading to what is popularly called the Grid and peer-to-peer (P2P) computing performance computing and high throughput computing. The former aims at minimizing the execution time of each network. These enable sharing, selection and aggregation of, suitable computational and data resources for solving large-scale data intensive problems in science, engineering, and commerce. Task scheduling is an integrated part of parallel and distributed computing. Intensive research has been done in this area and many results have been widely accepted. However, with the emergence of the computational grid, new scheduling algorithms are in demand for addressing new concerns arising in the grid environment.

2. Related Work

Resource selection is one of the important and key concepts in grid. In [1] the author has discussed about the anatomy of the grid enabling scalable virtual organizations. The real and specific problem that underlies the Grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations. The sharing refers to direct access to computers, software, data, and other resources, as is required by a range of collaborative problem-solving and resource brokering strategies emerging in industry, science, and engineering. This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. Its disadvantages are

- It needs multiple administrative domains.
- Grid software should define the operating system services to be installed on every participating system.

In [2] the author has discussed about the scheduling multiprocessor tasks with genetic algorithms Task scheduling which is the process of deciding which instructions will be run by which processor, and in which order. We assume that the multiprocessor system is uniform (homogenous) and non preemptive etc. The processors are identical, and a processor completes the current task before executing a new one. Task execution time can be non uniform. Every processor can communicate with each other and every processor has own memory. The drawbacks of this system are

- A processor can execute at most one task at a time.
- The processor availability is limited.

With the proliferation of grid, at least two new things need to be considered in a scheduling model. The first is the quality of service. In a grid environment, applications compete for the best Quality-of-Service (QoS) from the remote resources. The resources provide non dedicated services to the applications. The scheduler in the grid environment needs to consider the QoS to get a better match between applications and resources. The other issue is how to handle the non-dedicated network. For non-dedicated networks, since they have their own local jobs, they cannot provide exclusive service to remote jobs. So how to predict the job computation time for non-dedicated network needs to be addressed. In current grid task scheduling, tasks with or without QoS request compete for resources. While a task with no QoS request can be executed on both high QoS and low QoS resources, a task that requests a high QoS service can only be executed on a resource providing high quality of service. Thus, it is possible for low QoS tasks to occupy high QoS resources while high QoS tasks wait as low QoS resources remain idle.

There are three main phases of scheduling on a grid. Phase one is resource discovery, which generates a list of potential resources. Phase two involves gathering information about those resources and choosing the best set to match the application requirements. In phase three the job is executed, which includes file staging and cleanup. There are two different goals for task scheduling: high performance computing and high throughput computing. The former aims at minimizing the execution time of each application and generally used for parallel processing, whereas the latter aims at scheduling a set of independent tasks to increase the processing capacity of the systems over a long period of time. Our approach is to develop a high throughput computing scheduling algorithm.

3. Proposed Model

The proposed system, addresses the QoS requirements along with preemption-aware scheduling, it addresses the valuable external requests along with other requests, it minimizes the external request response time. It also minimizes the number of Virtual Machine preemptions both for more user satisfaction and also system utilization purposes. All request type gets benefited by the proposed model.

The concern of this paper is utilization of resources in the computational farm. Utilizing the resources effectively and efficiently is an important problem. Our proposed work depends on

- More resource utilization
- Minimizes the external request response time.
- Minimizes the number of virtual machine pre-emption both for more user satisfaction and also system utilization purposes

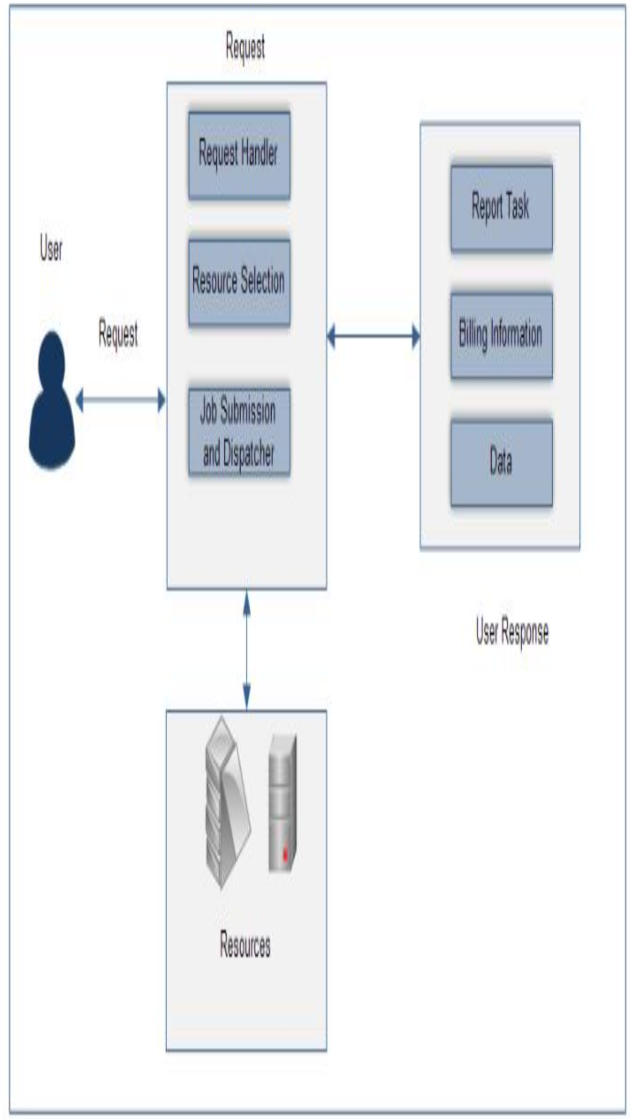

Fig. 1 Architecture diagram

User needs to authenticate to use the grid service and the user will give the task and budget and deadline to the grid resource. Request handler will handle the task and given deadline, the main purpose is used to select the resource for the requested task. User response will display the amount for the finished task and time taken.

Table 1: Notations used in the Algorithm

Symbols	Description
N	Number of clusters
Mj	Number of computing elements in cluster j where $1 \leq j \leq N$
ˆΛj	Arrival rate of external requests to cluster j after load distribution
Mj	Second moment of local requests service time on cluster j
P^{i}	τj + λj
M^{i}	$\hat{\Lambda}_j$ κj
u_j	Utilization of cluster j (γj + ρj)
η_j	Number of VM preemptions that happen in cluster j
T	Average response time of all external requests
T_j	Average response time of external requests on cluster j
v_j	Average number of VMs required by external requests
d_j	Average duration of external requests
s^{i}_{j}	Processing speed (MIPS) of processing element on cluster j

ALGORITHM

Output: Selected Cluster (js)
fastestCluster ← findFastestCluster(θ);
  For each Cluster j do
    $X_j \leftarrow 0$;
  For each RequestType i do
    $P^{i}_{j} \leftarrow Pj * GetProportion(i)$;
    $Y^{i}_{j} \leftarrow 0$;
$X_{fastestCluster} \leftarrow 1$;
For each external request received do
  i ← GetRequestType();
  min ← MaxValue;
  foreach Cluster j do
    if (P^{i}_{j} /= 0) then
    $D = (X_j + Y^{i}_{j})/P^{i}_{j}$;

Fig. 2 Resource Allocation Algorithm

4. Simulation and Results

The simulation is done using Java as the front end and mysql as back end.

The user Registration table maintains the user information such as first name, last name, email id, user name and password. These are provided by the user during the registration as shown in Fig 3

Fig. 3 Screenshot for client registration

The user enters the username and password.

This is shown in Fig 4.

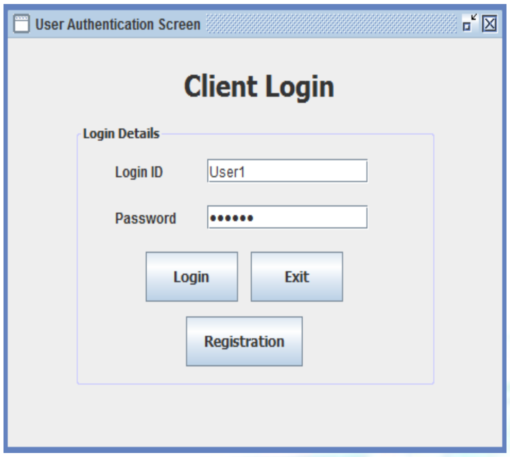

Fig.4 Screenshot for Client login

The grid service(task)request allows the user to select the type of request , task id and to give the deadline etc.Fig 5 displays the grid service(task)request.

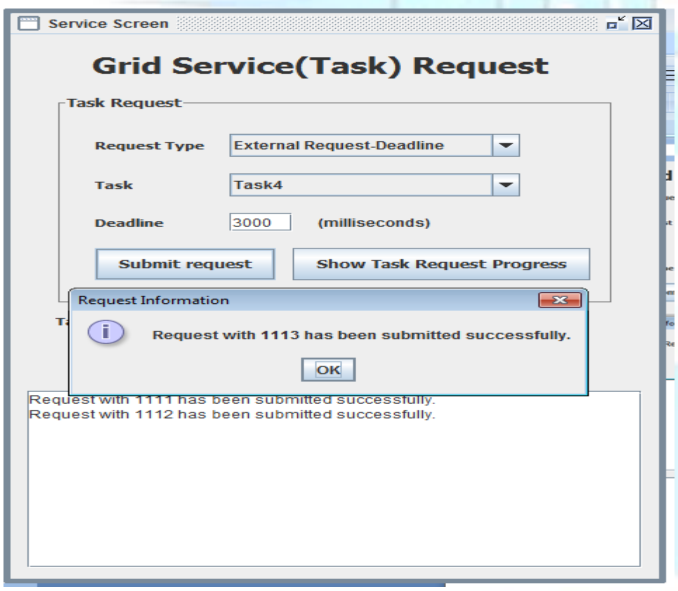

Fig.5 Screenshot for Grid Service(Task)Request for client

5. Conclusion and Future work

In this paper we presented how to select a set of resources that satisfies the user specification. In the proposed system, resource is utilized efficiently; external request response time is minimized. In future we have planned to extend our work based upon reservation and to concentrate on more QOS parameters.

References

[1] Ian Foster, Carl Kesselman, Steven Tuecke, "The anatomy of the grid enabling scalable virtual organizations "Information Sciences Institute, The University of Southern California, Marina del Rey,2003.

[2] Marin golub, Suad kasapovic, "Scheduling multiprocessor tasks with genetic algorithms" Department of Electronics, Microelectronics, Computer and Intelligent Systems. Unska 3, HR-10000 Zagreb, Croatia, 2004.

[3] M.Prakash, T.Ravichandran, "An Efficient Resource Selection and Binding Model for Job Scheduling in Grid", European Journal of Scientific Research, Vol. 81 No.4, 2012. Pp. 450-458.

[4] Haruna Ahmed Abba, Nordin B. Zakaria and Nazleeni Haron, "Grid Resource Allocation: A Review", Research Journal of Information Technology 4(2): 38-55, 2012 ISSN: 2041-3114, © Maxwell Scientific Organization, 2012.

[5] G. Sabin, R. Kettimuthu, A. Rajan, P. Sadayappan, Scheduling of parallel jobs in a heterogeneous multi-site environment, in: D. Feitelson, L. Rudolph,U. Schwiegelshohn (Eds.), Job Scheduling Strategies for Parallel Processing, Berlin, Heidelberg, in: Lecture Notes in Computer Science, vol. 2862, Springer,2003, pp. 87–104.