A Resource Allocation Architecture with support for Interactive Sessions in Utility Grids

Vanish Talwar, Bikash Agarwalla *, Sujoy Basu, Raj Kumar
Hewlett-Packard Laboratories
Palo Alto, CA 94304 USA

Klara Nahrstedt
University of Illinois at Urbana Champaign
Urbana, IL 61801 USA

E-mail: {vanish.talwar,sujoy.basu,raj.kumar}@hp.com, klara@cs.uiuc.edu

Abstract

Utility Grids implement a virtualization architecture and allow for sharing of infrastructure for improved Return on Investment(ROI). We consider extending the existing Grid infrastructure to support interactive sessions in an enterprise setting. This would allow users to remotely execute interactive applications in the Grid and view their output using remote display technologies. We propose a resource allocation architecture to support such interactive sessions in Utility Grids. End-users submit requests for a remote desktop session. A Utility Grid Site Resource Allocation System assigns a compute node for this request considering performance needs of such remote desktop sessions. We also propose a hierarchical admission control system. The system consists of a Site Admission Control System for admission control of remote desktop sessions, and a Session Admission Control system at the compute node for admission check of per-application interactive sessions. We also present discussion on mixed workloads consisting of requests for batch jobs and interactive remote desktop sessions.

1 Introduction

Utility Grids[1] provide geographically distributed heterogeneous resources to be accessed as a utility. They implement a virtualization architecture hiding the details of resource management from the end-user. Traditionally, such virtualization architectures have focused on supporting batch applications. In [8, 4, 2, 3], we introduced Grids which extend the supported application domains to also support interactive sessions using remote display technologies. Keyboard and mouse events are sent from the users' submission node to the remote compute nodes in the Grid, and the remote display is sent from the compute node to the user's submission node. The challenge is to architect a resource allocation system for such Grids. Traditional resource allocation systems cater to the needs of batch applications [5], and others like [7] cater to the needs of 3-tier applications. These existing systems in themselves do not address all the needs for interactive sessions in Utility Grids. Existing thin client system architectures[9] use remote display technologies but they do not have an architecture in the context of Grids. In our work, we consider the Utility Grid to be composed of several sites and we focus only on the resource allocation mechanisms for a single site. Interactive Sessions are more sensitive to performance requirements than batch jobs. In this paper, we propose some key components we are investigating for a resource allocation architecture considering the performance requirements for interactive sessions. Specifically, we describe a hierarchical session model, hierarchical admission control system, resource assignment system, and a discussion on mixed workloads consisting of requests for batch jobs and interactive sessions.

2 System Architecture

2.1 System Model

We are considering Utility Grids in an enterprise environment provided either by Application Service Providers (ASPs)[2] or by in-house IT departments. These Utility Grids have been extended to support interactive applications. Examples of interactive applications in such an environment are for CAD/CAM applications, office applications, and financial applications. In such a structure, the end-users would typically have a thin client machine, and would desire a machine for interactive use. The ASP or in-house IT department has several sites, each site hosts compute nodes and storage nodes. A hierarchy of schedulers exist as shown in Figure 1 to manage the scheduling of user requests in such an environment. At the highest level, we have a Grid Super Scheduler capable of scheduling user requests across different sites. Each of the sites has their own local Utility Grid Site Resource Allocation System which

*Work done during internship at HP Labs. Current affiliation: College of Computing, Georgia Tech, Atlanta, GA. Email: bikash@cc.gatech.edu

0-7803-8430-X/04/$20.00 ©2004 IEEE

in our context is responsible for allocating resources to end-users for interactive use. At the compute node level, the local OS scheduler is responsible for scheduling the per-application sessions on the local processor. In our work, we only consider a Utility Grid Site Resource Allocation System to which the user submits a request for interactive sessions. This is done either directly or indirectly through a Grid Super Scheduler that distributes user requests across several sites.

As an example, consider a user desires a desktop session for executing CAD design applications. He creates a request containing the list of all the applications that he may have to use during the desktop session. For example, he may choose AutoCAD, Matlab, software code editors and compilers. He also specifies in the request the static resource requirements of the compute node he desires eg. the processor speed, memory size etc. These requests are received at a Utility Grid Site Resource Allocation System as shown in Figure 1. On arrival, the requests are placed into an Input Queue. We first apply coarse grain filtering of compute nodes for the requests based on the static resource requirements like desired processor speed and type, the desired memory size etc. We then apply admission control through a Site Admission Control module for those compute nodes satisfying the static resource requirements, taking into consideration the performance requirements of the interactive sessions. Requests for which the Site Admission Control test fails for all eligible compute nodes are placed in a Pending Queue to be tried again later when resources get freed up. An compute node is finally picked by the Resource Assignment system from those satisfying the Site Admission Control test considering the heuristics for interactive sessions. Once a compute node is assigned to a request, the deployment infrastructure performs the configuration including the setup of users' data and required applications. Once the system is setup, the remote desktop session is started, and the user is given direct control of the compute node as shown in Figure 2. The user now submits requests for starting his desired applications directly to the compute node. These requests for starting applications go through a Session Admission Control system. If the session admission control test is satisfied, the application is started in the context of the remote desktop session. Monitoring and Management agents [4] monitor the remote desktop session and the applications within that remote session. The remote desktop session ends after a specified duration of time.

2.2 Hierarchical Session Model

In our utility environment, we consider a hierarchical session model consisting of (i) a top level remote desktop session, and (ii) per-application sessions within a remote desktop session. A remote desktop session constitutes a runtime environment like a VNC (Virtual Network Computing)[6] remote desktop session. The per-application sessions are started interactively by the end-user and they execute in the context of the remote desktop session. All the per-application sessions share the resources allocated to the remote desktop session in which they execute. The request submitted by the user to the Site Resource Allocation System is for obtaining a remote desktop session. The request specifies the list of applications desired within that remote desktop session. The accounting for the user is done for the remote desktop session and so is the coarse grain life cycle management. For example, from the perspective of lifecycle management, the per-application sessions can be considered as children of the remote desktop session. When the remote desktop session is killed, all the children are also killed. Each per-application session is represented by a profile $A_i = \{C_i, N_i, S_i, L_{N_i}, L_{S_i}\}$, where C_i, N_i, S_i represent the required CPU utilization (in cycles/second) , required network bandwidth, and required storage bandwidth respectively for the application. L_{N_i} represents the acceptable network latency between the end-users' submission node and the compute node, L_{S_i} represents the acceptable storage latency between the compute node and remote storage node. Using these profiles for the list of applications

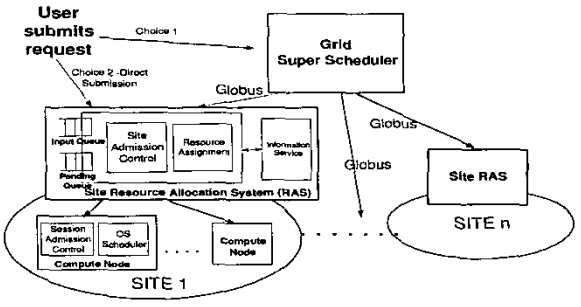

Figure 1. Considered Hierarchical Model

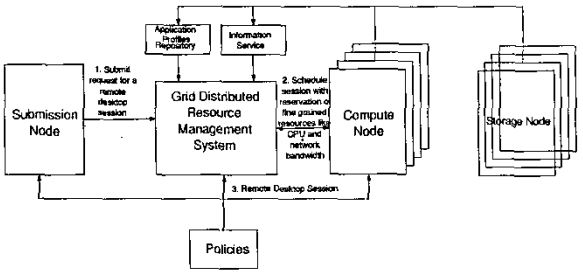

Figure 2. High level overview - A User's perspective

specified in the users' request, we determine the resource requirement for the remote desktop session.

2.3 Hierarchical Admission Control

We propose a hierarchical admission control system consists of a Site Admission Control system, and a Session Admission Control System. We describe them below: A site admission control module exists at the Site level. It is responsible for determining if a compute node meets the resource and latency requirements for a given users' request. These requirements are captured through the resource requirement for the remote desktop session. The resource assignment heuristics are then applied to only those resources that satisfy the admission control test. A Session Admission Control system is responsible for determining if a remote desktop session can meet the resource and latency requirements for a new per-application session. The Global Admission Control system makes an admission decision for the remote desktop session assuming the resource requirements derived from a resource model. Once the remote desktop session is started on the compute node, the end-user can interactively start the applications in various execution orders, and may start several instances of the applications. Hence, we need to perform a Session Admission Control check at the compute node level to check dynamically if there are enough resources available for the application without violating the resource availability for currently running applications.

2.4 Resource Assignment

A Resource Assignment system assigns compute nodes for the users' requests taking into consideration (a) performance needs of interactive sessions for Utility Grids, and (b) Wait time for the user. The Wait Time refers to the time it takes for the compute node to be assigned to a user since receiving the request. Unlike batch job submissions, a user after submitting the request for remote desktop session typically waits for the compute node to be allocated to him immediately. In our system, the wait time is dependent on (is the summation of) the wait time in the 'Input Queue', the wait time in the 'Pending Queue' waiting for resources to become available, and processing overhead of the admission control and assignment algorithms. The resource assignment algorithm must also consider all the three resources - CPU, network bandwidth, and storage bandwidth while making assignment decisions. Appropriate weights must be assigned for these three resources depending on which of these is the bottleneck resource for the set of applications desired by the user.

3 Mixed Workloads

Currently, Grids are being used pre-dominantly for running batch applications. For example, financial solutions in enterprises are using large clusters for running Monte Carlo simulations. Through our work, we have proposed to extend Utility Grids to also support remote desktop sessions. One approach could be to dedicate separate resource pools for the batch applications and remote desktop sessions respectively. However, allowing the sharing of resources for batch and interactive remote desktop sessions would improve efficiency. We are doing simulation studies for such mixed workloads consisting of (i) requests for compute nodes to run batch applications, (ii) requests for compute nodes for interactive remote desktop sessions. We are studying the overall performance of the system through metrics of Throughput and Wait time.

4 Conclusions

In this paper, we presented the key concepts for a resource allocation architecture with support for interactive sessions in Utility Grids. These were hierarchical sessions, hierarchical admission control, resource assignment, and mixed workloads. The hierarchical session model for interactive sessions consisting of remote desktop session and per-application sessions in the context of the remote desktop session. The hierarchical admission control system consists of a Site Admission Control system at the Site level and a Session Admission Control system at the compute node. A Resource Assignment system is responsible for assigning compute nodes to the users' considering the needs of interactive sessions. We are also looking at mixed workloads consisting of mixed batch and interactive session workload. As future work, we are refining and developing the concepts in more detail and also looking at simulation studies for mixed workloads.

References

[1] Global grid forum. http://www.ggf.org.

[2] S. Basu, V. Talwar, B. Agarwalla, and R. Kumar. Interactive grid architecture for application service providers. In *Proceedings of the International Conference on Web Services (ICWS)*, June 2003.

[3] J. Dwoskin, S. Basu, V. Talwar, R. Kumar, F. Kitson, and R. Lee. Scoping security issues for interactive grids. In *Proceedings of the Asilomar Conference on Signals, Systems, and Computers*, November 2003.

[4] R. Kumar, V. Talwar, and S. Basu. A resource management framework for interactive grids. In *Proceedings of the 1st International Workshop on Middleware for Grid Computing, Middleware 2003*, June 2003.

[5] J. Nabrzyski, J. M. Schopf, and J. Weglarz. *Grid Resource Management: State of the Art and Future Trends.* Kluwer Academic Publishers, 2003.

[6] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual network computing. *IEEE Internet Computing*, 2(1):33–38, 1998.

[7] J. Rolia, J. Pruyne, X. Zhu, and M. Arlitt. Grids for enterprise applications. In *Workshop on Job Scheduling Strategies for Parallel Processing*, June 2003.

[8] V. Talwar, S. Basu, and R. Kumar. An environment for enabling interactive grids. In *Proceedings of the IEEE International Symposium on High Performance Distributed Computing (HPDC-12)*, June 2003.

[9] S. Yang and et al. The performance of remote display mechanisms for thin-client computing. In *2002 USENIX Annual Technical Conference*, June 2002.