An Environment for Enabling Interactive Grids

Vanish Talwar, Sujoy Basu, Raj Kumar
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304 USA
{vanish.talwar,sujoy.basu,raj.kumar}@hp.com

Abstract

Traditional use of grid computing allows a user to submit batch jobs in a grid environment. We believe, next generation grids will extend the application domain to include interactive graphical sessions. We term such grids interactive grids. In this paper, we describe some of the challenges involved in building interactive grids. These include fine grain access control, QoS guarantees, and dynamic account management. In order to architect interactive grids, we propose and describe I-GENV, an environment for enabling interactive grids. I-GENV consists of GISH-'Grid Interactive Shell', Controlled Desktop, SAC-'Session Admission Control' module, GMMA-'Grid Monitoring and Management Agents', System Policies, and Dynamic Account Manager. We also present our testbed implementation of I-GENV using and extending Globus Toolkit 2.0 for the Grid middleware infrastructure, and VNC as the remote display technology.

1 Introduction

Grid Computing [1, 2] envisions a future where heterogeneous resources could be shared by users across geographical and administrative boundaries, and as a utility. Several efforts [6, 14, 15, 16, 17] are underway to architect and deploy a middleware infrastructure for Grid Computing. Commercial acceptance of Grid Computing technology is also steadily increasing. Traditional use of Grid Computing has been for the execution of batch jobs in the scientific and academic community. We believe that next generation grids will extend the application domain to include graphical interactive sessions. Such sessions would allow the end-user to interactively submit graphical, multimedia jobs to remote nodes in a Grid. The end-user will also be able to view the graphical and multimedia output of the submitted jobs and applications through such graphical interactive sessions. Example use cases for such interactive sessions could be for, but not limited to, graphics visualization applications, engineering applications like CAD/MCAD, digital content creation, streaming media, video games, text editing, command line interactions, e-mail applications. Some of the new problems posed for the design of such grids are: Fine Grain Access Control, QoS, and Dynamic Account Management.

Most of the ongoing work on Grids is for batch jobs. Other work like [13] does not focus on providing interactive job submission 'sessions', and [4] does not address the needs for graphical and multimedia sessions. Neither do the other related works provide a comprehensive framework for access control, QoS, and account management for graphical interactive sessions in a Grid Computing environment. In this paper, we propose I-GENV: a Grid environment for graphical interactive sessions to remote nodes. Our key contributions are as follows:

1. A framework for providing access control and QoS for graphical interactive sessions in a Grid, consisting of the following components:
(a) GISH - a 'Grid Interactive Shell', is a controlled shell providing fine grain access control. GISH also interfaces with SAC - 'Session Admission Control' system.
(b) Controlled Desktop - A Controlled desktop restricts the user to only launch allowed applications through a desktop.
(c) SAC - a 'Session Admission Control' module, provides admission control check for session resource usage parameters.
(d) GMMA - 'Grid Monitoring and Management Agents' monitor and enforce the session and resource parameters during a graphical interactive session.
The above components are tightly coupled to each other, and are designed with grid-enabled features.

2. A dynamic account management system for graphical interactive sessions to simplify the management of user sessions in a Grid Computing Environment.

Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03)
1082-8907/03 $17.00 © 2003 IEEE

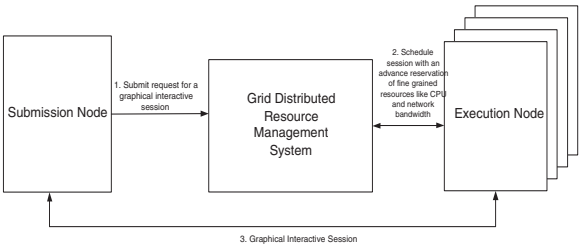

Figure 1. High level overview of the Interactive Grid computing system

Our architecture is proposed as an extension to the existing Grid middleware infrastructure. Our implementation uses Globus Toolkit 2.0 [6] as this Grid middleware infrastructure. We believe our proposed solution provides a comprehensive access and admission control methodology for graphical interactive sessions in a Grid context. The access control system is modular and policy based allowing for fine grained access control, easy extensibility and easy manageability. There is support for Quality of Service guarantees in an interactive graphical session, as well as support for dynamic accounts.

The rest of the paper is organized as follows. In Section 2, we describe Interactive Grids. We describe I-GENV in Section 3. In Section 4, we show how I-GENV solves the problems of fine grain access control, QoS, and manageability. In Section 5, we present an analysis of our solution. Section 6 presents implementation of the proposed solution. In Section 7, we present Related Work and we conclude in Section 8.

2 Interactive Grids

Figure 1 shows the Grid computing system that we are considering. The system consists of heterogeneous execution nodes distributed across multiple administrative domains. These nodes are managed by a Grid Distributed Resource Management system (DRM). An end-user submits a request for an interactive session to the Grid DRM through a submission node. On receiving the request from the user, the Grid DRM selects a remote execution node based on the session requirements, and reserves this node for the requested duration of the session. The Grid DRM also performs an advance reservation of fine grained resources like CPU and network bandwidth, for the users' session. At the requested time, the DRM would establish an interactive session between this remote execution node and the end-user's submission node. The end-user then interacts directly with this remote node[1], through the established session. During this session, the user can submit requests directly to the remote node, to launch multiple applications. A session thus constitutes the interaction of the end-user with the remote node involving the launching of one or more applications and, subsequently interactively using the launched applications. These interactions could either be graphical or text-based. We are more interested in addressing the problem for graphical interactive sessions to remote nodes. However, the solutions being proposed and developed by us are also applicable for text-only interactive sessions, as a special case. The interaction of the end-user with the remote node involves the execution of both installed applications and user specified binaries. It is also assumed that the user does not have a local account with the remote node apriori.

Such a grid computing model represents interactive grids - next generation grids supporting graphical interactive sessions. Interactive grids pose several new problems in the context of Grids. Some of the challenges include fine grain access control, QoS guarantees, and dynamic account management.

Fine Grain Access Control: Interactive sessions allow a malicious user to submit unauthorized jobs to the remote node and permit end-users unspecified time of access to the remote node. Further, interactive sessions allow a malicious user to probe for vulnerabilities in the Grid system, and launch attacks against other remote nodes in the Grid system. A fine grain access control in the Grid context is needed.

QoS: Interactive grids permit end-users to launch interactive graphics and multimedia applications. Such applications are sensitive to response time and real time requirements. There is a need to guarantee quality of service for such applications.

Dynamic Account Management: Interactive grids pose a challenge in terms of account management for arbitrary end-users of the grid.

3 Our Solution: I-GENV

Our solution is to provide I-GENV: an environment for graphical interactive sessions, in an Interactive Grid Computing system. I-GENV consists of the following components:

1. Controlled Shell: GISH - 'Grid Interactive Shell'
2. Controlled Desktop
3. SAC - 'Session Admission Control' module
4. GMMA - 'Grid Monitoring and Management Agents'
5. System Policies
6. Dynamic Account Manager

These components provide for fine grain access control, QoS, and account management in interactive grids. Figure 2 shows the interaction of I-GENV components in the

[1]The terms 'remote node' and 'remote execution node' are used interchangeably in this paper.

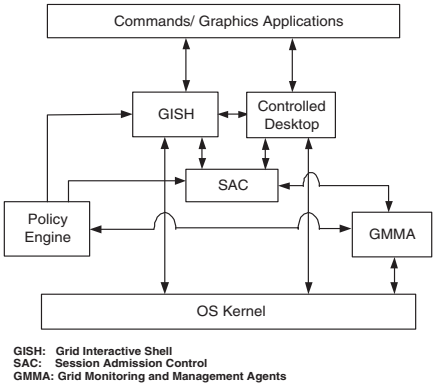

Figure 2. Interaction among I-GENV components

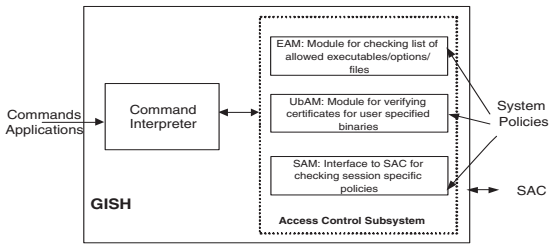

Figure 3. GISH Design

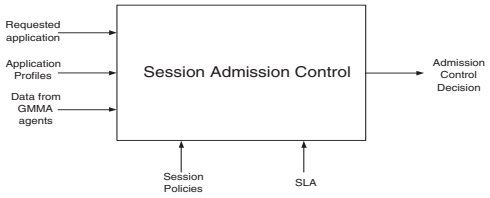

Figure 4. SAC Design

context of a dynamic account created by the Dynamic Account Manager. In the next few sections, we describe GISH, GMMA, SAC, and Account Manager. We then describe how these components provide for access control, QoS, and manageability.

3.1 Controlled Shell

A controlled shell provides a restricted interface to the end-user to submit requests for executing applications and commands to the remote node interactively. We have designed a controlled shell called GISH -'Grid Interactive Shell'. GISH accepts requests to execute two kinds of commands/applications:
1. Commands and applications that are already installed on the remote node by the system administrator.
2. Commands and applications that are NOT already installed on the remote node, and is a user specified binary file. Such applications must be certified to be non-malicious by a trusted certificate authority.

GISH allows grid-users to be logged on to the remote node through two kinds of user accounts:
1. Controlled normal user accounts. This corresponds to a normal user account given by the underlying operating system, restricted by the access control policy files.
2. Controlled super user accounts. This corresponds to a super user account given by the underlying operating system, restricted by the access control policy files.

Figure 3 shows the design of GISH. It consists of a command interpreter interfaced to an access control subsystem. The access control subsystem consists of access control modules described in detail below. The user submits a request to start a command or application to GISH. The command is first parsed by the command interpreter, and then passed onto the access control modules. Each of the access control modules performs an access control check. If the access control check fails for any of the modules, a failure message is returned back to the user and the request to start the application/command is denied. If the access control check succeeds for all the modules, then the command or application is started by GISH and the graphical output, if any, can be viewed through the remote graphical display. We describe briefly some of these access control modules below. In order to make the design modular, we choose to interface GISH with a Session Admission Control system (SAC). SAC is responsible for making admission control decisions for session parameters.

EAM: Executables and files Access control Module
This module is responsible for verifying that the requested command/application (i) Belongs to the list of allowed executables, (ii) Is invoked with a list of allowed arguments/options, (iii) Only accesses allowed files and directories. This verification is enforced through a policy file which enumerates the list of allowed executables, allowed executable arguments, allowed files and directories for the user. The policy files used by EAM is categorized based on whether the user is logged on as a controlled normal user or as a controlled superuser. For allowed executables, EAM would not be able to determine all of the files and directories that an application would access. In order to restrict the applications from accessing only the allowed files and directories at run-time, we supplement GISH, based on a policy decision, with systems like [19, 5] which compartmentalize the execution of processes, or with virtual machine sandbox environments like [3].

UbAM: User binaries Access control Module
This module is responsible for verifying the signature for

user specified binaries. We assume that there exists trusted services in our grid computing environment that checks user specified binaries and signs non-malicious binaries. UbAM verifies such signatures. If such trusted services are unavailable to the user, we provide based on a policy decision, a virtual machine environment [3] for executing the users's binaries, or supplement the system with runtime system call monitoring systems [4].

SAM: Session Access control Module
This module interfaces with SAC- 'Session Admission Control' module, for verifying session specific parameters. SAC is explained in detail in Section 3.4. SAM passes the requested command to the SAC. SAC replies back with an 'Allow' or 'Deny' decision for the requested command/application.

The GISH design shown in Figure 3 could be extended with other access control modules as seemed appropriate for a particular implementation. We have presented a few access control modules that we envision to be necessary in a Grid environment for graphical interactive sessions.

3.2 Controlled Desktop

The controlled desktop has to be identical to GISH in terms of the policies enforced. The desktop's menus and icons is customized by a desktop configuration file that enforces these policies. At the time of initialization of the session, this file is read in for customizing the desktop. The user is not given permission to modify this file, or to add or modify menu items or icons. Only the allowed executables with allowed arguments, and allowed files for the user is accessible through the controlled desktop.

3.3 SAC

SAC stands for Session Admission Control module. This module is responsible for making an admission control decision for a requested application, based on Service Level Agreements (SLAs), and session policies. Figure 4 shows the inputs to a SAC system. These are explained below:
1. *Requested application*: The graphics application which the user is requesting to be launched. This is provided to SAC by GISH through the SAM module.
2. *SLA*: The Service Level Agreement for the session in progress. The SLA is determined prior to the start of the session.
3. *Application profiles*: The application profiles contain the estimated CPU and bandwidth required for various classes of applications to meet their acceptable performance levels. Example classes of applications are engineering applications, visualization applications, video games etc. Such application profiles are determined by a system administrator, and refined by an application predictor system.
4. *Data from GMMA agents*: The resource usage data gathered by GMMA monitoring agents. The GMMA monitoring agents are explained in Section 3.4.
5. *Policies*: The session policies in place for the session.

Given these inputs, SAC checks the session parameters to verify availability of resources in compliance to SLAs. These session parameters are:
1. Number of processes launched during a session.
2. Usage time for a session.
3. Disk quota usage for a session.
4. CPU utilization percentage for a session.
5. Network bandwidth utilization percentage for a session.

SAC compares the current values for these session parameters with the limiting values agreed upon in the SLA. If there is a violation, or if a violation would occur upon executing the application, SAC decides on a 'Deny' decision for executing the application. Otherwise, an 'Allow' decision is made for the application by SAC. Figure 5 shows an algorithm for SAC to make an admission control decision, based on the CPU and network bandwidth utilization parameters for a session.

Input: Application request, reservation agreement from SLA, data from GMMA agents, application profiles

Output: Admission control decision (Allow or Deny)

1. Determine the class of applications that the requested application belongs to.
2. Obtain from the application profiles, the CPU and network bandwidth usage requirement for this application.
3. Use the data gathered by GMMA agents to obtain the current CPU and network bandwidth utilization values for the session.
4. Obtain from the SLA, the CPU and network bandwidth reservation values made for this session.
5. Compare the values from Step 3 and Step 4 to determine the CPU and network bandwidth available for the requested application, to comply with the SLA.
6. Compare the values from Step 2 and 5 to determine if executing the requested application would violate the SLA. If so, return "Deny". Else return "Allow".

Figure 5. An algorithm for SAC with CPU and network bandwidth utilization as the session parameters

SAC could be extended to support other session parameters as seemed appropriate for a particular implementation. We have presented a few session parameters that we envision to be necessary in a Grid environment for graphical, interactive sessions.

3.4 GMMA

GMMA stands for 'Grid Monitoring and Management Agents'. The monitoring agents collect dynamic monitoring data, which is used by the management agents to enforce

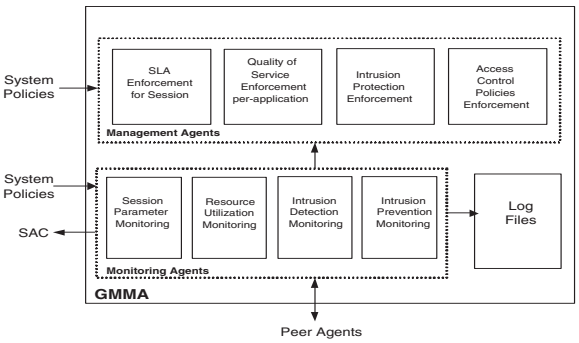

Figure 6. GMMA Design

session SLAs, QoS for applications, intrusion protection, and access control policies.

Some of the GMMA agents are associated with a specific session, while some others are system wide agents that monitor all the sessions started through the interactive Grid environment. The monitoring agents log their information in log files[2], interface with other peer agents, other monitoring systems, as needed. The management agents use the data gathered by monitoring agents for enforcement purposes. Figure 6 shows the design of GMMA. We describe below a few categories of the monitoring and management agents, focusing on the functionality to be provided by these agents in the context of interactive grids.

3.4.1 Monitoring Agents

Session Parameter Monitoring
The Session parameter Monitoring is for monitoring session specific parameters like (i) Usage Time for the session, (ii) Number of processes spawned during the session, (iii) Number of socket connections opened during the session, (iv) Disk quota usage for the session, (v) CPU Usage for the session, (vi) Network Bandwidth usage for the session. This data is then used by the management agents to enforce SLA guarantees for these session parameters.

Resource Utilization Monitoring
The resource utilization monitoring is for monitoring the overall as well as per-application utilization of resources like CPU, network bandwidth. This data is then used to provide QoS guarantees to applications. The resource utilization data is also used by the Grid DRM while making scheduling decisions onto this node.

Intrusion Detection Monitoring
The Intrusion Detection Monitoring is for monitoring the intrusion detection parameters. For example, these agents monitor the IP addresses of incoming connections, TCP connection information. These agents could also interface with their peer agents on other grid enabled nodes. This allows the agents to share intrusion detection information, thus forming a distributed intrusion detection system for Grid environments.

Intrusion Prevention Monitoring
The Intrusion Prevention Monitoring is for monitoring the intrusion prevention parameters. For example, these agents monitor the IP addresses for outgoing connections, and notify the management agents when certain selective connections are attempted. The agents are informed about the list of selective connections through policy files. The management agents take appropriate action for these selective connections, for example, they may block these connections as a precautionary measure to prevent a possible intrusion.

Other possible monitoring agents are for monitoring the files, directories, and system calls made by applications started through I-GENV. This monitored data is then used by management agents to enforce access control policies.

3.4.2 Management Agents

SLA Enforcement for sessions
The SLA Enforcement for session parameters is for enforcing the Service Level Agreements for the session parameters. The SLA agents obtain the session data from the monitoring agents, and use this data to verify for any violation of session SLAs. The SLAs are for resource reservations made per session, wall-clock usage time for the session, number of processes launched during a session. Based on policies, an appropriate action is taken on a violation of SLA, for example kill the processes started during the session, and end the session[3].

Quality of Service Enforcement Per application
The QoS Enforcement is for enforcing the QoS guarantees per application. The application profiles discussed in an earlier section contain the estimated CPU and network bandwidth required for various classes of applications for acceptable performance and response time to the user. The QoS enforcement agents obtain the data from the monitoring agents, and check for any violation of QoS specified in the application profiles. Once a violation is detected, an enforcement action is taken as one or the combination of the following: (i) Decrease the priority of applications that exceed their resource utilization levels.

[2]This logged data is used only for session and Grid management purposes. Any privacy issues regarding this information would be agreed upon as an agreement prior to the start of the session.

[3]A warning message may be optionally given to the end-user before killing the process.

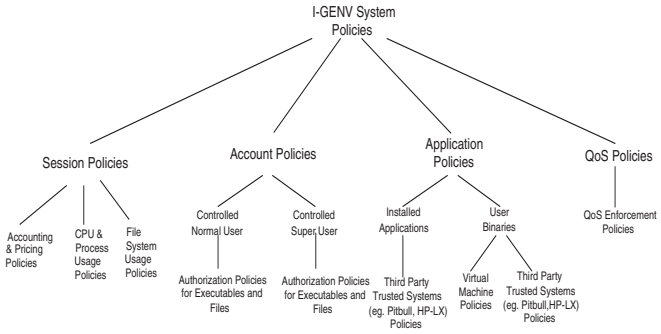

Figure 7. Taxonomy of I-GENV system policies

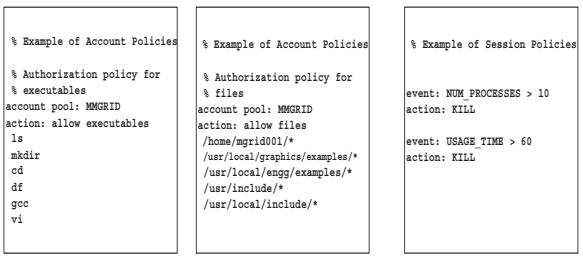

Figure 8. Example system policy files

(ii) Increase the priority of applications falling below their desired resource utilization levels. (iii) Kill applications that have violated their resource utilization levels by a large amount. The enforcement process for QoS is controlled by policies.

Intrusion Protection Enforcement
The intrusion protection enforcement is for enforcing intrusion protection mechanisms. An intrusion protection enforcement action can be triggered through (i) Intrusion prevention monitoring agents on detecting certain selective connections. In such cases, an example enforcement action is to block those connections. (ii) Detecting an intrusion using the data gathered through Intrusion Detection Monitoring Agents. In such cases, the IP addresses in question could be added to the list of selective connections monitored by the Intrusion Prevention Monitoring Agents.

Access Control Policies Enforcement
The Access Control Policies Enforcement is for enforcing access control policies like restricting the access to only the set of allowed files and executables for this grid user. The monitoring agents would gather the information about files and executables accessed by a grid-user during the interactive session, and this data is used to verify compliance with access control policies. On a violation of access control policies, an appropriate action is taken eg. killing of the violating application.

The GMMA design presented in Figure 6 could be extended with other monitoring and management agents as seemed appropriate for a particular implementation. As for GISH and SAC, we have presented a few categories of monitoring and management agents that we envision to be necessary in a Grid environment for graphical interactive sessions. However, the design can be extended with other categories of monitoring and management agents as well.

3.5 System Policies

The system policies can be classified into the following categories also shown in Figure 7:
(1) Session Policies. These specify policy information for each session. Examples of such policies are accounting and pricing policies, CPU and process usage policies, file system and disk quota usage policies. The policies specify the default action to be taken on a violation of the system parameters.
(2) Account Policies. These specify policy information associated with account pools. There are separate policies for controlled normal users and controlled superusers. Examples of such policies are the authorization polices for executables and files for a user of the account pool.
(3) Application Policies. These specify policy information for applications that are started by I-GENV. There are two kinds of applications: installed applications, and user specified binaries. The execution of these applications could take place in a secure environment using systems like PitBull [19], HP-LX [5], or virtual machine environments [3]. Such systems have their own policies.
(4) QoS Policies. These specify policy information for QoS metrics. Example policies are the QoS enforcement policies on violation of QoS metrics specified in the application profiles.

Each of the above policies are customized for a given grid-user of the system. Figure 8 shows examples of some system policy files.

3.6 Dynamic Account Manager

I-GENV uses dynamic or template accounts to make the resource virtualization more appropriate for grids. The scalability and manageability of the system are enhanced if we do not require grid users to have their personal user accounts on all the machines that are part of the grid. Instead the system administrator has to add the user once to a directory maintained by the virtual organization in which the user has obtained membership. Any site that participates in that Virtual Organization (VO) will check the user's membership with the directory during authentication, and authorize the

user as a dynamic account if she does not have a static account. The dynamic account is chosen from the pool of dynamic accounts maintained for that VO. Each dynamic account is a full-fledged Unix account created on the computer, but without a permanent real-world user associated with it.

Each pool is associated with a set of the policy files for I-GENV mentioned in Section 3.5, customized to the target users of that pool. Unlike normal user accounts that belong permanently to their real-world owners, a dynamic account is bound to a user temporarily. The selection of a pool and the binding of the user to an available dynamic account from that pool are based on the Grid credentials presented. After the successful selection and binding of user to a dynamic account, the graphical interactive session is started. A window manager, terminal windows running the GISH shell, and other programs specified in the window manager's startup files are started as processes owned by the allocated dynamic account. The appropriate GMMA agents are also simultaneously started for this session. The entire process can be described by a flowchart shown in Figure 9.

The dynamic account is freed at the termination time agreed upon for the session that is using the dynamic account. At the termination time, GMMA management agents kill the processes still running with this account as owner, and delete all files owned by the account. The account is then returned to the pool. We could also choose to archive the files created by the user as against deleting it, on a server maintained by the VO. Subsequent sessions for this user retrieve the files from the archive.

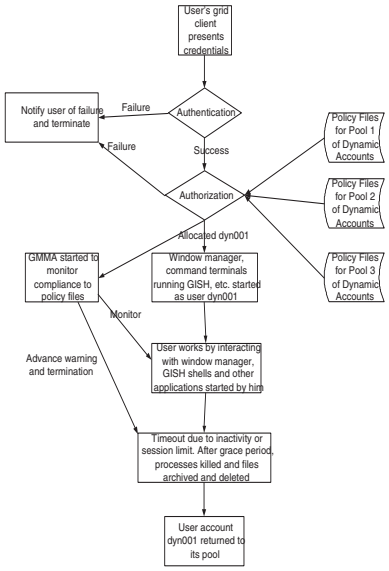

Figure 9. Flow Chart describing the process of account allocation, access control and session management

4 Discussion

Figure 2 shows the interaction among the components of I-GENV. As shown in the figure, there is a tight coupling among the components. These components exist in the context of a dynamic account created by the Dynamic Account Manager. Together, the I-GENV components achieve Access Control, QoS, and Manageability. We explain this below.

Access Control through I-GENV

Access Control for a grid-user is achieved through a combination of GISH, Controlled Desktop, GMMA, SAC, and system policies. Using these components, we can control the access of the user to (i) Executables, (ii) Files, (iii) Network interfaces, (iv) Network connections, (v) Resource usages decided in SLA.

QoS through I-GENV

QoS in I-GENV is achieved through a combination of SAC, GMMA, and system policy files. We assume that the the Grid middleware would have made CPU and network bandwidth reservation, before the session is launched on the remote node. We also assume the existence of application profiles which contain the estimated CPU and network bandwidth required for various classes of applications to meet acceptable performance levels. GMMA monitoring agents monitor the actual usage values of the CPU and network bandwidth utilization of applications. This data is used to enforce QoS for each application as specified in the application profiles. The data gathered by GMMA monitoring agents is also used by the GMMA management agents for enforcing the reservation limits for sessions stated in the SLA (policing). Further, before a new application is launched, the Session Admission Control System (SAC) verifies that the requested application would consume resources within the reservation limits agreed upon for the session. This check ensures that the SLA and QoS guarantees for currently executing applications, would not be violated upon launching the application[4].

Manageability through I-GENV

The system policy files are associated with a pool of dynamic user accounts. A grid user is mapped to one of these pools of dynamic accounts based on a VO-wide policy. The user is then dynamically allocated one of

[4]Even if the SAC does not perform this admission control check, the monitoring agents would detect the violation and an appropriate enforcement action would be taken. However, there would be a time delay before such an action can take place. Performing a check at the SAC itself ensures no violation of SLA even for this time delay period.

the accounts from the mapped account pool. The user is subject to the system policies associated with that account pool. After the session for the user expires, the dynamic account is returned back to the pool. Such a design coupled with the access control and monitoring agents system provides for easy manageability through the proposed Grid Environment, I-GENV.

5 Analysis

We now provide an analysis of our solution as described in the previous sections. While designing the system, we came up with a list of requirements for our solution to satisfy. These were: (1) Be applicable across heterogeneous platforms, (2) Extend existing general purpose tools, (3) Require minimal changes to the existing system software, (4) Be extensible and modular, (4) Address needs of graphics and multimedia applications, (4) Support self-managing capabilities, (5) Work for all application types, (6) Be flexible to interface and interoperate with other complementary and grid solutions, (7) Be driven by policies.

Towards this end, we avoided designing solutions specific to an OS, or in-kernel solutions, or solutions requiring drastically new tools. This is reflected in our design for example through GISH, which can be implemented by extending existing popular shells like bash. In order to make the design self-managing, we propose monitoring and management agents, which gather run-time information, and enforce appropriate enforcement actions to honor SLAs and access control policies. We also took a two-level approach of (1) filtering commands before execution to verify for access and admission control, and then (2) monitoring and managing the behavior of the system at run-time. In such a two-level approach, we introduce tight feedback loops from the GMMA agents to access control policy engine and SAC module, so that dynamic run-time information gathered can be used for subsequent admission and access control decisions. This helps in making the system self-learning and self-managing. The two-level approach is also expected to provide performance benefits by eliminating some of the non-compliant application requests at the shell itself. Since we desired to design the system to work for any application, we avoided QoS solutions like QuO [22] from the Quorum project [23], that specifically address needs of distributed object systems built over CORBA-like middleware infrastructures. Rather, in order to provide QoS management we provide (1) Functionality in the Interactive Grid DRM to allocate the most appropriate available resource(s) that would satisfy the QoS requirement of the application. (2) Monitoring and enforcement framework to adjust fine-grain resource allocations of applications at run-time to enforce QoS guarantees.

We also realized during the design process that an integrated set of components could be used to together solve the three problems of fine grain access control, QoS, and account management, compared to providing separate solutions to each of these problems. We believe this would help improve efficiency. At the same time, our solution is also designed to be modular. The solution is designed to build upon components providing common functionality that can be used for multiple purposes. For example, our GMMA agents provide input to both the first step of application admission control and the second step of QoS management at runtime. Dynamic account management also contributes to our theme of QoS enforcement by allowing us to customize the policy files governing the account that serve as input to QoS management.

In terms of performance analysis, we are interested in reducing the overhead caused by I-GENV during an interactive session in an interactive grid computing system. The overhead caused by Dynamic Account Manager is only at the beginning of a session, and is expected to be minimal since it would primarily involve accessing the *gridmapfile*, and executing a simple logic to map the user to the appropriate account. The overheads caused by GISH and SAC is also expected to be insignificant compared to the high human response time expected while the user is interactively submitting commands. The overhead caused by GMMA agents would incur in terms of filesystem read access, agent-agent communication, enforcement algorithms. These can be addressed through appropriate implementation techniques and as future work, we are designing the GMMA agents taking these factors into consideration, so as to not limit the usefulness of the system due to these overheads.

6 Implementation

Our implementation environment consists of Intel x86 machines running Red Hat Linux 7.3, as the remote nodes. The end-user can request a graphical interactive session to these remote nodes from any machine supporting a web browser. We use Globus Toolkit 2.0 [6] as the Grid middleware platform, and VNC [8] as the remote display technology for remote graphical sessions. GPDK [7] is used to provide a web portal to the end-user for submitting job requests. We have extended the functionality of Globus, so that it can also be used for submitting requests for starting a graphical interactive session to the remote nodes. Figure 10 in Appendix shows the job submission process. On a successful authentication, the appropriate account pool for the user is determined. A set of policy files is associated with this account pool. A dynamic account from this pool is then allocated for this user. We then start a VNC server and GMMA agents for this session. On a successful VNC authentication, the user is presented with a controlled KDE

Desktop environment containing only the applications and menus the user is allowed to access. The KDE desktop environment is pre-configured by the system administrator for each pool of accounts.

The session starts with default startup applications, including a GISH shell. The GISH shell has been implemented as an extension to the popular GNU bash shell for Linux and Windows. The shell source code was modified so as to include the access control modules. GISH currently checks for list of allowed executables from a file, before executing commands. The GMMA Agents started at the beginning of the session, run with super-user privileges. They record the session and system information and store them in pre-determined files. Currently, the GMMA agents check for the usage time for the session, number of spawned processes. The system policy files contain information about the session usage time, number of allowed processes etc. along with their maximum allowed values, and actions to be taken on violation of these policies. The current default action is to KILL all the processes and end the session, on violation of the session policies. Implementation for other modules and agents for GISH and GMMA is a work in progress. Figures 11 shows some interaction examples within GISH during a session in progress, and Figure 12 shows the session screen on termination.

To support dynamic accounts, we modified $globus\_gatekeeper$ and GSI-SSH daemon from Globus Toolkit 2.0 by linking them with a modified library for reading the $gridmapfile$. Normally the $gridmapfile$ contains entries mapping the distinguished name (DN) of the user to the local Unix account. As a first step, we modified the $gridmapfile$ to replace the local Unix account name with a predefined string for the user's VO, indicating that a dynamic account from the VO's pool should be used for this user. To get our modified library for reading the $gridmapfile$, we started with a patch to the $globus_gss_assi\ st$package, distributed by the Grid for UK Particle Physics, obtained from [11]. As a second step, we are extending the $globus\_gss\_ass\ ist$ library further. We are removing the requirement to have an entry in the $gridmapfile$. When the user authenticates herself, the DN obtained from her certificate will be queried against the directory maintained by the VO for membership. If the user is a member, she will be assigned a dynamic account from the pool customized for that VO.

7 Related Work

Majority of the work in the area of grid computing has been for batch jobs and hence do not address the problems as outlined in the paper. The same holds with recent projects on interactive applications like CrossGrid [13]. Solutions developed in Punch project [4] do not address graphical and multimedia sessions. gsissh [18] provides for encryption of interactive sessions and can be used with our solution. QoS solutions provided through Quorum project [23, 22] address the needs of distributed object systems and provide QoS support in underlying middleware infrastructure like CORBA. We propose QoS support for applications through appropriate initial resource allocation and subsequent monitoring and adjustment of resource allocations by the Grid resource management framework. We do not assume any available support for application adaptation through multiple application behaviors. Most of the other related work for access control and QoS are in the non-grid context do not completely satisfy the requirements for Grid. For example, traditional OS access control mechanisms do not allow to easily enforce fine grain access control for arbitrary end-users. Sudo [21] is not a replacement for shell and does not provide a complete solution for all our needs in Grid context. Our solution provides for an integrated and comprehensive solution for problems of access control, QoS, and account management in the context of graphical interactive sessions in Grids. Our solution is not specific to any remote display technology and can be used with systems like [8, 20]. Monitoring Systems like Network Weather Service [10] can be interfaced to the GMMA framework. Dynamic account management has been described in [9, 12]. However, we differ from the prior work in using the dynamic account as a component of our customizable grid environment, by associating each pool of dynamic accounts with its set of policy files that I-GENV enforces.

8 Conclusions

In this paper, we introduced interactive grids which allow end-users access to remote execution nodes belonging to a Grid, for graphical interactive use. Interactive Grids extend the application domain for Grid Computing Systems from traditional batch jobs to graphical, interactive sessions. We described some of the problems posed for the design of interactive grids, namely that of fine grain access control, QoS, account management. In this paper, we have presented I-GENV: an environment for enabling interactive grids, that addresses these issues. Our approach has been in building a set of components addressing these issues in an integrated but modular manner. We believe this leads to more efficiency. The components are GISH - 'Grid Interactive Shell', Controlled Desktop, SAC - 'Session Admission Control' system, GMMA - 'Grid Monitoring and Management Agents', System Polices, and Dynamic Account Manager. While designing the system, we realized the need to satisfy important requirements of heterogeneous platforms, easy extensibility, modularity, and self-managing capability. We identified the areas of overheads that would occur with a deployment of our solution, which would be consid-

ered for optimization in the future work. We also described our implementation of the system on Linux x86 machines, using and extending Globus Toolkit 2.0 for the base grid middleware infrastructure and VNC as the remote display technology.

References

[1] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kauffman Publishers, 1999.

[2] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of SuperComputing Applications*, 15(3), 2001.

[3] E. Bugnion, S. Devine, K. Govil, and M. Rosenblum. Disco: Running commodity operating systems on scalable multiprocessors. *ACM Transactions on Computer Systems*, 15(4):412–447, 1997.

[4] A. Butt, S.Adabala, N.Kapadia, R.Figueiredo, and J.A.B.Fortes. Fine-grain access control for securing shared resources in computational grids. In *IPDPS*, April 2002.

[5] N. Edwards, J. Berger, and T. Choo. A secure linux platform. In *5th Annual Linux Showcase & Conference*, November 2001.

[6] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of SuperComputing Applications*, 11(2):115–128, Summer 1997.

[7] J. Novotny. The grid portal development toolkit. *Concurrency-Practice and Experience*, 2000.

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

[9] T.J.Hacker and B.D.Athey. A methodology for account management in grid computing environments. In *2nd International Workshop on Grid Computing*, November 2001.

[10] R. Wolski, N. T. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 15(5–6):757–768, 1999.

[11] http://www.gridpp.ac.uk/gridmapdir.

[12] An Accounting System for the Datagrid Project version 3.0. http://server11.infn.it/workload-grid/docs/DataGrid-01-TED-0115-3_0.pdf.

[13] Crossgrid. http://www.crossgrid.org.

[14] NASA IPG. http://www.ipg.nasa.gov.

[15] Teragrid Project. http://www.teragrid.org.

[16] Eurogrid Project. http://www.eurogrid.org.

[17] Entropia. http://www.entropia.com.

[18] Gsi-ssh. http://www.ncsa.uiuc.edu/Divisions/ACES/GSI/openssh/.

[19] Pitbull lx white papers. http://www.argus-systems.com.

[20] Sgi opengl vizserver. http://www.sgi.com/software/vizserver.

[21] Sudo. http://www.courtesan.com/sudo/.

[22] J. Zinky, D.Bakken, and R. Scantz Architectural Support for quality of service for CORBA objects. *Theory and Practice of Object Systems*, 3(1), 1997.

[23] Quorum. http://www.dist-systems.bbn.com/projects/QuOIN/

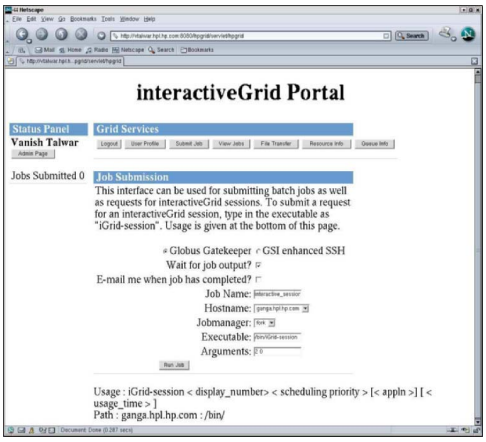

Figure 10. Job Submission screen for a graphical interactive session through Globus

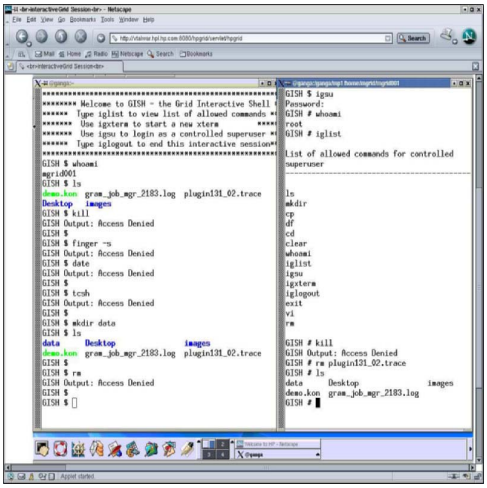

Figure 11. Screen during an interactive session in progress

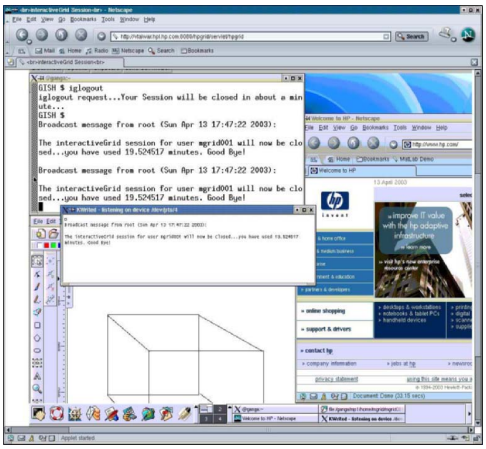

Figure 12. Screen at a session logout