Abstract

“Grid” computing has emerged as an important new field, distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications, and, in some cases, high-performance orientation.[1] Managing resources on the grid is extremely crucial for effective resource utilization. This paper presents an approach to resource monitoring and publishing using an information provider.

1 Introduction

Grid Monitoring Service is the activity of measuring significant grid resources related parameters in order to analyze usage, behavior and performance of the grid. It is also used to detect and notify fault situations contract violations (SLA) and user-defined events. Therefore, it is an essential part of the grid management activity.

The components of a grid monitoring service are:

1. Measurement Service: This service probes the resources for certain parameters (especially QoS related)
2. Discovery Service: This service finds out which resources are currently available and populates them on the grid.
3. Detection and Notification Service: This service deals with fault situations, SLA violations and user-defined events.
4. Data Analyzer: It prepares performance and usage reports and statistics
5. Presentation Service: It is the web-based graphic user interface used for presentation of the data.

Before I explain the architecture and implementation of my project I will briefly describe the external components used to build my system: Monitoring and Discovery Service(MDS) and the Ganglia Monitoring Cluster Toolkit.

2.1 Monitoring and Discovery Service(MDS)

Most of the development surrounding the information domain of Grid Computing has been related to Monitoring and Discovery Service (MDS) of the Globus Toolkit. MDS is designed to provide a standard mechanism for publishing and discovering resource status and configuration information. It provides a uniform, flexible interface to data collected by lower-level information providers. It has a decentralized structure that allows it to scale, and it can handle static or dynamic data. MDS includes a set of core GRIS information providers used to generate information such as platform type, host OS, system load, memory, file systems, etc. A user can also create an information provider to publish information to the MDS. However, the MDS does not run on individual nodes. It only provides aggregate cluster monitoring data. (Figure 1)

![Figure 1: Architecture of MDS](image-url)
3 Design of the system

The objectives of my design were to come up with a monitoring system, which effectively represents the data. It was also essential to ensure that there are no performance overheads caused due to the monitoring of resources.

The architecture of my project can be described with the aid of Figure 3. As the figure shows clearly my project comprises of two sections:

1. An information provider which publishes information from the Ganglia to the MDS
2. An MDS Browser (presentation layer), which provides a web interface.

3.1 Information Provider

The essence of creating an information provider involves three basic steps:

1. Decide what kind of information needs to be published into MDS. Then decide how that information should be represented in the Directory Information Tree (DIT). This requires the definition of a schema, an OID assignment [3], and naming conventions for that information. I decided to use the GLUE-CE schema which aims to provide interoperability between grid project efforts [4]. Only those metrics were chosen to be published which were crucial for performance monitoring and scheduling.

2. Create a program that adheres to the input and output interface requirements of LDAP. The program must be callable by the fork() and exec() facilities of the GRIS back end, and must return LDAP Data Interchange Format (LDIF) data objects according to the defined schema. I decided to write and information provider using C.

3. Enable the program by adding an entry for it in the grid-info-resource-ldif.conf file. The grid-info-resource-ldif.conf file defines all active GRIS providers, so when the provider is created a reference to it needs to be added to this file.

Fundamentally, an information provider is any computer program written such that it adheres to two specific
interfaces: the input and output interfaces of the GRIS back end (as shown in Figure 4).

3.1.1 Define Provider Schema

For this purpose the widely accepted GLUE-CE schema was used. Since it contains all the necessary attributes required it just needed to be mapped to the corresponding Ganglia metric (as shown in Table 1).

3.1.2 Create a Provider Program

The purpose of a user-created information provider is to get data from the Ganglia monitoring daemon (gmond) and put the data as an LDIF object. (see Figure 5)

Information Provider Input

The input for the information provider comes from the gmond. Gmond is a multi-threaded daemon which runs on each node of the cluster that needs to be monitored. Gmond has four main responsibilities: monitor changes in host state, multicast relevant changes, listen to the state of all other ganglia nodes via a multicast channel and answer requests for an XML description of the cluster state. This XML provides the input for the information provider and it can be used to publish it on the MDS.

<table>
<thead>
<tr>
<th>Common Name</th>
<th>Attribute (OID)</th>
<th>objectclass (OID)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hostname</td>
<td>GlueHostName 1.3.6.1.4.1.8005.100.3.2.3.1</td>
<td>GlueHost 1.3.6.1.4.1.8005.100.3.1.3</td>
</tr>
<tr>
<td>A Unique ID assigned to the host</td>
<td>GlueHostUniqueID 1.3.6.1.4.1.8005.100.3.2.3.2</td>
<td>GlueHost 1.3.6.1.4.1.8005.100.3.1.3</td>
</tr>
<tr>
<td>Processor Load, 1 Min Average</td>
<td>GlueHostProcessorLoadLast1Min 1.3.6.1.4.1.8005.100.3.2.10.1</td>
<td>GlueHostProcessorLoad 1.3.6.1.4.1.8005.100.3.1.10</td>
</tr>
<tr>
<td>Processor Load, 5 Min Average</td>
<td>GlueHostProcessorLoadLast5Min 1.3.6.1.4.1.8005.100.3.2.10.2</td>
<td>GlueHostProcessorLoad 1.3.6.1.4.1.8005.100.3.1.10</td>
</tr>
<tr>
<td>Processor Load, 15 Min Average</td>
<td>GlueHostProcessorLoadLast15Min 1.3.6.1.4.1.8005.100.3.2.10.3</td>
<td>GlueHostProcessorLoad 1.3.6.1.4.1.8005.100.3.1.10</td>
</tr>
<tr>
<td>SMP Load, 1 Min Average</td>
<td>GlueHostSMPLoadLast1Min 1.3.6.1.4.1.8005.100.3.2.11.1</td>
<td>GlueHostSMPLoad 1.3.6.1.4.1.8005.100.3.1.11</td>
</tr>
<tr>
<td>SMP Load, 5 Min Average</td>
<td>GlueHostSMPLoadLast5Min 1.3.6.1.4.1.8005.100.3.2.11.2</td>
<td>GlueHostSMPLoad 1.3.6.1.4.1.8005.100.3.1.11</td>
</tr>
<tr>
<td>SMP Load, 15 Min Average</td>
<td>GlueHostSMPLoadLast15Min 1.3.6.1.4.1.8005.100.3.2.11.3</td>
<td>GlueHostSMPLoad 1.3.6.1.4.1.8005.100.3.1.11</td>
</tr>
<tr>
<td>Number of CPUs</td>
<td>GlueHostArchitectureSMPSize 1.3.6.1.4.1.8005.100.3.2.4.2</td>
<td>GlueHostArchitecture 1.3.6.1.4.1.8005.100.3.1.4</td>
</tr>
<tr>
<td>Processor Clock Speed (MHz)</td>
<td>GlueHostProcessorClockSpeed 1.3.6.1.4.1.8005.100.3.2.5.4</td>
<td>GlueHostProcessor 1.3.6.1.4.1.8005.100.3.1.5</td>
</tr>
<tr>
<td>The name of the Network Interface</td>
<td>GlueHostNetworkAdapterName 1.3.6.1.4.1.8005.100.3.2.9.1</td>
<td>GlueHostNetworkAdapter 1.3.6.1.4.1.8005.100.3.1.9</td>
</tr>
<tr>
<td>Network Adapter IP address</td>
<td>GlueHostNetworkAdapterIPAddress 1.3.6.1.4.1.8005.100.3.2.9.2</td>
<td>GlueHostNetworkAdapter 1.3.6.1.4.1.8005.100.3.1.9</td>
</tr>
<tr>
<td>The amount of RAM</td>
<td>GlueHostMainMemoryRAMSize 1.3.6.1.4.1.8005.100.3.2.7.1</td>
<td>GlueHostMainMemory 1.3.6.1.4.1.8005.100.3.1.7</td>
</tr>
<tr>
<td>Free RAM (in KBytes)</td>
<td>GlueHostMainMemoryRAMAvailable 1.3.6.1.4.1.8005.100.3.2.7.2</td>
<td>GlueHostMainMemory 1.3.6.1.4.1.8005.100.3.1.7</td>
</tr>
</tbody>
</table>

Table 1: GLUE-CE schema mapping for Ganglia metrics
**Information Provider Output**

A user-created information provider must generate LDIF objects as its output. LDIF is a file format suitable for describing directory information or modifications made to directory information. LDIF is typically used to import and export directory information between LDAP-based directory servers, or to describe a set of changes that are to be applied to a directory. So in order to publish in on the MDS the XML data needs to be transformed to LDIF objects.

**3.1.2 Enable the Provider**

The grid-info-resource-ldif.conf file lists active objects, which show how to invoke an information provider. The created objects need to be added to grid-info-resource-ldif.conf. The GRIS back end reads grid-info-resource-ldif.conf to get the path name (path: and base: parameters) and the arguments (args: parameter) of the information provider. Then the GRIS back end forks and execs the information provider.

**3.2 MDS Browser – Presentation Layer**

In order to represent the hierarchical data a Java based directory browser was developed. Since a web interface is essential for resource monitoring an applet was programmed. Such a system is not only easy to visualize but also simple to browse and effectively represents the hierarchy of resources in a grid.

![Diagram](image)

**Figure 5: Functional Overview of the system**

**4 Implementation**

In order to implement such a system, it required a study of not only the design of the Ganglia Cluster Toolkit, but also the MDS and LDAP specifications.

**4.1 Information Provider**

I made use of the open source C Ganglia Library to implement the information provider. The first thing to do was to open a TCP connection to the gmond (default port: 8649) and get the XML data (see Figure 6). In order to parse this XML I made use of the Expat parser, a very powerful and fast stream parser [5]. The parsed data was then stored in a hash table, which...
could be searched easily for entries. After the data had been stored in the internal data structure, it needed to be converted to LDIF. This LDIF was required to pertain to the GLUE-CE schema and once the information provider was enabled it could publish data on the MDS. Since MDS serves as a common framework for information services this data can be utilized by any of the processes for scheduling, fault tolerance etc.

### 4.2 MDS Browser

The simple MDS browser was designed using the Netscape Directory SDK[7]. The idea was to query the MDS using the SDK and displaying it in a suitable format using Java Swing Classes. In order to achieve good network performance and fast response times the applet was carefully designed. It was made sure that the query for the LDAP data was executed only when a user clicked on one of the nodes of the tree. The whole tree data was not populated at the initial load time itself. Initially, there were some problems with network connectivity and applets.

One of the problems faced during the design of the applet was that it needed network connectivity. Typically an applet cannot have access to the network due to security considerations imposed by the Java sandbox security framework. In order to overcome this problem I had to create a digital certificate for the applet. This was done using the jarsigner and the keytool, which are a part of the J2SDK.

### 5 Future Works

Next-generation information architecture would use Open Grid Services Architecture mechanisms. With the launch of Globus Toolkit 3 (GTK3) integrated monitoring & discovery architecture will be interesting to explore. Information services in the future will be mostly XML-based and the emphasis will shift from host-level concept to service-level concept.

I intend to study the information model in greater detail with special emphasis on web services and standardization of protocols. Another possible extension could be the graphical representation of historical monitoring data. MapCenter, developed by DataGrid represents an interesting approach to implementation of such a system [7].

All said and done, one thing is certain- if grid technologies have to be applicable in today’s business world, they will need to be fault tolerant and robust. I believe effective resource monitoring and fault reporting systems can go a long way to solve many of the problems that present grid applications suffer from.

### References


