B 534 DISTRIBUTED SYSTEMS

A Survey Paper on Grid Information Systems

Anand Hegde
800 North Smith Road
Bloomington
Indiana – 47408
aghegde@indiana.edu

ABSTRACT
Grid computing combines computers from various administrative domains to achieve a common goal in order to solve a single task. Grid facilitates large-scale distributed resource sharing. The Grid consists of information which is both static and dynamic in nature. Hence, the selection and configuration of various algorithms and protocols used by different applications becomes very important. In order to achieve this one needs correct and up-to-date information regarding the state and structure of available resources.

A Metacomputing Directory Service provides access to dynamic and distributed information about the state and structure of available resources.

This paper provides a brief description of Grid Information Systems concentrating mainly on Globus Metacomputing Directory Service (MDS) and the usage of Lightweight Directory Access Protocol (LDAP) in the Globus Metacomputing toolkit. The paper discusses various aspects of MDS such as design issues and implementation of MDS with Globus. The paper also discusses where and why LDAP is used with Globus MDS toolkit.

1. INTRODUCTION
As mentioned in the abstract, selection and configuration of protocols and algorithms is needed in grid/distributed computing. This problem does not arise in traditional distributed computing where we can use default protocols and manual configuration. Manual configuration cannot be used in grid computing because it would demand in depth knowledge of remote systems which is not possessed by an average programmer. Hence, we need a dynamic configuration approach where decisions are taken solely based on the structure and state of available resources.

In order to implement such a dynamic configuration approach, one needs to have all the right information. The paper discusses one such approach, Metacomputing Directory Service (MDS) which provides a uniform and consistent interface to varied information sources. The paper discusses how Lightweight Directory Access Protocol (LDAP) helps in achieving the desired goal. In order to understand the implementation, we need to understand what a directory, directory service and LDAP are.

A directory is used to store and retrieve information. They can be related to databases. They offer a static view of data and updates in directories are simple without complex transactions.
Now, a directory service provides a directory that can be accessed by some network protocol. They usually consist of mechanisms to handle data distribution and replication. Domain Name System (DNS) is an example of a directory service.

Lastly, LDAP is a standard directory protocol. LDAP is a vendor and platform independent network protocol standard. Hence in a metacomputing system, LDAP can be ported transparently across huge and complex networks. LDAP also supports various authentication, integrity and privacy technologies.

The paper majorly discusses various design aspects of a directory service such as addressing issues of data representation, data model and implementation of a new architecture for high performance grid computing.

2. DESIGN ISSUES
Following are the requirements for the successful implementation of an information infrastructure for a distributed computing system. Some of the requirements are subjective while some are quantitative.

**Performance:** Caching is necessary because an information infrastructure must allow quick access to frequently used configuration details.

**Scalability and Cost:** The cost should be low and at the same time the infrastructure should scale to large number of components. It should also allow simultaneous access by different entities.

**Uniformity:** A uniform data model along with an API is required to perform common services on the data represented by the model.

**Expressiveness:** The infrastructure should be robust enough to represent all the relevant structures within the high performance distributed system.

**Extensibility:** No data model is going to be perfect for all applications. Hence it is important to be able to make necessary changes to improve the data model.

**Multiple information sources:** The information interface must be able to integrate different information from various sources.

**Dynamic data:** The information infrastructure should be able to provide dynamic data such as network load whenever required.

**Flexible access:** One should be able to read, update and search from within an information infrastructure.

**Security:** It is necessary to have control over who is accessing and updating configuration data.

**Decentralized maintenance:** Delegation of tasks such as creating and maintaining data regarding various resources must be possible.

3. METACOMPUTING DIRECTORY SERVICE
After analyzing all the design issues and various requirements a Metacomputing Directory Service (MDS) can be defined to be a systems consisting of the following components:

**Representation and data access:** The directory structure, data representation and API defined by LDAP.
**Data model:** A data model that is robust enough to encode details of types of resources found in super computers.

**Implementation:** A collection of steps in order to meet the desired requirements.

The above diagram describes the overview of the MDS architecture.

### 3.1 REPRESENTATION

The design of MDS is based on the API and data representation defined by the LDAP directory service because the LDAP data representation is extensible and flexible. Since LDAP is widely used one can expect wide usage and deployment of LDAP services. LDAP implementation does not restrict us from using the standard implementation techniques but as described earlier LDAP gives a better interface to work with.

In the MDS representation, information are carefully clubbed and organized into entries. Each entry describes the instance of some object such as a person, a network or a computer. Each entry is further described by its attribute. The association of an attribute to an entry is based on the type of object the entry represents.

Now, each MDS entry has a unique name known as distinguished name. These entries are organized in a hierarchical tree structure known as Directory Information Tree (DIT).

This simplifies the task of locating an entry in the MDS. The distinguished name of an entry is based on the path from the DIT root to the desired entry.

Each part of the path that defines the distinguished name must identify a specific entry in the DIT. Therefore, all the children of a DIT entry must have a unique attribute in order to differentiate it from its siblings. Thus an entry can be uniquely named based on the list of attribute names and values that identify its ancestors up to the DIT root.

Example of a MDS distinguished name is as follows:

```
<
HN = cold.mcs.anl.gov,
OU = MCS,
O   = National Laboratory,
O   = Globus,
C   = US
>
```

Where the distinguishing attributes are,

- **HN** = Host Name
- **OU** = Organizational Unit
- **O** = Organization
- **C** = Country

Thus it is clear that the country entry is at the root of the DIT and the host entries are beneath the organizational level of the DIT.

Now, each DIT entry has its own type known as object class. Details of the attributes of an entry are described by the object class. The object class consists of three major parts, the parent class, list of essential attributes and the list of optional attributes.

A simple inheritance mechanism is allowed by the SUBCLASS section of the object class. This allows the definition of an object
class in terms of an existing object class. The MUST CONTAIN section consists of the list of essential attributes found in an entry. The list of optional attributes are present in the MAY CONTAIN section of the object class. Attributes can be either case-insensitive strings (cis) or distinguished names (dn).

The DIT should be considered as a separate structure which is used to organize different entries. DIT also facilitates in distributing these entries over many physical sites.

### 3.2 DATA MODEL

A data model consisting of relevant information must be created in order to use the MDS representation for a particular purpose. The data model must consist of the DIT hierarchy and the various object class definitions. The data model is not network centric but it computer centric.

Entries corresponding to computers and people are placed under the organizational units. For e.g. as mentioned earlier organization and organizational unit are placed under the root node country.

Proper representation of computers and networks is very essential. The representation of computers and network should be such that it should be able to answer questions like whether two computers are on the same network, details of latency between two computers and the list of protocols available between two computers.

As described earlier regarding the DIT structure, both people and host are direct children of the organizations in which they are located. In the DIT structure, even the computational networks are represented as children of the organization.

Let us take an example of the following distinguished name,

```
<
NN = mcs-lan,
OU = MCS,
O    = National Laboratory,
O    = Globus,
C    = US
>
```

The above distinguished name represents a Local Area Network in MCS. This distinguished name defines an instance of type GlobusNetwork object. The attribute value of this object gives us details of the physical network link such as the network topology, physical media and details of the link protocols used.

The GlobusNetworkImage object consists of logical information such which network protocol is being used. Such information is not available in the GlobusNetwork object.

### 3.3 IMPLEMENTATION

As mentioned earlier, one can answer questions about system configuration based on the information represented by the MDS. Standard LDAP implementation is adopted because the data model is defined on the basis of the LDAP framework. TCP based wire protocol is used along with a distributed collection of servers for the implementation of MDS. Now, each server is responsible for all the entries located within a sub tree of the DIT. This approach works for a distributed environment which is loosely coupled but has three major disadvantages when applied to a high performance environment. They are as follows:

**Single information provider:** The LDAP implementation considers that all the information present in the DIT sub tree is
provided by a single provider. The design of the MDS data model becomes very complex if information is assumed to be provided by a single provider. For e.g. the IP address details can be provided by a system call whereas the available network bandwidth can be provided by Network Weather Service (NWS).

**Client and server architecture:** The implementation using LDAP requires at least one round trip network communication for each of the LDAP access. Hence, regular and frequent access to MDS becomes very expensive. Hence a mechanism where the MDS details can be cached will be favorable.

**Scope of data:** The implementation using LDAP assumes that any piece of data can be used from any point in the network. A more efficient implementation can be obtained by restricting the locations from which attributes can be accessed in order to perform an attribute update. Things like which information must be propagated to which information provider and when to cache the information can be determined with the help of scope.

All the above disadvantages or drawbacks are associated with the LDAP implementation and has nothing to do with its API. The LDAP API can be used for MDS without any modification. However, MDS needs a customized and specialized version of LDAP API in order to meet the requirements of high performance computing.

In the above implementation technique, information providers are specified on a per attribute basis which is not the case in standard implementation. The object class metadata entry consists of which protocol to use for each entry attribute. These Metadata entries are stored in MDS and are accessed via LDAP protocol.

The object class metadata also consists of a time to live (TTL) for attribute values. The TTL data is used to enable caching. The initial implementation considers three update scopes: process, computation and global. Process scope attributes are available only within the same process as the writer. Computation scope attributes are accessible by any process within a single computation. Attributes of type global scope can be accessed by any node or process on the network.

### 4. MDS APPLICATION IN GLOBUS

MDS information can be used in high performance computing. The Globus toolkit provides many service mechanisms such as communication, authentication, resource location, resource allocation, process management and information infrastructure in the form of MDS. The Globus toolkit is designed with the configuration problem discussed earlier in mind.

For each component the Globus toolkit tries to provide an interface which allows higher level services to manage lower level services. As an example we can take the problem of selecting network interfaces and communication protocol when executing communication code within a network.

The Globus communication module allows a user to specify the communication operations of the application by using a single notation. The Nexus API can be used for this purpose. The Nexus implementation configures a communication structure for the application during run time. It selects a communication method for each of the communication link which is then used for communication over that link. In order to do
this selection Nexus makes use of MDS information. It uses this information to determine which low level mechanisms are available between a pair of processors. Due to this selection process the source code of the application can thus run on many environments without making any changes.

Resource allocation is one more application that uses MDS information. “Resource broker” can be defined as a process that supports specialized searches against the MDS implementation. Instead of loading the MDS servers by incorporating these search capabilities in the MDS server, one can construct a broker that constructs and maintains the required indexes which obtain the required data by just querying the MDS from time to time.

5. CONCLUSION
We can conclude that for high performance computing, we need a dynamic approach for configuration of the systems involved. In the approach discussed in the paper, configuration choices are based on the information gathered from external sources rather than the information gathered from the programmers.

MDS is designed to provide uniform, efficient and dynamic access to information about the structure and state of the available resources. MDS defines a representation based on LDAP, a data model and an implementation. Various experiments conducted with the Globus toolkit show that MDS is very useful in practical situations.

6. RESOURCES
4. Usage of LDAP in Globus - Gregor von Laszewski and Ian Foster Mathematics and Computer Science Division Argonne National Laboratory, Argonne, IL 60439
5. A Novel Architecture for Grid Information Systems - Zoltán Balaton, Gabor Gombás, Zsolt Németh MTA SZTAKI Computer and Automation Research Institute P.O. Box 63., H-1518 Hungary
6. A Directory Service for Configuring High Performance Distributed Computations – Steven Fitzgerald, Ian Foster et al.