Implementation of OGSA Services in Grid Environment

By

Madhuri Vaghasia

08MCE023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

May 2010

Implementation of OGSA Services in Grid Environment

Major Project

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology in Copmuter Science and Engineering

By

Madhuri Vaghasia

08MCE023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

May 2010

Declaration

This is to certify that

i) The thesis comprises my original work towards the degree of Master of Technology in Computer Science and Engineering at Nirma University and has not been submitted elsewhere for a degree.

ii) Due acknowledgement has been made in the text to all other material used.

Madhuri Vaghasia

Certificate

This is to certify that the Major Project entitled **"Implementation of OGSA Services in Grid Environment"** submitted by **Madhuri Vaghasia** (08MCE023), towards the partial fulfillment of the requirements for the degree of Master of Technology in Computer Science and Engineering of Nirma University of Science and Technology, Ahmedabad is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Prof. Madhuri Bhavsar
Guide,Sr.Associate Professor,
Department of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. S. N. Pradhan
Professor,P.G. Coorinator,
Department Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad.

Prof. D. J. Patel
Professor and Head,
Department of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. K. Kotecha
Director,
Institute of Technology,
Nirma University, Ahmedabad.

Abstract

Grid systems and applications aim to integrate, virtualize, and manage resources and services within distributed, heterogeneous, dynamic, virtual environment. Here a service-oriented architecture, Open Grid Services Architecture (OGSA) vision of a broadly applicable and adopted framework for distributed system integration, virtualization, and management requires the definition of a core set of interfaces, behaviors, resource models, and bindings.

This thesis contains information about main features of OGSA,implementation of service Provisioning Directory Service which exposes the local file system's directory hierarchy to remote grid clients. It allows listing the contents of the current working directory and change the current working directory.

Other grid service Migration Service which is responsible for coordinating the migration of a selected service from one host at a service provider to either another host at the same service provider. To address these challenges,a migration facility is proposed, as part of automatic and transparent brokerage, focused on migrating Grid services. Such a facility addresses reconfiguration by providing mechanisms to migrate a service from one host to another automatically as the grid changes. A migration facility addresses reliability by providing mechanisms to move a service from a host which may be in the process of failing, or which must be taken offline. Finally, a migration facility addresses optimization. Services can be migrated to faster hosts as they become available or join the grid, improving the overall performance of a Grid service. Results are analyzed accordingly.

Acknowledgements

I am deeply indebted to my thesis supervisor for his constant guidance and motivation. She has devoted significant amount of her valuable time to plan and discuss the thesis work. Without her experience and insights, it would have been very difficult to do quality work.

I would also like to extend my gratitude to **Prof. Madhuri Bhavsar** for fruitful discussions during Modeling and Analysis meetings and for her encouragement.

I would certainly like to thank **Dr. S. N. Pradhan**, PG-Coordinator, Computer Engineering Department, Institute of Technology, Ahmedabad for his constant encouragement and motivation throughout the course of the project.

I am also thankful to **Dr.Ketan Kotecha**,Director,Institute of Technology for his kind support in all respect during my study.

Last, but not the least, no words are enough to acknowledge constant support and sacrifices of my family members and friends because of whom I am able to complete the degree program successfully.

- Madhuri Vaghasia
08MCE023

Contents

List of Figures

List of Tables

Abbreviation

OGSA Open Grid Services Architecture

OGSI Open Grid Services Infrastructure

WSRF Web Services Resource Framework

JSDL Job Submission Description Language

GSI Grid Security Infrastructure

GRAM Grid Resource Allocation & Management Protocol

GASS Global Access to Secondary Storage

WSN Web Service Notification

MDS Monitoring and Discovery Service

GGF Global Grid Forum

NTP Network Time Protocol

GSH Grid Service Handle

GSR Grid Service Reference

GAR Grid Archive

GSBT Globus Service Build Tool

WSDD Web Service Deployment Descriptor

GT4 WS-Core Globus Toolkit4 Web Service-Core

SMB System Management Broker

Chapter 1

Introduction To Grid Services

1.1 What is Grid?

The Grid is a conceptually simple idea and yet complex to implement. The aim is to be able to utilize computing resources wherever they are and in whatever form.

The simplest view can be depicted in the visionary phrase 'Boundaryless Computing'. That is, transparent access to computing power, irrespective or its location.

1.2 What is Grid Computing?

A grid is a system that

a. Coordinates resources that are not subject to centralized control Grid integrates and coordinates resources and users that exist within different control domains.

b. Uses standard open, general purpose protocols and interfaces : A grid is built from multipurpose protocols and interfaces that address such issues like authentication, authorization and resource discovery.

c. To deliver non-trivial qualities of services: A grid allows its constituent resources to be used in a coordinated fashion to provide various qualities of service like response time, throughput etc.

Grid computing (or the use of computational grids) is the combination of computer resources from multiple administrative domains applied to a common task, usually to

a scientific, technical or business problem that requires a great number of computer processing cycles or the need to process large amounts of data.

One of the main strategies of grid computing is using software to divide and apportion pieces of a program among several computers, sometimes up to many thousands. Grid computing is distributed, large-scale cluster computing, as well as a form of network-distributed parallel processing.

1.3 Introduction to our Globus Grid

1.3.1 What is GLOBUS?

The Globus Toolkit, currently at version 4, is an open source toolkit for building computing grids developed and it is provided by the Globus Alliance.

The Globus Toolkit is an implementation of the following standards:

- Open Grid Services Architecture (OGSA)
- Open Grid Services Infrastructure (OGSI) - originally intended to form the basic plumbing layer for OGSA, but has been superseded by WSRF and WS-Management.
- Web Services Resource Framework (WSRF)
- Job Submission Description Language (JSDL)
- WS-Management
- SOAP
- WSDL
- Grid Security Infrastructure (GSI)
- Resource management: Grid Resource Allocation & Management Protocol (GRAM)

- Information Services: Monitoring and Discovery Service (MDS)
- Security Services: Grid Security Infrastructure (GSI)
- Data Movement and Management: Global Access to Secondary Storage (GASS) and GridFTP

1.3.2 Configuration Details

* Operating System : Fedora Core 8.0

* Middleware : Globus 4.2.1

* Machines : Machines having following configuration

Node	IP Address	Host Name
A	10.1.3.14	nodeA.grid.nirma.com
B	10.1.3.13	nodeB.grid.nirma.com
C	10.1.3.19	nodeC.grid.nirma.com

1.4 What is OGSA?

"Open Grid Service Architecture" (OGSA) is the industry blueprint for standards-based grid computing. "Open" refers to both the standards development process and the standards themselves. OGSA is "service-oriented" because it delivers functionality among loosely-coupled interacting services that are aligned with industry-accepted Web service standards. "Architecture" defines the components, their organizations and interactions, and the overall design philosophy.[1]

What are the key architectural areas associated with OGSA?
OGSA specifies eight categories of services, each essential to coordinating the work of applications that interact with available resources in a shared, secure environment. End users require different subsets of these OGSA services to satisfy the requirements

of their particular use cases, enabling modularity and deployment flexibility.

The eight high-level categories of services specified by OGSA are:

- **Infrastructure Services:-** enable communication between disparate resources (computer, storage, applications,etc.), removing barriers associated with shared utilization.
- **Resource Management Services:-**enable the monitoring, reservation, deployment, and configuration of grid resources based on quality of service requirements.
- **Data Services:-** enable the movement of data where it is needed - managing replicated copies, query execution and updates, and transforming data into new formats if required.
- **Context Services:-**describe the required resources and usage policies for each customer that utilizes the grid-enabling resource optimization based on service requirements.
- **Information Services:-**provide efficient production of, and access to, information about the grid and its resources, including status and availability of a particular resource.
- **Self-Management Services:-**support the attainment of stated levels of service with as much automation as possible, to reduce the costs and complexity of managing the system.
- **Security Services:-**enforce security policies within a virtual organization, promoting safe resource-sharing and appropriate authentication and authorization of users.
- **Execution Management Services:-**enable both simple and more complex workflow actions to be executed,including placement, provisioning, and management of the task lifecycle.

1.5 List of OGSA Services

- reservation, brokering and scheduling
- installation, deployment and provisioning
- metering, accounting
- aggregation, migration
- VO management
- security management
- monitoring (performance, availability, etc.)
- control (start, stop, etc.)
- problem determination and fault management

1.6 Why use Grid Services?

The OGSI GGF recommendation introduced Grid services, built as an extension to Web services, offering benefits for data access and integration services over non-OGSI Web Services solutions. These benefits include:

$ **Service Data Elements:** allow the state of a service to be exposed and provide a standard interface for getting and setting service properties, for example whether a database query is in progress or complete. Higher-level services can then use these to query the underlying database state, e.g. a DQP service could find out the database schema or the server load through the service's SDEs.

$ **Dynamic Service Creation:** allows service instances to be created on demand according to need. Transient service instances are useful for managing aspects of an interaction, e.g. to provide context for a database connection, or to monitor a computing job.

$ **Lifetime Management:** can be used to ensure that services that are no longer required are cleaned up (so resources are not wasted) as well as providing lifetimes for notification subscriptions, cache lifetimes and other time-constrained services and resources.

$ **Service Groups:** facilitate the provision of registries that can be used by clients to identify service instances meeting application-specific requirements. This allows the assignment of unique global ids for data resources, location independence and ultimately true virtualisation of data.

1.7 OGSA Mechanisms for Services

- **Inteoperability**
  The freedom of implementation of services based on native platform facilities and the global rule for grid services to abide by uniform grid service interface pattern thus solve the problem of interoperability.

- **Discovery and Access of Resource**
  The requirement of discovering and accessing the resources was addressed in three parts in OGSA:
  - A standard representation of service data, containing information about grid service instance and represented in XML structures.
  - A standard operation, FindServiceData, to retrieve service data from individual grid service instances.
  - A standard interface registry, for registering the information about the grid service instances.

- **Independent Upgradability**
  Upgradability of grid service instances and maintenance of versioning information was addressed by defining OGSA mechanisms,to refresh the client's knowledge of service, such as an upgrade of services supported, an upgrade to a host platform or any other domain specific upgrade details applicable to the

client. The service description includes protocol-binding properties that will be used to communicate with the service. Two properties are often needed in such communications: reliable service invocation and authentication.

- **Transient life cycle management of resources**
  In a dynamic environment, services are created and need to be destroyed when no longer needed. The grid services architecture addresses this life cycle issue through a soft state approach, where grid services are created with a specified initial lifetime. The initial lifetime can be extended by a specified time period by explicit request of the client service or by another service, which has credential delegations of the client. In this architecture, client will be sending "Keep live" messages to keep the service active in a providers system. If the client does not need the service, it stops sending "keep alive" messages. This situation can also be due to the breakdown of the software system at the clients end or any component failure in the grid services workflow.

  The initial creation of service allows the client to send maximum and minimum acceptable expiration times for a service. If the provider agrees to a customer request and can provide a service whose expiration time falls between the maximum and minimum, then the factory service instance creates the service for the client. The clock synchronization used here for the process of determining time is based on Network Time Protocol (NTP). Using this protocol, time is synchronized between all grid services to an accuracy of tens of milliseconds.

- **Services state - grid service handle and reference**
  The state nature of every service actually changes throughout its lifetime, so it is necessary to have some process that manages these states. OGSA addresses this requirement by defining Service Data Elements that will store the service states and maintain them until the end of the service lifetime. The stored service state will be accessible via the grid service interfaces defined in OGSA. The change in service states will be passed on to co-services in the grid by asynchronous state change notification.

As the service instance is created, a unique service instance identifier is allocated to it, called Grid Service Handle (GSH). This is invariable and unique to the service instance even over a time period. If a co-service wants to re-start the terminated service instance and regain control of some resources in the client system, it can do so using a GSH, since it remains unique even over time.

There is also more information about that specific service stored by the Grid Service Reference (GSR). In a different way than GSH, (GSR) may vary over time for a single service instance over the lifetime of the service and has a set termination time after which it expires. For example, the versioning information of the grid service and protocol binding information to the grid service are not carried by GSH; instead, GSR maintains it.OGSA also defines mapping mechanisms for obtaining the updated GSR. The result of using an expired GSR is undefined.

- **Factory**
  OGSA defines an interface called Factory for creating new grid service instances. The Factory interface receives requests from client services and responds with a GSH and initial GSR after successfully creating the service instance. As complexity increases in the grid, from simple hosting environments to collective virtual hosting environments, factories will have multiple levels, for instance higher and lower. Higher-level factories will delegate work to one or many of the multiple lower level factories under its reign, to accomplish a specific task.

- **Dynamic resolution of transient references from permanent handles**
  After the generation of GSH and initial GSR by the factory, OGSA defines a dynamic way to resolve the references between GSR and GSH through a new interface called HandleMap. This allows the client service to identify the new GSR. This interface maintains the latest mapping between the Handles (GSH) and References (GSR). The handle map interface will not return references to

service instances that it knows have terminated.

The handle map interface will return the most recent and valid GSR. To identify the handle map interface, GSH will have the URL of the (home) HandleMap interface included in it. Thus, once GSH is obtained, GSR can be obtained by contacting the handle map interface.

We contact the HandleMap with the use of HTTP GET operation,to speak to the HandleMap interface using the supplied GSH. In return, HandleMap returns GSR for the GSH requested in WSDL format.

- **Service data element and registry interface**
  Every service instance in grid technology has its own unique information needed later during its lifetime (such as time-to-live information of the service, GSH, GSR, HandleMap, etc.). All these data elements are nothing but XML elements contained in a single wrapper called Service Data Element.

  Using the FindServiceData, a WSDL operation, one can retrieve the Service Data Elements for a particular service instance. But, for all this to happen,Service Data Element needs to be stored and maintained in a specific place. Hence the need for Registry Interface, which provides the service of supplying Service Data Elements for a given service instance.

- **Asynchronous notification of state changes**
  The changes to the state of the service will take place throughout the existence of the services lifetime. The co-services and services that are authorized to received and process information related to grid service state change will subscribe to the grid service for notification (source/sink).

  The OGSA framework addresses notification services in two parts. One is the Sender of notification, called the notification source, which will implement the

notification source interface to publish and receive subscriptions to its notification messages. The second part is the receiver of notification messages, called the notification receiver, which will use the notification sink interface to receive the notification message.

1.8 Major Feature of Grid Services

- **Factory**
  In object-oriented programming terms, a factory is used to create instances of a class. The factory is also used to isolate the creation of objects of a particular class into a single place so that new features or functions can be added without widespread code changes.

  In a grid context, a factory creates service instances and has a registry to keep track of those instances and to enable service discovery by clients or other services. Clients typically first locate the factory, and then request the creation of a service instance. On request, a factory creates an instance of a grid service and returns a GSH and a GSR to the client.

  The GSH is a unique identifier and the client uses it to communicate with the service instance. No further communication from the client necessitates the factory and communication is established directly with the service instance.
  A typical client scenario includes the following steps:

  a. Client discovers a factory by querying the registry service

  b. Client calls a factory operation to create an instance of a grid service

  c. Factory creates a new instance of the grid service

  d. Factory returns the GSH of the new grid instance to the client

  e. Client and service interact as result of the initial call

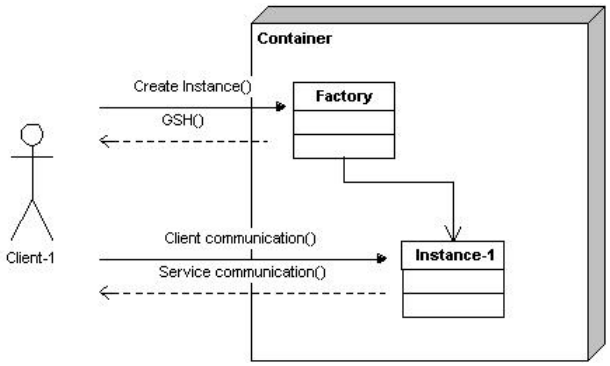

Figure 1.1: Client requests an instance from the factory

- **Service Data Elements**

  The service data is a structured collection of information that is associated with an instance of a grid service to expose a grid service instances state data to service requestors. The service data must be easy to query, so that grid services can be classified and indexed according to their service characteristics.

- **Life Cycle**

  Most entities have a life cycle. This typically refers to the states between the objects creation and destruction. Life cycle management is very important,especially in a robust environment where services should be capable of resuming operations in the event of a server or container restart. In order to be capable of resuming operations after a container restart, services must support checkpoints and persisting of its state information.

  These critical points during the grid service life cycle include:

  a. preCreate - called when a grid service starts the creation process, prior to loading configuration data

  b. postCreate - called when a grid service has been created and loaded its configuration data

  c. activate - called when a grid service is activated or loaded into memory space

d. deactivate - called before a grid service is deactivated or paged out of memory

e. predestroy - called before a grid service is destroyed

To support the requirement call back the service must log its internal state prior to being destroyed. The service must reload the previously saved state during object creation and be capable of resuming operations from that point. Instance deactivation time is also taken into consideration,because whenever instance is idle,it is to be deactivated.

- **Notifications**

  Notifications are a useful mechanism for tracking changes to service data. A party interested in a particular Service Data Element registers to be notified if that value changes. The interested party, to which the notification is sent, is called the notification sink. The service containing the Service Data Element of interest and which generates the notification to interested parties or subscribers is called the notification source.

  Figure illustrates the notification subscription flow of events.An application or grid service is interested in a particular Service Data Element of another grid service (notification source). The notification sink and the subscription service can interact to perform lifetime management tasks.

  a. An interested service subscribes to be notified if a Service Data Element changes. The interested service that will be notified is called the notification sink. The grid service that received the notification subscription is called the notification source.

  b. The notification source creates a subscription manager instance.

  c. The notification source returns the handle of the subscription manager instance to the notification sink.

  d. The notification sink can use the subscription manager handle to manage the subscription lifetime.

e. When the condition specified in the notification subscription is met, a notification message is sent to the notification sink.

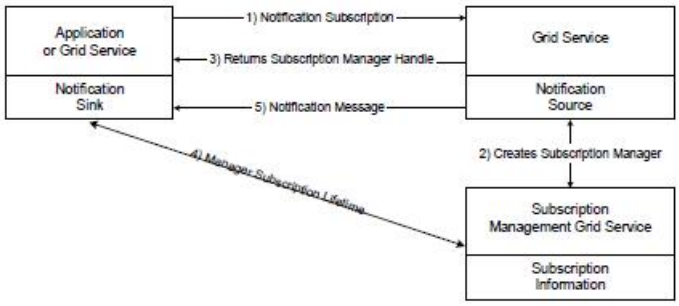

Figure 1.2: Notification Subscription Flow

1.9 Thesis Organization

The rest of the thesis is organized as follows.

Chapter 2, *Literature Survey*,includes literature survey related to grid services.

chapter 3, *Implementation of Provisioning Directory Service*, The five main steps for the grid services creation are presented and through this implementation for the service is done.

Chapter 4, *Implementation of Migration Service*, describes introduction about the service and proposed algorithm. Implementation of service is done migration service interface and finally calculation for the migration time for the known host and host selected by the SMB is done.

chapter 5, concluding remarks for the implemented services is presented.

Appendix A, gives the list of troubleshootings covered during installation of Globus Toolkit.

Appendix B, gives the list of useful web sites.

Chapter 2

Literature Survey

2.1 Migration of Web Services to Grid Services

The North-East Regional e-Science Centre(NEReSC) is involved in a large number of research projects [2] that rely on the design and development of a Grid-based infrastructure. For the migration of Web Services to Grid Services a common requirement as many UK e-Science research projects have built their infrastructure on Web Services and are now considering a move to Grid Services. Therefore, this paper discusses the experience gained by migrating two Web Services to the Core Grid Middleware and making them OGSI compliant.

The core consists of a set of key Grid Services. Each project can adopt the Core Grid Middleware and build its own application specific services on top of it. It will consist of four Grid services running on the Globus OGSI reference implementation. These services, shown in figure, were chosen by analysis of the requirements of the NEReSC Grid projects.

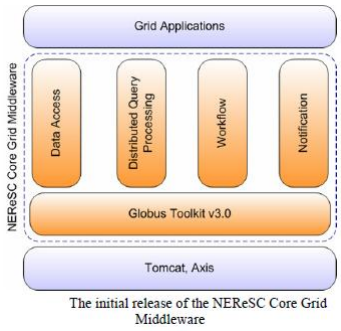

Figure 2.1: Coregrid Architecture

2.2 Grid Services for Distributed System Integration

The Open Grid Services Architecture enables the integration of services and resources across distributed, heterogeneous, dynamic virtual organizations-whether within a single enterprise or extending to external resource-sharing and service-provider relationships.[3]
Grid Services for Distributed System Integration has the following charactristics:
-Service orientation
-Virtualization
-Service semantics: The Grid service
-Role of hosting environments

2.3 The Open Grid Services Architecture, Version 1.5

Successful realization of the Open Grid Services Architecture (OGSA) vision of a broadly applicable and adopted framework for distributed system integration, virtualization, and management requires the definition of a core set of interfaces, behaviors, resource models, and bindings. The document focuses on requirements and the scope of important capabilities required to support Grid systems and applications in both e-

science and e-business. The capabilities described are Execution Management, Data, Resource Management, Security, Self-Management, and Information. The description of each capability includes, to some extent, possible interrelationships with other capabilities. Capabilities are, however, largely independent of each other and there is no requirement that they all be present in an OGSA system.[1]

2.3.1 GT4:software for service-oriented systems

The principal characteristics of recent web-services based GT4 release,which provides significant improvements over previous release in terms of roboustness,performance, usability,documentation,standard compliance and functionality. [4]

2.3.2 Resource Management in OGSA

The issues of management those are specific to a Grid and especially to OGSA. This paper includes Grid management, including both management within the Grid and the management of the Grid infrastructure.[5]
Resource Management in OGSA requirement includes notification,discovery, configuration and collections. All of these topics are critical to management, and must be supported as appropriate within OGSA services.
The following list enumerates the main requirements for management in OGSA. These requirements are especially important in a large-scale, distributed environment with no centralized notion of control, such as a Grid:
-Scalability
-Interoperability
-Security
-Reliability
-Policy
-Performance Monitoring
-Peer-to-Peer Management Requirements

2.4 Application-level Resource Provisioning on the Grid

The algorithms for Grid resource provisioning are presented that employ agreement-based resource management[6]. These algorithms allow user level resource allocation and scheduling of applications that are structured as a precedence constrained set of tasks.In this provisioning model where the resource availability in the Grid can be enumerated as a set of slots. A slot is defined as a number of processors available from a certain start time for a certain duration at a certain cost.

The term resource provisioning to imply creating a contract between the user and the resource owner specifying that a certain resource would be made available to the user for a certain time frame.it allows the user or work flow manager to control the scheduling and execution of the application on the provisioned resources.

2.5 Using Eclipse to develop grid services

GT4 grid services can be tedious to develop because they frequently require the developer to juggle many artifacts (source files, WSDLs, client and server stubs, etc., many of which need to be auto-generated) and configuration steps (various iterations of compilation, linking, deployment, etc.).
Without an integrated development environment (IDE) [7] such as Eclipse, you must switch between many tools (editors, command shells, file managers, build tools, application containers, etc.) while iterating through the development process. With the right plug-ins and configuration, the Eclipse IDE can be used to manage all of these artifacts within a single project abstraction and coordinate all of the useful development activities from coding to deployment to debugging. By embedding the Apache Tomcat Web services container within Eclipse, any updates to the grid service implementation can be immediately reflected in the actively running grid service.

2.6 Service Migration in Autonomic Service Oriented Grids

many problems are still open, e.g., grid reconfiguration, reliability and computing optimization. We argue here that a mechanism that could help solving these problems is Web Service migration, a part of automatic and transparent brokerage. Web service migration presents a number of new requirements not addressed in traditional process migration, being the outcome of Web services specific configuration of hosts and application servers, and availability of Web services/resources state.

The development of a Web service migration facility [8] focused on providing migration of services in a Service Oriented Grid environment. We present a novel approach to Web service migration, embodied in a System Management Broker, which is transparent, interoperable and flexible. We take the requirements of Web services into consideration when discovering suitable destination hosts and match services to suitable grid resources which are able to fulfil the needs of the service. A number of experiments conducted with different types of grid and Web service applications to highlight the feasibility and effectiveness of our migration facility and demonstrate how our facility significantly improves Service Oriented Grids.

2.6.1 A Problem Solving Environment in Grid for GECEM

An application such as Computational Electromagnetics(CEM) is a suitable candidate for use of the Grid. CEM is of increasing importance to the civil and defence sectors. It is central to important problems such as predicting the electromagnetic compatibility between complex electronic systems, and the response of systems to lightning strikes and electromagnetic pulses. These issues are of key concern in possible future platforms such as the More Electric Aircraft(MEA) and the All Electric Ship (AES). The Grid-Enabled Computational Electromagnetics (GECEM) portal shown in figure, and in particular show how the solver code may be migrated [9] on-the-fly to be executed at a remote location.

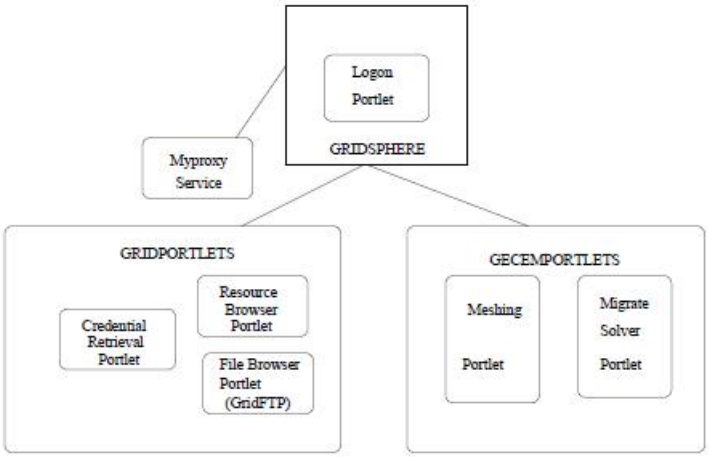

Figure 2.2: GECEM Portal overview

2.6.2 Importance of Migration for Fairness in Online Grid Markets

The intimate connection between job migration and fairness is important here. Computational grids offer users simple access to tremendous computer resources for solving large scale computing problems. Such grids are composed of shared resources owned by different organizational entities. Often in such shared settings each entity (owner) acts both as a provider and as a consumer.

Until recently, only few grids and cluster systems provided preemptive migration[10], which is the ability of dynamically moving computational tasks across machines during runtime. The emerging technology of virtualization becomes an important building block in grids. Virtualization provides off-the-shelf support for virtual machine migration, thus making the use of migration more accessible. It has been shown that migration with zero cost provides better theoretical worst case bounds. Even if the cost of migration is not negligible, preemptive migration is highly beneficial in practice.

Chapter 3

Provisioning Directory Service

3.1 What is Provisioning Directory Services?

This service exposes the local file system's directory hierarchy to remote grid clients. It allow to list the contents of the current working directory and change the current working directory. This WS-Resource consists of a Web service paired with only one resource. The singleton resource in this case is a Java class that keeps the current working directory resource property and returns listings for that directory.

3.2 The Five Main Steps

To create the stateful Grid Service,these are the five main steps[11].

a. **Define the services interface**. This is done with WSDL

b. **Implement the service**. This is done with Java.

c. **Define the deployment parameters**. This is done with WSDD and JNDI

d. **Compile everything and generate a GAR file**. This is done with Ant

e. **Deploy service**. This is also done with a GT4 tool

3.2.1 Step 1: Defining the interface in WSDL

Globus-specific features of WSDL

Following are three Globus-specific features of WSDL

- **Resource properties:**We use the wsrp:ResourceProperties attribute of the portType element to specify what our services resource properties are.The resource properties are where all state information is kept.
- **The WSDL Preprocessor:**The wsdlpp:extends attribute of the PortType element in which we can include existing WSRF portTypes in our own portType. A WSDL Preprocessor will use the value of that attribute to generate correct WSDL which includes our own portType definitions plus any WSRF portType.
- **No bindings:** Bindings are an essential part of a normal WSDL file. However, we dont have to add them manually, since they are generated automatically by a GT4 tool that is called when we build the service.

Namespace mappings

When any one want to refer to this interface from a specific language (like Java) and this is done through a set of stub classes which are generated from the WSDL file using a GT4 tool. A mappings file, maps WSDL namespaces to Java packages.
For example:`http\mskip\medmuskip//www.globus.org/namespaces/examples/core/MathService_instance=org.globus.examples.stubs.MathService_instance`

3.2.2 Step 2: Implementing the service in Java

The QNames interface

The first bit of code we need is a very simple Java interface using its qualified name, or Qname. This is a name which includes a namespace and a local name. For example, the QName of the Value RP is:
`{http://www.globus.org/namespaces/examples/core/MathService_instance}Value`

The service implementation

-To make our service implementation easy we will consist of a single Java class with

the code for both the service and the resource.
-Now, we need to implement the Resource interface. However, this interface doesnt require any methods. It is simply a way of tagging a class as being a resource.
-By implementing the ResourceProperties interface we are indicating that our class has a set of resource properties which we want to make available.

3.2.3 Step 3: Configuring the deployment in WSDD (and JNDI)

This step actually take all the loose pieces that we have written up to this point and make them available through a Grid services container. This step is called the deployment of the Grid service.

The WSDD deployment descriptor
One of the key components of the deployment phase is a file called the deployment descriptor. Its the file that tells the Grid Services container how it should publish our grid service (for example, telling it what the our services URI will be). The deployment descriptor is written in WSDD format.

The service name
This specifies the location where our web service will be found. If we combine this with the base address of our Grid Services container, we will get the full URI of our grid service. For example, if we are using the GT4 standalone container, the base URL will probably be:
`http://localhost:8080/wsrf/services`.
Therefore, our services URI would be:
`http://localhost:8080/wsrf/services/examples/core/first/MathService`

ClassName
This parameter refers to the class which implements the service interface. <parameter name=”className” value=”org.globus.examples.services.core.first.impl.MathService”/>

The WSDL file

<wsdlFile>share/schema/examples/MathService_instance/Math_service.wsdl</wsdlFile>
The wsdlFile tag tells the Grid Services container where the WSDL file for this service can be found. Filename.wsdl will be generated automatically by a GT4 tool when we compile the service.

Load on startup

<parameter name="loadOnStartup" value="true"/>
This parameter allows us to control if we want the service to be loaded as soon as the container is started. Since our service has a single resource, it is usually best to load it at startup.

The common parameters

<parameter name="allowedMethods" value="*"/>
<parameter name="handlerClass" value="org.globus.axis.providers.RPCProvider"/>
<parameter name="scope" value="Application"/>
-These are three parameters which well see in every gird service we program.

The JNDI deployment file

This JNDI file tells the Grid Services container how to present the WSDL interface and implemented java file to outer world.

3.2.4 Step 4: Create a GAR file with Ant

Using these three (1) a service interface in WSDL, (2) a service implementation in Java, and (3) a deployment descriptor in WSDD and JNDI files, a Grid Archive , or GAR file is generated. This GAR file is a single file which contains all the files and information the grid services container needs to deploy our service.
Creating a GAR file involves the following steps:

- Processing the WSDL file to add missing pieces (such as bindings)

- Creating the stub classes from the WSDL
- Compiling the stubs classes
- Compiling the service implementation
- Organize all the files into a very specific directory structure

Ant

Ant is very similar to the classic UNIX make command. It allows programmers to forget about the individual steps involved in obtaining an executable from the source files. We have to be ready with the service interface, the service implementation, and the deployment descriptor. Ant takes care of the rest.

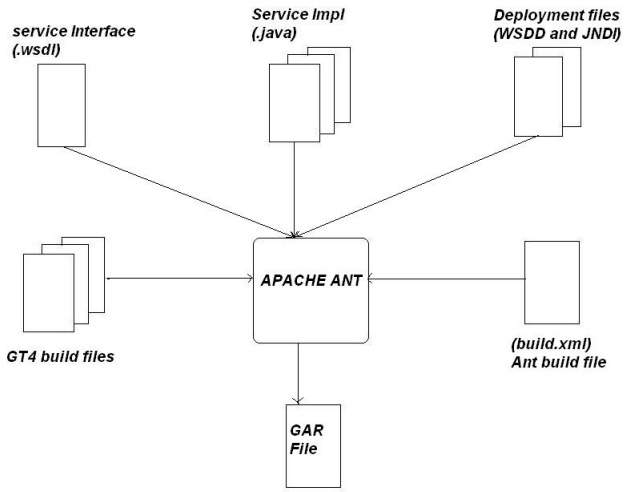

Figure 3.1: Creation of GAR

The globus-build-service script and buildfile

We wont have to write a separate buildfile for each of our services. We will be relying on the globus-build-service script and buildfile, one of the tools developed as part of the Globus Service Build Tools (GSBT) project. This tool will allow us to create a GAR file with minimal effort, and without having to modify an Ant buildfile every

time.

Creating the GAR
Using the provided Ant buildfile and the script, building a web service is as simple as doing the following:
./globus-build-service.sh -d <service base directory> -s <services WSDL file>

-globus-build-service also allows us to use a shorthand notation which is much easier to use. For example, to build our first example and generate its GAR file, we simply need to do the following:
./globus-build-service.sh servicename

3.2.5 Step 5: Deploy the service into a Grid Services container

The GAR file, contains all the files and information the web server needs to deploy the grid service. Deployment is done with a GT4 tool that, using Ant, unpacks the GAR file and copies the files within (WSDL, compiled stubs, compiled implementation, WSDD) into key locations in the GT4 directory tree. This deployment command must be run with a user that has write permission in $GLOBUS_LOCATION.

- globus-deploy-gar $EXAMPLES_DIR/org_globus_examples_services_core_first.gar
  There is also a command to undeploy a service:

- globus-undeploy-gar org_globus_examples_services_core_first

3.3 Implementation steps using Eclipse IDE

The Eclipse IDE is used here to develop, deploy, and debug a simple stateful Grid service that uses WSRF to keep state information [12].

To create the service Eclipse IDE taks the following steps:

- Setting up the required tools and components
- Creating the Eclipse project
- Adding the project files
- Creating the build launch configuration(assembling of GAR) and deploying the grid service into the Web services container
- Using the launch configuration
- Running and debugging the grid service

Tools and components required for service creation are:

- Sun Java SDK V1.4.2 (http://java.sun.com/j2se/1.4.2/download.html)
- Eclipse IDE (http://www.eclipse.org/downloads/index.php)
- Apache Jakarta Tomcat V5.0 (http://jakarta.apache.org/tomcat/)
- Sysdeo Eclipse Tomcat Launcher Plug-in V3.0 (http://www.sysdeo.com/eclipse/tomcatPlug
- GT4 WS-Core-4.0.7 (http://www.globus.org)
- globus-build-service (http://gsbt.sourceforge.net/content/view/14/31/)

The flow for the Grid Service Creation is as follows. Using this flow total grid service can be create, deploy and run within Eclipse IDE environment.

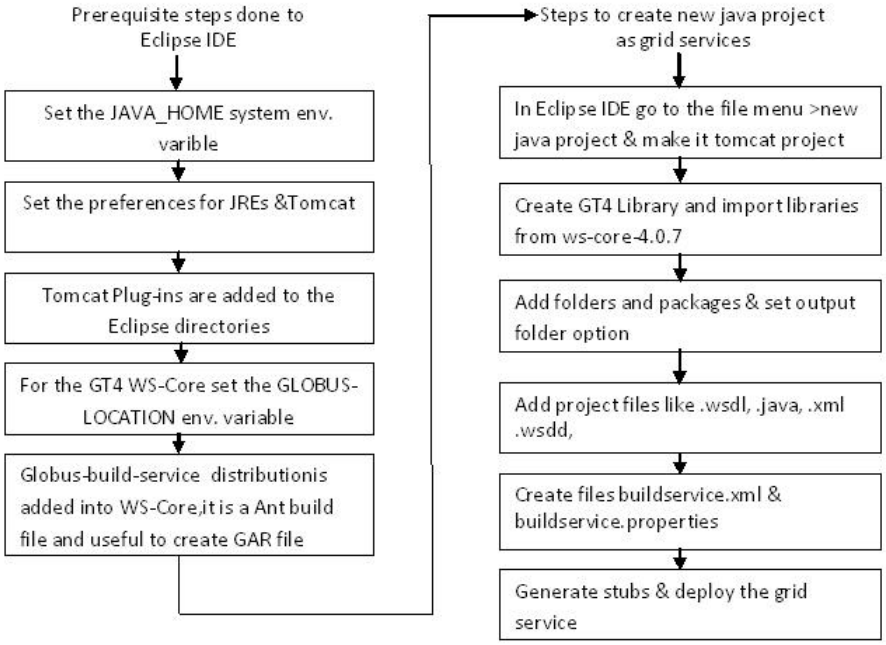

Figure 3.2: Grid Service Creation Flow

3.4 GT4 WS-Core

The GT4 WS-Core is the Globus Toolkit implementation of the WSRF. It provides the API and tools for building stateful Grid services. We need to set a GLOBUS_LOCATION environment variable to /dev/GTK. To build the WS-Core from source, We must first obtain the Apache Ant build tool from (http://ant.apache.org/bindownload.cgi).

Java WS Core is a Globus project that provides an implementation of the Web Services Resource Framework (WSRF), the Web Service Notification (WSN) family of standards, as well as WS security technology, and the Servicegroup implementation. The Java WS Core project is a contributor to the Globus Toolkit distribution. To build it, open a CMD shell and type the following:

D:/>cd %GLOBUS_LOCATION%/ws-core-4.0.7
D:/Dev/GTK/ws-core-4.0.7>ant all

- Select **File** > **New** > **Project...**,and select **Java** > **Java Project** from the selection wizard. Then type project name in particular directory. Here the project name is ProvisionDirService.

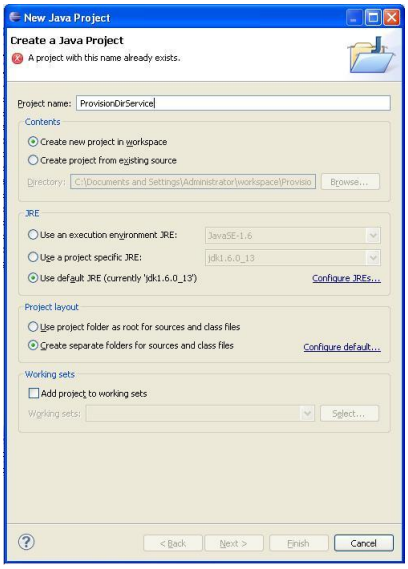

Figure 3.3: Create New Java Project

- Go to the project propertie's java build path and To create a user library from the GT4 library directory, use the **User Libraries...** button. Click **New...** in the User Libraries dialog (see Figure 11) and create a Library called GT4 Library.

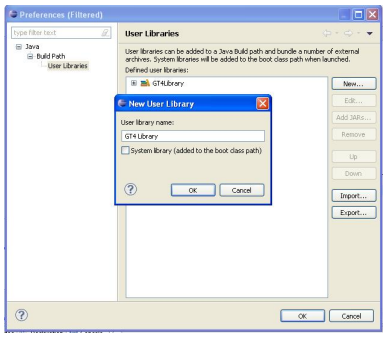

Figure 3.4: Add GT4 Library

- Add **folder and packages** and also set output folder.

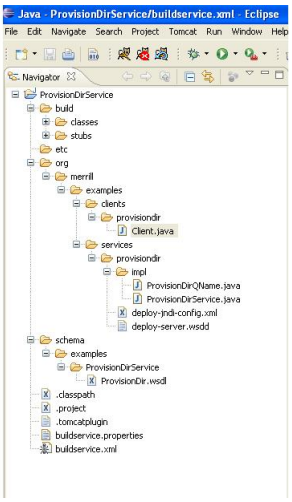

Figure 3.5: Adding Folders and Packages

- All Service Files are created like **.wsdl,.java,.xml,.wsdd** and put them in specific created folders.

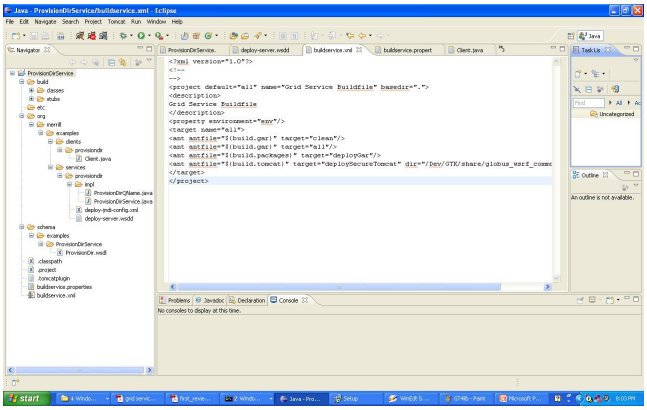

Figure 3.6: All Service files are created

- Create launch configuration to run the service. To do this right-click on build-service.xml file and select **Run** > **External Tools** as shown in figure.

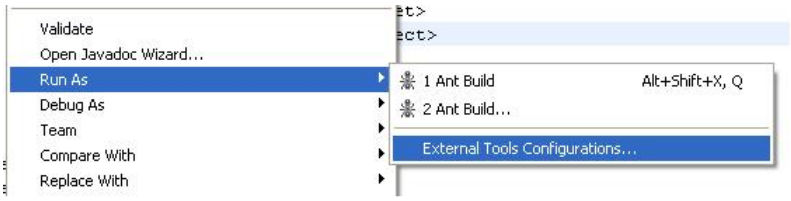

Figure 3.7: Launch configuration

- Generate stubs and deploy the grid service. By clicking the **External Tools** > **Build and Deploy ProvisionDir** launch configuration from the main toolbar.

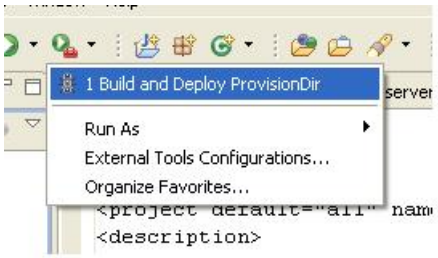

Figure 3.8: Build and Deploy the service

After the launch configuration BUILD SUCCESSFUL message is given to us.

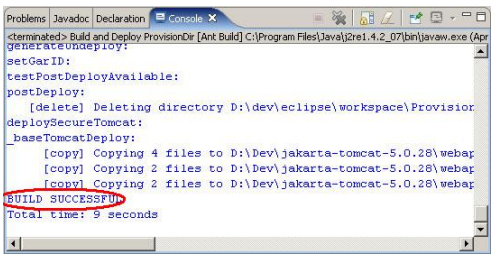

Figure 3.9: Build Sucessful Message

- After generating stubs, The Libraries tab of the Java Build Path page in the project Properties to put the stub class folder on the build path can be configured. Doing this will silence the Editor View's red compile errors. Go to the project Properties, select the Java Build Path page, and click the **Add Class Folder...** button as shown in figure.

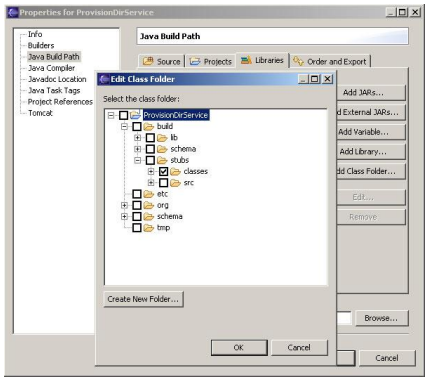

Figure 3.10: Edit Class Folder dialog

- Attach the source to the stub classes. We can now attach the source to the stub classes by right-clicking the "Source Attachment" item for the classes folder and specifying the /ProvisionDir/build/stubs/src folder in the dialog shown in figure.

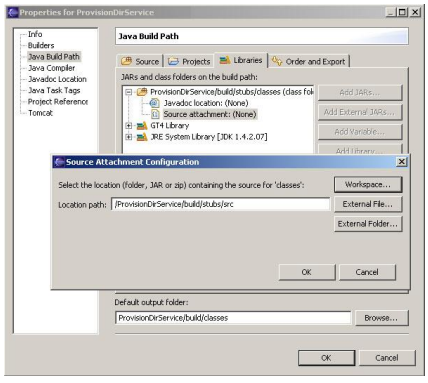

Figure 3.11: Source attachment configuration dialog

- Running (and stopping) the grid service is as easy as starting the Tomcat container by clicking the Start/Stop/Restart Tomcat toolbar buttons.

Figure 3.12: Tomcat toolbar buttons

- To test the client, simply right-click the Client.java file and select **Run** > **Run...** from the pop-up menu (Shown in Figure). In the Run dialog that is displayed, select the Arguments tab and enter http://127.0.0.1:8080/wsrf/services/examples/ProvisionDirService in the Program Arguments: textbox.

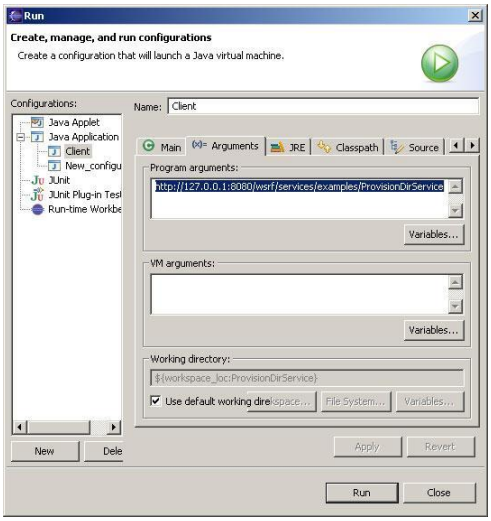

Figure 3.13: Run Dialog

Click Close to save the run configuration for this file. Once this is done, we can repeatedly run the client application by simply right-clicking the Client.java file and selecting **Run** > **Java Application**:

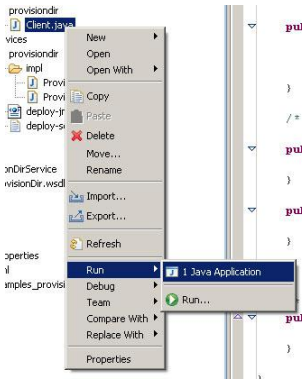

Figure 3.14: Run Java Application

The output from the client application shows up in the Console view (see Figure). The current working directory is seen to the remote grid client.

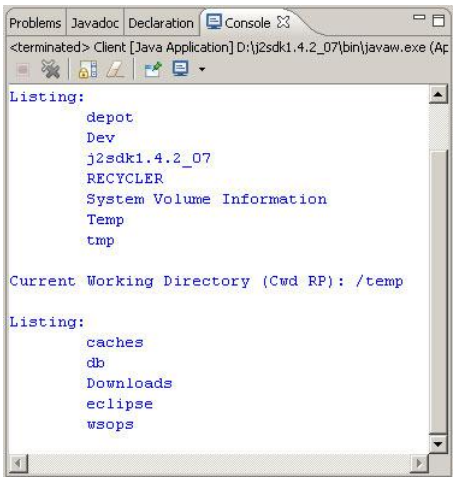

Figure 3.15: Final Output for ProvisionDirService

3.5 Analysis of the results

The user can see the local file system's directory hierarchy. In above figure the list of directories and the current working directory is shown. Which is useful to the remote grid client.

Chapter 4

Migration Service

4.1 What is Migration Service?

The migration service is responsible for coordinating the migration of a selected service from one host at a service provider to either another host at the same service provider.

4.1.1 Introduction

With the adoption of Grid services, grid systems have addressed two main obstacles, interoperability and usability. Interoperability has been addressed as Web services adopt standards for communication, allowing different system architectures to use well defined, standard communication mechanisms (SOAP/XML). Usability has been improved as Web services also provide standards to describe services (WSDL) MigrationWSDL allowing services to be easily published and discovered and used by client applications.

Despite these advantages of Web services some problems are still open and new challenges have been introduced which have not been addressed by current grid systems. These challenges include: transparent grid reconfiguration, reliability and optimization of grid services.

To address these challenges,a migration facility is proposed, as part of automatic and transparent brokerage, focused on migrating Grid services. Such a facility addresses reconfiguration by providing mechanisms to migrate a service from one host to another automatically as the grid changes. A migration facility addresses reliability by providing mechanisms to move a service from a host which may be in the process of failing, or which must be taken offline (e.g. for maintenance). Finally, a migration facility addresses optimization. Services can be migrated to faster hosts as they become available or join the grid, improving the overall performance of a Grid service[13].

Web services are typically provided by service providers.They offers services to the cilents. A service provider would benefit from a Web service migration facility for a number of reasons:

- A host leaves the grid or becomes offline either expectedly (i.e. for maintenance) or unexpectedly (due to a failure),
- A new host is added to the grid requiring the grid to be reconfigured, and
- A host is under utilised, to optimise the services offered.

Clients of grid service providers would benefit from a migration facility:

- Through the increase of availability when better suited hosts join the grid (faster, cheaper, etc.),
- Through the optimisation of their service request by completing their request faster, and
- Through the provision of reliable, failure free service.

We propose to solve the problem of Grid service migration within a broker that provides self configuration, self healing and self optimization.Here i have presented System Management Broker(SMB), responsible for providing autonomic characteristics for Web services and forming a platform for a Web service migration facility.

4.2 Migration Environment

Service providers have many different resources (hosts, clusters, databases etc). The resources have a number of attributes (architecture, computational ability, etc.) which are used by the services to fulfil a task.Clients utilise the resources through the Grid services and also have a number of requirements (price, Quality of Service, trust, etc.). Grid services must be hosted (executed) on hosts which fulfil their requirements and the hosts themselves are resources. A migration facility which migrates Grid services must therefore ensure that service requirements as well as the details of each resource are considered.

For all these resource management,some especial type of environment is required. So,for that broker approach is adopted, to provide the mechanisms for self discovery and negotiation, self configuration and self healing. The broker is a grid service able to match migration requests to available and suitable resources. The System Management Broker (SMB) provides grid clients and service providers with autonomic management facilities.

4.2.1 The System Management Broker

To provide adequate and transparent grid service migration in a grid, a supporting environment in the form of an SMB offers:

1. Transparent service discovery,
2. Data collection and Web service state management,
3. The negotiation of quality of service parameters,
4. An assessment of the trustworthiness of clients and grid service providers,
5. Automatic configuration of grid services,
6. Transparent identification of failures and healing of grid services, and

The architecture of the SMB and its communication with clients and service providers are shown in Figure. Each of the modules, apart from the interface, is

responsible for providing individual autonomic services. The self discovery and negotiation module provides services for transparent discovery, negotiation of quality of service attributes and the collection of grid knowledge. The self configuration module offers dynamic service deployment, a service provider advertisement facility, the migration service and the system change notification service. The self healing module provides the detection of service failure, the management of services state and the transparent restoration of failed services.

4.2.2 Proxies

There are two proxies that are required in an environment managed by a System Management Broker. These proxies are used to transparently communicate with the SMB.

Client Proxy

The client proxy is used by the clients application and is responsible for transparently interacting with the System Management Broker on behalf of the client. The client proxy assists with discovering SMB managed services, negotiating quality of service parameters, assessing trustworthiness and reporting failed services.

Service Proxy

The service proxy contains two components, the service management component and the service instantiation component. The service management component identifies services running on the host and transmits the state of the services on that host to the System Management Broker. The service instantiation component of the service proxy is responsible for instantiating a service on the given host.

Figure 4.1: Service Proxy

4.3 Web Service Migration

The service migration service is responsible for coordinating the migration of a selected service from one host at a service provider to either another host at the same service provider, or a host at an alternative service provider. The decision to migrate a service is made either automatically by the System Management Broker(SMB) based on knowledge of the grid environment, or specifically by a service provider. The decision to migrate a service is based on the following scenarios:

- A service proxy at a given service provider must migrate a service from one host to another, due to the expected removal or failure of a host in the grid,
- The failure detection service requests the migration of a service due to an imminent failure of a host in the grid,
- A new host or service provider joins the grid which is better suited to fulfil a current service request, and
- A host may have a required resource (such as adataset).

The migration service aggregates all the required information about a service prior to migration and moves this information to the destination. This includes the services state, binary files and service requirements. The migration service uses the services of the negotiation service, state management service, restoration service, provider registry and system knowledge service to fulfil a migration request.

When a service requires migration, if no destination host is specified, the migration service attempts to find a suitable host by communicating with the provider registry. If there is no other host of the same service provider the service is migrated to a host of another service provider. In this case the migration service must negotiate the terms of the migration with the destination provider to reach an agreement prior to service migration. For this agreement, not only resource requirements from the Web services point of view have to be satisfied but also QoS and trust of both the client and Web service provider must be addressed to the satisfaction of both parties.

4.3.1 Logical Flow of Service Migration

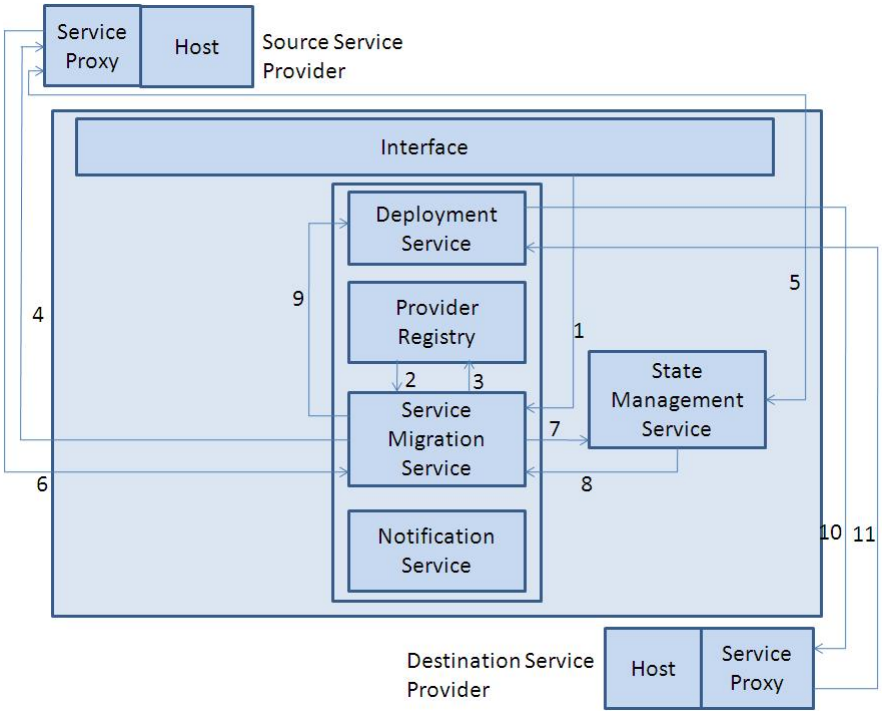

Figure 4.2: Logical Flow of Service Migration

The migration service is initiated by invoking the MigrateRequest method (Message 1). This request contains a parameter to identify the service which is to be migrated and optionally, a parameter which identifies the destination host. If the destination is not provided when invoking the migration request, the migration service must first invoke the FindSuitableHost method of the provider registry to find a suitable destination for the migration of the service (Messages 2 and 3). If a suitable destination is found, the assessment of trust and the negotiation of quality of service parameters must first be commenced. These assessments are done by invoking the system knowledge service, which performs trust assessment and invokes the negotiation service to perform negotiation. Once the migration service has a suitable destination, the migration service informs the source service proxy that a migration is about to commence (Message 4). The service proxy is then able to save the state of the service prior to the migration (Message 5) and once the service is ready for

migration, the service proxy returns a ready to migrate message (Message 6). The migration service must then check if the state exists by invoking the state management service (Message 7) and if the state of the service has previously been saved, the state is returned (Message 8).destination by invoking the destinations service proxy (Message 10). If the deployment is successful, the service proxy at the destination returns a Success message (Message 11). The service deployment service has then completed the migration and returns a Success message to the migration service (Message 12). This completes the migration of the service.

4.4 Proposed Algorithm to Migrate Service

Algorithm for Service Migration:

Table I: Service Migration Algorithm

```
public bool MigrationRequest(ServiceDetails Src, Host Dst, bool removeSrc)
{
        if(service has state)
                request state from state management service
        get details of service from service deployment service
        if(destination host not provided or host not available)
                request suitable host from provider registry
        request deployment of service on host
        if(deployment successful)
                if(RemoveSrc)
                        request removal of service from source
                        hosts service proxy
                        remove from known UDDI servers
                return(true)
        else
                return(false)
}
```

4.4.1 Algorithm for Finding a Suitable Host

Algorithm for Finding Suitable Host:

Table II: Finding Suitable Host Algorithm

```
Public Host FindSuitableHost(ServiceDetails ServiceRequirement)
{
        call GetHosts to receive a list of all available hosts
        for each host returned
                if(OS matches serviceRequirements)
                        for each required software in ServiceRequirements
                                if(software is available on host)
                                        set valid host-true
                                else
                                        disregard host
                        if(valid host)
                                return(host)
                else
                        disregard host
        if(no host found)
                for each known alternative
                        find suitable host
                if(host found)
                        return(host)
                else
                        return(error)
}
```

4.5 Implementation for Migration Service

The advanced option is shown in figure.Using the Migrate Solver Interface, in addition to the input file parameters, other parameters including the destination host name, the migration host name, the local host name, and the file path of the executable need to be specified. Using this interface we can migrate our service to other specified host.States for that service are sent via FTP port to the destination host.

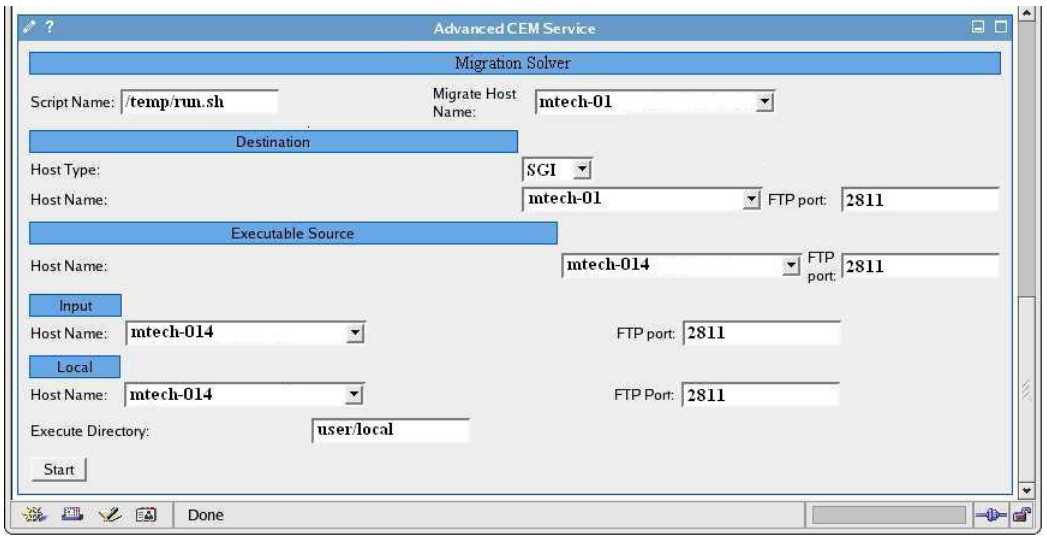

Figure 4.3: Service Migration Interface

4.5.1 Testbeds for the Migration Service-Auction Service

One of the typical commercial Web services is an auction application which is used for the experiment purpose. The auction service has a number of methods to allow clients to create new auction items and bid on those items. The client application provides an interface to use the auction service. To get the flexibility of our approach, several versions of the auction service are developed including:

- A bare bones auction service that does not use any state management facilities,
- An auction service which uses the state management facilities of the SMB,
- An auction service which uses the state management facilities of the Web Service Resource Framework (WSRF), and
- An auction service which uses the state management facilities of both the SMB and WSRF.

We perform a number of different tests involving the migration of the auction service between hosts locally (same service provider) and remotely (different service providers) and measure the time taken to perform the migration. Two different scenarios are tested for each of these cases, migrating the service to a specified destination

and migrating the service to a suitable host selected by the SMB.

The results of these tests are shown in the following tables.In first case destination host is known.

Table III: Time taken when destination host known

Auction Service	Time taken to migrate by host(Second)	
	Remote Host	Local Host
No state management	0.32	0.92
state management of SMB	0.48	1.32
state management of WSRF	0.67	2.40
state management of SMB and WSRF	1.02	2.67

From the following graph clearly seen that time required for migrate on local host takes more time than remote host.

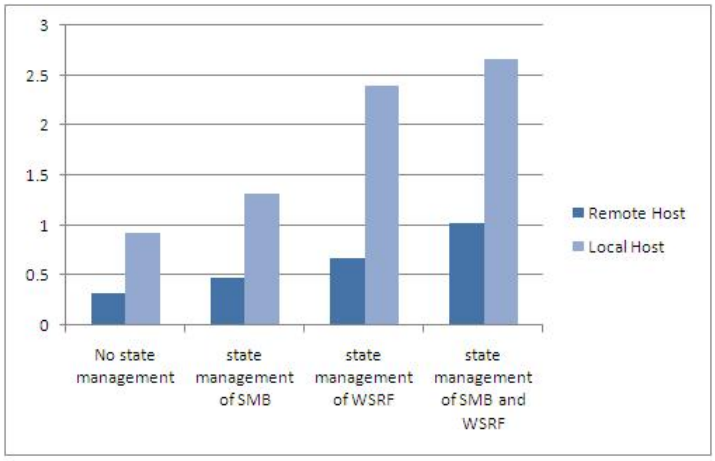

Figure 4.4: Migration time taken when destination host known

In second case SMB chooses the destination host,so time taken is slightly more because SMB have to process the service's requirements to select a suitable host and also to make sure that host is available and accessible. Results for this are as follows:

Table IV: Time taken when destination host chosen by SMB

Auction Service	Time taken to migrate by host(Second)	
	Remote Host	Local Host
No state management	0.65	1.40
state management of SMB	0.80	1.70
state management of WSRF	0.90	2.50
state management of SMB and WSRF	1.30	3.60

When SMB choose destination host,time require to migrate is increased.It is seen by the following graph.

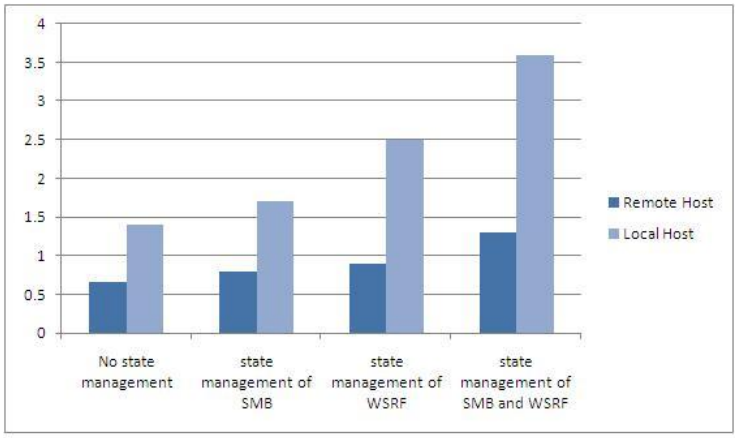

Figure 4.5: Migration time taken when destination chosen by SMB

4.6 Analysis of the results

The significant performance improvement is seen to grid through the migration facility. The time taken to run the service on migrated host is less than the time taken on the local host. Through experimentation we can see that the facility is fast,automatic and fully transparent to the client.

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

The most important improvements with regard to web Services are Stateful and potentially transient grid services obtains to the user. "Stateful" means we can perform a chain of operations in one instance. And "Transient" means we can set the lifetime of the services by our own,So the results of one client are not seen by another client. New service can be created at any time using Factory.

Through Provisioning Directory Service the local file system's directory hierarchy can be seen to remote grid clients. It will list the current working directory and also can change the current working directory.

The development and experimentation of a migration facility capable of migrating Web services. Novel approach to Web service migration is presented through System Management Broker environment. By supporting self configuration and self optimization, our migration facility provides Web service migration automatically and transparently from both clients and service providers. We have demonstrated the significant performance improvement to grids through the introduction of the migration facility. Through experimentation we can see that the facility is fast,automatic and fully transparent.

5.2 Future Scope

In the future the framework to broader areas that would help carry out more detailed performance benchmarks tests would be applied. Also to investigate implementation of the merged Web Service based management specifications. Finally, more metrics such as CPU utilization, available memory and locality need to be taken into account when assigning managers to services.

Appendix A

TroubleShooting

Error: while GT4 make - UNTAR FAILED
Sol: It can be done by editing the file in $GLOBUS_LOCATION/var/lib/perl/Grid/GPT/ where $gunzip is being set,its called LocalEnv.pm.The root cause of the error is that gunzip version was returning gzip in the version string instead of gunzip.

Error: Starting Container gives error Check if port number and host are correct and postmaster is accepting tcp/ip connections.
Sol: Check if another instance of postgresql is not running.Check the log file for hint.if this is so change user to postgres use kill -SIGTERM process id and start postmaster again.

Error: cannot create regular file /opt/globus-4.0.1/etc/hostcert.pem: Permission denied (while sigining certificate)
Sol: when grid-cert-request is run it creates three files hostcert request.pem, hostkey.pem, hostcert.pem. Here hostcert.pem file is empty remove that file.

Error: while running command globus-url-copy -vb gsiftp gives globus xio:Unable to connect to nodeb.grid:2811
Sol: start gridftp server using command globus-gridftp-server -S -p 2811

Error: globusrun-ws -submit -streaming -F https://hostname:8443/wsrf/services/ManagedJobFactoryService -c /usr/bin/whoami ERROR:Delegating user credentials... Failed. globusrun-ws: Error trying to delegate globus xio: Unable to connect to hostname:8443 globus xio: System error in connect: Connection refused globus xio: A system call failed: Connection refused
Sol: copy globus-host-ssl.conf , globus-user-ssl.conf , grid-security.conf to /etc/gridsecurity/certificates and globus-user-ssl.conf.hashno and globus-host-ssl.conf.hashno to /etc/grid-security/.

Error: globus-mds-start is not found in $GLOBUS_LOCATION/sbin
Sol: GT4 uses web-mds other than MDS follow webmdsAdminGuide to start MDS service.

Appendix B

List of Websites

- http://www.globus.org/
- http://www.globusconsortium.org/tutorial/
- http://www.globus.org/alliance/publications/papers.php/
- http://www.globus.org/toolkit/docs/development/4.0.1/
- http://www.gridlab.org/
- http://gsbt.sourceforge.net/content/view/14/31/
- http://www.eclipse.org/downloads/index.php/
- http://jakarta.apache.org/tomcat/

References

[1] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui*"The Open Grid Services Architecture, Version 1.5"*, http://forge.gridforum.org/projects/ogsa-wg,24th July,2006.

[2] S. Parastatidis and P. Watson.*"Experiences with Migrating Grid Web Services to Grid Services"*,Nottingham,UK,2003.

[3] I. Foster, C. Kesselman, J. Nick, S. Tuecke,*"Grid Services for Distributed System Integration"*,Argonne National Laboratory,Computer, 35(6),2002.

[4] I. Foster, *"GT4:software for service-oriented systems"*, IFIP International Conference on Network and Parallel Computing,Argonne National Laboratory, University of Chicago,Argonne, Springer-Verlag LNCS 3779,pp 2-13,2006.

[5] F. B. Maciel,*"Resource Management in OGSA"*,http://www.ogf.org/documents/GFD.45.pdf,2005.

[6] G. Singh, C. Kesselman, E. Deelman,*"Application-level Resource Provisioning on the Grid"*,IEEE,Page:83,2006.

[7] D. Merrill,*"Using Eclipse to develop grid services"*,ibm,University of Virginia,http://www.ibm.com/developerworks.

[8] M. Messig and A. Goscinski,*"Service Migration in Autonomic Service Oriented Grids"*, Deakin University,Australia, http://www.gaoang.com/files/Eclipse_grid.pdf.

[9] M. Lin, D. Walker, Y. Chen and J. Jones,*"A Grid-based Problem Solving Environment for GECEM"*,IEEE,volume-5,pages 347-370,2005.

[10] L. Amar, A Mualem, J. Stober,*"On the Importance of Migration for Fairness in Online Grid Markets"*,IEEE,Aug,2008.

[11] B. Sotomayor,*"The Globus Toolkit 4 Programmers Tutorial"*,University of Chicago Department of Computer Science,Nov,2005.

[12] L. Ferreira, A. Thakore, M. Brown, F. Lucchese, H. RuoBo, L. Lin, P. Manesco,*"Grid Services Programming and Application Enablement"*,http://www.ibmredbooks.com,2004.

[13] L. Chen, Q. Zhu, G. Agrawal,*"Supporting Dynamic Migration in Tightly Coupled Grid Applications"*,Ohio State University, Columbus,IEEE,Nov,2006.

Index