Working with Grid-services
Bina Ramamurthy

CSE4/586 Distributed Systems **Due Date: 12/6/2004 by mid-night.**

"The Illiterate of 21st century will not be those who cannot read and write, but those who cannot learn, unlearn and relearn". – Alvin Toffler

In our social setting, "Knowledge is change"- and accelerating knowledge-acquisition, fueling the great engine of technology, means accelerating change. – Alvin Toffler in Future Shock.

Purpose:

1. Understand the components and functions defined by Open Grid Services Architecture (OGSA).
2. Get hands-on experience, working with an implementation of OGSA in Globus Toolkit 3.0.2 (GT3).
3. Understand the concepts of virtual organization (VO), service definition and service oriented architectures (SOA).
4. Design and implement a grid service(s) and an application using the services.
5. (optional) Enhance the features of the service by adding logging, notification, security and other persistence services offered by grid framework.

Preparation:

For all the preparatory work you will work in your project space.

1. Download GT3 and install it project space. Work with the samples in the download. You should be able to run grid services in the samples directory by starting the GUI browser for Globus services. Use GT3 version 3.0.2 even though a later stable version is also available.
2. For Java use JDK1.4.2_04 version. This has been installed at /projects/bina/cse486/fall2004. Update your JAVA_HOME to point to this.
3. Understand the technology underlying Globus: its architecture and application models. Work with examples given in the samples directory of the Globus distribution.
4. Download the GT3 tutorial that explains how to write a real grid service, experiment with the tutorial on basic service and features such as service data and notification.
5. You are also given accounts on LinuxGlobusGrid assembled by KenSmith at the CSE department. Make sure you have accounts on this grid by logging into "cerf", "mills" or "vixen" from host machine. You will "ssh" into these machines. GT3 3.0.2 is the version installed on the LinucGlobusGrid.

Technology details:

Open Grid Services Architecture (OGSA) defines the components of a grid service and Open Grid Services Infrastructure (OGSI) specifies the infrastructure. Globus Toolkit

3.0.2 is an OGSA compliant implementation of the OGSI. For details of a grid service, grid service architecture (Open Gird Services Architecture: OGSA), grid services infrastructure (Open Grid services Infrastructure: OGSI) and the hosting environments see the comprehensive paper on this topic in [1]. The software that we will be using is the core of the Globus Toolkit 3.0.2. The core of the Globus can be downloaded from http://www-unix.globus.org/toolkit/download.html#core . The details of the core are available in a white paper on the core services at http://www-unix.globus.org/core/. This white also contains a javadoc-style Grid Services API description, User's Manual and a Programmer's Manual. The user's manual provides the instructions to compile, build, convert, deploy and test a grid service. The programmer's manual provides the details of writing a grid service, the various programming choices available, and deployment description. A samples directory in the core package provides a numerous examples illustrating the various grid services features.

A grid service is a web service with features as shown in Figure 1. Basic service is enhanced by standard functionality specified by OGSA. In other words, a grid service can provide in a standard fashion logging, notification, service data, routabilty, security etc. These standard functionalities enable the seamless interaction of grid services in a global large scale and high density distributed system. Basic application model is also enhanced by collaborative models, and competitive models with such higher level capabilities as negotiation and mediations. These are initial steps towards commoditization of services and their availability as transparent utilities similar to electricity and water utilities. Such a model will certainly impact the society in a very significant way. Benefits of computers will be experienced by masses without any need to explicitly learn about computers or computing.

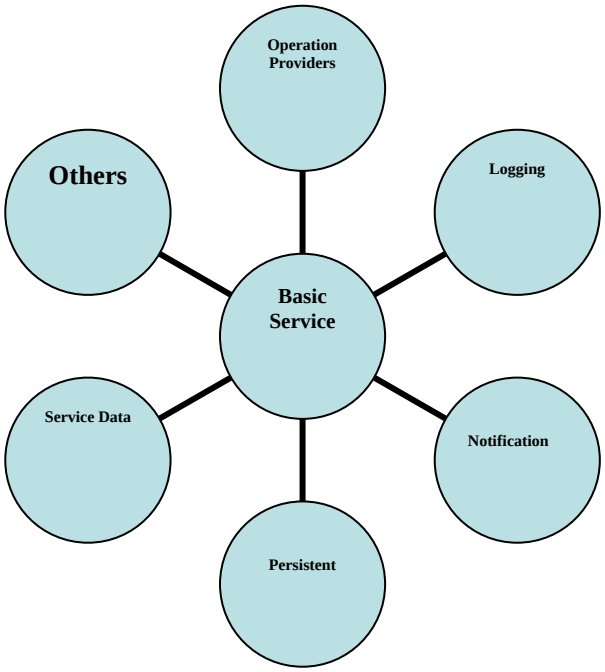

Figure 1 Features of a Grid Service

The concept of a virtual organization (VO) shown in Figure 2 is central to grid service-oriented architecture and it supports one or more grid services by sharing resources from various organizations. Architecture of a VO is defined by a *registry* for service handles, a *handle map* to translate service handles to actual service references, factories for creating and managing transient services, different types of services and a hosting environment. The self-explanatory notation used in Figure 2 is an original notation designed by the author of this document. Figure 2 shows a generic VO for the various components: (service) Factory, Registry, HandleMap, three types of services (simple, complex, and end-to-end service) and the hosting environment. Figure 3 shows a sample architecture in IRS Tax Filing application that uses three representative VOs PersonalVO, BankVO and EmployeeVO.

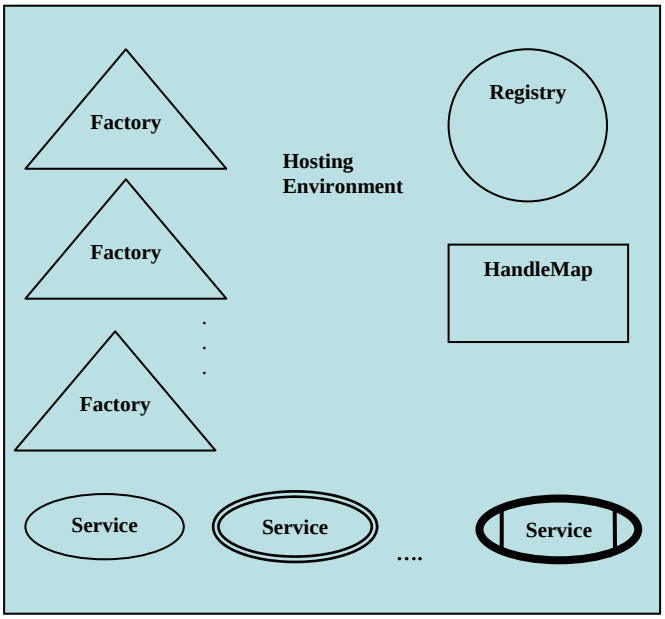

Figure 2: Virtual Organization with Distinct Symbols for Components

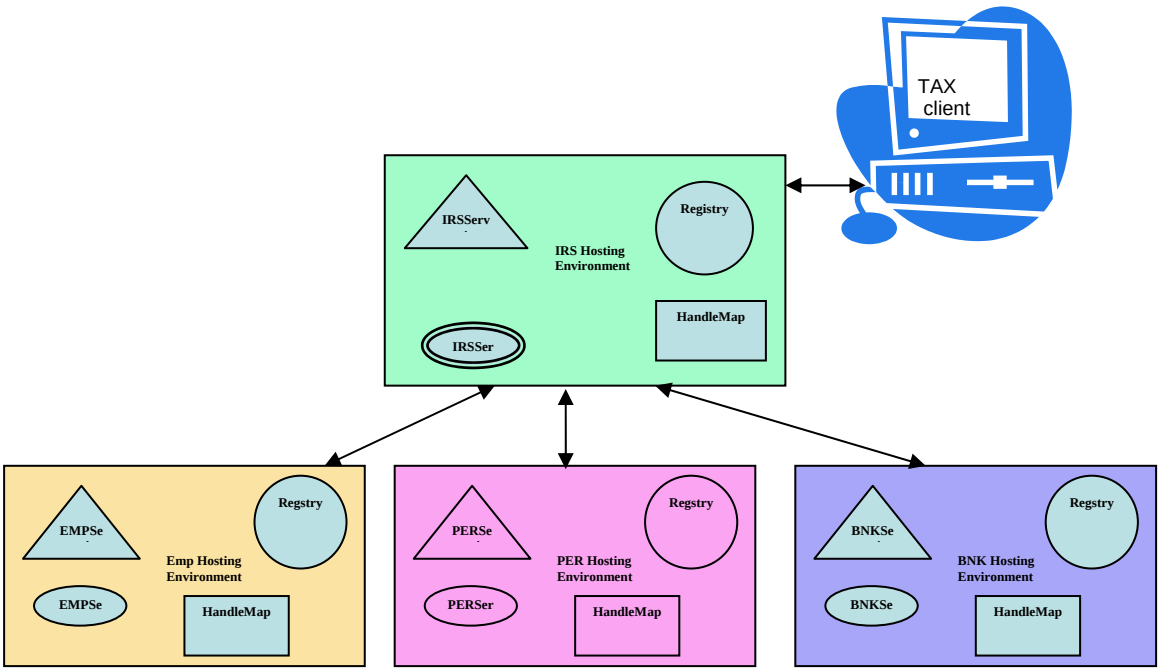

Figure 3: Tax Filing System Architecture

A Usage Scenario for IRS Application:

Any client who wants to file tax returns uses his/her communication device (a computer, cell phone, pager, telephone, PDA etc.) and authenticates himself/herself by sending appropriate information such as social security number. Then he/she authorizes filing of returns. Tax client then acts as a proxy for the user discovers and instantiates complex service IRSService which in turn invokes the EMPService, PERService and BNKService to accomplish the tax return filing.

Your Assignment:

For the application domain I have chosen the financial market and portfolio building. (This follows up on the Markowitz model we discussed and demonstrated during lecture.) You will build a service that manages a portfolio of stocks for a customer. Management involves (i) determining the list of stocks in a portfolio based on Markowitz analysis or customer input (ii) determine (based on user input/risk levels) acceptable high and low values of the stocks held, (iii) monitor by subscribing for notifications from the stock trading service, (iv) buy, sell or hold based on the notifications, customer input and/or internal policy of the service, (v) log all the transactions and decisions made using various levels of logging, and (iv) simulate the operations for a reasonable set of data. Swing in stock prices beyond the limits can be specified as one of the service data item. Your services need to be properly designed to accommodate this feature.

You research the issue by studying related literature, design an appropriate solution and implement using the grid development environment in your project space and on the CSELinuxGrid.

Analysis and Design:

Server side: Research and analyze the problem to understand the requirements. Represent the system requirements using UML (Unified Modeling Language) diagram. Identify the entities, processes and rules. Discover classes needed to implement the processes and entities. The rules are typically represented by methods in the classes. Represent the classes and relationship among them using a UML class diagram. Implement very simple grid services with appropriate relational tables for persistence.

Use real data and data feeds where ever possible.

Client side: Write a simple web client. You may reuse some of the web interfaces from the last two projects. You should be able to run the client from pollux or anyother machine outside cerf.

Implementation steps and details:

1. **Getting used to building grid services:** Work with Globus tutorial and understqand building grid services. You may use directory structure used by the tutorial or Globus core. Name every deployable service you build, prefix the service name with your user name. For example, my stock servcie will be named binaPortfolioBuilder.

2. **Building systems using build tools such as Ant:** In order to tackle complexities in configuration and deploying server-side applications, you will need to use special build tools. Apache Ant is a XML-based build tool which similar to "make" utility that most of you are familiar with. This topic will be covered during the recitation this week. Work on simple simple files to familiiarize yourself with the Ant build tool.

3. **Study and understand** grid services building and deployment.

4. **Design, implement and test** your services and sample applications and test them. Start with the basic service and then add the extra features.

5. **Deploy the integrated system:** The various components listed above were deployed and tested individually. Test the individual modules before assembling into a VO application.

6. **Create .gar** (grid archive) for each VO. Please follow strict naming conventions: Examples: usernameStockService.gar, usernameTradeService.gar, etc. Name other files based on this naming convention.

7. **Work in Groups:** You may form groups of three for this project.

8. **Practice good programming style**: Finally, practice all the good programming styles that you learned in the lower-level courses.

9. **Deploying on the CSE grid**: we will arrange for the deployment of your services on the grid. Details will be given to you in a different document. You submit your gars for deployment, and they will be automatically deployed. Mohit, Bina or Schubert will also be able to manually deploy the services for you.

10. **Port numbers**: We will assign port number ranges once groups are decided. This is to avoid collision among groups and services within groups.

Submission Details: Use the electronic submission program that corresponds to your class (cse4/586). Submit all the file using the command below:

submit_cse586 xyz.gar at the Unix prompt.

Documentation and Report: See **report details**.

[1] The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. I. Foster, C. Kesselman, J. Nick, S. Tuecke, Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002. (extended version of Grid Services for Distributed System Integration) [Citation, PDF]