# Design and implementation of an enterprise grid

by D. S. Meliksetian
J.-P. Prost
A. S. Bahl
I. Boutboul
D. P. Currier
S. Fibra
J.-Y. Girard
K. M. Kassab
J.-L. Lepesant
C. Malone
P. Manesco

This paper presents the design and implementation of intraGrid, an experimental grid based on the Globus Toolkit™ and deployed on the IBM intranet. The architecture and the main components of intraGrid are described. Then, the major technical challenges and their solutions are reviewed, including software packaging and distribution, the interface for administrative tasks, and the design and implementation of the three major services: information services, management services, and job submission services. The paper also describes the extensions and modifications to intraGrid that were required to create the ISD grid, a grid that is used for joint projects with customers and thus requires access by external users. The paper reviews the use of intraGrid by various teams of IBM researchers to date and outlines the plans for future applications. The work in progress to migrate the intraGrid to an OGSA-based (Open Grid Services Architecture-based) grid is also described.

Grid computing[1] is defined as flexible, secure, coordinated resource sharing among a dynamic collection of individuals, institutions, and resources. The sharing of these resources must be tightly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. Therefore, grid computing presents unique authentication, authorization, resource access, resource discovery, and resource management challenges. This paper presents the design, architecture, and implementation of an experimental grid infrastructure within the IBM Intranet, referred to as the *intraGrid*. The intraGrid is based on Globus Toolkit** Version 2[2] (GT2), with work underway to upgrade to Globus Toolkit Version 3[3] (GT3). The same design and architecture is also used as the basis for the *IBM Solution Design* (*ISD*) grid, a customer-facing grid that is used as a test bed for customer proof-of-concept engagements and for experimenting with new grid services in the customer environment.

The main objectives of the intraGrid initiative were: 1) the creation of an experimental grid-computing infrastructure that could be used as a test bed for experimentation and prototyping of pilot applications, tools (administration, monitoring, accounting), and policies, and 2) the creation of a community in which people involved in grid-computing activities would share expertise, ideas, and solutions.

It was recognized early that the success of this initiative depended highly on making the participation in the intraGrid activities as easy and nondisruptive as possible. This included the ability to easily deploy grid middleware, to easily manage the resources available on the grid, and to access and use these resources in a natural and easy way. The remainder of this paper is a detailed description of the attempts to solve these issues.




0018-8670/04/$5.00 © 2004 IBM

The intraGrid activity started in the IBM Research Division laboratory in Yorktown Heights, New York, where the Globus Toolkit Version 1.1.3 was installed on a number of machines in early 2001. Other IBM Research Division sites joined this nascent grid, and ad hoc mechanisms were put into place to distribute and configure the necessary components. Soon thereafter, when it became clear that these ad hoc mechanisms were not sufficiently robust and scalable to support the growth of the grid, work started on designing methods that would facilitate the deployment and management of grid components. In early 2002, a number of custom installation packages of the Globus Toolkit for various combinations of operating systems and hardware platforms were created, and the configuration of the installation was automated through a centralized management application deployed as a Web application. Since then, the deployment packages and the corresponding management applications have undergone several revisions.

Another aspect that the intraGrid team focused on was the ease of use of this grid infrastructure. This included the ability to view and monitor the resources available on the grid, as well as the ability to submit jobs to those remote resources. It was determined that the best way to approach this was to develop a Web-browser-based interface for these operations. In late 2002, the intraGrid Job Submission Portal was deployed.

The intraGrid team decided from the start to base the intraGrid infrastructure on the Globus Toolkit because its openness and malleability suited the intraGrid requirements and philosophy. The Globus Toolkit provides an open-source reference implementation of basic services for building a grid infrastructure and for deploying grid applications. It relies on existing open, standard protocols and provides application programming interfaces (APIs) to allocate and manage shared resources in a secure framework. It includes the following core services:

- Grid Security Infrastructure (GSI),[4] which provides authentication, single sign-on, delegation through proxy certificates, and other security interfaces,
- Resource allocation and job management, provided by the Grid Resource Allocation Manager (GRAM),[5]
- File I/O staging provided by the Global Access to Secondary Storage (GASS)[6] interface,
- Information services, provided by the Monitoring and Discovery Services (MDS),[7] composed of a Grid Resource Information Service (GRIS) at each resource and of one or more information aggregation services referred to as Grid Index Information Services (GIIS), and
- File-transfer and file-replica management, provided respectively by GridFTP and the Replica Location Service (RLS).[8]

Several other large projects have been leveraging the services provided by the Globus Toolkit to deploy enterprise (e.g., NASA Information Power Grid[9]), national (e.g., UK eScience Core Programme,[10] DutchGrid[11]), continental (e.g. TeraGrid,[12] Network for Earthquake Engineering Simulation Grid,[13] Grid Physics Network[14]), and worldwide (e.g., European Data Grid,[15] Large Hadron Collider Computing Grid[16]) grid infrastructures aimed at solving problems in various scientific disciplines.

Whereas the intraGrid GRAM-based Job Submission Portal only provides basic scheduling capabilities, a number of open-source projects and commercial grid offerings, such as the Condor[17] project, the UNICORE[18] project, the Open Portable Batch System[19] (OpenPBS), Platform's Load Sharing Facility (LSF),[20] IBM's LoadLeveler*,[21] and Altair Engineering, Inc.'s PBS Pro**,[22] provide more extensive job scheduling capabilities (the GRAM service provides interfaces to these cluster-oriented schedulers and could be leveraged by the intraGrid project if there were cluster resources on intraGrid). Other commercial grid offerings, such as Platform Symphony**,[23] GridSystems' InnerGrid,[24] DataSynapse, Inc.'s GridServer**,[25] and United Devices' Grid MP**,[26] provide task-scheduling capabilities. In contrast to job schedulers, in order to take advantage of the finer grain distribution enabled by task schedulers, modifications to the application source code are usually required. The decision to use only open-source grid offerings or IBM internal technologies in the intraGrid project ruled out the use of any of these task schedulers. Work is now underway to add desktop grid capabilities for PCs based on Linux** and Microsoft Windows** onto intraGrid, using technologies developed by the Melody project[27] team.

This paper presupposes a familiarity with grid technologies in general and the Globus Toolkit concepts in particular. It is organized as follows. The next section presents the overall architecture of the intraGrid, defines users and resources, and describes the major components. The section that follows, "Software packaging and distribution," describes the custom packaging implemented to automate the deployment, installation, and configuration of the intraGrid

components. The section, "Interface for administrative tasks," describes the commands available for creating a resource pool, setting up a resource, and performing similar tasks. The subsequent three sections detail the major grid services: the information services, the management services, and the job submission services. The section "ISD grid" details the modifications and extensions that were required to adapt the intraGrid architecture in order to build a grid accessible to customers. The section "The intraGrid experience and future plans" describes some of the applications deployed on this infrastructure and plans for future development of intraGrid. The final section summarizes the contents of the paper.

## Architecture

This section presents the basic concepts of the intraGrid, users, and resources, and then describes the intraGrid architecture, the major components, and the services they provide.

**Users and resources.** To access the intraGrid, users must first belong to a group known as an intraGrid *project*. A project is created by a *project leader*, who has the authority to change the composition of the project by adding or removing users. The intraGrid users can access the intraGrid either through the intraGrid Job Submission Portal or by running the Globus Toolkit client software on their desktops.

There are two predefined users, the intraGrid *administrator*, who is responsible for the overall administration of the intraGrid, and the intraGrid *guest user*, a role that is used to test the correct installation of an intraGrid resource. The intraGrid guest user can also be used to experiment with intraGrid services without having to set up a new project and add a member to that project.

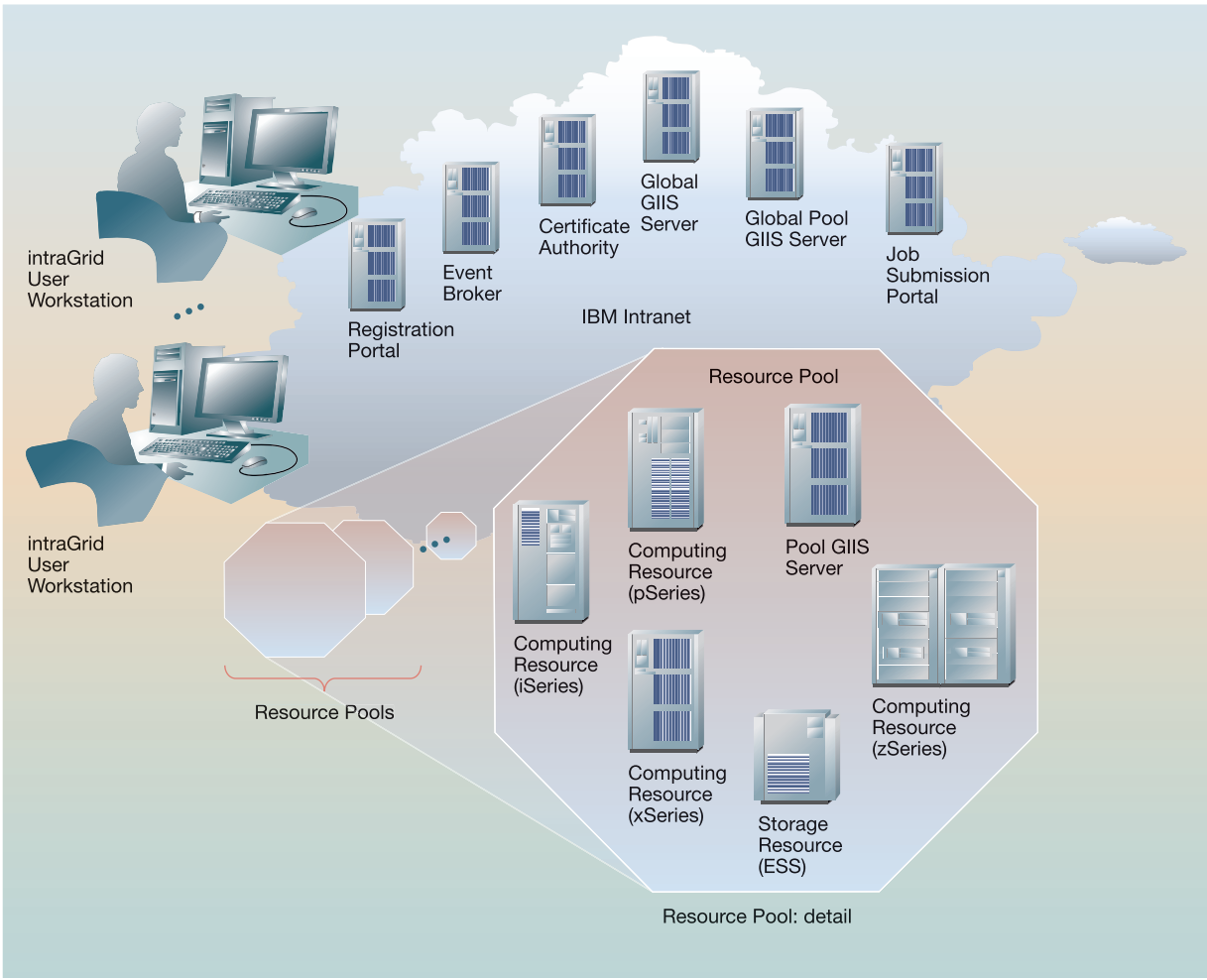
An intraGrid *resource* is any computer or storage system that is registered to the intraGrid. Every intraGrid resource has a resource owner who has root access to that resource and is responsible for installing and setting up the Globus** and intraGrid packages on the resource. *Pools* are arbitrarily defined sets of geographically collocated resources. There are two types of pools: *private* pools and *public* pools. Whereas the pool administrator of a private pool owns all the resources in that pool and has the unique authority to add or remove resources from that pool, the pool administrator of a public pool may assign others the authority to add or remove resources from the pool. The default authorization policy is to allow any member of any project to have access to any intraGrid resource at any time. However, this policy can be modified by the resource owner through the intraGrid administration graphical user interface (GUI). For example, it is possible to prevent a specific user from accessing the resource during a specified time window or to restrict the use of certain resources to members of specific projects. Upon creation, a pool contains one resource, which is configured as the pool GIIS server at the time the intraGrid package is set up. This GIIS server maintains detailed information about all the pool resources.

**intraGrid components.** The intraGrid components are depicted in Figure 1. The figure shows a number of resource pools; each pool has a GIIS server and a number of computational and storage resources. Also shown are a number of intraGrid user workstations that are used by intraGrid users to access the intraGrid. The intraGrid information services are provided by the Global GIIS server and the Global Pool GIIS server; management services are provided by the Registration Portal, the Event Broker, and the Certificate Authority (CA); job submission services are provided by the Job Submission Portal. The underlying fabric that connects all these components together is the IBM intranet.

A resource pool is *homogeneous* if all the resources are of the same type; for example, all computing resources are IBM iSeries* servers. In a *heterogeneous* resource pool the resources are of various types; for example, the computing resources could include iSeries as well as xSeries* servers. The storage resource could be ESS (IBM TotalStorage* Enterprise Storage Server). A *specialized* resource pool includes "specialized" resources, such as pSeries* SP* racks of servers.

To create projects and resource pools users access the Registration Portal by using their favorite Web browser. Users of the intraGrid can submit jobs to any intraGrid resource by using either the Job Submission Portal or their own workstations on which they have installed the Globus Toolkit client software. Each user and each resource is uniquely identified by a private key and its associated public certificate, which is signed by the intraGrid CA, following the Public Key Infrastructure (PKI)[28] security authentication scheme. Authorization to access any intraGrid resource is controlled by the Globus gridmapfile, which maps PKI identities to local userids on the resource. The intraGrid authorization scheme

Figure 1 The intraGrid architecture




maps all members of a project to the same local userid, which is used on all intraGrid resources for a given project. A master copy of the Globus grid-mapfile is managed by the Registration Portal application and stored on its associated repository. The Event Broker is used to advertise the existence of a new update of the master copy to all intraGrid resources; these resources retrieve the update asynchronously, using a pull-model scheme. The Global GIIS server gathers detailed information about all intraGrid resources from the various pool GIIS servers. Similarly, the Global Pool GIIS server gathers aggregate pool information from the pool GIIS servers.

The CA accepts user certificate requests from the Job Submission Portal, a user desktop running the Globus Toolkit client software, or from resources during resource setup. The CA signs the newly created certificate and sends it back to the requestor. The communication with the CA is through an HTTP (HyperText Transfer Protocol) Web server, and the requests are sent as SOAP (Simple Object Access Protocol) messages. The intraGrid mechanism of obtaining certificates differs from other grid implementations. The fact that each potential user can be uniquely identified through corporate identity management mechanisms allows the design of this automated system to distribute digital certificates—a

mechanism that bypasses the traditional method of authentication through e-mail and thus minimizes user intervention.

### Software packaging and distribution

The intraGrid software is distributed as two packages: the Globus Toolkit package and the intraGrid package. The intraGrid package complements the Globus package and requires that the Globus Toolkit package be installed first.

Because the target platforms are predominantly Linux- or UNIX**-based, Red Hat Package Manager (RPM)[29] is the tool selected for delivering a consistent set of software packages across the many supported operating systems. In order to ensure compatibility with native libraries (e.g., glibc) and compiler versions (e.g., gcc) on each operating system, the Globus Toolkit source code bundles were used in preference to the platform-specific binary toolkit bundles. The installation of the packages using RPM requires no special parameters and will deliver the software to the appropriate directories and set up the necessary environment changes.

The full Globus Toolkit is compiled as part of the RPM build. In order to ensure that the Globus Toolkit package stays generic and thus can be used in any grid environment, the full toolkit is installed as part of the RPM installation. The only difference from a Globus installation is that the Globus GSI configuration setup is not performed as part of the RPM install.

The intraGrid package provides a set of scripts and tools that allows users to quickly and easily add resources into the intraGrid. In particular, the Secure Update (SUP) function of the intraGrid supports automatic updates of code and configuration data on grid resources.

The intraGrid package provides the following functions:

1. GSI configuration:
   a. Generation of the security configuration files in directory /etc/grid-security.
   b. Dynamic download of the intraGrid CA public certificate into directory /etc/grid-security/certificates.
   c. Automated CA signing of the resource gatekeeper certificate via a secure Web service.
2. MDS configuration:
   a. Additional intraGrid information providers (discussed in the section, "Information services") are delivered which report, among other things, more meaningful usage, load, and availability attributes about the grid resource.
   b. The MDS registration chain is dynamically determined at configuration time based upon whether or not the resource is a pool GIIS server. On a pool GIIS server, both GIIS and GRIS are configured, whereas only GRIS is configured on other resources.
   c. The MDS host certificate is also automatically signed by the intraGrid CA secure Web service.
3. SUP agents are configured and started (see details in the subsection, "Update service" later).

The Globus Toolkit package and the intraGrid package are distributed as RPM packages for various architecture and operating-system combinations such as Linux on Intel** IA32, Linux on PowerPC**, Linux on S/390*, and AIX* on PowerPC. These packages are available for download from the intraGrid site.

### Interface for administrative tasks

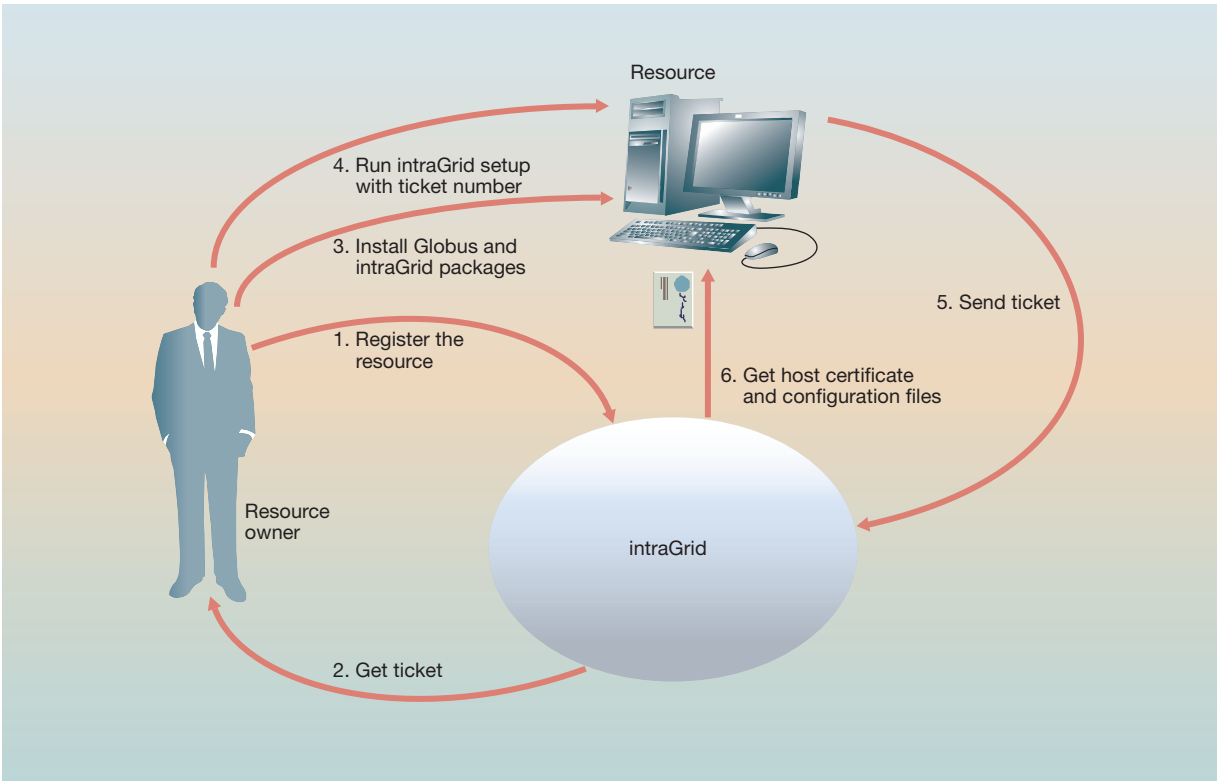
This section describes user interactions with the intraGrid infrastructure from the perspective of a pool administrator creating a resource pool, a resource owner setting up a resource, a project leader creating a project, and a user setting up a client machine.

**Creating a resource pool and setting up a resource.** In order to create a resource pool, a pool administrator first logs on to the registration service (provided by the Registration Portal) using a valid intranet ID (identifier) and password. The administrator then specifies a name for the pool, the type of the pool (private or public), a fully qualified domain name of a machine that will act as the pool GIIS server, and the IBM site where the pool resources are located.

After the resource pool is created, the pool administrator receives a *ticket* to be used in setting up the pool GIIS server as an intraGrid resource. The procedure for setting up the pool GIIS server is the same as that for setting up any other resource and is described below.

After the pool is created, additional resources can be added to the pool. If the pool is private, only the pool administrator can register additional resources

Figure 2 Configuring a resource




to the pool; root authority is required to perform the installation and configuration tasks. If the pool is public, any user that has root authority for a machine located at the site of the pool can register that machine as a member resource of the pool. The pool administrator of a public pool has to approve the addition of a resource before the ticket is issued that allows the resource owner to perform the setup tasks.

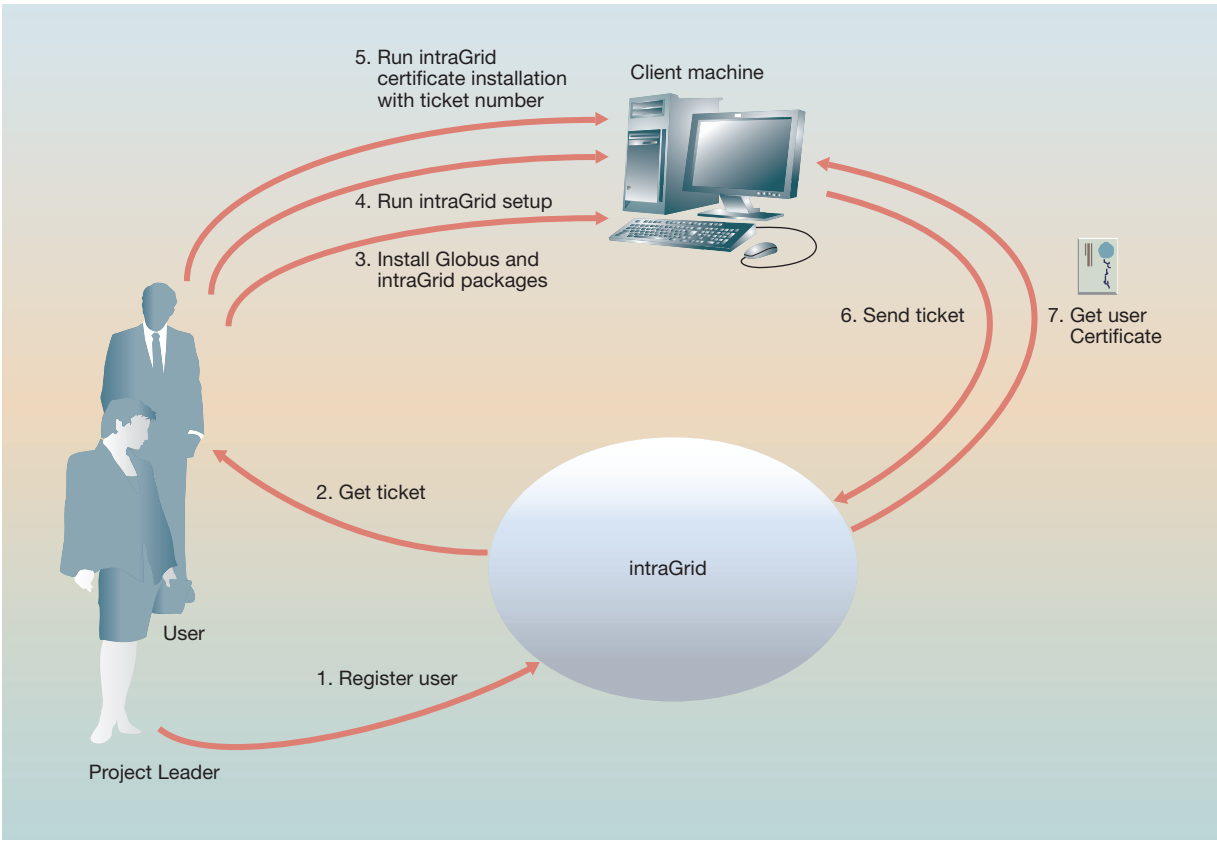
Figure 2 illustrates the steps of the resource setup process. The first step in setting up a resource is to register it (step 1). The resource owner logs on to the registration service using a valid ID and password and specifies the fully qualified domain name of the resource and the pool. As described above, there are slight differences between the processing of private and public pools. As a result of a successful registration, which in the case of a public pool requires the approval of the pool administrator, the resource owner obtains a resource ticket (step 2).

Next the resource owner installs the Globus and the intraGrid packages on the resource (step 3). The installation requires that the resource owner be logged on as root on the resource. Then, the resource owner runs the command intragrid-setup, specifying the resource ticket number obtained in the registration step (step 4). The script contacts the intraGrid registration service and the CA (step 5) to obtain the host certificates for the job submission service and for MDS. In addition, a copy of the master Globus grid-mapfile is retrieved from the registration service FTP server and stored on the resource (step 6).

**Creating a project and adding a user to a project.** To create a project, a project leader logs on using a valid ID and password. The project leader then specifies a project name, a brief description of the purpose of the project, and an optional URL that links to the project Web site. The project leader adds a member to the project by specifying an appropriate intranet ID.

The project leader need not be a member of the project. In order to act as a member of the project, the

Figure 3 Setting up an intraGrid client machine




project leader must register as such. A user can be a member of multiple projects; each such membership carries with it a distinct identity and security certificate associated with its project.

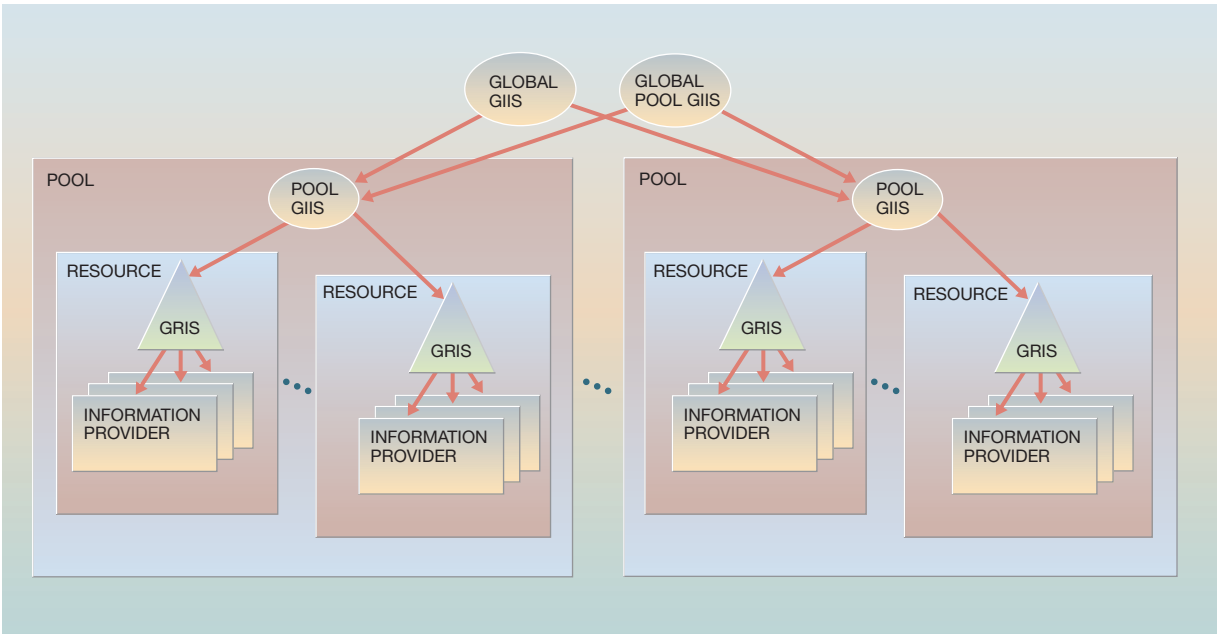
**Setting up an intraGrid client machine.** The machine to be set up as an intraGrid client machine must be either an already registered intraGrid resource or an independent machine. In either case, the user must have a local account on this machine.

Figure 3 illustrates the steps involved in setting up an intraGrid client machine. The project leader registers a user as a new member of the project (step 1) and the user receives a ticket by e-mail (step 2). The user installs the Globus and the intraGrid packages on the client machine (step 3) and runs the command intragrid-setup (step 4) without specifying a ticket number (if the client machine is an already registered intraGrid resource, these steps are not needed because the software packages have already been installed and configured).

At this point, the user logs on to the client machine and the user certificate can be obtained from the intraGrid CA. For this the user enters the command intragrid-cert-install and specifies the ticket number obtained in the registration step (step 5). The script generates a pair of private/public keys for the user and requests a certificate from the intraGrid CA (step 6). The returned certificate and the previously generated key are stored in the .globus subdirectory under the home directory of the user (step 7).

As previously mentioned, a user can be a member of multiple projects with a distinct Globus identity for each project. In this case, the user obtains a distinct certificate for each project by performing the last step multiple times on the client machine. The certificates and associated keys are then stored un-

Figure 4 The intraGrid information services architecture




der the .globus subdirectory in various directories identified by the corresponding project name.

## Information services

As depicted in Figure 4, the intraGrid information services use the LDAP[30]-based (Lightweight Directory Access Protocol-based) Globus MDS, according to the following hierarchical distributed layout:

- Each resource runs a GRIS server that maintains both static (e.g., operating systems, processors, storage devices, network devices) and dynamic (e.g., CPU load, file systems, job queues) information about the resource.
- All resources within a pool register with the pool GIIS server, which aggregates detailed information about all the resources in the pool.
- All pool GIIS servers register with the Global GIIS server and the Global Pool GIIS server; this information can be used by intraGrid users or the intraGrid Resource Broker (or Broker, for short) to select resources based upon static or dynamic criteria. As previously mentioned, the Global GIIS server aggregates detailed information about every resource whereas the Global Pool GIIS server gathers general information about every pool.

Each GRIS server uses local *information providers* to gather information about the hosting resources and to store this information in the local LDAP directories. It also contacts its pool GIIS server at regular intervals to let it know it is alive. Upon user request the pool GIIS server retrieves information from the GRIS servers that have registered with it and stores this information in memory for a period specified in its configuration. Beyond this period, referred to as Time-To-Live (TTL), the stored information is considered stale, and a subsequent request for the same information will trigger a retrieval of the information by its information provider. At regular intervals each pool GIIS server also contacts the Global GIIS server and the Global Pool GIIS server to notify them that it is alive. The Global GIIS server retrieves, upon user request, information from the pool GIIS servers that have registered with it and stores this information in memory for a period of length TTL. Similarly, the Global Pool GIIS server retrieves, upon user request, the aggregate information about each pool from the local information provider running on each pool GIIS server.

The primary consumer of the information gathered by the information services is the intraGrid Resource

Broker. The role of the Broker is to identify a set of resources that satisfy user-specified criteria. The intraGrid Resource Broker uses the information aggregated by the Global Pool GIIS to first select certain pools that contain the resources that match user-specified selection criteria. This aggregate information includes in particular the number of resources in each pool on a per-operating-system basis. After this preselection, the Resource Broker contacts specific pool GIIS servers to obtain more detailed information about each resource in the pool and to determine the scoring of that resource vs. the selection criteria. This two-step process was instituted after it was discovered that the Global GIIS server experienced scalability limitations. As the number of registered resources on the intraGrid increases, the retrieval of information from the Global GIIS server incurs longer delays. Because in the two-step scheme the Global Pool GIIS server maintains only a summary of the information about each pool, it scales better with the number of pools and resources. Maintaining the Global Pool GIIS server as a single repository for information about all resources has additional benefits for third-party tools such as graphical LDAP browsers.

## Management services

The main objective in the design of the intraGrid management services was to facilitate the interactions of users with the intraGrid. The management services consist of the *registration service*, which supports the registration of users and resources, and the *update service*, which supports automated updating of components.

We identified three categories of users: the resource providers/managers, the resource consumers, and the intraGrid administrator. The management services provide the following mechanisms:

- The ability for any IBM employee to register a machine as an intraGrid resource.
- The ability for any IBM employee to become a registered user of the intraGrid.
- The ability to automatically update and configure the intraGrid resources as new users get registered or new code releases are available.

**Registration service.** For the proper administration and management of an enterprise grid, it is necessary to be able to manage and control the resources available on the grid and to be able to track their use. The intraGrid registration service maintains information about all intraGrid projects, registered users, pools, and resources. It allows users to create projects and pools and to add members to projects and resources to pools. An interface to an e-mail Web service supports mail delivery to the intraGrid users and administrators for notification of actions to perform and confirmation of a status change (e.g., approval of a user registration, a project creation, addition of a user to a project, and addition of a resource to a pool). Although any user can access the intraGrid resources as guest user, to obtain greater access to intraGrid resources, a user must be registered as a member of a project. Any user can be a project leader. The earlier section, "Creating a project and adding a user to a project," described the registration of projects and users. Upon successful registration of each member, the project leader receives a ticket, a copy of which is also mailed to the intranet address of the new project member. This ticket is used to request an x509[28] certificate authenticating the identity of the project member. Members of the project are responsible for making sure the intraGrid packages for the appropriate platform are installed on their desktops and for setting up these machines as intraGrid client machines (as described in the section, "Setting up an intraGrid client machine," earlier). The section, "Creating a resource pool and setting up a resource," describes the registration of pools and resources. Upon registration of each resource, the resource owner receives a ticket. This ticket is used to request x509 certificates authenticating the resource GRAM and MDS services. After registration, the resource owner is responsible for installing the Globus and intraGrid packages for the appropriate platform on the resource and for setting up the resource as an intraGrid resource using the corresponding ticket.

The intraGrid registration service is implemented as a Web application and is based on the Model-View-Controller (MVC) design pattern. The Controller, which manages and controls all interactions between the user and the application, consists of several Controller servlets that implement methods specific to each user action. The Controller passes the input parameters to the Model, which consists of session Enterprise JavaBeans**,[31] to fetch and store data from internal databases. When called by the Controller, the View component uses the results of the business process to construct the response to be presented to the user, usually implemented through JavaServer Pages**[32] (JSPs**) that build dynamic HTML code and send it as an HTTP response to the user's browser.

**Update service.** It is a challenge to update code and maintain a consistent configuration among the potential thousands of resources available on a grid. It is especially cumbersome in an environment like the intraGrid because users contribute various resources and any update or configuration change usually requires the manual intervention of many individuals. What is needed is a system that would automate most of these tasks or, at worst, would involve only the intraGrid administrator and exclude all resource owners. The SUP function of the intraGrid is built to do just that: to securely and efficiently update intraGrid code and configuration data on intraGrid resources with minimal human intervention.

***Design.*** There are several types of updates required for the intraGrid. First, the propagation of the Globus grid-mapfile to all nodes is essential for the proper operation of the system. As new users are added to or removed from projects, the grid-mapfile has to be updated on all machines. Also, updates to both the Globus code and the intraGrid code may be required as new versions are released.

The code update operation involves the intervention of the intraGrid administrator. First, update files are placed on a secure file server (central repository) and MD5 CRC checksums[33] are generated. Through a simple Web interface, the update is then triggered by the intraGrid administrator. Unless a problem with the update arises or a reboot is required, the update to the node is transparent.

Although a grid-mapfile update can also be triggered manually by the intraGrid administrator, it is usually initiated by user changes performed by the registration service. Any time a user is added to or removed from a project, this update action takes place. Every aspect of the update needs to be secure. A username and password are required to manually trigger an update. The file transfers between the central repository and the nodes are encrypted. The data/code are verified for consistency at each endpoint, and when root privilege is required for update installation, it is used in an isolated fashion. In order to accommodate the potentially large number of resources that need to be served by this system, a highly scalable and lightweight solution based upon HTTP, and a publish/subscribe system provided by an MQSeries*[34] Event Broker was designed.

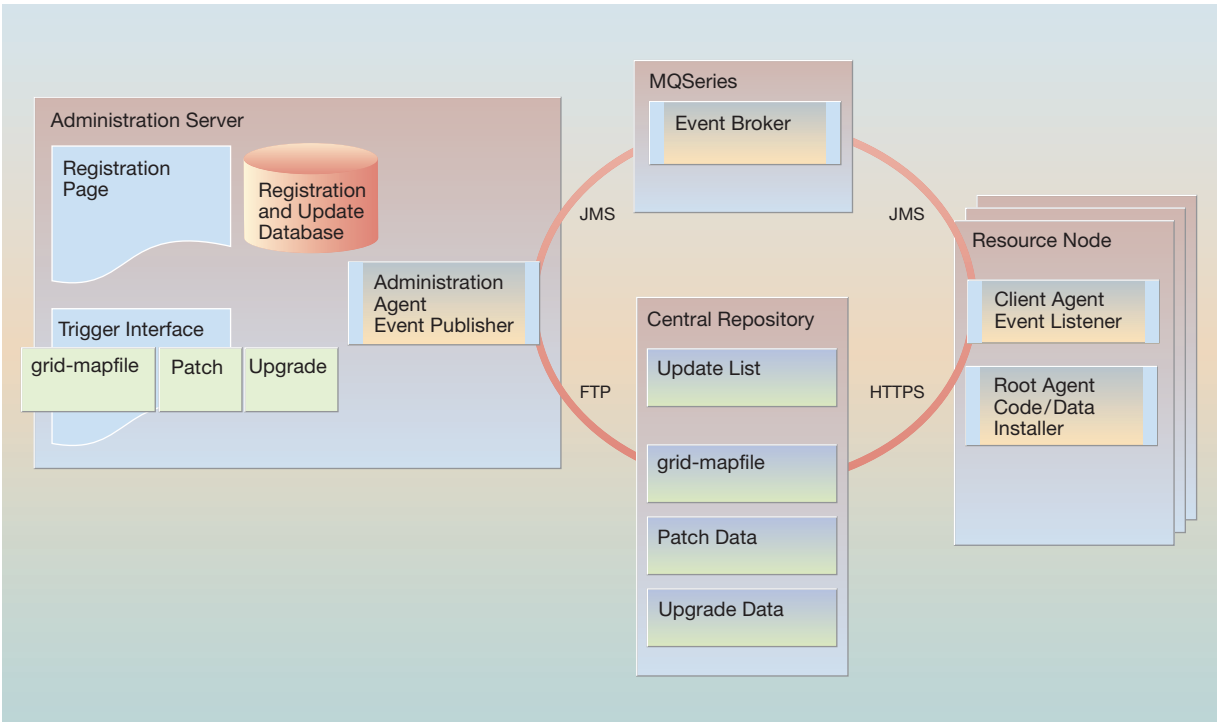
***Implementation.*** In order to implement a responsive SUP function, multiple agents are used. These agents run as system daemons, each with a specific task and narrow authorization. All these agents are monitored on a regular basis to make sure they are behaving as expected. As illustrated in Figure 5, the SUP infrastructure consists of an administration server running WebSphere*[35] and DB2*,[36] a central repository running the IBM HTTP Server and ProFTPD, an Event Broker running MQSeries Event Broker, and the various resources running a Java** Runtime Environment[37] (in addition to the Globus Toolkit). Because each server is a critical component of SUP, redundancy throughout the system is used to ensure a reliable update system. There are three agents: the Administration Agent, the Client Agent, and the Root Agent.

During the startup process, either manually through the secure Web interface or automatically, triggered by a registration update, the UpdateActions database table is updated with the task to be done. The Administration Agent monitors this table and creates a list of new updates. If the update corresponds to a grid-mapfile change, the Administration Agent builds a new grid-mapfile from data within the corresponding table of the database. If the update corresponds to a code change, then the update specifics are retrieved from the Web interface and inserted next into the appropriate update list in the central repository.

At this point, update lists (update_<platform>.xml) based on XML (Extensible Markup Language) are created, one for each of the various platforms available on the intraGrid. Within these lists are the specifics of how the resources will retrieve and validate the data. These lists, along with the grid-mapfile, are transferred to the central repository where the resources can access them and extract the update information. Finally, the Administration Agent triggers an alert on the intragrid_sup channel of the Event Broker specifying that an update is to be installed and the target platform or platforms for the update. Whereas a grid-mapfile update is intended for all resources regardless of their platforms, a code update can be targeted to one platform or a subset of the available platforms.

Each resource on the grid is listening to the Event Broker via the Client Agent. When the listener receives an alert that an update is ready to be downloaded, it first checks that the resource is of the platform type targeted by the update; if not, the Client Agent goes back to listening on the channel; otherwise it sleeps for a random period of time in order to stagger the connections back to the central repos-

Figure 5 Update service components




itory as a form of load distribution. Next an SSL connection to the central repository is opened to receive the XML update list. The Client Agent then compares the list of required updates to the list of updates already installed on the machine. The resulting updates are then systematically downloaded. As the data/code is received, MD5 CRC checksums are calculated and compared with the original checksums entered during the startup process. If they are identical (validating that the data/code has not been tampered with), the code is placed on the resource in a new directory with an XML installation file tailored for that update.

Due to the GT2 requirement that files (grid-mapfile and application code) be installed by root, SUP has a separate and isolated process called the Root Agent that performs these installations. This limits the tasks performed with root authority and minimizes security exposures.

The Root Agent monitors the installation directories for new XML installation files. These files contain information pertinent to the actual installation process. Information such as Restart Globus, Restart Client Agent, Restart Root Agent, can be found here. If the update is a simple grid-mapfile, then the new grid-mapfile is compared to the existing grid-mapfile. If there is any difference, the grid-mapfile is updated. During this update, any modification, which the resource owner may have made in order to restrict which users/projects are authorized to access the resource, is preserved. If the XML installation file does not specify a grid-mapfile update, the Root Agent performs the update instructions contained in the XML installation file.

The outcome of the update operation is recorded by the Root Agent in the local XML update file, which maintains a history of updates performed on the resource and a log file. If there are problems, the resource owner is notified with detailed instructions on how to proceed.

## Job submission services

Accessing grid resources usually requires the installation of software and credentials on a machine that

will act as a client machine. One of the objectives of the intraGrid project is to facilitate access to grid resources for casual users; hence, it was necessary to implement mechanisms that would not require the distribution and maintenance of code and credentials on the client machines.

The intraGrid *job submission services* provide such a mechanism. With this scheme, services are provided to project members through the Job Submission Portal. It allows project members to execute a job in a secure way on one or several intraGrid target resources through a browser-based interface. It also controls and manages user credentials, thus relieving the user of this task. In addition, it provides a resource broker which allows for target resource selection based on user-specified criteria.

**Job Submission Portal.** The process of job execution contains the following steps:

- Create credentials
- Define job characteristics
- Select resources
- Retrieve execution results

All user interactions with the Job Submission Portal occur only after the user is authenticated. For this purpose, the Job Submission Portal is integrated with the IBM intranet common authentication mechanism. Thus, IBM employees can log on to the portal using their common intranet userid and password.

A user who logs on to the portal could be either a registered member of one or more projects or a first-time user, not a member of a project. In the former case, a project the user is a member of must be specified (this is done from a single JSP page displaying the list of projects the user belongs to). In the latter case, the user is automatically assigned the grid credentials of the intraGrid guest user.

For job submission, a user must acquire a user certificate for the selected project; the certificate identifies the user as a member of that project. In addition to this certificate, the user must also generate a proxy certificate to be used by the portal application to authenticate the user to the gatekeeper services of the target resources for the job. Both user certificate and proxy certificate are stored by the portal in its internal DB2 database.

Proxy certificates are managed from a popup window that allows the user either to generate both user and proxy certificates with the same passphrase or to just renew the proxy certificate. Note that the user certificate for a given project needs to be generated only once, whereas the proxy certificate needs to be generated whenever a new user certificate is created or whenever the validity date of the proxy certificate has expired. The Job Submission Portal application uses Java APIs provided by the Globus Java CoG (Commodity Grid) Kit[38] to create certificates signed by the intraGrid CA. Note that user certificates generated by the portal are distinct from user certificates generated from an intraGrid client machine (as previously mentioned, users access the grid either through the portal or through their own client machine).

Once a project has been selected and the proxy certificate generated, a set of jobs (collectively referred to as "a submission") can be submitted to intraGrid target resources. The portal application uses the GRAM APIs of the Globus Java CoG Kit to submit the jobs. The characteristics of each job are described to GRAM using the Globus Resource Specification Language[39] (RSL). Because the RSL syntax may be cumbersome for some users, the portal application provides either a Web page to directly enter an RSL string that may be used by experienced users or a form that allows the novice user to enter the name of an executable file and all the required parameters. In the latter case, the portal application maps this information into an RSL string before the job is submitted. Parameters that can be specified in the second method include the list of arguments required by the application, a working directory, the names and values of environment variables to be assigned on the target resources before running the executable, and the execution mode (interactive or batch).

An executable and one input file may be staged for each job within the submission. "Staging" means that the files are first uploaded from the user client desktop to a dedicated directory on the WebSphere* Application Server hosting the portal application, and then sent to the target resources before the job execution is initiated.

In the interactive-execution mode, jobs are submitted, and the portal application waits for the results. This mode should only be used for short jobs. In batch mode, the portal application submits the jobs and returns. For these batch submissions, the Globus Java CoG Kit provides a listener class that is used by the portal application to receive end-of-job notifications.

The portal application offers two ways of selecting target resources for the jobs to be submitted: the *registered resources selection method* and the *broker selection method*. In the first method, the portal application provides to the user the list of all intraGrid registered resources. In the broker selection method, the user specifies selection criteria to the portal application, and the intraGrid Resource Broker determines a list of resources that best match these criteria. In both cases, the user then selects a set of target resources for the job out of the resulting list.

After selecting the resources (either manually or through the Broker), the user maps each job to be submitted to one or several target resources. Once the job-to-target resource mapping is completed, the portal application uses the Globus Java CoG Kit GRAM API to submit the jobs.

The portal application provides a page that displays a history list of recent submissions performed by the user. From this history list, the user can display the details of a specific job submission, which are stored in the portal internal database. If the submission mode was batch, a link is provided from the submission details page for querying the job output. The job output is displayed in a separate popup window using the Globus Java CoG Kit GASS API.

Similar to the intraGrid registration service, the intraGrid Job Submission Portal application is hosted in a WebSphere Application Server, and its implementation conforms to the MVC design pattern.

**The intraGrid Resource Broker.** As described earlier, the portal application offers two ways of selecting target resources for a job, one of them being the Resource Broker method. In this method, the user specifies selection criteria to the portal application, and the intraGrid Resource Broker determines the resources that best match these criteria. The user interacts with the broker through JSP pages generated by the portal application, and the Controller servlet interacts with the Broker using the Broker's Java API.

The task of the Broker is to find resources according to user criteria, which relate to resource attributes published by the Globus MDS and to the geographical location of the resource. The Broker scores the available resources based on these criteria and returns to the user a list of resources sorted by the scores. The user criteria can be represented as the conjunction of constraints on resource attributes. The currently supported constraints include equality, "less than," and "greater than" operations. The resource attributes currently available include static resource attributes, such as CPU type and clock frequency, total memory, file-system size, operating system, and network interfaces, and dynamic resource attributes, such as CPU load, free memory, and file-system usage.

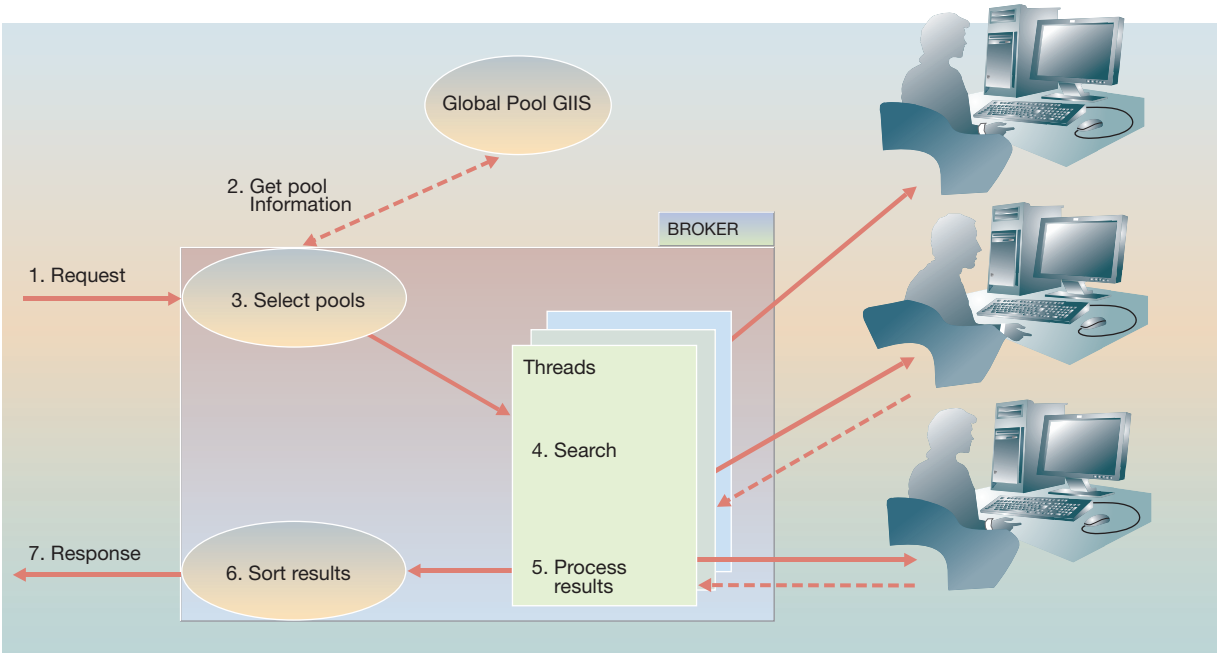
As described in the section "Information services," the intraGrid has a two-level hierarchy of GIIS servers. Resources are grouped within geographical pools. Each resource registers with its pool GIIS server, and each pool GIIS server registers with the Global Pool GIIS server. By contacting each pool GIIS server independently, the Broker can split the search space and avoid unnecessary refreshing of information at the Global GIIS server level.

Figure 6 illustrates the basic processing steps for selecting a resource. The Broker receives the user request (step 1), which includes a set of criteria (e.g., CPU speed greater than 1 GHz) and specifies a minimum score. This set of criteria is used as input to the Broker search function, whereas the score figure is used to truncate the result set. The broker contacts the Global Pool GIIS server to get information about all the pools registered by active pool GIIS servers (step 2). The Broker orders the list of pools in order to speed up the search for this request (step 3). The Broker allocates search threads (one per pool), and each thread contacts the pool GIIS server corresponding to its assigned pool to get information about the resources within that pool (step 4). The Broker processes the results returned by each thread and computes their scores (step 5). The Broker sorts the results according to their scores (step 6) and returns the sorted results to the portal application, which in turn displays them to the user (step 7).

***Algorithms.*** In this subsection the algorithms used for the implementation of the resource broker are presented.

In step 3 of Figure 6, the Broker has to determine an ordering of the pools in order to optimize the search. For load balancing across searches, it is preferable to target different pools in a way that distributes the search load. It is thus useful to apply some randomization and also to consider pools in an order that depends on their characteristics and on the criteria of the request. A preprocessing step could order pools according to the number of their re-

Figure 6 Resource Broker processing steps




sources. If the user asks for 20 resources, the Broker first considers pools that are likely to have sufficient resources (the Broker gets this information from the Global Pool GIIS server). Though this is not mandatory, it does make the search more efficient. Contacting only one pool GIIS server instead of a number of pool GIIS servers makes the search faster and consumes fewer resources. Another refinement to be applied to this step is ruling out the pools that do not satisfy the user requirements. For example, if the request specifies the operating system, the Broker only searches the pools that have resources running this operating system; this information is available at the Global Pool GIIS server.

In step 4, in which multiple searches are executed in parallel for speed and scalability, it is important to optimize the number of parallel threads. One solution is to query the information from every pool GIIS server that has been identified as a candidate in step 3 and process the received data in parallel. This provides the most speed, but at the highest cost. Alternatively, it is possible to start this process with a preselected subset of pool GIIS servers and continue through every one of them only if the search objective is not achieved earlier. This second approach was chosen by implementing a thread manager that takes care of running necessary search threads. Basically, at any given time, the search engine spawns a number of threads that depends on the number of resources yet to be found, satisfying the equation:

$$\sum_{i=1}^{N(t)} K_i \geq Velocity \times M(t)$$

where $M(t)$ is the number of resources yet to be found at time $t$, $K_i$ is the number of resources in pool $i$, $N(t)$ is the number of running threads at time $t$, and *Velocity* is a constant controlling the speed of the search (its default value is set to 3).

The number of active threads does not necessarily decrease over time, even when the number of resources yet to be found grows smaller. Indeed, this depends on the number of resources in the pools being searched. However, because the pools are preordered according to their numbers of resources, the number of active threads is likely to remain constant.

The *Velocity* parameter is configurable through the Broker API. The higher this number, the faster the search (but also the more resources consumed). Af-

ter some experimentation the default value 3 was judged to be a reasonable compromise.

In step 5 of Figure 6, the Broker scores the resources that satisfy the set of criteria specified by the user. The user typically specifies multiple criteria when searching for a set of resources. The matching resources are ordered according to the degree to which they satisfy the request. Obviously, the criteria are not all given the same importance. The user can order the criteria within the request by assigning weights to them. For example, if the user requires a machine with CPU speed greater than 1000 MHz, a 500 MHz machine should score below a 900 MHz machine. However, a 1.2 GHz machine should score the same as a 2 GHz machine because they satisfy equally the user's request. In order to obtain good scores, the scores are normalized using a Gaussian function, and each criterion is scored on a scale of 0 to 1000, using the following equation:

$$Score = 1000 \times e^{-TOLERANCE\frac{(v-r)^2}{(v+r)^2}}$$

where $r$ is the user-specified value for a criterion, $v$ is the actual value of the criterion, and *TOLERANCE* is a variable associated with each type of criterion to control the spread of the Gaussian function. A *TOLERANCE* value of 8 generally gives good results. The overall score is computed as the weighted average of the scores of all the criteria.

## ISD grid

In early 2003, the IBM Design Centers for on demand business were looking into creating a worldwide grid infrastructure, referred to as the ISD grid, that could be used for customer proof-of-concept projects and for grid technology evaluation and integration. Although the intraGrid appeared to be a good fit, because it was deployed over the IBM intranet, it did not allow access by external users, which was a requirement for the ISD grid. Several adaptations to the intraGrid were necessary to create the ISD grid and are described in this section.

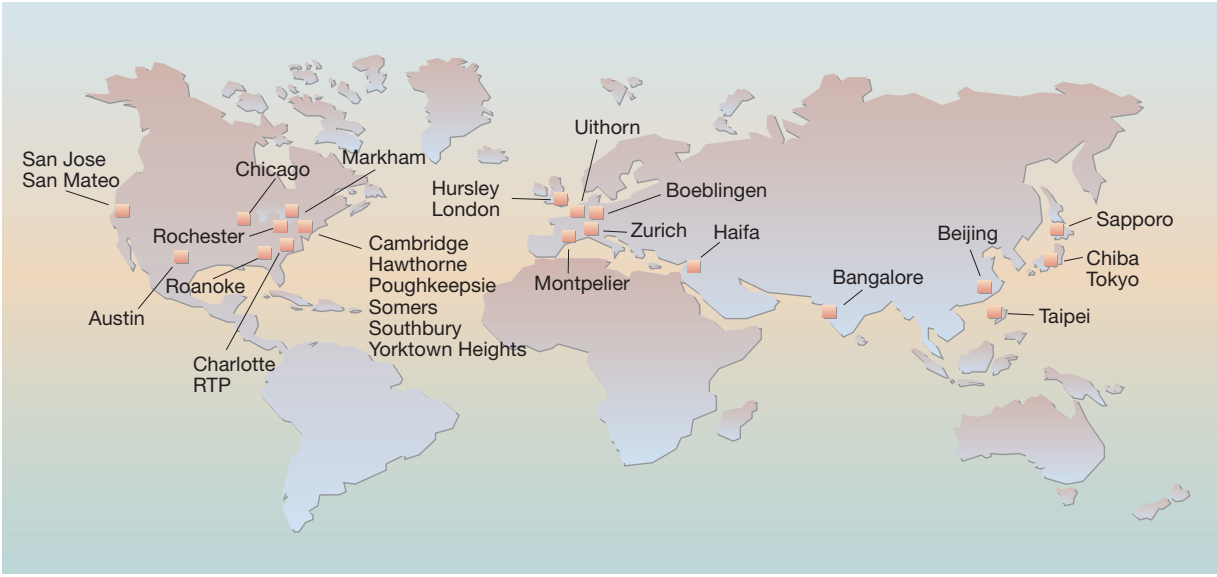
The first adaptation involved creating a virtual private network (VPN) over the IBM intranet that connected the Design Centers and deployed firewalls at each entry point to the VPN. These firewalls allowed access to the ISD grid from the IBM intranet by IBM employees and from the Internet by external users (i.e., who are not IBM employees). Note that the Registration Portal and the Job Submission Portal were placed in a "demilitarized zone" (DMZ) and access to them was through the HTTPS (HyperText Transfer Protocol Secure Sockets) protocol.

The second adaptation consisted of modifying the registration process for users. Whereas intraGrid users make use of their intranet user IDs to log on to the Registration Portal and to the Job Submission Portal, for external ISD grid users a new identification method was necessary. A new user registration function was added to the Registration Portal whereby external users were identified by a password-protected e-mail address. All users, including intraGrid users, had to register before they could become members of an ISD grid project. The registration process is a two-phase process that allows for e-mail verification.

The third adaptation involved tighter control on pool and project creation. Authority to create pools and projects is limited to IBM employees, subject to approval by an ISD grid administrator. However, once a pool (project) is created, the pool administrator (project leader) can add resources (members) without further authorization. Users' access to information is limited to projects they are members of.

The fourth adaptation consisted of setting up in each Design Center a Job Submission Client machine, in order to allow users programmatic access to the ISD grid in addition to the access through the Job Submission Portal; that is, access to the grid using the Globus Toolkit C API, the Globus Java CoG Kit APIs, or the Globus commands in shell scripts. (Note that IBM employees can use their own desktops as intraGrid client machines because they are on the IBM intranet.) The solution involves the creation of one resource pool in each Design Center and the use of the pool GIIS server as the Job Submission Client at that site. Access to Job Submission Clients is secured by secure shell[40] connections. The creation and deletion of user IDs on the Job Submission Client is automated and coordinated with the automatic update of the Globus grid-mapfile on the pool GIIS server at each site. Users must first log on to a Job Submission Client using their own IDs before obtaining programmatic access to any other ISD grid resource. User passwords are locally managed on each Job Submission Client even though they could be centrally managed and synchronized with user login passwords to the ISD grid Registration Portal using Pluggable Authentication Modules.[41] From the Job Submission Client, users can also browse ISD grid resources by using a graphical LDAP client that con-

Figure 7 Location of intraGrid resources




nects anonymously to the Globus MDS. A wrapper around the grid-proxy-init Globus command, called gdcgrid-proxy-init, allows for user selection of the appropriate project before the creation of a new proxy certificate.

## The intraGrid experience and future plans

The intraGrid infrastructure has been used for a variety of grid computing projects, from sandbox-like experiments to deployment of enterprise applications. It consists of resources located throughout the world as illustrated in Figure 7, which identifies the locations of resource pools.

A team of IBM researchers created the IBM MathSci Grid Demo Desktop (IMGD) using the intraGrid infrastructure.[42] IMGD, which consists of a GUI, client toolkit, and supporting back-end code, simplifies the use of grid applications that require access to legacy libraries and modules. Another team of IBM researchers tested extensions to the GridFTP protocol, analyzing the granularity of download sizes for optimal performance.[43] In another project, a team of IBM developers prototyped an autonomic management system for mixed workloads. The system optimizes resource allocation for a mix of Web requests and background computations, while maintaining service-level commitments for the response time of Web requests.

A completely different use of the intraGrid infrastructure involves employee productivity applications that benefit from a grid environment. One such application, the IBM downloadGrid, consists of a network of specialized download servers distributed over the intraGrid infrastructure, tied together through a management service and accessed through various specialized clients. The result is a scalable, reliable, and secure system that provides an efficient and adaptable download service.

The typical use case of the downloadGrid is as follows. Upon a client request for downloading a particular file, the management service creates an optimized download plan that specifies the number of download servers to be used for downloading the requested file and their locations. The determination of the optimized plan is based on the location of the client, the availability of the servers, and their statuses. Upon receipt of the optimized plan, the client initiates the download process by contacting the servers listed and starts the download of a chunk of the file from each server. During the course of the download, the client adjusts the chunks downloaded from the different servers, allocating new chunks to the servers as they finish their previous allocations. As the chunks are downloaded, they are written to the destination file at the appropriate offsets. During the download process and upon its completion, the cli-

ent sends feedback information to the management service about its connections to the downloadGrid servers. This feedback information is taken into account during the creation of subsequent optimized plans.

The intraGrid and ISD grid teams are currently focusing on the use of these grids for processor design simulations and financial risk assessment applications. A prototype under development will be used to study whether the underutilized CPU power of z/Series* mainframes can be used to create virtual grid nodes that are created dynamically as jobs arrive and reclaimed when jobs are completed.

Another important activity now in progress is upgrading the intraGrid and the ISD grid to GT3, thus enabling the deployment of OGSA (Open Grid Services Architecture)[44,45] services. This migration mainly consists of new GT3 grid services such as GRAM[46] Master Managed Job Factory Service, Managed Job Factory Service, Managed Job Service, MDS3[47] Index Service, and GT3 Service Data Providers, as packaged in the IBM Grid Toolbox[48] and hosted on the embedded WebSphere container. In this migration process, fault tolerance and scalability issues in the current design of the intraGrid information services are also being addressed. The policy-based dynamic creation and management of a fault-tolerant information-service-aggregator hierarchy is being designed, and new algorithms for information aggregation are being defined.

Furthermore an effort is in progress to add desktop grid capabilities for Linux-based and Microsoft Windows-based PCs onto intraGrid, leveraging IBM Research Division technologies developed by the Melody project team. This desktop grid will be used as an adjunct to the enterprise grid deployed as part of the intraGrid.

## Conclusion

This paper presents the architecture of the IBM internal enterprise grid, the intraGrid, which provides an experimental worldwide grid infrastructure accessible by all IBM employees. It is currently based on the GT2 and allows users to submit computation-intensive jobs on selected resources of the intraGrid. The resource selection is supported by a broker component that determines the best candidate resources based on user-specified criteria. Portal technology is used to create a user-friendly interface that hides from the user the complexity of the grid.

The intraGrid design was adapted and extended for the implementation of the ISD grid, which allows access to IBM resources by external users. These experimental grids have been used to develop and apply grid technologies in the enterprise and continue to be used as test beds in IBM research activities and customer engagements.

The intraGrid architecture presented in this paper is being used as the blueprint for building the IBM production grid. The IBM production grid will enable the sharing of resources located at IBM service delivery centers and will provide novel mechanisms for executing engineering and business applications. As the intraGrid architecture evolves and extends, the changes will be incorporated in the IBM production grid and in various IBM offerings.



## References

1. I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of Supercomputer Applications* **15,** No. 3 (2001).
2. Globus Toolkit 2.4, The Globus Alliance, http://www.globus.org/gt2.4/.
3. Globus Toolkit 3.2, The Globus Alliance, http://www-unix.globus.org/toolkit/download.html.
4. Globus Security Infrastructure, The Globus Alliance, http://www.globus.org/security/.
5. Globus Resource Allocation Manager, The Globus Alliance, http://www-unix.globus.org/api/c-globus-2.2/globus_gram_documentation/html/.
6. J. Bester, I. Foster, C. Kesselman, J. Tedesco, and S. Tuecke, "GASS: A Data Movement and Access Service for Wide Area Computing Systems," *Sixth Workshop on I/O in Parallel and Distributed Systems*, May 5, 1999, ACM, New York (1999), pp. 78–88.
7. Monitoring and Discovery Services, The Globus Alliance, http://www.globus.org/mds/mds2/.
8. Globus Data Management, The Globus Alliance, http://www-unix.globus.org/developer/data-management.html.
9. A. Lisotta, *Information Power Grid*, NASA Ames Research Center (2003). PowerPoint presentation available at http://hmi.stanford.edu/Presentations/May_2003_Team_Meeting/presentations/nasa.ipg.lisotta.ppt.
10. UK eScience Core Programme, Research Councils UK, http://www.rcuk.ac.uk/escience/.
11. DutchGrid, http://www.dutchgrid.nl/.
12. TeraGrid, http://www.teragrid.org/
13. Network of Earthquake Engineering Simulation Grid (NEESgrid), http://www.neesgrid.org/.
14. Grid Physics Network (GriPhyN), http://www.griphyn.org/.
15. The DataGrid Project, EU-DataGrid, http://www.eu-datagrid.org/.

16. Large Hadron Collider Computing Grid (LCG), European Organization for Nuclear Research (CERN), http://lcg.web.cern.ch/LCG/.
17. Condor Project Homepage, Condor High Throughput Computing, http://www.cs.wisc.edu/condor/.
18. UNICORE Forum, http://www.unicore.org/.
19. Open Portable Batch System (OpenPBS), http://www.openpbs.org/.
20. Load Sharing Facility (LSF), Platform Computing, Inc., http://www.platform.com/products/LSF/.
21. LoadLeveler, IBM Corporation, http://www-1.ibm.com/servers/eserver/pseries/library/sp_books/loadleveler.html.
22. PBSPro, Altair Engineering, Inc., http://www.pbspro.com/.
23. Platform Symphony, Platform Computing, Inc., http://www.platform.com/products/Symphony/.
24. InnerGrid, GridSystems, http://www.gridsystems.com/products/innergrid.html.
25. GridServer, DataSynapse, http://www.datasynapse.com/solutions/gridserver.html.
26. Grid MP, United Devices, http://www.ud.com/solutions/.
27. V. K. Naik, S. Sivasubramanian, and S. Krishnan, "Melody: A Desktop Grid Architecture for Computational Workloads," http://www.cs.indiana.edu/~srikrish/talks/Melody.ppt. This work was performed at IBM during the summer of 2003.
28. *Information Technology - Open Systems Interconnection - The Directory: Public-Key and Attribute Certificate Frameworks*, ISO Standard 9594-8:2001, ITU-T Recommendation X.509, International Organization for Standardization and International Telecommunications Union (March 2000).
29. RPM Package Manager, http://www.rpm.org.
30. Yeong, W., Howes, T., and S. Kille, *Lightweight Directory Access Protocol*, RFC 1777 (March 1995), http://www.faqs.org/rfcs/rfc1777.html.
31. J2EE Enterprise JavaBeans Technology, Sun Corporation, http://java.sun.com/products/ejb/.
32. J2EE JavaServer Pages Technology, Sun Microsystems, Inc., http://java.sun.com/products/jsp/.
33. R. Rivest, The MD5 Message-Digest Algorithm, RFC 1321, The Internet Engineering Task Force (April 1992), http://www.faqs.org/rfcs/rfc1321.html.
34. WebSphere MQ Family, IBM Corporation, http://www.ibm.com/software/integration/mqfamily/.
35. WebSphere Software Platform, IBM Corporation, http://www.ibm.com/software/websphere/.
36. DB2 Product Family, IBM Corporation, http://www.ibm.com/software/data/db2/.
37. Java Technology, Sun Microsystems, Inc., http://java.sun.com/.
38. G. von Laszewski, I. Foster, J. Gawor, and P. Lane, "A Java Commodity Grid Toolkit," *Concurrency and Computation: Practice and Experience* **13,** Nos. 8–9, 643–662 (2001).
39. The Globus Resource Specification Language RSL v1.0, Globus Consortium, http://www.globus.org/gram/rsl_spec1.html.
40. See drafts on Secure Shell (secsh), The Internet Engineering Task Force (2004), http://www.ietf.org/home.html.
41. D.-E. Smørgrav, Pluggable Authentication Modules, The FreeBSD Project (2003), http://www.freebsd.org/doc/en_US.ISO8859-1/articles/pam/.
42. H. Xi, H. Cao, L. Berman, and D. Jensen, "Distributed Supply Chain Simulation Using a Generic Job Running Framework," *Proceedings of the 2003 Winter Simulation Conference*, (2003), pp. 1305–1312.
43. M. Silberstein, M. Factor, and D. Lorenz, "DYNAMO - DirectorY, Net Archiver and MOver," *Proceedings of the Third International Workshop on Grid Computing*, Baltimore, MD, USA, November 18, 2002, Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, Germany (2002), pp. 256–267.
44. I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration," *Global Grid Forum 5 (GGF5)*, July 21-24, Edinburgh, 2002, Global Grid Forum (2002).
45. I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "Grid Services for Distributed System Integration," *Computer* **35,** No. 6, 37–46 (2002).
46. Resource Management, The Globus Alliance, http://www-unix.globus.org/developer/resource-management.html.
47. Information Services in Globus Toolkit 3, The Globus Alliance, http://www.globus.org/mds/.
48. IBM Grid Toolbox, *alphaWorks*, IBM Corporation, http://www.ibm.com/grid/solutions/grid_toolbox.shtml.

*Accepted for publication June 11, 2004.*

**Dikran S. Meliksetian** *IBM Systems and Technology Group, 150 Kettletown Road, Southbury, Connecticut 06488 (Dikran_Meliksetian@us.ibm.com).* Dr Meliksetian, a Senior Technical Staff Member, leads the design and development of the IBM internal grid. He received his Ph.D. degree in computer engineering from Syracuse University in 1992. Prior to joining IBM, he had served on the faculty of the Electrical and Computer Engineering Department at South Dakota School of Mines and Technology and on the faculty of the Department of Electrical Engineering and Computer Science at Syracuse University. In 1999 he joined an IBM team that designed and developed advanced content management systems. In 2004 for his contributions to that work, he received an IBM Outstanding Technical Achievement Award.

**Jean-Pierre Prost** *IBM France ATS PSSC, Montpellier Center, P.O. Box 81021, Montpellier, 34000, France (jpprost@fr.ibm.com).* Dr. Prost is an IT architect at the EMEA (Europe/Middle-East/Asia) Design Center for e-business on demand within the Products and Solutions Support Center (PSSC) in Montpellier, France. He received a Ph.D. degree in computer science from Institut National Polytechnique de Grenoble in 1989. He subsequently joined the IBM Thomas J. Watson Research Center in 1990, where he worked on parallel file systems and high-level parallel I/O interfaces. In 1995 he received an IBM Outstanding Technical Achievement Award for his work on MPI-IO/PIOFS (Message-Passing-Interface I/O/Parallel I/O File System). Since 2001 he has been involved in grid computing. Dr. Prost was on the team that started the intraGrid project, and in 2002 he joined his current organization where he is leading the technical grid computing activities.

**Amarjit S. Bahl** *IBM Systems and Technology Group, 150 Kettletown Road, Southbury, Connecticut 06488 (abahl@us.ibm.com).* Mr. Bahl is a Software Engineer in the Advanced Internet Technology Lab (WebAhead), which he joined in 2001. He has a B.S. degree in electronics and computer engineering and an M.B.A. degree, both from India. He has been working in the IT industry for more than eight years and has gained wide experience with various platforms and software systems.

**Irwin Boutboul** *IBM Systems and Technology Group, 150 Kettletown Road, Southbury, Connecticut 06488 (boutboul@us.ibm.com).* Mr. Boutboul received an M.S. degree in computer science from Ecole Nationale Supérieure d'Informatique et de Mathématiques

Appliquées (ENSIMAG) in Grenoble, France. He began his career in the IBM Research Division in 2002 on a team working in grid computing. In 2004 he received an IBM Outstanding Technical Achievement Award for his contribution to a content delivery grid.

**David P. Currier** *IBM Systems and Technology Group, 150 Kettletown Road, Southbury, CT 06488 (currierd@us.ibm.com).* Mr. Currier is a Senior Technical Project Leader on the WebAhead team. He received an A.S. degree in computer science from Three Rivers Technical College, Norwich, Connecticut, and has been involved in architecting IT solutions for the past 15 years. He joined Aetna, Inc. in 1993 where he developed an electronic insurance claim system that eliminated paper processing and helped process 50 million claims a year. In 1997 he joined the Web Services team of IBM Global Services where he helped design IBM's production Web-hosting environment known as the Global Web Architecture. Subsequently, he led the deployment of ShopIBM, the company's main e-business site. In 1998 and 1999 he received two IBM Leadership Excellence awards for his global leadership and design vision. Recently, he has been involved in a variety of projects, including the intraGrid Secure Update Project.

**Sebastien Fibra** *IBM France ATS PSSC, Montpellier Center, P.O. Box 81021, Montpellier, 34000, France (fibra@fr.ibm.com).* Mr. Fibra is a grid specialist at the IBM Advanced Technical Support (ATS) Design Center for on demand business within the Product and Solutions Support Center (PSSC) in Montpellier, France. He received an engineering degree from ENSEEIHT in Toulouse in 1996. After his graduation, he worked for Matra, providing consulting and support for CAD solutions for space applications and aeronautics, in which role he spent 16 months in Boston, Massachusetts. Prior to his current position in IBM, he was a senior technical consultant for a Web agency. He joined IBM in 2003 and is the lead for the deployment and operations of the grid connecting the network of IBM Design Centers worldwide.

**Jean-Yves Girard** *IBM France ATS PSSC, Montpellier Center, P.O. Box 81021, Montpellier, 34000, France (girardjy@fr.ibm.com).* Mr. Girard, who has been with IBM for six years, is an Advisory IT Specialist at the Advanced Technical Support (ATS) Grid Design Center within the Product and Solutions Support Center (PSSC) in Montpellier, France. His current work involves grid computing and Linux solutions. He received his degree of "élève ingénieur" from École Centrale Paris, France, in 1995. His areas of expertise include the Linux operating system, high-performance computing, grid technologies, and Web services.

**Kevin M. Kassab** *IBM Systems and Technology Group, 150 Kettletown Road, Southbury, CT 06488 (kkassab@us.ibm.com).* Kevin, who joined IBM in 1998, is a staff software engineer with the WebAhead team, responsible for infrastructure. He received a B.S. degree in management information systems from the University of Connecticut in 1998. He went on to obtain several certifications in operating systems, networking, and security.

**Jean-Luc Lepesant** *IBM France ATS PSSC, Montpellier Center, P.O. Box 81021, Montpellier, 34000, France (lepesant@fr.ibm.com).* Jean-Luc is an IT Specialist at the IBM Advanced Technical Support (ATS) Design Center for on demand business within the Product and Solutions Support Center (PSSC) in Montpellier, France. He has more than 25 years of experience in application development. His areas of expertise include database management, Web applications, and grid computing.

**Colm Malone** *IBM Research Division, Thomas J. Watson Research Center, 1101 Kitchawan Rd., Yorktown Heights, New York 10598 (colm@us.ibm.com).* Mr. Malone, a researcher in the IT Department at the Thomas J. Watson Research Center, received a B.S. degree in computer engineering from Trinity College in Dublin, Ireland in 1993. Since joining IBM in 1994 he worked on the development of an OS/2® managed client and later on Microsoft Windows and Linux client solutions. In 2002 he received a Chairman's Award for developing a standard Linux client solution and for helping transfer the technology to the Global Services Division, which resulted in its widespread use throughout the company. His current work is concentrated in the area of grid computing and in particular in leveraging the computing power of the many Linux machines within IBM.

**Paul Manesco** *IBM France ATS PSSC, Montpellier Center, P.O. Box 81021, Montpellier, 34000, France (paulmanesco@fr.ibm.com).* Mr. Manesco, who has been with IBM for five years, is an IT specialist at the IBM Advanced Technical Support (ATS) Design Center for on demand business within the Product and Solutions Support Center (PSSC) in Montpellier, France. He has been involved with grid computing since 2002, working on Globus-Toolkit-related projects. He holds an M.S. degree in computer science from the University of Montpellier, France. His areas of expertise include the Linux operating system, grid technologies, and performance benchmarking on IBM xSeries® platforms.