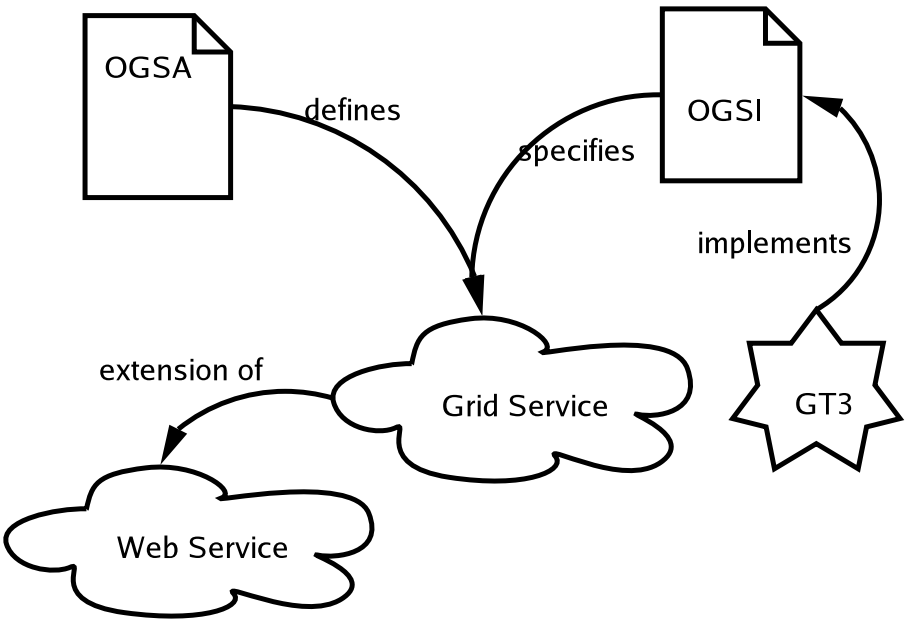
Grid and related technologies

Norwegian Computing Center
APPLIED RESEARCH AND DEVELOPMENT

REPORT



ISBN 82-536-508-4

Wolfgang Leister
Shahrzade Mazaher
Jørn Inge Vestgården
Bent Østebø Johansen
Bjørn Nordlund

16th June 2004

Norwegian Computing Center
APPLIED RESEARCH AND DEVELOPMENT

NR Report

Title: Grid and related technologies

Author: Wolfgang Leister
Shahrzade Mazaher
Jørn Inge Vestgården
Bent Østebø Johansen
Bjørn Nordlund

Date: 16th June 2004
Year: 2004
ISBN 82-536-508-4
Publication no.: 1000

Abstract: We look into grid technologies with respect to applications and implementations. Grid technologies promise to make various resources available seamlessly, allowing a more secure operation at lower costs. The key to grid is resource sharing, which leads to a virtualisation of organisations and resources.

Besides a definition of the grid technology we take a look at web services, grid services and related distributed technologies. We also present our hands-on experiences with an implementation of grid, namely the Globus Toolkit. Experiences with MSDotGrid and an overview with initiatives from industry are presented.

During the project we organised a work shop on grid technologies, with some twenty attendants. The abstracts of the presentations are included in this report.

Keywords: grid, grid services, resource sharing, Globus Toolkit

Target group:

Availability: Open

Project: Grunnbevilgning GRID

Project no.: 105002

Research field: distributed systems

No. of pages: 38

Norwegian Computing Center
Gaustadalléen 23, P.O. Box 114 Blindern, NO-0314 Oslo, Norway
Telephone: +47 2285 2500, telefax: +47 2269 7660, http://www.nr.no

Contents

Preface

This document is the result of a project where NR researchers investigated **grid** technologies. Besides a survey on the field we report hands-on experiences with several implementations and technologies. The following goals and questions will be answered by the project:

- definition of grid computing;
- use cases of grid computing;
- the Globus Toolkit, with an emphasis on GT3 as a specific implementation;
- grid initiatives from SGI, IBM, SUN, and others;
- relation to other technologies, e.g. Internet technologies, Web services, .NET, CoG distributed technologies;

This document is meant to conclude the findings of our project. The document is also meant to give the partners of NR an overview of grid technologies, in order to consider the use of grid technologies. Since several research efforts must be done for a successful introduction of grids in businesses, we would like to get any feedback concerning the use of a grid.

This document has been edited by Wolfgang Leister, wo contributed most parts of the text. The other authors were: Shahrzade Mazaher contributed Section 1.7, Jørn Inge Vestgården most parts of Section 2, Bjørn Nordlund Section 1.8, and Bent Østebø Johansen Section 3. During the course of the project we discussed many grid-related subjects with our colleagues Håvard Hegna, Dalip Dewan, and Dag Haug.

1 Grid

For many applications the use of extended resources is required in order to solve a task. In the vision of the designers of grid computing the resources necessary for an application are accessed using standardised access methods through a grid infrastructure. The metaphor "grid" comes from power grids, where electrical current can be accessed in a seamless way using a standardised plug.

The www can be seen as a specialised predecessor of a grid, which allows access to documents. In general the accessible services include resources like computing cycles (computational grid), special computing capabilities, storage, devices, or even human expertise.

Many publications on the subject of grid is available, and since the subject is emerging new publications are appearing. A good starting point is the section on overview papers at `http://www.globus.org/research/papers.html`. A comprehensive introduction is published as IBM Redbook [1].

1.1 Introduction

Our modern and mobile society depends on fast and stable access to information and computing resources, at all time and in all places. Grid systems enable this seamless access, and allow us to achieve better qualitative and quantitative results by coordinating

the resources we depend on. Additionally, grid systems are designed to give a more secure, and more dependable/accessible (up-time) access to these resources.

Grids are defined as persistent environments that enable software applications to integrate various resources in widespread locations. While this definition is very generic, the incentive to use grids is sharing of resources globally. Therefore the term Global Computing is frequently used. Resources to be shared can be computation resources (CPU cycles, storage, etc.), information resources (data bases, application data), displays, instruments, etc. These resources can be accessed seamlessly by means of the grid technology and the infrastructure of a grid.

In practise a grid is an infrastructure to control sharing of resources, i.e., being able to access others' resources, and giving access to own resources to others. The single entities are called virtual organisations, whether they are users or donors of resources. For practicability the access to resources should be organised seamlessly, and the complexity of the access to resources should be hidden from the end users. Important functionality in a grid network includes accounting, security issues, resource allocation and administration, and QoS. A grid can be said to be a cooperation for computation resources, in the same manner as cooperations were founded for rural purposes over a century ago.

The pioneer-work within grid computing has been done by the Globus Alliance, who develop the Globus Toolkit as a testbed for their ideas. Several companies have adopted the grid technologies and employ the Globus Toolkit and other developments in order to implement grids for various applications. The Global Grid Forum (GGF) is a community-initiated forum of thousands of individuals from industry and research leading the global standardisation effort for grid computing.

Historically, the grids were developed from the needs for computing power in the 1990ies. The goal was to build high-performance computing facilities in order to solve so-called grand-challenge applications, that are extremely storage- and computing-intensive. The terms meta- and hyper-computing were often used when planning these early predecessors of grid.

Checklist for grid. To decide whether an infrastructure can be considered as a grid, the coordinated resources cannot be subject to a central control[1], the protocols and interfaces must follow standards, be open, and general purpose. Additionally, the service quality of the services must be non-trivial. These characteristics can also be expressed as local autonomy, support of heterogeneous resources, scalability, and adaptivity to changes.

1.2 What is a grid?

The definitions for **grid** and grid computing vary somewhat; different visions and business models for using a grid may be the cause. Sometimes the term "grid" is confused with the term "distributed computing", and peer-to-peer techniques (e.g., SETI@Home). However, these technologies are complementary to each other. In the following paragraphs we give an overview about grid definitions.

[1] in contradiction with some examples given later in this report.

The term "grid computing" suggests a computing paradigm similar to an electric power grid-a variety of resources contribute power into a shared "pool" for many consumers to access on an as-needed basis.

What distinguishes Grid computing from conventional distributed computing is its focus on coordinated resource sharing among a dynamic set of entities. The problems that the conventional distributed computing does not seem to solve and are aimed at by the Grid technology are:

- highly flexible sharing relationships ranging from client-server to peer-to-peer;
- precise levels of control over how shared resources are used, such as fine-grained and multi-stakeholder access control;
- delegation of credentials and application of local and global policies.

Coordinated resource sharing among a dynamic set of entities places the issue of resource management at the centre. Resource management functionality, such as locating, initialisation, coordination, query, monitoring, control, accounting and secure access, across organisational boundaries thus becomes a major aspect of the grid technology.

Definition by grid.org: The definition from www.grid.org/about/gc covers the most important aspects. However, the definition is quite generic which leaves a too broad space for interpretations:

> Grid computing is a form of distributed computing that involves coordinating and sharing computing, application, data, storage, or network resources across dynamic and geographically dispersed organisations.

Definition by Foster and Kesselman: In 1998, when the term grid computing came up, Carl Kesselman and Ian Foster gave the following definition [2]:

> A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities.

This definition evolved, leading to a checklist given by Ian Foster [3], which requires that (1) resources are not subject to a centralised control, (2) open, standard, and general-purpose interfaces and protocols are used, and (3) the Grid delivers a non-trivial QoS.

Definition by NGG Expert Group on European Grid Research. They give a definition that seems to be inspired by an overall view, rather than technical considerations.

> A Grid provides an abstraction for resource sharing and collaboration across multiple administrative domains ...

Definition by globus.org: According to the Globus Alliance (globus.org)

> grids are persistent environments that enable software applications to integrate instruments, displays, computational and information resources that are managed by diverse organisations in widespread locations.

Most of the grid initiatives today have their origin in applications which solve large problems where the computational resources are distributed. The term of a *virtual organisation* is introduced [4]. The emphasis is to give a user access to all kinds of resources in a seamless manner.

Definition by IBM: IBM have several definitions on their web site `www.ibm.com/grid`. However, we emphasise on the one given at the Grid Workshop at NR (see Section A):

> Grid Computing is distributed computing over a network, using open standards, to enable heterogeneous operations.

On their web pages on grid IBM emphasise on unleashing the latent and unused power that can be used for a gain in power, speed, collaboration, and acceleration of compute-intensive processes to decent costs. They see the grid as the latest evolution of distributed computing, the Web, peer-to-peer computing and other technologies with the following characteristics:

- Like the Web, grid computing keeps complexity hidden: multiple users enjoy a single, unified experience. Unlike the Web, which mainly enables communication, grid computing enables full collaboration toward common business goals.
- Like peer-to-peer, grid computing allows users to share files[2], and additionally grid allows many-to-many sharing for all types of resources.
- Like clusters and distributed computing, grids bring computing resources together. Unlike clusters and distributed computing, which need physical proximity and operating homogeneity, grids can be geographically distributed and heterogeneous.
- Grid computing enables the vitalisation of IT resources for more than one single system.

Definition by whatis?com: According to the web site `whatis.com` (a web dictionary) the emphasis for grid is very much on distributed computing:

> Grid computing (or the use of a computational grid) is applying the resources of many computers in a network to a single problem at the same time - usually to a scientific or technical problem that requires a great number of computer processing cycles or access to large amounts of data. A well-known example of grid computing in the public domain is the ongoing SETI@Home project [...]
>
> Grid computing requires the use of software that can divide and farm out pieces of a program to as many as several thousand computers. Grid computing can be thought of as distributed and large-scale cluster computing and as a form of network-distributed parallel processing. It can be confined to the network of computer workstations within a corporation or it can be a public collaboration (in which case it is also sometimes known as a form of peer-to-peer computing).
>
> Described as a distributed resource management (DRM) tool, GridEngine [by Sun Microsystems] allows engineers at companies like Sony and Synopsys to

[2] To our opinion peer-to-peer is more than file sharing, which seems to be the basis for IBM's definition of peer-to-peer technologies.

> pool the computer cycles on up to 80 workstations at a time. (At this scale, grid computing can be seen as a more extreme case of load balancing.)
>
> Grid computing appears to be a promising trend for three reasons:
>
> - its ability to make more cost-effective use of a given amount of computer resources,
> - as a way to solve problems that can't be approached without an enormous amount of computing power, and
> - because it suggests that the resources of many computers can be cooperatively and perhaps synergistically harnessed and managed as a collaboration toward a common objective.

This definition by whatis?com mentions SETI@Home explicitly. However, the SETI@Home architecture is build to support the computing space needs of one user on many resources seamlessly. It belongs rather to the fields of distributed and autonomic computing, and could therefore be considered as complementary technologies to grid computing.

Definition by Wolfgang Gentzsch: He gives a quite visionary definition (see http://www2.cio.com/ask/expert/2002/questions/question1566.html):

> Grid computing is the next wave of the Internet. Think of it like a time machine. Grid computing enables you to effectively compress time by expanding computing space.

Definition by Brian MacKinnon: He defines grid as follows [5]:

> A computational grid is a distributed infrastructure that appears to an end user as one large computing resource.

Definition by grid.org: Together with their definition some technologies called *United devices* and *Grid MP* are presented. The definition of grid in their eyes is:

> The basics Grid computing is a form of distributed computing that involves coordinating and sharing computing, application, data, storage, or network resources across dynamic and geographically dispersed organisations.

Conclusion on a definition. Extracting the essence of the definitions above, we dare to define grid as follows:

> Grid is an infrastructure (consisting of hardware, software and communication premises) which can be accessed by users as one resource in a seamless way. The access is typically done across the boundaries of virtual organisations, typically on a global basis. The tasks a grid network performs include issues like administration, accounting, security, etc.

What the grid is NOT? While a grid provides seamless access to resources, it is very often confused with other distributed technologies. Note, that these technologies can be used in a grid, or might be important parts of a grid. However, these subjects do not cover all aspects of a grid. We give a short list of subjects that often are used when grids are discussed:

- A grid does not (necessarily) deal with **dividing a problem into partial problems**.
- Grid computing is not similar to the use of **computing clusters**, though these might be considered as resources in a grid.
- Grid is not similar to **distributed computing in general**. However, techniques from distributed are used to implement grids.
- **P2P**, **autonomic**, etc. are possible underlying implementation techniques for building grids.
- **compile once, run everywhere** is one aspect used in some grids.

Categories of Grid. Reinefeld and Schintke [6] distinguish three categories of grid systems: the Information Grid, the Resource Grid, and the Service Grid.

The **Information Grid** is equivalent with today's world wide web that delivers information on any kind of topic to any place in the world. Information can be retrieved mediated by many different kinds of networking infrastructure, PCs, advanced mobile phones, etc. Also file sharing services are part of the information grid.

The **Resource Grid** provides mechanisms for the coordinated use of resources like computers, data archives, application services, and special devices. The Globus Toolkit (see also Section 2) is an example for such a resource grid giving access to resources without bothering the user with the name, location and other attributes of the resources used. In contrast to the data supplied by the Information Grid, the facilities of the Resource Grid are supplied to authorised users only.

The **Service Grid** delivers services and applications independent of their location, implementation, hardware platform, etc. The services are built on the concrete resources available in the Resource Grid. While the Resource Grid gives access to concrete resources, the Service Grid provides abstract, location-independent services.

1.3 Benefits

The use of a grid gives advantages for both users and owner/maintainer of an infrastructure. For the end user a grid gives seamless, transparent, and secure remote access to computing resources, data, experiments, instruments, sensors, etc. The users access computing and data services rather than specific servers, without knowledge of the infrastructure (vitalisation). The resources are accessed on demand, and at the quality the service agreement specifies. In case of system failure, applications are migrated and restarted automatically (failover). Grids enable collaboration of virtual teams over the Internet, and support joint work on complex tasks. For the owner of the infrastructure the degree of utilisation increases (typical numbers suggest a utilisation increase from 20% to 80%). Additionally grids also support heterogeneity. The advantages of using a grid include the reduction of

risks, higher flexibility, reduced costs, return on investment, total cost of ownership, access to larger infrastructures, etc.

Focus on total costs of ownership

The use of grids promises to give a reduction on total costs of ownership compared with using dedicated resources for each task. This is especially relevant for resources that are used only partially or part-time, e.g., PCs, expensive special-purpose devices, supercomputers, etc. Obviously, the unused parts of the resources can be used otherwise, and possibly give an additional revenue. Grids are built to make use of resources by others possible in a seamless way. We mention several examples:

- **Supercomputers:** These are extremely expensive, but their use is necessary since some problems cannot be solved on other machines (e.g., clusters of cheaper machines) reasonably. Therefore, institutions invest in such a supercomputer, and sell the unused time to others with similar needs. This is the way physicists and meteorologists get enough computing power for their experiments and calculations, e.g., CERN.
- **Office Computers:** These are usually used during work hours, hence part-time. The usage profile often shows short and intense needs for processing resources for applications, while the PC stays rather idle in the remaining periods. These unused computing resources can be used with benefit for other tasks.
  One of the authors was involved in a project in the area of computer-animation, which could not be performed without the use of spare-time from office computers since the purchase of special purpose resources was not affordable [7]. A predecessor of a grid system was implemented. Experiences in that project showed that the resources could be used without interference with other users.
- **Special Devices:** These devices are quite expensive, and often only used part-time. Therefore, grid technologies can be used to perform tasks from others on these devices.
- **Human Experts:** The use of the term grid is not limited to technical resources. Grid technologies can also be used to give access to human expertise, e.g., in health care. Remote specialists that do not have enough tasks in their own environment (e.g., country) could be used to contribute to environments where this kind of expertise is missing.

The term of virtual organisations makes it possible to define business and service contracts between these virtual organisations.

Focus on Security and QoS

The standard ISO/IEC 17799 [8] defines security and integrity requirements. A key aspect of Information Security is to preserve the **confidentiality**, **integrity** and **availability** of an organisation's information. Other properties can be defined; however these are dependant on these former three.

Grids provide explicit and implicit security gain for users of services. Since grids are distributed systems which control several resources there is redundancy in the system. In

case of one resource becoming unavailable there is the possibility to use other resources instead. Therefore grids can provide transparent failover, and increase the availability of services.

Grid systems also provide the other aspects of security, using PKI with certificates and authentication methods between the virtual organisations. However, grids are based on trust between these organisations. Note, that the data have to be unencrypted in order to be processed, and therefore, in general confidentiality agreements must be signed between the partners in a grid.[3]

Focus on the end-user

The benefits for the end user are the seamless access to various resources without needing the knowledge of where these resources are provided. The end user also benefits from increased security, especially availability of services.

Focus on business

For businesses the participation in a grid can accelerate time to results by improving productivity and collaboration, thus being able to perform tasks that were either impossible to perform, or too expensive using alternative means. The participation in a grid also enable collaboration and promote operational flexibility, by bringing together not only IT resources, but also experts in their fields. Widely dispersed departments and businesses create virtual organisations to share data and resources.

Grids are suited to efficiently scale to variable business demands. This is achieved by creating flexible, and resilient operational infrastructures that address rapid fluctuations in customer demands and needs. Grids also give instantaneously access to resources, and provide for an optimisation of the infrastructure, by consolidating workload management, providing capacity for high-demand applications, and reduce cycle time.

Productivity increases since grids can help to give end users uninhibited access to computing, data and storage resources when they need them. This also increases flexibility in the end users daily work. Grids also provide for a highly available infrastructure, offering workload balancing, and enabling transparent recovery from failures.

Grids leverage existing capital investments since they improve optimal utilisation of computing capabilities, can help avoiding common pitfalls of over-provisioning and incurring excess costs, and can free IT organisations from the burden of administering disparate, non-integrated systems.

Concerns of negative effects when using a grid

Since grids are designed to be seamless and transparent, end users whose computer is part of a grid do not need to worry of negative effects when they need to access their desktop PC. Grids should be implemented in a manner that the grid runs in the background, utilising available resources when needed by the system. If the PC user decides to run an

[3] However, a research subject might be to provide technologies that allow processing of data in encrypted state.

application that requires more processing power, the work currently being processed by the grid on that machine will be dynamically reallocated to another machine in the grid with available processing power.

1.4 Market Models

Grids can have different architectures, varying from many users accessing one large resource, many users accessing many resources, to one user accessing many resources (e.g., SETI@Home). One business model for grid is that several (virtual) organisations invest in one or several infrastructure entities, which can be used by all the investors, paying for their share according to a payment scheme. Note that this model is similar to cooperations[4]. Instead of the cooperation model, the computing resources could also be owned by a company selling these resources to buyers, still giving seamless access to the end users.

Resources that are made accessible in a grid include CPU (computing cycles), storage, data, services, and special devices. The task to transport data from location A to location B is not a typical grid task; however a Content Distribution Network (CDN) might be, since it provides seamless access to multimedia-resources independent from where the content is provided. Note that enough capacity needs enough transport resources between location in order to make a working grid.

The main issue behind the market models of grids is the foundation of virtual organisations, which are needed for several reasons:

- The VOs are the partners that sign contracts and service level agreements.
- The VOs are sharing resources, i.e., they own their part of the infrastructure; other VOs are using the infrastructures; or both.

Organisations having unused resources can make a benefit of these by renting out these resources by the means of a grid. Since the grid infrastructures (should) support payment models, authentication, security, and SLM, the possible revenue can be defined in usual business models.

Several of these issues are based on Service Level Agreements (SLA) and Service Level Management (SLM). Therefore metrics for QoS, security semantics, availability, resource management, coordinated failover, etc. are essential.

1.5 Applications for grid

The idea of using networking resources has been around for a while, especially for computation-intensive tasks, like in physics, meteorology, or computer animation. As outlined in [7], already in 1987 several other competing grid-predecessor systems could be identified.

In order to identify application areas suitable for grid and global computing the problems to be solved must need resources that are not available locally, while these resources are available at other places, and communication infrastructure to these resources can be accessed.

[4]Norwegian readers could think of a grid as a kind of OBOS, Coop, Felleskjøp or Samvirkelag for computing resources. For German readers the paradigm would be a “Genossenschaft” for computing resources.

Examples where applications use grid technologies include: Automotive and aerospace, for collaborative design and data-intensive testing; financial services, for running long, complex scenarios and arriving at more accurate decisions; life sciences, for analysing and decoding strings of biological and chemical information; government, for enabling seamless collaboration and agility in both civil and military departments and agencies; higher education for enabling advanced, data and compute intensive research.

Applications with heavy use of computing resources (e.g., simulations, number crunching, the so-called grand challenge applications), applications using large information resources (e.g., multimedia databases), applications using special sub-applications (e.g., visualisation), and applications using special devices (e.g., expensive scanners, laboratory equipment) are candiates for grids. Especially we mention the following important application areas:

- **Medical Applications:** In diagnostics huge amounts of data are generated at one place by specialised devices. These data have to be transported to the specialists, possibly located at several locations, while the patient might be at a third location. The task of a grid in this scenario is to prepare and transport the medical data, so that they are available at the right location at the right time. Challenges within this group of applications include security (integrity of data, and making sure that the patient's data do not fall into unwanted hands), synchronisation (deliver the right data at the right time to the right destination), etc.
- **Support for multinational enterprises:** Multinational enterprises work at several locations in several time zones. Data, e.g., multimedia data from inspections, must be pre-processed and forwarded to specialists who can take decisions.
- **Multimedia Applications:** Several Multimedia Applications make use of a grid for processing media streams. Within multimedia QoS control is very important. Applications often include the handling of Digital Rights Management. E.g., multimedia data can be watermarked, scrambled etc. Other typical Grid applications include **indexing** and **data mining** of media streams, **Multiplayer Games**, (scientific) **visualization** and **computer graphics**. Concerning the subject "Computer Graphics and Grid" see also the special issue of IEEE CG&A, Vol. 23, No. 2.
- **CDN** are Content Delivery Networks, which give access to multimedia content in a seamless way independent from the underlying infrastructure. Therefore, CDN can be seen as a grid application for delivering multimedia content.
- Applications from **bio-informatics**, **seismology**, **meteorology**, etc. are data- and computing-intensive, and need often other information resources.

Explicit transport of data from A to B is NOT a grid application. Applications that are less suitable for global computing includes tasks to transfer data between locations. Additionally all tasks that are well-suited for being performed on a single PC are not in the target for grids[5].

[5]However, there are initiatives to organise the use of documents and office applications in portals, behind which grid technologies can be employed.

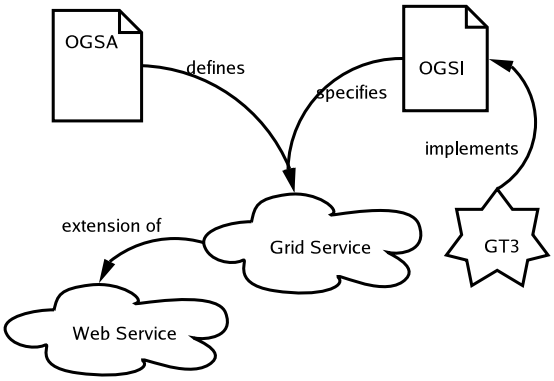


Figure 1: Relationships between OGSA, OGSI, Grid Services, Web Services, and The Globus Toolkit

1.6 Grid Architecture – OGSA and OGSI

The architectural aspects of a grid are based on the Open Grid Services Architecture (OGSA) [9]. Upon the architecture the infrastructure is built, defined by the Open Grid Services Infrastructure (OGSI) [10]. OGSI defines mechanisms for creating, managing, and exchanging information between Grid Services. Grid Services are extensions to Web Services, which are conformant to certain interfaces and behaviors. The Globus Toolkit in its Version 3 (often referred to as GT3) is an implementation of the OGSI. The motivation behind these standards is the attempt to commercialise grid computing, i.e., targeting non-scientific communities such a businesses, and make the grid technologies available to them. The relationship between a Grid Service to other terms is illustrated in Figure 1.6.

The GT3 is an implementation of OGSI (and OGSA). The Globus Toolkit is a collection of tools that can be used to build a grid. It is not a complete toolkit, since several aspects of the entire architecture are not yet implemented. In many occasions it makes sense to exchange some components if this is suitable for the applications. Other technologies, e.g., Web Services, .NET, have components that can be used to implement certain characteristics and functionality needed in a grid. Besides the Globus Toolkit other frameworks dedicated to implement grids can be employed, e.g., UNICORE, or SGI Visual Area Network (VAN). The SUN GridEngine can be used in grid implementations as a basis, but many components of a grid are not addressed.

The OGSA Service model includes service discovery, creating transient services using factories, service life time management, managing handles and references, notification, change management etc. The most important parts of a grid infrastructure includes:

- the **Information Infrastructure** (how to find a resource);
- the **Resource Management** (how to run a job);
- the **Data Management** (how to distribute data); and
- the **Grid Security Infrastructure**.

The focus is on the notion of a Grid Service and a grid is defined as an extensible set of Grid Services aggregated in various ways to meet the needs of VOs [9]. Before implementing a grid virtual organisations must be defined, e.g., (virtual) departments that can sign a

kind of service contract with other virtual organisations. All these virtual organisations agree to go together in a grid with the goal to share resources. The virtual organisations could be defined within one company, across company borders between different project partners, or on a global level between random users and providers of resources. The latter is called Global Computing. The service contract governs also fees to be paid for the use of others resources.

The grid technologies add the dimension of seamlessness to this scenario. Instead of negotiating every single access to computing resources, the technology employs certificates, resource management, service level management, secure transfer technologies, accounting technologies, etc. in order to perform a task for an application with optimal use of resources.

Technically, grids and the Globus Toolkit are based on the use of Web Services. However, extensions to to Web Services [11], especially WSDL are necessary, and have been the focus of newer developments, e.g., the WSRF (Web Services Resource Framework).

All Grid Services are Web Services. Additionally, Grid Services must support routine operations which are defined in OGSI, which include aspects like machine-readable interface definitions of services; extendability of services; lifetime management; discovery and referencing of services; instantiation of services (factory); error handling; notification; and manageability.

Interfaces. Service interfaces are defined with the Web Service Description Language (WSDL). A service containing functions and data elements is described by a so-called **portType**, since the WSDL specification describes a service to be accessed through a network port. Since WSDL 1.1 does not provide extensions of the portType, the OGSI standard defines an extended portType in the GWSDL (Grid WSDL) name space.[6]

States. Grid Services can contain state information. Therefore, for data elements (service data) it can be defined how their behaviour should be; i.e., whether they are static, constant, extensible, or mutable. These data elements are introspective, and can be queried from the outside.

Lifetime management. Grid Services include an explicit and an implicit lifetime management. For implicit lifetime management timestamps indicating the expiry time of the service are specified. GMT is used as a global time format, which is extended by an infinity-value. Every service contains the values after and before. A service terminates after the timestamp marked with after, but always before the timestamp marked with before. Using an explicit destroy method services can be terminated. New instances are produced using a Factory.

Dynamic Service References. For referencing services in dynamic environments OGSI uses a two-layer referencing system. A Grid Service Handle (GSH) is a pointer to a service independent of an instance. A GSH can be used to get a Grid Service Reference (GSR) which points to the implementation of a service including all necessary informations. The

[6]This extension is on its way into WSDL 1.2, and can be removed from OGSI thereafter.

format of the GSR is dependent on the calling mechanism, e.g., WSDL is used for a SOAP service, or IOR is used for a CORBA service.

Grid Services vs. Web Services. Grid Services are built on Web Services, using the WSDL, SOAP and Web Services security layer. For Grid Services, it must be agreed on the scheme for services, (using GSH and GSR), the mandatory data elements (Service-Data, which contain meta data and state information), and a unified instantiation and life management.

Web Services Resource Framework. Recently (January 2004), the Web Services Resource Framework (WSRF) was presented [12], which unifies the concepts of OGSI and Web Services.

1.7 Grid and Distributed Computing

In this section we describe how grid relates to conventional distributed computing techniques, as realised by CORBA, J2EE, DCOM, etc. In distributed environments security is an important ingredient. Authentication between users and resources, security of the data in transfer, fine grain control over who is accessing what, delegation of credentials from users/programs to programs, management of policies across multiple institutions are all core functionality in grid computing.

A Model for Grid Computing

A grid architecture (OGSA) is described in [4], which is depicted in Figure 3 (left). The architecture is layered and identifies the fundamental system components, their purposes and how they interact with each other. Grids are heterogeneous by nature and operate in networked environments. To provide interoperability, the architecture defines a set of common protocols used as the basic mechanism by which users and resources negotiate. OGSA uses the following layers:

The Fabric Layer. This layer provides the resources to which shared access is mediated by grid protocols. Fabric components implement the local, resource specific operations that occur on specific resources (physical or logical). A minimum that a resource should provide is enquiry mechanisms that allow discovery of their structure, state and capabilities, e.g., whether they support advance reservation.

The Connectivity Layer. This layer defines core communication and authentication protocols required for grid-specific network transactions. Communication protocols enable exchange of data between Fabric layer resources, while Authentication protocols provide cryptographically secure mechanisms for verifying the identity of users and resources. Authentication solutions must provide for

- single sign on: a user logs on (is authenticated) just once, and then have access to multiple grid resources;

- delegation: a user is able to endow a program with the ability to run on that user's behalf.

The Resource Layer. This layer builds on the Connectivity layer communication and authentication protocols, and define protocols for the secure negotiation, initiation, monitoring, control, accounting and payment of sharing operations on individual resources. Note that Resource layer protocols are concerned entirely with individual resources and ignore issues of global state and atomic actions across distributed collections.

The Collective Layer. This layer is concerned with what was ignored by the previous layer, namely, a collection of resources. It contains protocols and services that are global in nature and capture interactions across collections of resources. The collective layer components build on the protocols of the Resource and Connectivity layers and implement a variety of sharing behaviours without placing new requirements on the resources being shared. Some of the services included in this layer are Discovery services, Co- allocation, scheduling and brokering services, Monitoring and Diagnostics services, Data Replication services, Community authorisation servers, etc.

The Application Layer. This layer comprises the user applications that operate within a VO (virtual organisation) environment.

Grid and Commodity Distributed Computing

In grid literature Commodity Distributed Computing refers to existing distributed computing technologies. In the past decade grid and commodity computing have evolved in parallel, but with different goals and focus. There exists therefore some overlap between the grid and commodity computing technologies, but essentially, they are complementary.

Investigation on how the two worlds of grid and commodity computing can be combined has resulted in Commodity Grid Toolkit (CoG Kit). These CoG toolkits allow for the use of already familiar technologies, e.g., CORBA or Java, for grid services. A CoG Kit is defined as follows in [13]:

> A Commodity Grid Toolkit (CoG Kit) defines and implements a set of general components that map grid functionality into a commodity environment framework.

CoG Kits have been prototyped for different commodity distributed computing technologies. A brief description of some of the existing CoG Kits are given below.

Java CoG Kit. The components of the Java CoG Kit can be classified in categories, as depicted in Figure 2. The following components can be identified:

- **Low-level Grid Interface Components.** They provide mappings to commonly used Grid services, such as the Metacomputing Directory Service (MDS) and the resource management service (GRAM).

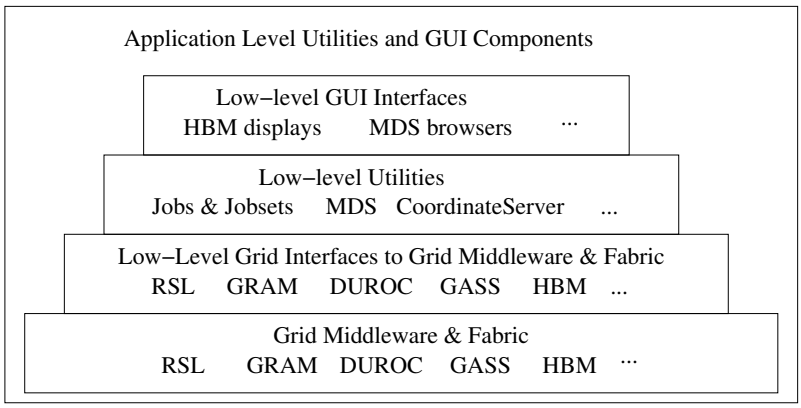


Figure 2: Categories of the Components of the Java CoG Kit

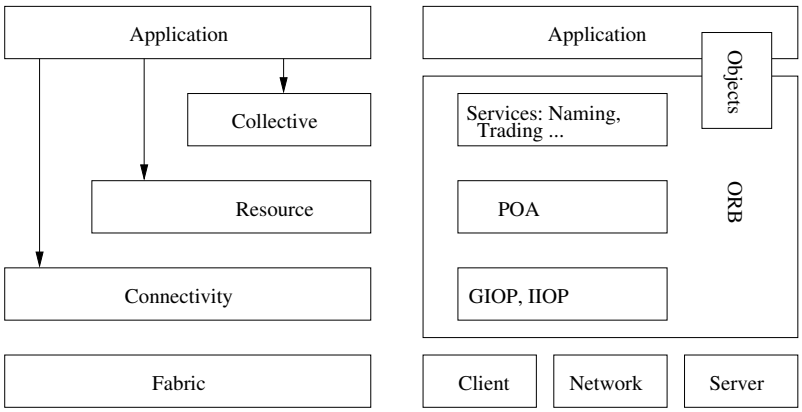


Figure 3: Relationship between the Layered Grid Architecture (left) and CORBA (right).

- **Low-level Utility Components.** They provide functionality needed in different types of applications, such as components that use the information service functions to find all computing resources that a user can submit to, or locate geographical coordinates of compute resources.
- **Low-level common GUI Components.** They provide low-level GUI entities that can be reused either by application developers or by components providing more sophisticated GUI features.
- **Application-specific GUI Components.** They provide more application domain oriented GUI entities and components.

CORBA CoG Kit. The relationship between CORBA CoG Kit and the grid architecture is illustrated in Figure 3. CORBA offers a number of features that can be exploited in a CoG Kit, such as modular programming mode, a number of available advanced services (e.g., security, naming, trading, event, transaction, etc.), interoperability, location transparency, etc.

Integration and interoperability between CORBA and grid applications/services can be achieved at two different levels:

- a high-level integration where CORBA wrappers are put around grid services, and
- a low-level integration wherein CORBA services are extended and new services are implemented to support grid applications.

In a CORBA CoG Kit a combination of the two levels is needed. The first level makes possible for CORBA application to make use of existing grid applications/services, while the second level will make CORBA a grid technology in its own right.

Work on the high-level integration has been going on and CORBA server objects that interface to services on the grid are developed, i.e., these server objects are wrappers around the corresponding Globus services. These server objects are accessible through CORBA's ORB (Object Resource Broker).

Other CoG Kits. In addition to the CoG Kits described above there exists Perl and Python CoG Kits as well. However, these are not directly related to distributed computing technologies. The Perl CoG Kit makes it possible to access resource information from the Web, and Python CoG Kit make the Globus toolkit available from the Python language.

1.8 Implementation of a Grid Service

The relationships between OGSA, OGSI, Web Services, and Grid Services has been explained in Section 1.6. Since a Grid Service is an extension of a Web Service, we start with explaining how ordinary Web Services are implemented:

Implementing a Web Service. A Web Service consists of a `WSDL` file which is available on the web, and which describes in detail how a service looks like. This includes the description of available methods, in- and out-parameters, how complex objects are defined, etc. Additionally, stub classes on both client and server side are necessary in order to facilitate a communication with a Web Service. These stubs are typically generated automatically.

In order to implement a `WSDL` file tools are available. As an example, we show how to implement an interface in Java, and then use the tool `Java2wsdl` from the axis-library of Apache.

```
public interface ImageProcess {
    void addImage(byte[] image);
    byte[] getImage();
}
```

From the interface that is to be exposed as a Web Service the WSDL file is generated. After the WSDL file is generated, the interface (i.e., the actual service) is implemented:

```
public class ImageProcessImpl extends GridServiceImpl
                              implements ImageProcessPortType  {
   public void addImage(byte[] in0) throws RemoteException {
          // post to java space
```

```
   }

   public byte[] getImage() throws RemoteException {
          //  get from java space when finished
   }
}
```

Stub classes must be generated in order to communicate using SOAP. A tool from the axis-library of Apache can be used to generate these from the WSDL file. Then a deployment descriptor, a so-called WSDD is generated:

```
<?xml version="1.0"?>
<deployment name="defaultServerConfig"
            xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

<service name="tutorial/core/image/ImageProcessService" provider="java:RPC">
   <parameter name="allowedMethods" value="*"/>
   <parameter name="className" value="gt3tutorial.core.image.impl.ImageProcessImpl"/>
</service>
</deployment>
```

The class files and the XML descriptors are packed to gether in a Java archive, and can be deployed on the application server, e.g., Tomcat.

In order to avoid typing all commands manually, we use *ant* as a build-system for Java. Since the Globus Toolkit uses *ant*, we use the GT files for this task. However, these ant-files require that the directories of the Globus Toolkit are used with a similar structure, this is somewhat inconvenient for a user. This being said, the method works well as long as the rules for the Globus Toolkit are followed.

Implementing a Grid Service. A Grid Service is an extension of a Web Service. While a Web Service is a rather large stateless service shared by all users, a Grid Service has a central factory which has the responsibility of a number of service instances. First a client contacts the factory in order to get a new instance of a service. Then the client uses the service directly from the server of the service. This service can be stateful.

In order to implement a real Grid Service, the interfaces and the WSDL file must be generated as before. However, for a stateful service a method to post the problem to the service must be implemented together with a method for getting the result when the solution is available. The biggest change is in the deployment descriptor.

```
<?xml version="1.0"?>
<deployment name="defaultServerConfig" xmlns="http://xml.apache.org/axis/wsdd/"
            xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

<service name="tutorial/core/factory/MathFactoryService" provider="Handler"
         style="wrapped">
   <parameter name="name" value="MathService Factory"/>
   <parameter name="instance-name" value="MathService Instance"/>
   <parameter name="instance-schemaPath"
```

```
      value="schema/gt3tutorial.core.factory/Math/MathService.wsdl"/>
   <parameter name="instance-baseClassName"
      value="gt3tutorial.core.factory.impl.MathImpl"/>

   <!-- Start common parameters -->
   <parameter name="allowedMethods" value="*"/>
   <parameter name="persistent" value="true"/>
   <parameter name="className" value="org.gridforum.ogsi.Factory"/>
   <parameter name="baseClassName"
              value="org.globus.ogsa.impl.ogsi.PersistentGridServiceImpl"/>
   <parameter name="schemaPath" value="schema/ogsi/ogsi_factory_service.wsdl"/>
   <parameter name="handlerClass" value="org.globus.ogsa.handlers.RPCURIProvider"/>
   <parameter name="factoryCallback"
              value="org.globus.ogsa.impl.ogsi.DynamicFactoryCallbackImpl"/>
   <parameter name="operationProviders"
              value="org.globus.ogsa.impl.ogsi.FactoryProvider"/>
</service>
</deployment>
```

As before, the class files and the XML descriptors are packed together in a Java archive with the extension `.gar` for a grid archive. This is deployed to an application server, e.g., a modernised Tomcat server.

There are two possible implementation approaches for a Grid Service: It can be implemented either by inheriting from a skeleton class or by using a delegation model, where incoming calls are delegated to a series of classes called operation providers.

- Life cycle management: Grid Services provide the necessary tools, such as callback functions during special moments in a Grid Service's lifetime (creation time, destruction time, etc.), to effectively manage its life cycle (for example, to make Grid Services persistent).
- Service Data: A Grid Service can have a set of associated service data that describes it. This is not to be confused with WSDL, which describes details like methods, protocols, etc. Service data is particularly useful to index Grid Services according to their characteristics and capabilities.
- Notifications: We can configure a Grid Service to be a notification source, and certain clients to be notification sinks (or subscribers). This means that if a change occurs in the Grid Service, that change is notified to all the subscribers. In the MathService example, suppose that all the clients perform certain calculations using a variable called InterestingCoefficient which is stored in the Grid Service. Any of the clients can modify that value to improve the overall calculation. However, all clients must be notified of that change when it occurs. We can achieve this easily with the Grid Services notifications.

Implementation of a client.

```
public class MathClient {
   public static void main(String[] args) {
      try {
         // Get command-line arguments
         URL GSH = new java.net.URL(args[0]);
```

```
            int a = Integer.parseInt(args[1]);

            // Get a reference to the MathService instance
            MathServiceGridLocator mathServiceLocator = new MathServiceGridLocator();
            MathPortType math = mathServiceLocator.getMathService(GSH);

            // Call remote method 'add'
            math.add(a);
            System.out.println("Added " + a);

            // Get current value through remote method 'getValue'
            int value = math.getValue();
            System.out.println("Current value: " + value);
         } catch (Exception e) {
            System.out.println("ERROR!");
            e.printStackTrace();
         }
      }
   }
```

Conclusions. We showed in this chapter how to implement a Grid Service by implementing a Java interface, and generating a WSDL file. While the use of a toolkit is convenient, it can be necessary to manually implement all files in order to make use of all possibilities. The people behind the Globus Toolkit have implemented their own variant of WSDL, called GWSDL, which contains some extensions to be included in the next version of WSDL.

Since there are restrictions on how to implement a Grid Service (e.g., same internal structure and location in the Globus Toolkit directories), the process of generating a Grid Service is still too complicated. Work must be done in order to make this process easier, and suitable for the mass market.

1.9 Related Technologies

Several technologies are related to grid; sometimes even confused for being identical to grid. Therefore, we give a short overview of some selected technologies.

Autonomic Computing. The term "Autonomic Computing" [14] is defined to be an approach to self-managed computing systems with a minimum of human interference. Autonomic Computing deals in the first line with research areas that emerge for managing highly complex systems, as well as technical progress in developing embedded devices. Applications are within many areas of information processing, including telecommunication and security. Major players in the market use this area as a major challenge in future computing.

An autonomic system has the following properties:

- Several aspects of autonomic systems (processing power, memory, transfer, etc.) are distributed in space, while data transfer between these is performed with certain properties (QoS, loosely coupled systems, peer-to-peer transfer, etc.).
- The system has no centralised control of tasks.

The goal of autonomic computing is to realise the promise of information technology: increasing productivity while minimising complexity for users by making computing systems capable of running themselves, adjusting to varying circumstances, and preparing their resources to handle most efficiently the workloads the users put upon them. This field emerges for several reasons:

- Many systems grow so big that the number of hierarchy levels gets too large, which causes inefficiencies. Additionally, hierarchical systems tend to be vulnerable to failures. Large systems in biology and nature follow a more decentralised structure, which could be an inspiration for this work.
- Hardware progress is a factor for computers being used everywhere, often as embedded entities. There is an indication of these embedded devices being networked, likely in a rather decentralised manner.
- Todays communication devices (e.g., mobile phones) work in a rather centralised way (GSM), which works well in areas where infrastructure is available, and the providers have a commercial interest. On the other hand, WLAN and peer-to-peer technology emerge in the decentralised pattern, which can give more control and independence to the individual, while the area of operation can be enlarged and costs for the user could be reduced.

Peer-to-Peer. Peer-to-Peer (P2P) [15] technologies are built on the paradigm of equal partners. P2P technologies can be used for communication, file sharing, computing, and many other areas. One definition of Peer-to-Peer system (according to searchNetworking.com) is as follows:

> Peer-to-peer is a communications model in which each party has the same capabilities and either party can initiate a communication session. Other models with which it might be contrasted include the client/server model and the master/slave model. In some cases, peer-to-peer communications is implemented by giving each communication node both server and client capabilities. In recent usage, peer-to-peer has come to describe applications in which users can use the Internet to exchange files with each other directly or through a mediating server.
>
> On the Internet, peer-to-peer (referred to as P2P) computers can form a type of transient Internet network that allows a group of computer users with the same networking program to connect with each other and directly access files from one another's hard drives. Napster and Gnutella are examples of this kind of P2P software. Corporations are looking at the advantages of using P2P as a way for employees to share files without the expense involved in maintaining a centralised server and as a way for businesses to exchange information with each other directly.

Eternity systems. An eternity system [16, 17] is a peer-to-peer system that focuses on persistent and secure storage of data for a long time period. Another issue is the storage of data that are highly available and durable.

Ad-hoc networks. Many devices have the possibility for ad-hoc networking via Bluetooth, WLAN, or other technologies. These devices can be used for building a temporary communication infrastructure. Ad hoc networks can be used for messaging applications (e.g., HikerNet [18]), ad-hoc networked multimedia messaging, or other technologies to build infrastructure cooperatively without using standard infrastructure.

Sensor Networks. Sensor networks [19, 20, 21] are made of a group of sensors exchanging information with each other, and the environment. The devices have very limited capabilities with respect to processing power, communication, and even power consumption. There are many applications, including environmental techniques. Such networks can also be formed by **embedded devices**.

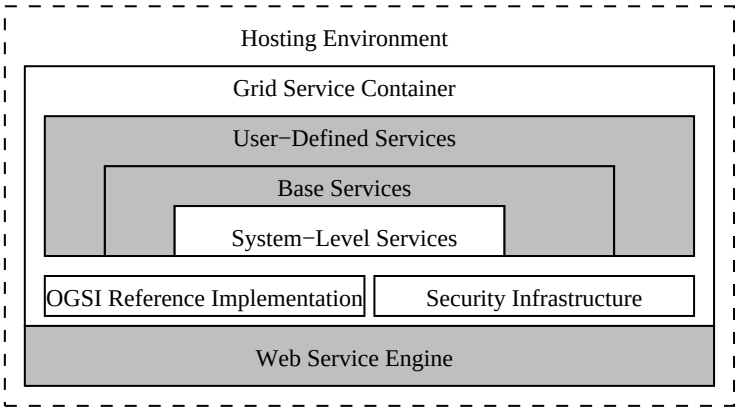


Figure 4: Kernel Architecture of GT3

2 Globus Toolkit

We take a closer look at the Globus Toolkit (often referred to as GT) as one concrete technology used to build grids. The Globus Toolkit (see www.globus.org) is developed by the Globus Alliance, and used as a testbed for academic research, and is the platform for some industrial grid infrastructures.

The Globus Toolkit is a collection of numerous tools for implementing, administrating, and maintaining grid services, not one single monolithic application. Moreover, the Globus Toolkit depends on a vast number of third party tools. Therefore, the GT consists of a high number of components; the distributed install-file is very large.

The Globus Toolkit is suitable to build grid infrastructures, and be used by grid applications. The GT3 is based on Grid Services, as outlined in Section 1.8, and implements OGSI (and OGSA).

2.1 Infrastructures of the Globus Toolkit

The Globus Toolkit implements the infrastructures mentioned in Section 1.6. The **Information Infrastructure** includes the MDS (Globus Monitoring and Discovery Service) of GT2, and an LDAP-based index service for GT3: the Grid Resource Information Service (GRIS) allows querying resources for their current status, while the Grid Index Information Services (GIIS) is knitting together arbitrary GRIS services.

For **Resource Management** the GT2 uses the Grid Resource Allocation Management (GRAM), while the GT3 uses the Master and Managed Job Factory Service (MMJFS), whose task is resource allocation, submitting jobs, and managing job status and progress. For **Data Management** the GT2 uses GridFTP (Grid Frile Transfer Protocol), while the GT3 employs the Reliable File Transfer Service (RFT).

The **Security Infrastructure** with the GSI is based on PKI standards. Transport-level security uses the httpg-protocol[7] based on https, while message-sevel security is based on SOAP, and provides security per message.

[7]The httpg-protocol phased out!

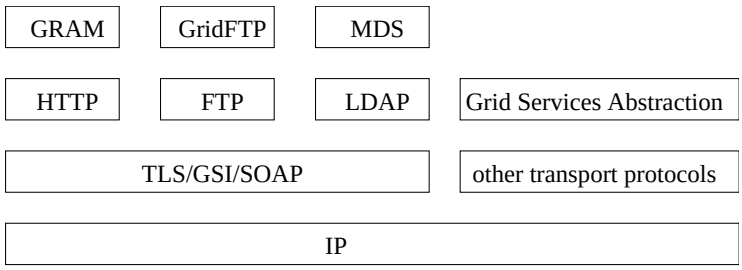


Figure 5: Protocols of Globus Toolkit 2

2.2 Globus Toolkit applications

Traditionally, the Globus Toolkit has been used in (academic) high performance scientific computing, by gluing together computational resources in one grid. Examples of such grids are the Nordic *NorduGrid* with core in Sweden www.nordugrid.se, the Norwegian NorGrid, etc. The middleware used in many cases is the Globus Toolkit version 2. In most implementations third-party additions were necessary for the specific task, replacing some parts of the grid middleware with other developments.

The academic partners of the Globus Alliance are currently mainly interested in computational applications. Globus Toolkit testbeds include National Technology Grid, The European DataGrid, NASA Information Power Grid, ASCI Distributed Resource Management (DRM) Testbed, GUSTO, and EMERGE.

Applications where the Globus Toolkit is used include (list from www.globus.org):

- Cactus - A simulation using parallel code to visualise black holes colliding in space.
- MM5 - A meteorological simulation program that uses networked computing resources via Globus.
- Nimrod - A tool that uses networked resources to perform parameter studies on distributed computing resources.
- Tardis - A three-dimensional simulation of astrophysical thermonuclear flashes.
- Neph - A simulation that takes satellite data and visualises weather patterns.
- HTB - High-throughput computing applied to quantum Monte Carlo simulation.

IBM have installed the Globus Toolkit in several computational grids, e.g., in the AIST project for all super-computer resources in Japan. One of IBM's high profile projects is Butterfly.net, a solution for massive-multi-player online games, where the Globus Toolkit is used to distribute tasks within a large server park. The fundamental problem solved with a grid solution is distribution of jobs between the servers. Normally this would have been solved by a server cluster, but a a grid solution offers higher degree of flexibility and easier upgrade and maintenance of the hardware and software resources.

2.3 Installation

The installation of the Globus Toolkit packages is a quite laborious task. The installation includes an *Apache* web-server. Most of the other parts are Java-based. An example of

an installation procedure can be found on http://www-unix.globus.org/toolkit/3.0/ogsa/docs/admin/example.html. IBM have published a very comprehensive installation guide [22].

2.4 Grid Security

Grid applications cross traditional organisation boundaries. Standard security thinking has been focused on protecting system boundaries, e.g. with a limited number of externally available machines, firewalls, and intrusion detection (IDS), while information flow has been free inside the organisation. In the same way the standard security concepts as identity, authentication, and authorisation are just meaningful inside the organisation, **Interoperability** must be possible with various kinds of security implementations.

While working with the Globus toolkit, we concluded that there were some missing parts regarding grid security. The Globus Toolkit security infrastructure lacks the following:

- Role-based access control, which could be achieved using a policy server that takes the user certificate as input and generates a valid role certificate with limited lifetime.
- The security could be integrated with a resource broker that keeps track of which and how much a resource is accessed. The broker is a part of the Grid accounting system.

Grid Security Infrastructure (GSI) is a public-key-based security protocol, using X.509 certificates, a widely employed standard. The protocol provides single sign-on authentication, which allows a user to create a proxy credential that can authenticate with any remote service on the user's behalf, as well as communication protection and initial support for restricted delegation.

2.5 The Security Infrastructure in GT3

The Globus Toolkit security uses GSS-API (Generic Security Services Application Programming Interface) which is, as the name implies, a security infrastructure interface independent of programming language and underlying security mechanism. Currently, there are two choices of underlying security mechanisms: Kerberos and GSI (Globus Security Infrastructure) of the Globus Toolkit, which is used in most cases.

GSI is a public-key infrastructure based on standard X.509 certificates [23]. The Certificate Authority (CA) in the infrastructure is independent from the grid installation. For our experimental grid environment we chose to have our own local grid CA, with self-signed certificates.

The authentication of a grid-service request consists of two separate parts: host-authentication and user-authentication. To be allowed to run a remote grid service, the user must be accepted by the remote machines, and the grid user must be mapped to a local user on the remote machine. Therefore, two certificates are necessary: one for the user, and one for the host. The user certificate is placed in the `.globus/` directory of the user's home directory while the root-owned host certificate is located at `/etc/grid-security/`.

The installation procedure of a local grid CA is well-described in [24]. However, we encountered some problems during the installation and test-use of the package:

- User confusion. The grid CA can be any user. Since we installed the Globus Toolkit as root, and the GT3 CA uses many of the GT3 build tools, several of the installation steps must be performed as root.
- Both the user certificate and the host certificates must be signed for a particular machine to become part of a grid. User certificates are put at the `$HOME/.globus` directory, host certificates are put (by root) at `/etc/grid-security/`.
- Time synchronisation is important. Should the clocks not be synchronised proxy-grid services can be repelled.
- The Grid Resource Manager (GRAM) is pedantic with regards to user permissions and responds with errors if the permissions or owner of a key file or certificate is not as expected.
- Correct setting of environment variables. Before the grid security services are run, make sure `LD_LIBRARY_PATH=/opt/gt3/lib`.

2.6 Signing a Certificate for the Globus Toolkit

We present a hands-on description on how to sign self-signed certificates. In our example we set up two machines *glad.nr.no* and *laks.nr.no* running Linux. The CA of the NR test-grid is located at *glad.nr.no* and the CA is the user *jornv*. First get the CA package `globus_simple_ca_[CA-HASH]_setup.tar.gz`, where the `CA-HASH` is characteristic for this particular CA.[8]. To install the package type

```
% /opt/gt3/sbin/gpt-build globus_simple_ca_[CA-HASH]_setup.tar.gz
% /opt/gt3/sbin/gpt-postinstall
```

To generate a sign-certificate request type

```
% grid-cert-request
```

When the sign certificate request is generated the program asks for the users proper name. The certificate request will create a *.globus* directory in the user's home directory. The *.globus* directory contains three files:

- *userkey.pem* - the (private) user key,
- *usercert_request_.pem* - a certificate request directed to the specified CA, and
- *usercert.pem* - an empty file which later will contain the user's own certificate.

The sign-certificate request must be sent to the CA. In our special case, the CA is a system user and can directly fetch the certificate request file and sign the certificate.[9] After receiving the signed certificate in return from the CA, overwrite the file *.globus/usercert.pem*. Be sure the replaced file has the correct user permissions; with wrong permissions (or owner) the GT3 authentication will not work. The last step before the certificate is active is to map the GT3 user to a Unix user, which is handled by the map file */usr/grid_security/grid-mapfile* where each line contains the GT3-user and a Unix user. This step must be performed by the local system administrator. To get your GT3 user-identity write

[8] Example: the jornv@glad CA's hash is 1bf594a1

[9] Usually, email will be used to send the certificate.

```
% grid-cert-info -subject
```

At this stage you are able to run grid services on one single machine. For running the GT3 grid services on several machines we have two options: On rather homogeneous domains with identical users on other machines, simply copy the entire *.globus* directory to the new machines, and update the local grid-mapfile. The other option is to have a new certificate on each machine. However then there cannot be two different GT3-users with same CA, the same proper name, and same domain. This problem can be handled by using a different proper name on each machine.

To test your freshly created GT3 grid-users you must first start a grid proxy with the command *grid-proxy-init* and then you can for example run

```
sei@laks> globusrun -o -r "glad.nr.no" '&(executable=/bin/hostname)'
```

which will hopefully output the name of the remote machine.

2.7 Example of Globus Toolkit usage

We present a simple example for distribution of one single C program to be run on a remote machine using the Globus Toolkit. The Globus Toolkit command *globusrun* is used to distribute and run the application and two parameters must be specified: the machine to run the application and the machine to fetch the executable from. The file can be transfered with a variety of protocols. In our example the *gridFTP* protocol is used and the executable is located at the same machine as call *globusrun* from.

First we implement our example application. In our case we use a "hello world" C-program, hence the executable is a binary and the program can only be distributed to machines with the same architecture. Let the local machine be *laks.nr.no* and the binary to distribute is */home/mio/hello*. Then the command can be run at the remote machine *glad.nr.no* with the command:

```
% globusrun -r glad.nr.no -o '&(executable=gsiftp://laks.nr.no/home/mio/hello)'
```

3 Grid Dot Net

In this section we describe how .Net can be used to implement an OGSI compliant grid application. The OGSI standard represents an extension of the current Web Service standard that include stateful Web Services, inclusion of mechanisms for asynchronous notification, a definition of how to reference instances of services and introduction of service data. Since one of the design goals of the OGSI standard was to enable exposure of functionality in a homogeneous and platform independent way, OGSI compliant Web Services in principle can be implemented on any platform.

3.1 Web Services in .Net

.Net is (among other things) the name of Microsoft's new development platform. It is envisioned that Web Services will be one of the key architectural elements of distributed application built on this platform. The support of Web Services is extensive in the standard .Net class library and in the development tools.

3.2 Implementation philosophies in Java and .Net

It is possible to develop OGSI compliant services based exclusivly on the standard support for Web Services in the libraries and tools. Using this strategy, all the methods required by OGSI must be implemented. State management is not a part of the current implementations and must be handled by the developer.

To make development of web services easier, a common strategy is to implement a set of classes and base services for the application developer to extend. This strategy is used in the the Java implementation provided by the Globus Toolkit. In the Globus implementation they have followed the same philosophy as with other java containers: The user have access to a set of base classes to extend when the functionality of the Grid Service is implemented. In addition the developer needs to write a set of XML files that the container application can read to establish the service when the application is introduced in the container.

The design philosophy in .Net is quite different. Instead of using external XML files to configure the application container it uses custom attributes. Custom attributes can be regarded as meta data connected to classes, methods and fields. To define a new attribute type the developer defines a new class as a subclass of the Attribute class. The naming of the attribute class must follow the naming convention <name>Attribute. Constructors have to be provided to allow setting of the meta data values. Getter and setter methods should also be provided to allow communication with the outside world. After an attribute type is introduced it is possible to use it to provide meta-information related to classes, methods and fields. To define attributes the developer uses the syntax:

```
[MyAttribute(SomeType)]
public int myMethod(){
}
```

The effect of this is to attach an attribute of type `MyAttributeAttribute` with the value defined by SomeType to the method myMethod. The attribute and the attribute values are available through introspection after compilation of the class.

The .Net platform uses custom attributes to facilitate the development of Web Services. The Web Service implementation uses Microsoft Internet Information Server as a container and has the same structure as a standard ASP .Net application. To expose a Web Service the developer must declare a subclass of the provided Web Service base class and mark the methods to be exposed with the attribute [WebMethod]. When the application is deployed on the server the container can inspect the assembly by the means of introspection, find the methods to be exposed and generate the necessary code to expose the service. Visual Studio .Net provides good support for Web Service development.

3.3 Implementations of OGSI compliant Web Services in .Net

Because of the general support for Web Services in .Net, both on the platform side and in the development tools, there exist several projects with the ambition to enhance the basic functionality of the platform in order to make development of OGSI-compliant services easy. There are at least two stable implementations of such frameworks: the MS.NetGrid project by EPCC in Edinburgh and the OGSI.Net implementation of University of Virginia.

3.4 MS.NetGrid as basis for OGSI-compliant Web Services

We show how the MS.NetGrid can serve as a basis for the implementation of OGSI compliant Web Services. The implementation of MS.NetGrid is available from `http://www.epcc.ed.ac.uk/~ogsanet/software.html`. The download contains the source code and the necessary project files, and can be installed following the installation instructions.

To demonstrate the use of the framework we developed a small service that transforms an image and sends it back to the client. The implementation follows a factory pattern where a persistent factory instantiates a transient service.

To implement the service we must write the transient service class as a subclass of TransientGridServiceInstanceAspProxy. The methods exposed by the Web Service are implemented here with an attached custom attribute [WebMethod], to indicate to the application container to expose it as a Web Service. The TransientGridServiceInstanceAspProxy class is a subclass of .Net´s WebService class and adds functionality to do calls on port types using late binding by the method CallMethodOnPortType. This method uses the method invocation facility in the introspection mechanisms of .Net. The net effect of the call is that an object of the portType defined by the string is instantiated and that the method defined in the second parameter is executed. The method returns an object containing the return value of the call, parameters to the call are provided by the means of an object array.

The actual implementation of the functionality is done in a subclass of PortBaseType. This class is instantiated by the transient service class. Since late binding is used in the method invocation no type checking can be done at compile time.

The factory objects are implemented using the same strategy. Custom property must be provided to indicate the services to create.

To implement a client that uses the Web Service we use the .Net tool WSDL. This tool generates the necessary proxy classes on the client side when invoked with the URL of the WSDL file as a parameter. The generated classes contain the functionality needed to access the Grid Service.

3.5 Execution of user-provided code

Another interesting experiment done on the .Net implementation of the OGSI framework was execution of user-provided code. In the experiment an assembly was compiled on the client side and sent to an OGSI compliant service. In the service the assembly was loaded and executed and the result was sent back to the client. The implementation of such a service was quite simple given the infrastructure of the MS.NetGrid and the introspection and late-invocation facilities of the .Net platform. The rich security model of the .Net platform makes it possible to execute unknown code on the server side without compromising server security.

3.6 Conclusions

The main focus of MS.NetGrid implementation has been to test how the .Net architecture can be used to implement a framework for .Net services. From a developer's perspective, the implementation provides a good starting point for the development of OGSI services on .Net. Configuration is done with custom attributes which in most cases gives cleaner and less error prone code compared to the globus/java configuration based on XML configuration files. On the down side it can be mentioned that the framework, in some cases, integrates poorly with the Visual Studio tool. The most annoying being that the tool refuses to show any file containing a class that is a sub-class of the .Net WebService class.

The main focus in the MS.NetGrid implementation has been feasibility testing and is not meant to be used in mission critical implementation. It is not to be expected that it is stable enough to host mission critical applications, especially since the applications are executed under the control of Internet Information Service.

References

[1] Viktors Berstis. Fundamentals of Grid Computing. Red paper, IBM, 2002. http://www.redbooks.ibm.com/redpapers/pdfs/redp3613.pdf. 1

[2] Ian Foster and Carl Kesselmann. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, Elsevier, 1998. 3

[3] Ian Foster. What is the Grid? A Three Point Checklist. Web pages, http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf, 2002. 3

[4] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid. *Intl J. Supercomputer Applications*, 2001. http://www.globus.org/research/papers/anatomy.pdf. 4, 13

[5] Brian MacKinnon. Commercial computational grids: A road map. Web pages, http://www.acm.org/ubiquity/views/b_mackinnon_1.html, 2003. 5

[6] A. Reinefeld and F. Schintke. Concepts and Technologies for a Worldwide Grid Infrastructure. In *Euro-Par 2002, LNCS 2400*, pages 62–71. Springer, Aug. 2002. 6

[7] Wolfgang Leister, Thomas Maus, Heinrich Müller, Burkhard Neidecker, and Achim Stößer. Occursus cum novo – computer animation by ray tracing in a network. In N. Magnenat-Thalmann and D. Thalmann, editors, *New Trends in Computer Graphics*, pages 83–92. Springer Verlag, 1988. 7, 9

[8] NS-ISO/IED 17799. *Informasjonsteknologi: Administrasjon av informasjonssikkerhet (ISO/IEC 17799:2000)*. Norsk Standard, 2000. 7

[9] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. http://www.globus.org/research/papers/ogsa.pdf, june 2002. 11

[10] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, P. Vanderbilt, and D. Snelling. Open grid services infrastructure (ogsi) version 1.0. Draft recommendation 6/27/2003, Global Grid Forum, 2003. http://www.globus.org/research/papers/Final_OGSI_Specification_V1.0.pdf. 11

[11] M. Brandner, M. Craes, F. Oellermann, and O. Zimmermann. Grid Services. *Informatik Spektrum*, 27(2):129–135, April 2004. 12

[12] Ian Foster. Modeling Stateful Resources with Web Services. Whitepaper, http://www.globus.org/wsrf, 2004. 13

[13] Gregor von Laszewski, Ian T. Foster, and Jarek Gawor. CoG kits: a bridge between commodity distributed computing and high-performance grids. In *Java Grande*, pages 97–106, 2000. URL citeseer.ist.psu.edu/article/vonlaszewski00cog.html. 14

[14] IBM. Autonomic Computing: IBM's Perspective on the State of Information Technology. Web pages, http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf, 2001. 19

[15] M. O'Reilly. O'Reillys Peer-to-Peer Summit. Web pages, http://www.oreillynet.com/pub/a/linux/2000/09/22/p2summit.html, 2000. 20

[16] J. Anderson. The Eternity System. Web pages, http://www.cl.cam.ac.uk/users/rja14/eternity/eternity.html, 2000. 20

[17] L. Vepstas. The Next Few Decades of Computing. Web pages, http://linas.org/theory/eternity.html, 2001. 20

[18] W. Leister. HikerNet. Web pages, http://home.nr.no/~wolfgang/hikernet.pdf, 2001-2003. 21

[19] Wu chi Feng, Jonathan Walpole, Wu chang Feng, and Calton Pu. Moving towards massively scalable video-based sensor networks. In *Proc. Workshop on New Visions for Large-Scale Networks: Research and Applications*. March 12-14, 2001. 21

[20] David Steere, Antonio Baptista, Dylan McNamee, Calton Pu, and Jonathan Walpole. Research challenges in environmental observation and forecasting systems. In *Proc. Mobicom*, 2000. 21

[21] E. Culler and W. Hong (ed). Special issue on Wireless Sonsor Networks. *Communication of the ACM*, 47(6), june 2004. 21

[22] F. Luis, B. Jacob, S. Slevin, S. Sundararajan, M. Brown, J. Lepesant, and J. Bank. Globus Toolkit 3.0 Quick Start. Red paper, IBM, 2003. http://www.redbooks.ibm.com/redpapers/pdfs/redp3697.pdf. 24

[23] ITU-T. Recommendation x.509 (1997 e): Information technology - open systems interconnection - the directory: Authentication framework. Technical report, ITU-T, june 1997. 24

[24] Globus simple ca package. http://www.globus.org/security/simple-ca.html, 2003. 24

[25] SGI. SGI on the Grid, Visual Area Networking in Grid Environments. White Paper, www.sgi.com, 2002. 33

[26] SGI. SGI on the Grid, Unique Capabilities for Grid Computing. White Paper, www.sgi.com, 2002. 33

[27] SGI. Open GL Vizserver 3.1. White Paper, http://www.sgi.com/pdfs/3263.pdf, 2003. 33

[28] Wolfgang Gentzsch. Grid Computing Adoption in Research and Industry. In Ahmar Abbas, editor, *Grid Computing: A Practical Guide to Technology and Applications*, pages 309–339. Charles River Media, 2003. 35

A The Grid Workshop at NR

NR organised a workshop on the subject of grid on march 12^{th} for Norwegian companies. Several of the partners in the project held a presentation. The slides from the workshop are available at http://nr.no/pages/dart/grid_seminar_march_2004. We give a short overview on these presentations:

Wolfgang Leister, NR gives an introduction to grid technology as outlined in this report, and presents results from the grid project at NR so far.

Jan Petter Strømsheim, UFD, presents grid and high speed Internet initiatives at national, Nordic, and EU level. He presents an action line which consists of two parts:

- program for utilising the high speed research networks (building on the experiences of the Nordic Council of Ministers project NORDUnet2, the objective is to have a NORDUnet2-like program for the whole region).
- improved and upgraded inter-connection of the Russian and the accession countries high bandwidth research networks.

He also presents the NORDUnet3 initiative, which supports applications in Nordic Internet research, particle physics, environmental research, digital libraries, telemedicine, virtual universities, and eGovernment. Forskningsnet in Denmark, FUNET in Finland, RHnet in Iceland, UNINETT in Norway, and SUNET in Sweden are participating in NORDUnet3.

Within the 6th framework programme of the EU it is a clear objective to realise grid-empowered infrastructure to create new enhanced facilities for performing research and for fostering innovation in Europe. Programmes like EGEE (enabling grids for eScience in Europe) and DEISA (Infrastructure for supercomputing facility) is mentioned.

Loek Vredenberg, IBM, presented IBMs Grid Computing activities. IBM distinguish four types of grids: Processing Grids, Data Grids, Resiliency Grids, and On Demand Grids. Grid is a first step to on-demand computing, defined by business transformation, flexible financial and delivery offerings, and operating environment. Grids are built on integrated, open, virtualised, and autonomic environments and elements. Focus areas for grids include business analytics, engineering and design, research and development, government development, and enterprise optimisation. Several examples from IBM projects were presented.

The eDiaMoND project was selected as a show case, and implements a UK digital mamography national database. The goal is to replace traditional secreening techniques for UK breast screening with digital alternatives, giving seamless access to data for medical specialists, storing the data, and perform processing of the scans.

Fabrizio Magugliani, SGI, presents "SGI for the Grid: A Value-added Infrastructure for Science and Engineering". SGI focus on processing, visualising, and managing extraordinary amounts of data on the grid. The hardware and software used for

visualisation is fully grid-enabled, using mostly Linux on Altix and Onyx machines. He also presents the Visual Area Networking (VAN), which outsources storage and visualisation capacities, where the end user receives a constant video-stream over the Internet. Finally, he states that grids are built, not bought.

Andreas Botnen, USIT, started with presenting the NOTUR (Norsk Tungregning) activities, where USIT participates. The activity has its roots in astrophysics computations and particle physics. The grid paradigm was introduced in 2003 into NOTUR using GT2 as a middleware. GT2 appears to be a mature technology, while the successor GT3 is more at an experimental state. However, some functionality (e.g., broker functionality) is missing in GT2. For the next version of NOTUR GT3, GT4, and UNICORE are evaluated. An integration to FEIDE (authentication service by UNINETT) is planned, and a Nordic cooperation is maintained. The layout of the planned NorGrid cooperation was also presented.

B Cooperation with Companies

During the course of the project several companies who are major players in grid technologies supported our project. These companies provided us with information about their grid initiatives, and participated at the Grid Workshop at NR (see Section A).

B.1 SGI

We had several meetings with representatives from SGI (Office in Bergen), and SGI participated in the Grid Workshop at NR.

We looked into grid technologies presented by SGI (http://www.sgi.com) in several white papers [25, 26, 27]. White paper [27] promotes the OpenGL Vizserver computing solution, and Visual Area Networks (VAN). Visual Area Networking enables remote users and distributed teams to manipulate and visualise large data sets with a much higher speed than on a desktop system. Access from tablet computers, and handheld devices is possible, though these devices might not have the computing power for visualisation. The bandwidth needs for a visualisation application is constant and thus independent from the data set size during the visualisation. The VAN approach supports centralised visualisation servers, whose resources are used by many others. Centralisation gives advantages in terms of maintenance, and security.

To our opinion, the VAN approach is somewhat different from other grid approaches, since it uses centralised powerful resources. However, the metaphor of a remote grid resource being accessed still holds to a great extent, since virtualisation and seamless access to certain resources is provided by this concept.

The OpenGL Vizserver Architecture consists of a client and a server. While the server renders a compressed multimedia stream, the clients task is to show this stream, and forward input events to the server. This architecture enables also collaboration of several parties sharing the same data set.

Within the frame of the project we planned to use computers by SGI, which are located at SGI offices in Bergen, and accessible through UNINETT and the University of Bergen. The machines were

- SGI Altix 3700 8 CPU Itanium2 1.3Ghz, 64GB memory (Linux), see http://www.sgi.com/servers/altix/.
- SGI Onyx2 IR3 visualisation server; 4 CPUs 300Mhz MIPS R12000, One InfiniteReality3 graphics pipe. This system can run the SGI OpenGL Vizserver for testing of graphics applications over the network.

Within the frame of this current project SGI were interested that NR look into the subjects of CPU rendering over the Internet, heavy duty 3D-visualisation over the Internet (as implemented in the SGI OpenGL Vizserver), user interfaces for grid computing, and effective operation of a grid network. However, current funding situation of our project did not allow us to continue the concrete work on these issues.

B.2 IBM

We had several meetings with representatives from IBM (Office in Oslo), and IBM participated in the Grid Workshop at NR. We also prepared for a visit at IBM labs in the U.K., which unfortunately was cancelled.

According to their web pages (see http://www.ibm.com/grid/) IBM contributes to and uses grid technologies to a great extent. Related subjects, like **autonomic** computing are also part of IBM's portfolio. IBM uses own developments and the Globus Toolkit.

IBM's intraGrid, based on the Globus Toolkit, is a research and development grid that allows IBM to leverage many worldwide assets for research purposes, and to understand the complexities of managing a grid infrastructure on an enterprise scale. One example of IBM's use of grid technologies is the IBM Böblingen Lab Grid, composed of three IBM pSeries[tm] clusters running AIX and LoadLeveler, a cross-departmental grid used to run zSeries processor unit simulations. Jobs are submitted through a web portal, presenting users with the same interface as the one they used when running simulations on an isolated cluster. The WebSphere based portal uses the Globus Java CoG Kit to pre-select candidate queues for submitting each simulation, using Globus Metacomputing Directory Service. This pre-selection is based on cluster loads and job characteristics. Access to a shared DB2 database allows for the automated generation of proxy certificates and for the monitoring and reporting of user jobs.

According to IBM "Virtualisation" has been a key factor since the earliest days of electronic business computing. They draw the line from the mainframe (by creating virtual memory, virtual storage and the virtual processor) via timesharing (this development enabled the computer to do many processing jobs simultaneously for hundreds and eventually thousands of users) to grid solutions.

The figures IBM presents show that almost every organisation is sitting on top of enormous, unused computing capacity, widely distributed. Mainframes idle 40% of the time. Unix servers are actually "serving" something less than 10% of the time. And most PCs do nothing for 95% of a typical day, which is an intolerable situation for customers.

Current IBM customer references for grid include Butterfly.net, a development studio,

online publisher and infrastructure provider for massively multiplayer games that connect players on PC's, consoles and mobile devices. Butterfly Grid consists of two clusters of approximately 50 IBM [tm] xSeries[tm] servers running in IBM hosting facilities. Specialised game servers and database servers are fully meshed over high-speed fiber-optic lines, enabling transparent routing of players to different servers in the grid.

Another current reference for IBM Grid Computing is the University of Pennsylvania's National Digital Mammography Archive, which gives rapid retrieval of digital patient files from multiple locations in a secure environment. The University of Pennsylvania Grid manages this huge data volume, schedules traffic and encrypts all image and information transmission using portal systems running almost exclusively on IBM hardware - including sixteen distributed IBM Netfinity servers running Linux and Windows 2000.

B.3 Sun Microsystems

We had an email exchange with representatives from Sun Microsystems [28], and some talks with representatives in Sweden. Since Sun Microsystems developed the GridEngine which is used as a part to build grids, we were interested in looking into their technology. However, since we ran out of funding, we could not continue with this work.

B.4 First NorGrid Workshop

NR researchers participated in the First NorGrid Workshop in Oslo, see www.hpc.uio.no/hpc/NorGrid/. NorGrid is a Norwegian initiative where mostly academic organisations (UNINETT, universities, NOTUR) join to build a grid infrastructure.

Index