Concepts and Technologies for a Worldwide Grid Infrastructure

Alexander Reinefeld and Florian Schintke

Zuse Institute Berlin (ZIB)
{ar,schintke}@zib.de

Abstract. Grid computing got much attention lately—not only from the academic world, but also from industry and business. But what remains when the dust of the many press articles has settled? We try to answer this question by investigating the concepts and techniques grids are based on. We distinguish three kinds of grids: the HTML-based *Information Grid*, the contemporary *Resource Grid*, and the newly evolving *Service Grid*.
We show that grid computing is not just another hype, but has the potential to open new perspectives for the co-operative use of distributed resources. Grid computing is on the right way to solve a key problem in our distributed computing world: the discovery and coordinated use of distributed services that may be implemented by volatile, dynamic local resources.

1 Three Kinds of Grids

Grids have been established as a new paradigm for delivering information, resources and services to users. Current grid implementations cover several application domains in industry and academia. In our increasingly networked world, location transparency of services is a key concept. In this paper, we investigate the concepts and techniques grids are based on [7,8,10,13,17]. We distinguish three categories:

- Information Grid,
- Resource Grid,
- Service Grid.

Figure 1 illustrates the relationship and interdependencies of these three grids with respect to the access, use and publication of meta information.

With the invention of the world wide web in 1990, Tim Berners-Lee and Robert Calliau took the first and most important step towards a global grid infrastructure. In just a few years the exponential growth of the Web created a publicly available network infrastructure for computers—an omnipresent **Information Grid** that delivers information on any kind of topic to any place in the world. Information can be retrieved by connecting a computer to the public telephone network via a modem, which is just as easy as plugging into the electrical power grid.

B. Monien and R. Feldmann (Eds.): Euro-Par 2002, LNCS 2400, pp. 62–71.

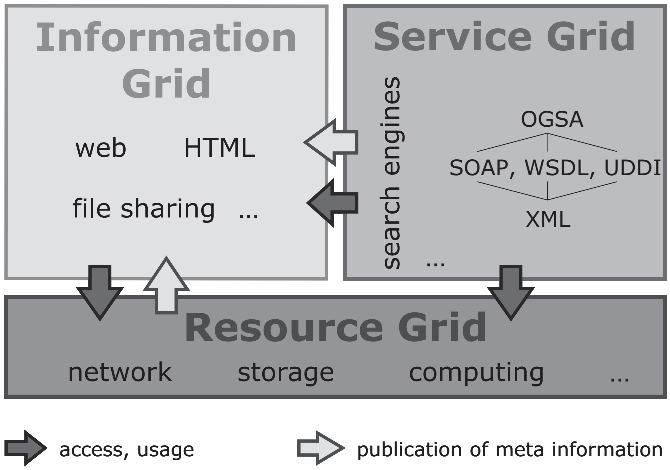

Fig. 1. The three Grids and their relation to each other.

File sharing services like *Gnutella, Morpheus* or *E-Donkey* are also part of today's Information Grid. In contrast to the Web, the shared data is not hosted by an organization or Web site owner. Rather, the file sharing service is set up by individuals who want to exchange files of mp3 audio tracks, video films or software. The bartering service is kept alive by the participants themselves; there is no central broker instance involved. Data is simply referenced by the filename, independent of the current location. This is a distributed, dynamic, and highly flexible environment, which is similar to the *Archie* service that was used in the early years of the Internet to locate files on ftp servers for downloading.

The **Resource Grid** provides mechanisms for the coordinated use of resources like computers, data archives, application services, and special laboratory instruments. The popular *Globus* toolkit [6], for example, gives access to participating computers without the need to bother which computer in the network is actually being used. In contrast to the data supplied by the Information Grid, the facilities of the Resource Grid cannot be given away free of charge and anonymously but are supplied for authorized users only. The core idea behind the Resource Grid is to provide easy, efficient and transparent access to any available resource, irrespective of its location. Resources may be computing power, data storage, network bandwidth, or special purpose hardware.

The third kind of grid, the **Service Grid**, delivers services and applications independent of their location, implementation, and hardware platform. The services are built on the concrete resources available in the Resource Grid. A major point of distinction between the last two grids lies in their different abstraction level: The Service Grid provides abstract, location-independent services, while the Resource Grid gives access to the concrete resources offered at a computer site.

2 Current Status of the Three Grids

2.1 Information Grid

Since its invention in 1990, the Information Grid became one of the biggest success stories in computer technology. With a data volume and user base steadily increasing at an extremely high pace, the Web is now used by a large fraction of the world population for accessing up-to-date information.

One reason for the tremendous success of the Web is the concept of the *hyperlink*, an easy-to-use reference to other Web pages. Following the path of marked hyperlinks is often the fastest way to find related information without typing. Due to the hyperlink the Web quickly dominated ftp and other networks that existed long before. We will show later how this beneficial concept can be adapted to the Resource and Service Grids.

Another important reason for the success of the Information Grid lies in the easy updating of information. Compared to traditional information distribution methods (mail of printed media), it is much easier and more cost-effective for vendors to reach a wide range of customers through the web with up-to-date information.

2.2 Resource Grid

The Internet, providing the means for data transfer bandwidth, is a good example of a Resource Grid. Wide area networks are complex systems where users only pay for the access endpoint, proportional to the subscribed bandwidth and the actual data throughput. The complex relationship between the numerous network providers whose services are used for transmitting the data within the regional networks are hidden from the user. Note that the Internet and other wide area networks were necessary for and pushed by the development of the Web.

Other Resource Grids are more difficult to implement and deploy because resources are costly and hence cannot be given away free of charge. *Computational Grids* give access to distributed supercomputers for time-consuming jobs. Most of them are based on the Globus toolset [6], which became a de-facto standard in this field. Today, there exist prototypes of application-specific grids for CFD, pharmaceutical research, chemistry, astrophysics, video rendering, post production, etc. Some of them use Web portals, others hide the grid access inside the application.

Data Grids provide mechanisms for secure, redundant data storage at geographically distributed sites. In view of the challenges of storing and processing several Petabytes of data at different locations, for example in the EU Datagrid project [4] or in satellite observation projects, this is an increasingly demanding subject. Issues like replication, staging, caching, and data co-scheduling must be solved. On the one hand, the quickly growing capacity of disk drives may tempt users to store data locally, but on the other hand, there are grand challenge projects that require distributed storage for redundancy reasons or simply because the same data sets are accessed by thousands of users at different sites [4].

For individuals *Sourceforge* provides data storage space for open source software projects and *IBP (Internet Backplane Protocol)* [1] provides logistic data management facilities like remote caching and permanent storage space.

Embarrassingly parallel applications like SETI@home, Folding@HOME, fightcancer@home, or distributed.net are easily mapped for execution on distributed PCs. For this application class, no general grid middleware has been developed yet. Instead, the middleware is integrated in the application which also steers the execution of remote slave jobs and the collection of the results. One interesting aspect of such applications is the implicit mutual trust on both sides: the PC owner trusts in the integrity of the software without individual checking authentication and authorization, and the grid software trusts that the results have not been faked by the PC owner.

Access Grids also fall into the category of Resource Grids. They build the technical basis for remote collaborations by providing interactive video conferences and blackboard facilities.

2.3 Service Grid

The Service Grid comprises services available in the Internet like search engines, portals, active server pages and other dynamic content. Email and authorization services (GMX, Hotmail, MS Passport) also fall into this category. They are mostly free of charge due to sponsoring or advertising.

The mentioned services are separate from each other without any calling interface in between. With web services and the *Open Grid Service Architecture OGSA* [8], this state of affairs is currently being changed. Both are designed to provide interoperability between loosely coupled services, independent of their implementation, geographic location or execution platform.

3 Representation Schemes Used in the Three Grids

Because of the different characteristics the representation schemes in the three grids have different capabilities and expressiveness.

In the **Information Grid** the hypertext markup language HTML is used to store information in a structured way. Due to its simple, user-readable format HTML was quickly adopted by Web page designers. However, over the time the original goal of separating the contents from its representation has been more and more compromised. Many Web pages use non-standard language constructs which can not be interpreted by all browsers. The massive growth of data in the Web and the demand to process it automatically revealed a major weakness of HTML, its inability to represent typed data. As an example, it is not possible to clearly identify a number in an HTML document as the product price or as the package quantity. This is due to the missing typing concept in HTML.

An alternative to HTML would have been the *Standard Generalized Markup Language SGML*[1]. However, SGML parsers were found to be too complex and

[1] SGML is a generic descriptive representation method. Used as a meta-language, SGML can be used to specify other languages like XML or HTML.

time-consuming to be integrated into browsers. Later XML [2] started to fill the gap between HTML and SGML and is now used as a common data representation, especially in e-business, where it is now replacing older standards like Edifact and ASN.1.

Although bad practice, XML is often transformed to HTML for presenting data in the Information Grid. Only when the original XML content is available can users process the contents with their own tools and integrating it into their work flow. Meta information conforming to the *Dublin Core*[2] are sometimes included into documents, but mostly hidden from the user, which still restricts their usefulness.

For **Resource Grids** several specification languages have been proposed and are used in different contexts. Globus, for example uses the Resource Specification Language RSL for specifying resource requests and the Grid Resource Information Service GRIS for listing available Globus services. This asymmetric approach (with different schemes for specifying resource offer and request) might be criticised for its lack of orthogonality but it was proven to work efficiently in practice. Condor, as another example, builds on so-called classified advertisements for matching requests with offers. ClassAds use a flexible, semi-structured data model, where not all attributes must be specified. Only matching attributes are checked. A more detailed discussion of resource specification methods can be found in [15].

In the area of **Service Grids** it is difficult to establish suitable representation schemes because there exists a wealth of different services and a lack of generally agreed methods that allow future extension. Hence Service Grids have to restrict to well-defined basic services like file copy, sorting, searching, data conversion, mathematical libraries etc., or distributed software packages like Netsolve. Work is under way to define generic specification schemes for Service Grids.

In cases where remote services are accessed via customized graphical user interfaces, tools like *GuiGen* [16] may be helpful. GuiGen conforms to the Service Grid concept by offering location transparent services, no matter at which site or system they are provided. Data exchange between the user and the remote service provider is based on XML. The user interacts with the application only via the graphical editor—the remote service execution is completely transparent to him.

XML is the most important representation scheme used in grids. Several other schemes build on it. The *Resource Description Framework RDF* is used in the Semantic Web as a higher-level variant. Also the *Web Service Description Language WSDL* [20] for specifying web services [11] has been derived from XML. For accessing remote services, the *Simple Object Access Protocol SOAP* [3] has been devised. Again, it is based on XML.

[2] The Dublin Core Metadata Initiative (DCMI) is an organization dedicated to promoting the widespread adoption of interoperable metadata standards and developing specialized metadata vocabularies for describing resources that enable more intelligent information discovery systems.

4 Organizing Grids

Locating entities is a problem common to all three Grids. Search engines provide a brute force solution which works fine in practice but has several drawbacks. First, the URL of the search engine must be known beforehand, and second, the quality of the search hits is often influenced by web designers and/or payments of advertisement customers. Moreover, certain meta information (like keywords, creation date, latest update) should be disclosed to allow users to formulate more precise queries.

For the proper functioning of Resource Grids, the possibility to specify implicit Web page attributes is even more important, because the resource search mechanisms depend on the Information Grid. The Globus GRIS (grid resource information service), for example, consists of a linked subset of LDAP servers.

4.1 Hyperlinks Specify Relations

As discussed earlier, hyperlinks provide a simple means to find related information in the Web. What is the corresponding concept of hyperlinks in Resource Grids or Service Grids? In Service Grids there is an attempt to modularise jobs to simpler tasks and to link the tasks (sub-services) by hyperlinks. These should not be hidden in the application. Rather, they should be browsable so that users can find them and use them for other purposes, thereby supporting a workflow-style of programming. Figure 2 illustrates an example where boxes represent single or compound services and the links represent calls and data flow between services. With common Web technologies it is possible to zoom into compound boxes to display more details on lower-level services. Note that this approach emulates the Unix toolbox concept on the Grid level. Here applets can be used to compose customized services with visual programming.

The *Universal Description Discovery & Integration UDDI* and the *Web Service Inspection Language WSIL* are current attempts for discovering Web services together with the *Web Service Description Language WSDL*. UDDI is a central repository for Web services and WSIL is a language that can be used between services to exchange information about other services.

UDDI will help to find application services in future grids. When the first Web Services are made publicly available, additional human readable Web pages should be generated from the WSDL documents so that Web search engines can index them just like normal Web pages and people can find them with established mechanisms.

5 Open Grid Service Architecture

For Service Grids the *Open Grid Service Architecture (OGSA)* was proposed [8]. In essence, OGSA marries Web services to grid protocols, thereby making progress in defining interfaces for grid services. It builds extensively on the open

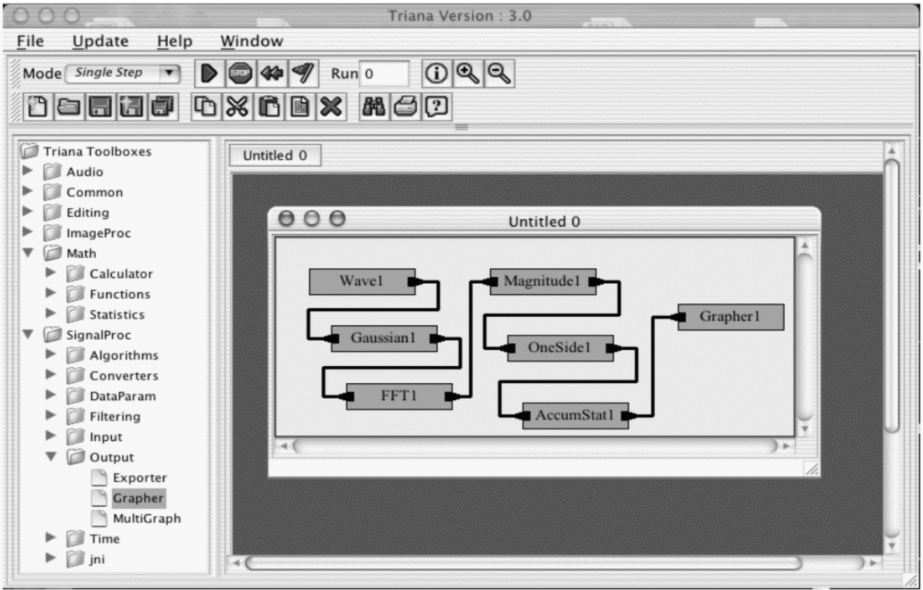

Fig. 2. Representing workflow in Triana [18].

standards SOAP, WSDL and UDDI. By this means, OGSA specifies a standardized behavior of Web services such as the uniform creation/instantiation of services, lifetime management, retrieval of metadata, etc.

As an example, the following functions illustrate the benefits of using OGSA. All of them can be easily implemented in an OGSA conformant domain:

Resilience: When a service request is sent to a service that has just recently crashed, a "service factory" autonomously starts a new instantiation of the service.

Service Directory: With common, uniform metadata on available services, browsable and searchable services can be built.

Substitution: Services can be easily substituted or upgraded to new implementations. The new service implementation just has to conform to the previous WSDL specification and external semantics.

Work-Load Distribution: Service requests may be broadcasted to different service endpoints having the same WSDL specification.

Note that there is a correspondence between the interaction concept of Web services and the object oriented design patterns [9]. The mentioned service factory, for example, corresponds to the Factory pattern. Transforming other design patterns to Web services scheme could be also beneficial, e.g. structural patterns (Adapter, Bridge, Composite, Decorator, Facade, Proxy), but also behavioral patterns like (Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Visitor). These patterns will be used in some implementations of services or in the interaction between services. This makes the development and the communication with grid services easier because complex design choices can be easily referred by the names of the corresponding patterns.

Another aspect that makes grid programming easier is virtualizing core services by having a single access method to several different implementations. Figure 3 depicts the concept of a capability layer, that selects the best suited core service and triggers it via adapters.

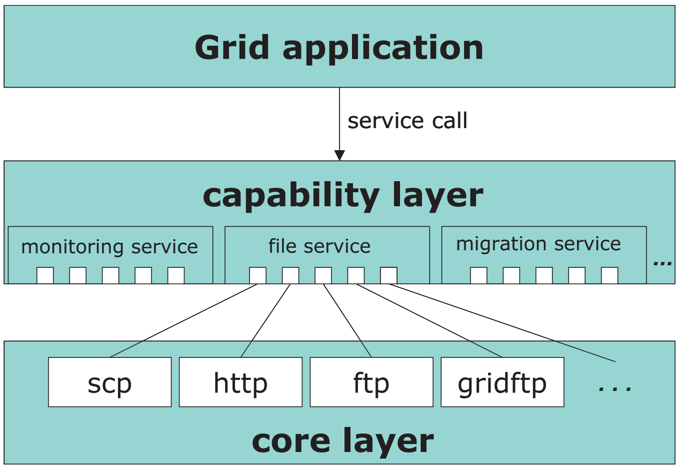

Fig. 3. Virtualizing core services makes grid programming easier.

5.1 OGSA versus CORBA and Legion

Both, CORBA and Legion [14,10] have been designed for the execution of object oriented applications in distributed environments. Being based on object oriented programming languages, they clearly outperform the slower XML web services. Typical latencies for calling SOAP methods in current implementations range from 15 to 42 ms for a do-nothing call with client and server on the same host. This is an order of magnitude higher than the 1.2 ms latency of a JavaRMI call [5]. In distributed environments this gap will be even more pronounced.

In essence, the OGSA model assumes a more loosely organized grid structure, while CORBA and Legion environments are more coherent and tightly coupled. As a result, remote calls in the latter should be expected to be more efficient than in the former model.

6 Outlook

Grid environments provide an added value by the efficient sharing of resources in dynamic, multi-institutional virtual organizations. Grids have been adopted by academia for e-science applications. For the coming years, we expect the uptake of grids in industry for the broader e-business market as well. Eventually, grid technology may become an integral part of the evolving utility network, that shall bring services to the end user in the not so far future.

"Our immodest goal is to become the 'Linux of distributed computing"' says Ian Foster [12], co-creator of the Globus software toolkit. In order to do so, open standards are needed which are flexible enough to cover the whole range from distributed e-science to e-business applications. The industrial uptake is also an important factor, because academia alone was in history never strong enough to establish new standards.

References

1. Alessandro Bassi et al. The Internet Backplane Protocol: A Study in Resource Sharing. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2002)*, pages 194–201, May 2002.
2. R. Anderson et al., Professional XML. Wrox Press Ltd., 2000.
3. Francisco Curbera et al. Unraveling the Web Services Web - An Introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing*, pages 86–93, March 2002.
4. EU Datagrid project. `http://www.eu-datagrid.org/`.
5. Dan Davis, Manish Parashar. Latency Performance of SOAP Implementations. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2002)*, pages 407–412, May 2002.
6. I. Foster, C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications*, Vol. 11, No. 2, pages 115–128, 1997.
7. Ian Foster, Carl Kesselman, Steven Tuecke. The Anatomy of the Grid - Enabling Scalable Virtual Organizations. *J. Supercomputer Applications*, 2001.
8. Ian Foster, Carl Kesselman, Jeffrey M. Nick, Steven Tuecke. The Physiology of the Grid - An Open Grid Services Architecture for Distributed Systems Integration. draft paper, 2002.
9. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns - Elements of a Reusable Object Oriented Software. Addison Wesley, 1995.
10. Andrew S. Grimshaw, William A. Wulf, the Legion team, The Legion Vision of a Worldwide Virtual Computer. *Communications of the ACM*, Vol. 40, No. 1, pages 39–45, January 1997.
11. K. Gottschalk, S. Graham, H. Kreger, J. Snell. Introduction to Web services architecture. *IBM Systems Journal*, Vol. 41, No. 2, pages 170–177, 2002.
12. hpcWire, `http://www.tgc.com/hpcwire.html`, 03.02.2002.
13. Michael J. Litzkow, Miron Livny, Matt W. Mutka. Condor - A Hunter of Idle Workstations. In *Proceedings of the 8th International Conference on Distributed Computing Systems*, pages 104–111, IEEE Computer Society, June 1988.
14. Object Management Group. Corba. `http://www.omg.org/technology/documents/corba_spe`
15. A. Reinefeld, H. Stüben, T. Steinke, W. Baumann. Models for Specifying Distributed Computer Resources in UNICORE. *1st EGrid Workshop, ISThmus Conference Proceedings*, pages 313-320, Poznan 2000.
16. A. Reinefeld, H. Stüben, F. Schintke, G. Din. GuiGen: A Toolset for Creating Customized Interfaces for Grid User Communities, To appear in *Future Generation Computing Systems*, 2002.

17. D.B. Skillicorn. Motivating Computational Grids In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CC-Grid2002)*, pages 401–406, May 2002.
18. Ian Taylor, Bernd Schutz. Triana - A quicklook data analysis system for gravitational wave detectors. In *Proceedings of the Second Workshop on Gravitational Wave Data Analysis*, Editions Frontières, pages 229–237, Paris, 1998.
19. Steven Tuecke, Karl Czajkowski, Ian Foster, Jeffrey Frey, Steve Graham, Carl Kesselman. Grid Service Specification, 2002.
20. Web Services Description Language (WSDL) 1.1. *W3C Note*, `http://www.w3.org/TR/wsdl/`, 15 March 2001.