GridLab: Enabling Applications on the Grid

Gabrielle Allen[1], Dave Angulo[2], Tom Goodale[1], Thilo Kielmann[3],
André Merzky[4] Jarek Nabrzysky[5], Juliusz Pukacki[5], Michael Russell[1],
Thomas Radke[1], Ed Seidel[1], John Shalf[6], and Ian Taylor[7]

[1] Albert Einstein Institute, Golm (AEI/MPG)
[2] Argonne National Lab (ANL)
[3] Vrije Universiteit, Amsterdam (VU)
[4] Zuse Institute, Berlin (ZIB)
[5] Poznań Supercomputing and Networking Center (PSNC)
[6] Lawrence Berkeley National Laboratory (LBNL)
[7] University of Wales, College of Cardiff (UWC)

`www.GridLab.org`

Abstract. Grid technology is widely emerging. Still, there is an eminent
shortage of real Grid users, due to the absence of two important catalysts:
First, a widely accepted vision on how applications can substantially
benefit from Grids, and second a toolkit of higher-level Grid services,
tailored to application needs. The GridLab project aims to provide fun-
damentally new capabilities for applications to exploit the power of Grid
computing, thus bridging the gap between application needs and exist-
ing Grid middleware. We present an overview of GridLab, a largescale,
EU-funded Grid project spanning over a dozen groups in Europe and the
US. We first outline our vision of Grid-empowered applications and then
discuss GridLab's general architecture.

1 Introduction

Computational Grids are becoming increasingly common, promising ultimately
to be ubiquitous and thereby change the way global resources are accessed and
used. However, presently there is a dearth of real Grid users, in part because
the whole concept is new, but also because there are few applications that can
exploit Grid resources. Although some application developers are interested in
writing Grid-enabled applications, there are few user-level tools, while tools for
application developers are nonexistent, and well-understood Grid usage scenarios
are unavailable.

It is therefore imperative to attract real users into the Grid community (for
example the Global Grid Forum (GGF)) and ultimately onto the Grid. The Ap-
plications Research Group (ARG) of the GGF (and formerly of the European
Grid Forum, EGrid) has been addressing questions about users requirements,
problems, and usage scenarios for several years now. In 2000, the group estab-
lished a pan-European testbed [1], based on the Globus Toolkit, for prototyping
and experimenting with various application scenarios. These testbed experiences
gave inspiration for an application oriented project, called *GridLab*, funded by
the European Commission.

The primary aim of GridLab is to provide users and application developers with a simple and robust environment enabling them to produce applications that can exploit the full power and possibilities of the Grid. The GridLab project brings together computer scientists with computational scientists from various application areas to design and implement a *Grid Application Toolkit (GAT)*, together with a set of Grid services, in a production grid environment. The GAT will provide functionality through a carefully constructed set of generic high-level APIs, through which an application will be able to call the underlying grid services. The project will demonstrate the benefits of the GAT by developing and implementing real application scenarios, illustrating wild, exciting, new uses of the Grid. We will make extensive use of specific application frameworks, namely Cactus [2] and Triana [4], as powerful and broad reaching, real-world application examples for developing GridLab, but the GAT will be useful for applications and users of all types. Our aim is to make Grid computing accessible for the widest possible spectrum of applications and users. This paper first presents our vision of Grid-empowered application scenarios. Then we motivate and discuss the global architecture of the project.

2   A Vision of Grid-Empowered Application Scenarios

The advocates of Grid computing promise a world where large, shared, scientific research instruments, experimental data, numerical simulations, analysis tools, research and development, as well as people, are closely coordinated and integrated in "virtual organizations". This integration will be fostered through web-based portals, woven together into modular wide-area distributed applications. One hypothetical scenario in astrophysics, described in the following, illustrates such an integration. Although sounding futuristically, many individual components have already been prototyped, and through the GridLab project we are striving to make such a scenario a common-day occurrence.

Gravitational wave detectors will rely on results from large-scale simulations for understanding and interpreting the enormous amounts of experimental data they collect. The Grid infrastructure is used both to share expensive and centralized resources among many scientists, as well as to integrate experimental data sources with the simulation codes necessary to analyze them. For example, the GEO600 detector in Hanover detects an event characteristic of a black hole or neutron star collision, supernova explosion, or some other cosmic cataclysm. Astronomers around the world are alerted and stand by, ready to turn their telescopes to view the event before it fades, but the location of the event in the sky must first be found. This requires a time-critical data analysis with a number of templates created from full-scale simulations.

In a research institute in Berlin, an astrophysicist accesses the GEO600 portal and, using the performance tool, estimates the resources required for cross-correlating the raw data with the available templates. The brokering tool finds the fastest affordable machines around the world. Merely clicking to accept the portal's choice initiates a complex process by which executables and data files are

automatically moved to these machines by the scheduling and data management tools. Then the analysis starts.

Twenty minutes later, on her way home, the astrophysicist's mobile phone receives an SMS message from the portal's notification unit, informing her that more templates are required and must be generated by a full-scale numerical simulation. She immediately contacts an international collaboration of colleagues who are experts in such simulations. Using a code composition tool in their simulation portal, her colleagues assemble a simulation code with appropriate physics modules suggested by the present analysis. The portal's performance prediction tool indicates that, due to memory constraints, the required simulation cannot be run on any single machine to which they have access. The brokering tool recommends that the simulation be run across two machines, one in the U.S. and the other in Germany, that are connected to form a large enough virtual supercomputer, to accomplish the job within the required time limit. The simulation begins, and after querying a Grid information server (GIS), the simulation decides by itself to spawn off a number of time-critical template generating routines, and to run asynchronously on various other machines around the world.

An hour later, the network between the two machines degrades and the simulation again queries the GIS, this time deciding to migrate to a new machine in Japan while still maintaining connections to the various template generators at other sites. All the while, the international team of collaborators monitor the simulation's progress from their workstations or wireless devices from an airport (where several team members happen to be), visualizing the physics results as they are computed. The template data are assembled and sent to the GEO600 experimenter in Germany for analysis, which finally yields the likely source location for the gravitational wave signal. This triggers an other Grid application which utilizes a different virtual organization and its infrastructure to direct the Hubble Space Telescope and various other available instruments at this source location. The entire process, which could not be performed on any single machine or at any supercomputing site available today, takes only a few hours.

3   Requirements for a Grid Software Environment

The main goal of GridLab is to provide a software environment for Grid-enabling scientific applications. In the advent of the Open Grid Service Architecture (OGSA) [3], GridLab's architecture will revolve around the notion of services.

It is our aim to provide an API through which applications access and use available resources. This API directly reflects application needs; among the intended functionality is the exploration of available resources (CPU, storage, visualization, etc.), remote data access, application migration, etc. The API will be concentrated in the *Grid Application Toolkit* (GAT). The functionality behind the API will be provided by interchangable service providers, being GridLab services as well as third-party services.

In this section, we will briefly define important types of services. We then summarize application requirements and constraints on a software architecture for a Grid Application Toolkit, and its service providers.

3.1 Service Categories

Service: "**A service is a network-enabled entity that provides a specific capability.** [...] A service is defined in terms of the protocol one uses to interact with it and the behavior expected in response to various protocol message exchanges (i.e., service = protocol + behavior)." [5]

Web Service: "The term **Web services describes an important emerging distributed computing paradigm** [with] **focus on simple, Internet-based standards** (e.g., eXtensible Markup Language: XML [...]) to address heterogeneous distributed computing. Web services define a technique for describing software components to be accessed, methods for accessing these components, and discovery methods that enable the identification of relevant service providers." [3]

Grid Service: "**A Web service that provides a set of well-defined interfaces and that follows specific conventions**. The interfaces address discovery, dynamic service creation, lifetime management, notification, and manageability; the conventions address naming and upgradeability." [3]

GridLab Service: A service provided by the GridLab project, normally a Grid Service.

Third-party Service: A service provided outside the scope of GridLab, either from underlying Grid middleware or from legacy software.

3.2 Application Requirements

The ultimate goal of the GridLab project is to provide application programmers and users with an environment which enables scenarios as the one from Section 2. From such scenarios, the main requirements to the GridLab architecture can be drawn.

One of the most important requirements of the GridLab user groups is that applications utilizing the GAT in order to become Grid applications are able to run in *all kinds* of real world environments, including *todays* Grid environments, disconnected environments (e.g. Laptops, developer machines), and firewalled resources.

Application execution inside a Grid installation or on isolated, disconnected resources should simply become two special cases of the same thing: an application running on whichever resources are available – without the need for both a "normal" and a "Grid-enabled" version of the application code. This requirement demands a single GAT API with which applications can be programmed. The actual service providers need to be instantiated at runtime.

Another user requirement is that of robustness, implying that "smart" adaptivity, complete control and fail safety are available on all levels.

3.3 Architecture Constraints

From the general design considerations for the GridLab project, the following constraints should apply to the GridLab architecture:

a) it must be cleanly layered,
b) it must incorporate security mechanisms on a global level,
c) all services should be dynamically swappable,
d) third-party services are to be easily incorporated.

3.4 Security Implications

It may not be obvious at first sight, but the above requirements and constraints partially contradict each other. In particular, the mandate to incorporate security mechanisms on a global level, for all components, is incompatible with respect to third-party services (one cannot mandate anything for those) and for disconnected environments. For third-party services, this could be solved by requiring access to any third-party service to be performed via a (secured) Grid-Lab service. But that again is not possible if our application is running in a minimalistic environment, or just *deliberately chooses* to contact the suspicious (e.g., legacy) component on its own behalf – as applications need full control on all levels.

We assume that contradictions of this type are very natural in Grid environments: the middleware developer wants to have control over all aspects of the environment, in particular over security, but also over scheduling, network interfaces and so on. The end user in turn wants to be able to run an application in *any* environment – benefiting from a Grid infrastructure if available.

In our GridLab project, we learned to distinguish between these two cases. Our *"General Architecture"* reflects both variants: on the one hand it will allow the GAT and hence the applications to utilize whichever service provider they need, be it secure or not, be it in a Grid or not; on the other hand it strongly recommends a cleanly layered architecture for the actual scope of the project, which is to provide application-oriented services as an abstraction of Grid environments, including, e.g., global security.

4 The GridLab Architecture

As discussed above, the GridLab project distinguishes between GridLab services and third-party services (e.g. low-level Grid services like GIS or GRAM, system services, libraries). This section will describe the implications of this distinction for the actual architecture.

Figure 1 is divided into a user space and a service space, reflecting the possible boundary between the application programs and the network-enabled (thus possibly remote) services. Applications should be programmed using the GAT API, and the GAT itself uses *adapters* for all kinds of services it can make available through the GAT API.
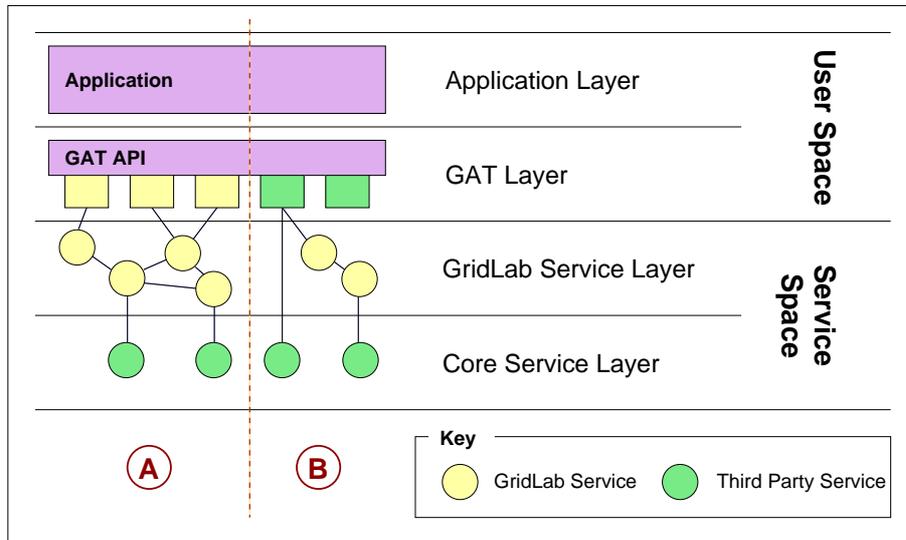
Fig. 1. The General GridLab Architecture: The GAT will be designed to interact with both types of service providers (cases A and B).

In addition to the GAT itself, GridLab will provide a large set of GridLab services, tailored to the needs of the GAT API. GridLab services will integrate lower-level services to build more powerful functionality (like smart adaptivity, control, and fail safety). In the recommended (Grid-enabled) case A in the figure, the GAT only uses adapters to GridLab services which in turn draw their functionality from lower-level, system services. In case B (for disconnected use or for legacy services), the GAT will also directly interface to third-party services, at the price of possibly sacrificing security concerns on the application's behalf.

5   Conclusions

In this paper we have briefly described the overall architecture of the GridLab project, which aims to provide application oriented Grid services for users and developers alike, covering the whole range of Grid capabilities as required by applications, such as resource brokering, monitoring, data management etc. These services will abstract lower level Grid functionality and will hence ease the development and deployment of Grid aware applications. This consistent service infrastructure will be the first major deliverable of the GridLab project.

These services will be made accessible to any applications running on the Grid through a Grid Application Toolkit (GAT), the second main project deliverable. The GAT will abstract those services needed by the Grid applications. In this way, applications can utilize service discovery at runtime, making use of whatever services are available, including different implementations of the same

service. This will enable users and application developers to easily develop and run powerful applications on the Grid, without having to know in advance what the runtime environment will provide. Such applications should then run on a laptop or an intercontinental Grid, taking advantage of whatever services are actually available (or unavailable). In particular, the GAT will not depend on the existence of GridLab services.

Although we collaborate with the developers of powerful application frameworks such as Cactus and Triana for the development of GridLab, the project is designed to enable *any* application not only to run on the Grid (or without a Grid), but to endow it with new capabilities uniquely possible on a Grid, such as those described above in our usage scenario. With such an architecture, we expect many new and powerful applications to be developed to exploit the Grids of today and tomorrow alike.

Acknowledgments

References

1. G. Allen, T. Dramlitsch, T. Goodale, G. Lanfermann, T. Radke, E. Seidel, T. Kielmann, K. Verstoep, Z. Balaton, P. Kacsuk, F. Szalai, J. Gehring, A. Keller, A. Streit, L. Matyska, M. Ruda, A. Krenek, H. Frese, H. Knipp, A. Merzky, A. Reinefeld, F. Schintke, B. Ludwiczak, J. Nabrzyski, J. Pukacki, H.-P. Kersken, and M. Russell. Early experiences with the EGrid testbed. In *First IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 130–137, Brisbane, Australia, May 2001.
2. Cactus Computational Toolkit home page: `http://www.cactuscode.org`.
3. I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Draft Document, `http://www.globus.org/research/papers/ogsa.pdf`, June 2002.
4. Triana home page: `http://www.triana.co.uk`.
5. S. Tuecke, I. Foster, and C. Kesselman. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputer Applications*, 15(3), 2001.