

## PREFACE

The term Grid was coined in the mid-1990's<sup>1</sup> to describe a technological vision of a shared computing infrastructure for researchers. This vision was later refined and described the Grid as an infrastructure for “coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations”<sup>2</sup> Recently the Network of Excellence CoreGRID (<http://www.coregrid.net/>) produced a definition of Grid as a fully distributed, dynamically reconfigurable, scalable and autonomous infrastructure to provide location independent, pervasive, reliable, secure and efficient access to a coordinated set of services encapsulating and virtualizing resources (computing power, storage, instruments, data, *etc.*) in order to generate knowledge.

Grid is made possible through a specialized middleware, which refers to the security, resource management, data access, instrumentation, policy, accounting, and other services required for applications, users, and resource providers. Middleware acts as a sort of ‘glue’, which binds all these together.

Various Grid middleware systems have been built so far, such as the Globus Toolkit, Condor, Unicore *etc.* All these systems provide similar Grid services, although the features, technologies and standards used may differ. Still, the complexity of these systems are high, and functionality provided to application users and developers are very limited.

In order to build and develop applications for the Grid there is a strong need for solid higher level middleware, that would provide richer, more automated features such as resource brokering, job scheduling, automatic file transfer *etc.* In addition to that users often call for easy to use Grids and easy to use APIs to deploy their applications on the Grid. The applications need to access Grid services and resources in a simple, transparent way. But, on the other hand, they want to have a full Grid functionality somehow abstracted to the high level. And this can be delivered by providing high level API and Grid programming environments.

The challenge for such environments and for the middleware is to provide applications with APIs that make applications more or less Grid unaware and the Grid itself transparent to its users. There are many efforts worldwide to provide such environments. The motivation behind this issue of CMST was to give a solid overview of not too many, but different and most important approaches. The idea was to collect some of the most interesting environments that make the life of Grid application users and developers easier, giving also access to the reach and advanced Grid technologies.

The first paper presents the structured parallel programming environment ASSIST, whose design is aimed at raising the level of abstraction in Grid programming and discusses how it can support transparent Grid programming while implementing Grid adaptivity. ASSIST environment can be considered a programming environment fulfilling the “invisible Grid” idea, *i.e.* allowing programmers to write efficient grid programs without actually being concerned with all the details related to efficient usage of the Grid middleware.

Following ASSIST we have the SAGA (Simple API for Grid Applications) paper. SAGA is a GGF standardization effort that addresses the gap between existing grid middleware and application-level needs by providing a simple, stable, and uniform programming interface that integrates the most common grid programming abstractions. These most common abstractions were identified through the analysis of several existing and emerging grid applications. The paper describes the SAGA effort, its relationship to other grid API efforts within the GGF community, and introduce the first draft of the API using some application programming examples.

Further, in the third paper of the journal we present ProActive. ProActive proposes a grid programming strategy that addresses several grid concerns, which we have classified into three categories: (i) Grid Infrastructure which handles the resource acquisition and creation using deployment descriptors and Peer-to-Peer. (ii) Grid Technical Services which can provide non-functional transparent services like: fault tolerance, load balancing, and file transfer and (iii) Grid Higher Level programming with: group communication and hierarchical components. The approach was validated with several grid programming experiences running applications on heterogeneous Grid resource using more than 1000 CPUs.

---

<sup>1</sup> I. Foster, C. Kesselman, *Globus: A Metacomputing Infrastructure Toolkit*, Intl J. Supercomputer Applications, **11(2)**, 115-128 (1997).

<sup>2</sup> I. Foster, C. Kesselman and S. Tucke, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, International Journal of High Performance Computing Applications, **15(3)**, 200-222 (2001).

In the next chapter the paper from the EGEE project gives a brief overview on how the EGEE Grid infrastructure can be programmed using the services provided by the gLite middleware distribution. Authors focus on the information monitoring, workload management, and data management services which are the most frequently used ones. A discussion of the security framework is also given in the paper. The paper contains also a very interesting discussion on using emerging Grid standards in production infrastructures.

The next paper presents PSNC's Grid toolkit called Gridge – The Grid Enterprise Solution. Gridge is a set of Grid services, tools and a portal framework for enabling scientific, business and industrial applications on the Grid. The approach, although similar to gLite offers different set of services and functionality especially important for dynamic Grid scenarios. Most of the Gridge services with their API were presented and illustrated in a simple task farming scenario.

Next chapter presents the example methodology of Grid-enabling a real Quantum Mechanics / Classical Molecular Dynamics (QM/MD) Simulation application using Grid RPC/Ninf-G together with MPI. GridRPC allows dynamic resource allocation and migration and automatic recovery from faults while MPI provides high-performance parallel processing on each cluster. The QM/MD simulation was run on an international Grid testbed in the Asia Pacific Region for about 52 days. The experimental results indicated that the hybrid QM/MD simulation could (1) adapt to the dynamic behavior of the simulation and can change the number of CPUs and the number of clusters, (2) adapt to unstable Grid infrastructure and recover from faults automatically, and (3) manage hundreds to thousands of CPUs on distributed locations, and (4) survive for several weeks without interrupting manual operation.

In the final paper we focus on higher-level user interfaces to Grid such as portals. The GridSphere portlet framework, which is currently one of the most commonly used portal frameworks for Grids is presented.

Grids are already common in eScience. But Grid provides a means by which not only research, but also commerce and industry can co-operate and share resources. Grid by its nature is an enabler for research, commerce and industry because they are increasingly collaborative, resource sharing, and multi-site and often multinational. So, we can expect good results of Grid deployment in such areas as eHealth, eGovernment, banking, finance *etc.* The importance of Grids is well noticed by big ICT vendors who transfer the Grid concept to their portfolio products. We think that Grids will also have an important impact on computational methods for science and technology. Moreover, their importance will grow with development of new computing technologies on clusters as well as new telecommunication technologies such as broadband mobile networks and sensor networks.

All the above-mentioned aspects made the CMST Editorial Board looking at Grid technologies and deciding to bring them closer to our readers. The following issue is our first of series on the topic. We hope you will enjoy this series.

*Jarek Nabrzyski and Maciej Stroiński*

*The Editors*