# Table of Contents

# Abstract

While the Grid provides a uniform interface to very large computational and storage resources, porting, deploying and running applications in the Grid environment is very difficult. The main reasons/obstacles are: heterogeneity of the environment in the WNs, runtime-constraints (e.g. connectivity), global job scheduling constraints (due to very complex system of sharing resources across many users and providers) which leads to unpredictable and unreliable service. This work addresses the problem of efficient enabling of a large class of applications and improving of the quality of service in the Grid.

In this dissertation we develop a concept of application-centric middleware which is dynamically deployed in a user space i.e. without special administrator privileges. The middleware acts as an application hosting environment and a reactive scheduler with a fault-tolerance engine which is designed for application-specific customization and fine tuning. Optimal selection of computing resources is controlled in a virtual cluster of the Grid worker agents, which enables to improve the responsiveness and reliability of the Grid infrastructure and may provide a low-latency access to the Grid resources suitable for interactive applications. Contrary to other approaches which rely on the modification of the underlying Grid infrastructure or persistent deployment of specialized services, the virtual cluster is created as a volatile overlay with minimal connectivity requirements, which enables seamless integration of Grid and non-Grid resources. This in turn allows an easy transition between local and Grid environments - an essential concept in supporting the every day's work of the scientists.

- Grid-enabling applications (sequential, several parallel, legacy black-box and at source code level) - a robust handling of the inherently heterogeneous and dynamically changing Grid environment - contrary to approaches which focus on specific application patterns (parameter-sweep, m/w) a virtual

(thus making it easier to enable applications for developers and running them the users), a reactive and adaptive scheduler to cope with dynamic changes in the underlying distributed environment (improvement of QoS, dynamic selection of the resources e.g. task placement for optimized data access, definition of the task interdependencies), a fault-tolerance engine to cope with runtime errors. The middleware imposes minimal requirements on the underlying computing infrastructure and allows an easy transition between a local environment and the Grid -

# Introduction

Grids enable scientific computing at unprecedented scale in terms of computing, storage capacity, resource integration and scientific collaboration.

Last years have seen countless of Grid projects and initiatives, ranging from small experimental testbeds to large production infrastructures. [any good reference?]

High Energy Physics (HEP) is one of the applications which have been driving the development of one of the largest-scale Grid infrastructures - WLCG (Worldwide LHC Computing Grid) - for the needs of the LHC (Large Hadron Collider) experiments at CERN.

Other communities, such as bio-informatics, medical physics, earth observation, telecommunications have joined the WLCG and created the EGEE (Enabling Grids for E-sciencE) Grid - the world's largest production Grid infrastructure to date.

The EGEE Grid must reach its full operational capacity by the startup of the LHC experiments at CERN (end of 2007). In contrast to many, small-scale experimental Grid installations which are research playgrounds for future usage, the EGEE Grid applications must be deployed in a robust, reliable and performant production Grid of an unprecedented scale - now! In order to understand the challenge of enabling the applications in a large production Grid one must understand the idiosyncrasies of the development of a large Grid infrastructure itself.

## The development of large production Grids (EGEE example)

The development of a large, production Grid in the scientific community is driven by a long-term negotiation process in a complex organizational matrix of resource providers (computing centers), research institutes and users (experiments).

Hardware resources (servers, storage devices, scientific instruments, networks) are in different administrative domains and controlled by different local policies. Virtual Organization (VO) brings a subset of resources into a single virtual administration domain and allows to set global policies which are mapped onto the local ones. However the ultimate control of the resources is retained by the resource providers which require the accountability and traceability of the user activities.

The deployment and maintenance of the middleware is at the heart of the operations in a large production Grid. The deployment process is not centralized because resource providers have local schedules and obligations towards their local non-Grid communities. This affects the maintenance of the Grid protocols and services and the introduction of the new ones. The operational changes, such as allowing certain local resources under a sole VO control (for example VOBox), must typically go through a negotiation process.

The introduction of new middleware components or operational changes are propagated very slowly in a large Grid and it may take several months until the entire system is updated. In addition the baseline middleware evolves in parallel to the deployment processes and application specific functionalities from different VOs may be conflicting. Therefore typically the application-specific functionalities are not introduced at the generic middleware level, but at a higher level, in the application-specific middleware.

The advantage of application-specific middleware is developed and deployed within the VO boundaries, thus at a smaller scale, but at a shorter timescale "in-sync" with the VO community needs.

The most prominent examples are the production systems of the LHC experiments such as DIRAC in LHCb, AliEn in Alice and AtProd in Atlas. These systems implement many features such as centralized task queue, file and metadata catalogs and generally improved reliability which are necessary for large data productions.

VO-specific middleware is setup and maintained in the central services (such as home made Workload Management Systems - WMS) as well as at the sites on dedicated, VO-controlled machines (called VO-boxes). Such an approach can only be afforded by relatively large VOs both in terms of neccessary human resources for maintainance as well as the political power to convince the site administrators to allow VO-boxes. Additionally, such systems in practice tend to be very specialized (despite the efforts to design them generically) and they derive from a specific VO-activity such as data production. Data production is a centrally managed activity much different from end-user data analysis and therefore the initially deployed and tested systems must be refactored and reengineered to cover other areas.

**In a large production Grid it is hard to modify generic-middleware and a higher-level, application-oriented middleware is neccessary to accomodate specific application needs in the timeframe convenient for the user community.**

## Enabling and support of the applications in a large production Grid

Significant efforts go to enabling and supporting of the applications. This includes the testing of the baseline infrastructure, the development and testing of the application-specific middleware, development of user interfaces [Ganga] and gathering monitoring information [Dashboard] for the operational fine-tuning and infrastructure debugging. Nevertheless user communities face several problems.

### Improvement of the Quality of Service

Large variations in the performance and reliability of the EGEE Grid are confirmed by independent observations [ARDA] and everyday user experience. This may be accounted for the fact that a large distributed system such as Grid is inherently dynamic: resources are constantly reconfigured, added and removed; the total number of failures is correlated with the size of the system; the user activities are not coordinated and the load may change rapidly. Still, there is a common feeling in the user community that currently the Grid infrastructures do not provide a required level Quality of Service (QoS).

Similarly to the classical batch systems, the Grid is designed for optimizing throughput of long, non-interactive jobs. One of the reasons for this is that the underlying Computer Elements are batch farms. The overhead of hierarchical Grid scheduling (in the WMS->CE->BS chain) is large (and has been measured) [C.Germain-Renaud et al.: Scheduling for Responsive Grids]. However a large number of applications, from physics data analysis to medical image processing, require *responsiveness*,_interactivity_ and *low-latency* access to resources.

The EGEE Grid is a large-scale federation of computing centers which allows for sharing of the computing resources across institutional boundaries. The resource sharing is typically implemented using well-established technologies based on high-throughput systems oriented for batch-processing (such as LSF, PBS, SGE, Condor). A higher-level application-oriented service which is an add-on layer above the batch oriented infrastructure allows to add the required capabilities and at the same time enables the Grid sites to continue to use the existing, well-established technologies to configure the access and sharing policies for the local resources. Correct balancing of the needs of local and Grid users and the retention of control is essential for the Grid sites and their funding agencies.

Predictability is another important aspect for grid users to reliably estimate and the time in which the various stages of processing are completed (including the total turnaround time). Because of performance and reliability variations, hierarchical scheduling and heterogeneity of the resources, the predictability on the Grid is more complex than in classical batch farms.

**The improvement of the Quality of Service is one of the most urgent issues for the successful applications on the Grid.**

**(Adding new capabilities (interactivity) as well)**

### Exploitation of application execution patterns

In traditional batch farms jobs are monolithic and typically unrelated executables. Limited features exist to support applications beyond a simple batch model. Some systems, such as LSF, allow a synchronized execution of a group of jobs (typically MPI) when the appropriate number of resources becomes available. Other systems, such as Condor DAGman, allow to introduce dependencies between the jobs using meta-scheduling i.e. an external service which submits new jobs to the Grid when needed. Similarly in the Grid, it is possible to specify special type jobs (such as MPI) or use meta-scheduling for job dependencies or workflows. However metascheduling performes well only in long timescales. The newest versions of the gLite middleware support bulk job submission which allows to reduce the submission time and share the common part of the input sandbox between multiple jobs. Bulk submission allows to increase the performance of job splitting i.e. submission of a large similar jobs which read different parts of a dataset but it cannot improve the job turnaround time. Condor glide-ins [more].

Grids are constantly evolving and new standards for the middleware are proposed to support advanced functionalities including service oriented architecture and the QoS. To date, however, the least common denominator of the largest, production Grids still remains amazingly similar in functionality to the traditional batch farms and does not provide the adequate (in terms of performance) support for applications beyond bunches of loosely-related jobs. AppLeS provides a framework for parameter-sweep applications [more].

On the other hand many applications exhibit striking structural similarities. Data-driven parallelism is very common in physics analysis applications (Atlas/Athena). Independent event simulation is possible with Monte-Carlo methods in medicine (radiotherapy). Workflow and parameter sweep applications are common in bio-informatics and exemplified by docking algorithms (Autodock). Iterative patterns and pipe-lines are present in many image processing applications (xmipp).

Because of the lack of convenient tools and mechanisms in the Grid to manage more complex execution patterns, users build more-or-less ad-hoc solutions to manage their applications in a better way (e.g. [LJSF]). However even more structured efforts suffer from several drawbacks. For example, a MPI-BLAST is a MPI-flavored version of the Basic Local Alignment Search Tool (BLAST) - a widely used tool for searching protein and DNA databases for sequence similarities [NCBI]. BLAST is a typical Master/Worker application and the appropriate management and bookkeeping layer has been implemented in MPI, and later ported to the mpich-g2 environment. There are a number of problems with such an approach:

- the mainstream Grid installations limit the support of the MPI jobs to a single Grid cluster, so the Grid capabilities cannot be fully exploited;
- the MPI was designed to build complex parallel applications rather than job management layers so the cost of development is relatively high;
- the management layer must be constantly maintained and adapted to the changing Grid environment.

*insert MPI Grid support information from Cal email*

**The exploitation of the recurrent application patterns in a structured and generic way would have an important impact on enabling new applications in the Grid.**

### Support of multiple environments

Grid is only one of the environments for the everyday work of scientists. Development and testing of algorithms and formulas is an intrinsic part of many types of scientific research activities such as data analysis in particle physics. A typical pattern is to develop and debug an algorithm locally, then to test it on a larger scale using local computing resources and locally available data before harnessing the full computational power of the Grid. The transition between local and Grid environments may also happen in another direction as the research process may involve multiple algorithm development cycles.

Some user communities use local resources which are not part of Grid: sometimes for the lack of human resources to maintain a Grid site, sometimes for the conservative policy in embracing new trends. It is clear that if such users had a possibility to mix local resources with the Grid ones it would be beneficial not only for them, but also, in the long term, for the whole Grid community.

User interfaces such as [Ganga] facilitate the transition between the local and Grids environments and provide the necessary aid for the users . However they do not provide the means to integrate both local and Grid resources into a single, useful computing platform.

**Application hosting environment (Ulrik's mail).** On-the-fly installation of the lightweight add-on layer avoids the service deployment issues.

**A lightweight method of integration of Grid and local resources to run the applications in a robust way would be beneficial for smaller user groups as well as the whole Grid community in the long run.**

## Definition of the User Level Scheduling

## The view-point and goals

Given the discussion above we can define the view-point for the dissertation: from the perspective of a user of EGEE, the world's largest production Grid, we try to maximize the user's return-of-investment into a Grid technology by:

- improving the Quality of Service and increasing application performance and reliability in the Grid;
- enabling the capabilities currently unavailable in the Grid such as interactivity and leveraging on recurrent application execution patterns;
- reducing user's investment (time) needed to interface the applications which can take advantage of improved QoS and new capabilites (application hosting environment).

In the following subsections we analyze the constraints, objectives and strategy for a **User Level Scheduling** system which could meet the goals listed above.

### Constraints

As explained in the previous sections it is not practical to expect that the middleware release cycle is coupled to the applications' activity schedules. Therefore many of the generic middleware concepts which have been proposed to handle hard (deterministic) Quality of Service such as [G-QoSm], [GARA] or advanced reservations systems [G-RSVPM] are useful given the contraints. If the main contraint is that the generic middleware cannot reliably be modified, one must resort to a higher-level, application-oriented middleware on top of unmodified generic infrastructure. However with such a pragmatic approach only the statistical or best-effort Quality of Service may be implemented.

The trust relationship in the Grid betwen resource providers and consumers is based on accounting and traceability at the user level. This requirement is not always met in the systems which form permanent overlays above the regular infrastructure and rely on glide-in techniques shared by all users in the VO (e.g. HEP production systems). The multi-user pilot jobs make the user accounting impossible which is a serious drawback for the Grid resource providers. Therefore accountability and traceability at the user level is an important constraint.

### Objectives

The objectives for a User Level Scheduling middleware:

- improvement of the Quality of Service in statistical sense;

- accomodation of short and very short jobs within batch infrastructure oriented for long jobs (responsiveness);
- accomodation of new capabilities such as interactivity;

### Strategy

A sucessful strategy for a User Level Scheduling middleware must incorporate:

- expoitation of recurring parallel execution patterns such as data-driven, iterative, workflow, parameter sweep, divide-and-conquer by decoupling application-specific functionality from the execution patterns;
- flexibility in adaptation to different application deployment models i.e. how applications are installed, configured and executed;
- minimization of the integration time of existing applications (also legacy);
- allowing resources from multiple environments be easily combined

# Quotes

*The development of user-level middleware will be critical to the wide-spread use of and application performance on the Computational Grid.* [AppLes]

# References

[GARA]

[G-QoSm] 2004 IEEE International Symposium on Cluster Computing and the Grid: QoS Support for High-Performance Scientific Grid Applications, R.Al-ali et al.

[G-RSVPM] G-RSVPM: A Grid Resource Reservation Model; First International Conference on Semantics, Knowledge and Grid (SKG'06); By Zeng Wandan, Chang Guiran, Zhang Dengke, Zheng Xiuying, November 2005

---

AN INTERESTING CGW06 Reference - FULL TEXT IN THE INDEX FILE (PDF)
http://www.cyfronet.pl/cgw06/posters.pdf

An Approach for Intra-VO Differentiation of Computing Services in Grid Systems

Sergio Andreozzi, Marco Cecchi, Vincenzo Ciaschini, Andrea Ferraro, Francesco Giacomini, Antonia Ghiselli, Alessandro Italiano, Gian Luca Rubini, Davide Salomoni INFN-CNAF, Bologna, Italy

In this paper, we describe a proposal for differentiating the access to computing resources among users of the same VO (this approach is described in the context of the gLite middleware).

Our approach grounds on a rigorous definition of service classes in terms of characterizing attributes that describe different quality of service levels other than the best effort. Examples of parameters characterizing a service class are the target share in the utilization of the site resources, policies related to the maximum walltime of a single job or a priority level. The set of service classes defined by a VO is called service model. According to the needs of each VO, resource providers will configure at the physical level the service classes to access computing resources relying on the local resource management system capabilities. The important constraint is that jobs submitted to the same computing facility under the same service class must follow a FIFO approach. As initial design, we envision three main service classes: best effort, guaranteed and express. For each service class, we describe the set of mandatory and optional parameters. Once the service classes are defined and configured at each resource provider, users part of the same VO, but with different credentials

regarding their group or role membership can be assigned to different service classes, thus enabling a differentiation of the access to resources of a VO. Given the geographical distribution of a Grid, the high number of sites and resources, and the dynamics of usage of the system, it is essential for a VO to have mechanisms to dynamically change how the VO users are assigned to the different service classes. In order to achieve this goal, we enrich our model with a distributed policy framework such as G-PBox (it relies on XACML for policy definition). By means of this component, the VO manager can define mapping policies that assign a set of Grid credentials to a certain set of instances of service classes. When a job is submitted to a site, the authentication and authorization layer queries the distributed policy framework by passing the Grid credentials of the user (Grid identity and VO membership information). The authorization framework will answer with the service class to which the user should be mapped, if any. The proposed approach is currently under testing in the INFN-Grid and relies on key middleware components like VOMS, G-PBox and LCMAPS. The proposed system enables a flexible intra-VO computing resources differentiation. By means of this approach, a VO can enforce that different groups of users have a guaranteed share of resources and the assignment can be dynamically changed over time without intervention at the sites.l

[LJSF] Light Job Submission Framework, https://twiki.cern.ch/twiki/bin/view/Atlas/LJSF

---

-- JakubMoscicki - 16 Dec 2006

---

This topic: Main > PhD
Topic revision: r14 - 2007-04-15 - JakubMoscicki