The Hungarian ClusterGrid Project: Challenges of a Production Grid

Tamás Máray[1], Péter Stefán[1], Ferenc Szalai[1,2], Gábor Vitéz[1]
[1]Office for National Information and Infrastructure Development NIIF/HUNGARNET
Victor Hugo 18-22, H-1132 Budapest
[2]Research Institute of Solid State Physics and Optics
Hungarian Academy of Sciences
P.O.Box 49, H-1525 Budapest
Hungary
E-mail: grid-tech@niif.hu

Abstract - **The paper aims at providing brief architecture overview of the Hungarian ClusterGrid, a production grid since September, 2002, as well as gives some key features that distinguish it from traditionally built grid systems, such as the use of private computer networking, intelligent resource brokering, dynamically mapped task execution environment, and a "one-job corresponding to one-directory structure" environment. The paper concludes operation experience and production grid service organizational models.**

Key-Words - **grid, cluster, high-performance computing, supercomputing**

1 Introduction

The initiation of the Hungarian ClusterGrid Infrastructure project dates back to July, 2002, when the Hungarian Ministry of Education donated 2000 PCs at national universities, polytechnics and public libraries. The PC labs involved were considered to satisfy two criteria: on one hand the labs were intended to support education and training tasks during the work hours of the institutes every day, on the other hand they were proposed to be used for solving high performance computation tasks whenever they are not used for their primary purpose, i.e. for education, during the nights and the week-ends.

In September 2002, a preliminary grid test-bed was installed gathering the "free-cycles" of the PC labs (or using supercomputing terms PC clusters), which was put into production after 2 months of testing period in November, 2002. The initial architecture was referred to as the first generation of the ClusterGrid infrastructure (CG).

In May 2003, major revisit over the architecture took place partly to process and recycle accumulated experience, and partly to involve new features which were completely missing from the first generation architecture. The improved architecture referred to as the second generation architecture, having been in production since July, 2003 now involves 1100 interconnected compute nodes and serves more than 25 scientific projects at the accumulated computation performance of 400 billion floating point operations per second.

2 The layered architectural model

Since the PC labs are used for dual purposes, and the way how they can be used for either is traditionally different, it is straightforward not to mix, but separate the two functions from each other over the same hardware as much as possible. This principle is called spatial and temporal separation, when spatial separation means using different operating systems, on different disk partition, in a different network segment, and temporal separation means using the PC labs during the day in an office like environment, and using them as a compute engine during the nights and week-ends when they would be turned off anyway.

While the classical grid architectures focus mainly on the connectivity among super-computational resources specifically at the application layer, contemporary grid development principles gradually start to recognize that many problems emerging from the production grid requirements cannot be solved only at the application level, but lower layers, such as computer networking need to be investigated [1]. The grid architecture used in the ClusterGrid infrastructure (CG) design corresponds to the simple model shown in Figure 1.

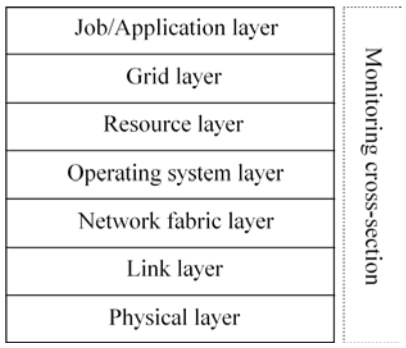

Fig.1 The grid reference model.

2.1 The physical layer

The physical layer involves the hardware: PCs and servers. Each piece of the hardware pool conforms to an architecture role among the following five basic roles:

- Resource nodes: Resource nodes are compute nodes which need to be strong machines in terms of CPU performance, caching and memory.
- Local master nodes: Each cluster of resource nodes contains a local master node to provide local services, such as network boot service, network file system service, or connectivity service. Local masters may not perform computation at all.
- Service nodes: The purpose of service nodes is basically to provide fault-tolerant grid-level services like operating system mirroring, maintenance, statistics collecting, logging or monitoring.
- Entry nodes: The entry nodes are the user interfaces in the system.
- Grid job gateway nodes: Job gateways have the same basic functionality as entry nodes, with the difference that they are expected to transfer jobs from/to external grid systems without user intervention.

2.2 The link layer

The link layer corresponds to the data link layer in the ISO/OSI networking reference model [2]. In the contemporary network design it means connecting the different nodes with special network devices, called switches.

Separation at this level can be done by using virtual local area networks (VLAN), or secure networks satisfying IEEE 802.1q and 802.1x standards. Each VLAN represents a communication pool in virtual connectivity within the resource provider institutes.

2.3 The network fabric layer

Introducing a network fabric layer within the grid middleware was one of the key innovative ideas of CG: in order to improve secure connection among the different clusters, they are not connected through the public data network, but a separated and dedicated communication channel, called the virtual private network (VPN).

Virtual private networks (VPN) can be implemented in many ways: One possible implementation is to use multi-protocol label switching (MPLS) technology by exploiting the capabilities of the high-quality academic network [3]. The customer edge (CE) - provider edge (PE) layout used in MPLS fits perfectly to the grid network fabric requirements, with the only extension of connecting the CE and the PE routers with each other via either intra-institutional VLAN, or via tunneling.

2.4 The operating system layer

Managing just a couple of PCs and administering several hundreds or thousands of nodes are inherently different. Different installation, configuration and maintenance policies are needed to both.

In the CG architecture, Linux is used in general as the grid operating system (OS), and the compute nodes are configured as diskless thin-clients to save management work. By using network root file system, and network boot procedure the number of OS images to maintain can be significantly reduced.

2.5 The resource layer

The resource layer basically transfers a bunch of PCs into a single large computational resource. It encapsulates both job management utilities, like Condor or SGE, and library routines that support exploiting the benefits of parallel processing, e.g. parallel virtual machine (PVM) or message-passing interface (MPI).

2.6 The grid layer

The grid layer implements abstraction over the cluster pools, and addresses issues like global job handling, resource brokering, and global grid information service. More on the grid layer functions and modules can be found in the forthcoming section.

2.7 The application layer

The application layer is responsible for the environment that the users access. The number of tasks within this layer is extremely large ranging from software development (graphical parallel software development environment, distributed make utilities, etc.), compilation, or in a more complete view the preparation of the executables, like code optimizing, packaging, and job management (job submit, job query).

2.8 The general view

Fig.2 shows the general view of the architecture. It is easy to see that besides the local PC clusters there is a global entity referred to as the "backbone" of the grid (GBone) which involves local master nodes, entry nodes and service nodes connected through the VPN cloud, and which determines the quality of the grid service. Note that the quality of the grid service does not tell anything about the computational performance which is determined by the compute nodes, but defines the quality of the higher level software services such as the throughput among clusters, the guarantee to find the right resource for a particular job, or security.

3 Innovative elements in the architecture

3.1 The grid resource broker

The grid resource broker used in the ClusterGrid infrastructure (CG) aims at pursuing a "minimal, but required by the users" approach and puts special emphasis on supporting user and job privacy and security.

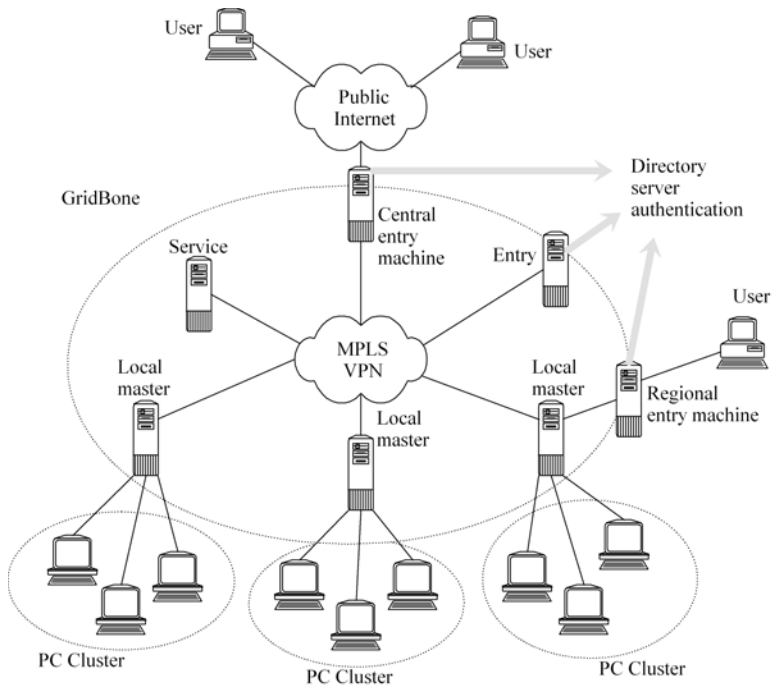

Fig.2 The general view of the ClusterGrid Infrastructure project.

The broker implementation is based on a special web-transaction protocol suite that was designed to include the following fundamental transaction elements:

- job submission,
- job removal and migration,
- job result and failure propagation,
- resource status inquiry.

The transactions use web POST method over secure communication channel to transfer communication messages and data. Hosts mutually authenticate themselves by using X509 host certificates[1].

Since the broker views the local clusters as atomic resource units, the scope of broker communication is restricted to the nodes within the grid backbone only. On one hand the resource and job information is exchanged between entry nodes and local master nodes, on the other hand the entry nodes synchronize their information with one another.

3.2 The user authentication and job identification

In the CG user and job entities are treated separately: users are authenticated, authorized only on the entry points of the system, while the jobs execute under dynamically mapped credentials independent from the users credentials. The advantage is that no user authentication is needed on the cluster nodes, nevertheless the jobs can be traced through the system due to the dynamic registering. The result is that user identifiers can easily be determined for local jobs making job migration among the different resources flexible.

The user authentication database can either be local, using classic password file, local information service, or global, integrated, external LDAP system depending on the user management policy. For example in the case of CG entry nodes a high-availability clustered, distributed directory server system provides the user database and the authentication service.

Since user identifiers exist on the entry nodes only, when the broker takes over a submitted user job and prepares it to execute on one of the clusters, the broker assigns a globally unique job identifier to the job, and maps it to local cluster credentials before execution. Thus the jobs, even those which were submitted by the same user, execute under different local credentials, preventing "malicious jobs" to access unauthorized data from other jobs. The local credential assignment is stored in a database backend, and managed by a special application called IDregister [7], dynamically.

3.3 The "job-dir" execution format

In traditional grid systems a job is usually defined as a statically linked executable equipped with input and parameter files. In the practice this single-file approach has proved to be too restrictive, since users may wish to transfer not only the executables, but license files, dynamic libraries, temporary files, or even the source of the code to exploit the strength of a cluster during the compilation. These demands which are not addressed in the case of clusters, due to the mandatory uniform file system representation used, have led to the formulation of

[1] The X509 based authentication makes the broker useful even outside the underlying VPN.

the "one job-one directory structure" mapping (or briefly to the "job-dir" format). As a historical remark, treating a task as a directory structure is not a novel approach; early implementations of the OpenStep standard (Next, GNUStep) also applied directories for treating environmental resources of the different applications [6].

A job-dir basically defines a static framework to how the user application appears in the file system. A job directory encapsulates all the environment elements needed to execute the jobs remotely, such as binaries (either in statically or dynamically linked formats), necessary libraries in the appropriate version, license files, environment variables, digital signatures to enable data protection, etc. The job directory, as an atomic unit, also allows defining operations over jobs, like job start, job finish, job transfer, job removal, or more complex operations, like execution sequence definition (workflow), children-jobs within a parent job, etc. A sample job directory can be seen in Fig.3.

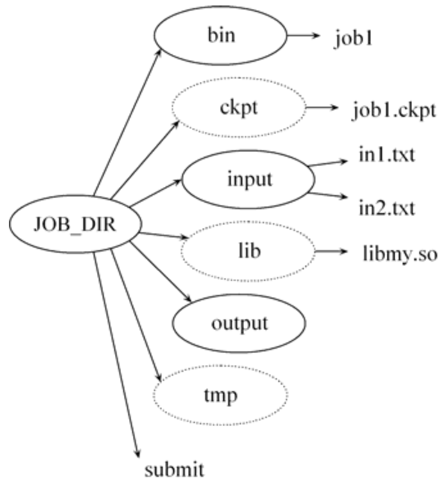

Fig.3 A sample job in "job-directory" format. Solid ovals represent mandatory subdirectories, dotted ovals optional ones. The submit text file describes properties of the job.

By using "job-directory" format the most sensitive parts of the job in terms of privacy can be encrypted and can be decoded on specific nodes or clusters only. In this way, the grid resources can be segmented to involve user groups on the same infrastructure who are working on the same research topic, or simply rely on each other's infrastructure. If the segmentation capability is extended over all types of grid services, virtual organization can be created.

3.4 External interfacing

As the grid is fundamentally a collaboration tool, it is of high importance to allow CG to be interoperable with those grid systems which are built on different principles like Globus grid, or Large Headron's Collider (LHC) grid. Since in the CG architecture design the jobs are defined as static and dynamic entities, and there are job operations defined over them, it is straightforward to implement these operations on a standard web-service interface, such as Simple Object Access Protocol (SOAP).

4 The grid in operation

4.1 Organization structure

To build a production grid, the motivations of different role-players need to be clarified: It was a reinforced experience throughout the CG development, that cooperation as a driving force of interest among the different partners is not enough to have and a much more organic structure among the role-players is necessary. For example, the user community and the resource provider community can be quite different with totally different motivations, and those who have the valuable computational resources do not necessarily want to use them.

When talking about a production grid service, the following participant roles can be identified:

- grid user: a person or an institute who is structurally compute power consumer,
- resource provider: an institute who offers its compute resources into for common pool, and is basically motivated in getting money for his offering,
- grid service provider: a special institute who plays the role of the resource coordinator, gathers and re-distributes local clusters for the user community extending them with extra services like support, coordination over the large common pool, etc.

The former grid roles anticipate a 2-layered model, in which the users take services provided by the grid service providers, grid service providers take, or buy the raw resources from resource providers, and there is no direct connection between the users and the resource providers. The 2-layered model can be used for formulating a semi-market organizational structure, in which the grid service providers can be centrally financed institutes at the beginning, and as the compute resource demand gradually increases, at the same time industrial and business users start to discover and start to use the computational facilities, the financing model can be gradually shifted from central-budget financing to a purely market-based model. Thus, on the long run, the resource market involving resource providers and resource consumers may evolve.

4.2 Monitoring

Monitoring is a cross-section module in the grid reference model, spans over multiple layers providing relevant information on the appropriate operation in real-time.

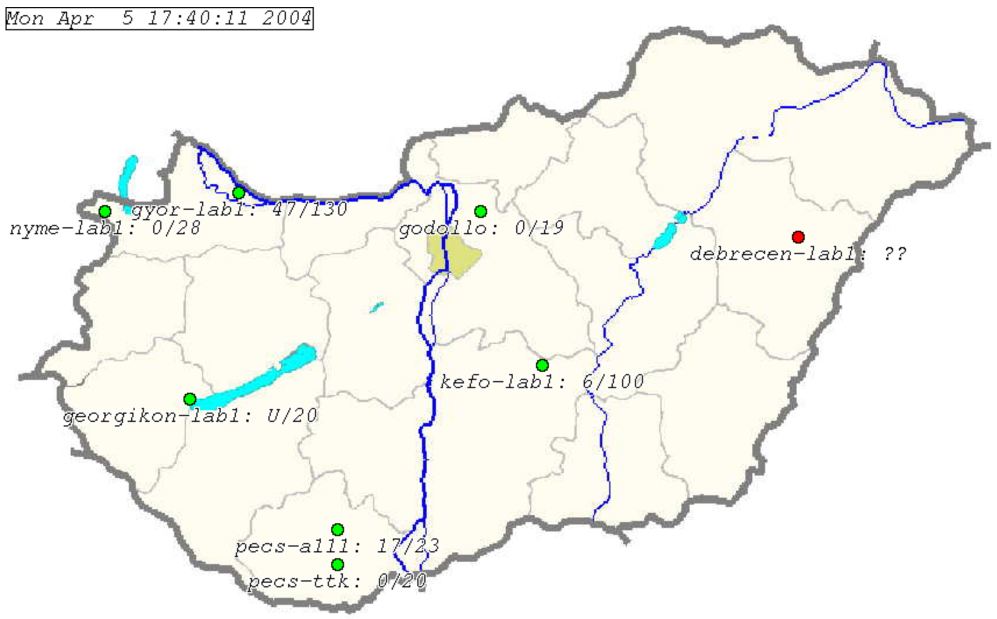

Fig.4 Provincial “weather-map” of the Hungarian ClusterGrid Infrastructure. The spots mark the locations of different clusters; the x/y numbers represent the currently operational nodes with respect to the total number of nodes in a particular cluster. (There are detailed maps for some towns as well.)

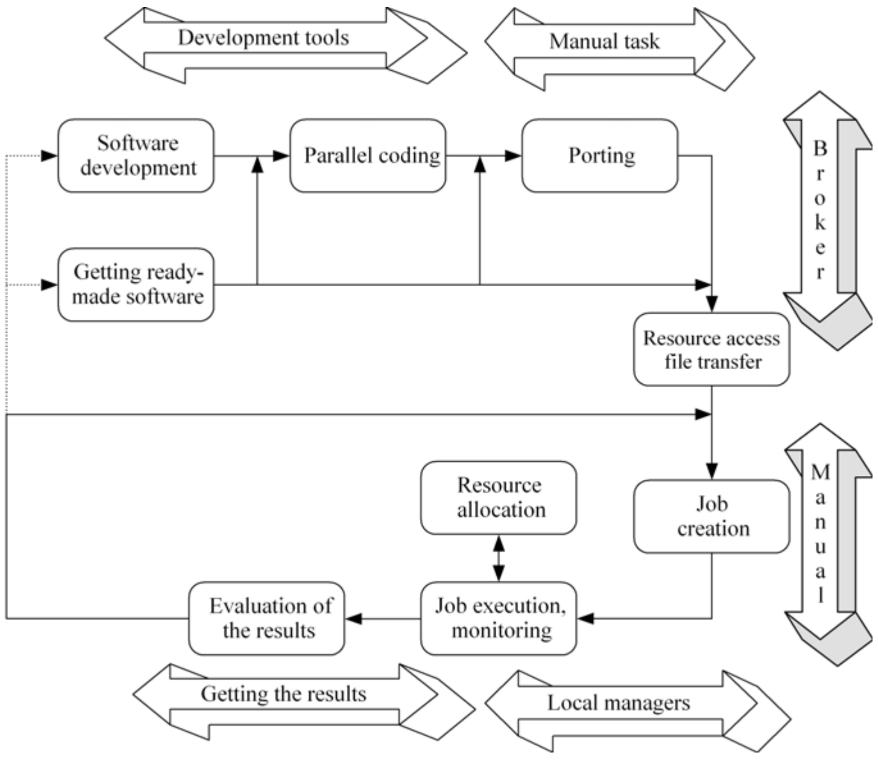

Fig.5 The user cycle that illustrates the process what a user typically does in the system after getting into the entry nodes.

Monitoring helps forecasting errors, finding and repairing malfunctioning components and measuring the overall usability and performance of the system.

In the CG project an open-source monitoring system, MON [5] is used for supervising various aspects of the grid: running services, system utilization, network utilization, free disk space, CPU accounting, availability of resources, etc.

For storing the CPU accounting information RRD database is used. RRD implements high quality data storage, and is also redundant against missed database updates without explicit user-level coding.

Monitoring is supported by graphical tools such as "weather-maps" well-known from network traffic representation tools [7]. Fig.4 shows part of the CG.

4.3 The users' cycle

CG currently supports two classes of user jobs: parameter scanning tasks, and tasks parallelized with PVM library. A user may opt between two access interfaces on entry nodes: a web-portal interface [4] and a command line interface (CLI). By using either, he gets through the sub-processes of the user cycle represented in Fig.5.

The very first step to solve an application problem is to have an algorithm that is capable of exploiting the distributed grid environment, and to have an implementation of it preferably in C, C++ or Fortran. The code itself can either be a self-developed application, by using development tools such as P-Grade [8], or an application most commonly used in the specific scientific research field.

The next step is the porting which basically means to compile and optimize the source code in the Linux environment. Porting is a tedious manual task for those applications which were developed for a different architecture and it needs lots of experience. To make the job to be capable to checkpoint and migrate, sequential checkpoint libraries are used in the case of serial applications, and user-level checkpoint methods are used in the case of parallel applications.

Having compiled the binary executables, the user copies the job execution environment to the entry node, and then, configures a job in the job-dir format.

Properly prepared jobs are then submitted into the grid, where the resource broker finds the appropriate cluster for them. The broker transfers the job environment to the execution cluster, and resubmits it into the local job manager. Whenever the users want, typically when the job is completed, but not necessarily, the results are explicitly drawn from the execution cluster to the entry point, and then the user may analyze the execution results.

The whole cycle then repeats either from the job re-execution, or from the application development.

5 Conclusions

Implementing a production grid is a tedious task, but is an elementary requirement not only inside of Europe, but all over the world. It is getting more and more important to put the results of fundamental grid (and related infrastructure) research into production to provide appropriate feed-back about the real users' real demands to the grid research.

Either as a research field or as an infrastructure development challenge, the area faces a great perspective: there are a lot of questions need to be addressed, a lot of problems to be solved, and a lot of users to be served.

Using virtual private computer networking, revised job structuring, dynamic execution environment mapping, intelligent, distributed, and web-service based resource brokering are some of the contributions that have been developed during the ClusterGrid Infrastructure project.

Although the infrastructure and grid architecture presented in this paper is available as a service, it is far from being ready both in terms of size and implementation quality. Utilizing storage chunks, processing and aggregating grid information to aid brokering, making the solution generally applicable to a wider range of operation systems, enhancing the combined nature of the computation facilities providing interoperable and distributed facilities are just some of the mainstream of the development directions.

6 Acknowledgements

The authors would like to express their gratitude to the ClusterGrid development community (the former Technical Board) involving the members of the Hungarian grid community and representatives of the Hungarian Grid Competence Center: *István Botka*, *István Farkas*, *Gábor Gombás*, *Péter Halász*, *Zoltán Kalmár*, *Bence Kiss* and *Imre Szeberényi*.

The work and moral support of NIIF/HUNGARNET colleagues, namely *Lajos Bálint* and *Miklós Nagy* is also highly appreciated.

References

[1] Foster, I., Gannon, D., Kishimoto, H.: Open Grid Service Architecture, 2003, http://www.globus.org/research/papers/ogsa.pdf

[2] The ISO/OSI Reference Model, ISO/IEC10026, http://en.wikipedia.org/wiki/List_of_ISO_standards

[3] Rosen, E., Rekhter, Y.: BGP/MPLS VPNs, RFC2547, 1999, http://www.ietf.org

[4] The GridSphere project, 2003, http://www.gridsphere.org

[5] MON, Service Monitoring Daemon, 2003, http://www.kernel.org/software/mon/

[6] The GNU OpenStep project specification, 1994, http://www.gnustep.org

[7] The ClusterGrid Infrastructure homepage: http://www.clustergrid.iif.hu

[8] The P-Grade Development Tool: http://www.lpds.sztaki.hu