IST-2001-32133
GridLab - A Grid Application Toolkit and Testbed

Grid of the Grids

Author(s):	Luděk Matyska and Miroslav Ruda
Document Filename:	GridLab-5-D5.10-0008-Tool
Work package:	Testbed
Partner(s):	Masaryk University
Lead Partner:	Masaryk University
Config ID:	GridLab-5-D5.10-0008-1.0
Document classification:	DELIVERABLE

Abstract: This document describes issues related to running a heterogeneous, but production-level Grid. It is based on experience gained running and maintaining heterogeneous GridLab testbed as well as large ad hoc demonstration testbeds. The Grids discussed in this document are built not using dedicated machines or clusters, but their building blocks are other Grids or their parts.

Last amendment date: 2004/12/03 & time: 15:17:00

Contents

1 General overview

Building a Grid that provides production like environment for running complex jobs is still very challenging task. Despite many efforts to facilitate this task, including simplification and standardization of middleware interfaces, software packaging and web and grid service frameworks, no general acceptable methodology exists. The problem can be to some extent simplified when building a uniform Grid, with explicitly defined requirements on resources (worker nodes), their operating system and grid middleware layer. However, as experience from the EU DataGrid and recently EGEE projects does show, still a huge manual effort is needed to build a Grid with a sufficient level of robustness and reliability.
Grid built and maintained as part of the GridLab project posed additional challenges. The most important has been the heterogeneity and the bottom up approach. Each individual partner allocated some resources and the task of the grid management team has been to connect these resources into a production-level Grid without changes at the operating system level. The coordination of these resources has been achieved through appropriate unifying middleware layer, but its correct installation and particularly configuration has been proven as very error prone and difficult task. Also, no resources were fully dedicated to the GridLab. Some were parts of larger local systems, some were even parts of other Grids. This created a unique environment to test and further develop infrastructure monitoring and management procedures and tools.
The experience gained through the GridLab testbed management has been further extended in taking care of the Grid infrastructure for several large scale achievements, usually associated with the SC (formerly Supercomputing) conferences. The first has been a demonstration testbed built as part of the European Grid Forum activities (before it merged with GF creating Global Grid Forum) [1]. The most prominent has been the largest and the most heterogeneous grid (with more than 7000 CPUs, ranging from Sony Playstation through PC based clusters up to the largest US supercomputing facilities) that won two out of three challenges on the SC02 conference in Baltimore (MD). Another example is the large testbed running distributed cactus application during the SC03 conference in Phoenix (AZ).
These large scale Grids were built using (and enhancing) the experience gained from the bottom up grid construction. The building blocks in these cases were not "simple" resources (individual computers or clusters), but whole, independently built and maintained Grids. They were put together using extended versions of procedures and methods developed for the "classical" Grid construction.

1.1 Basic Grid types

There is no specific characterization of individual Grid types. However, based on the middleware used, several basic types can be recognized. The classification is important for understanding differences between them and for specification of interface characteristics. It also helps understanding of problems encountered when combining too different types into one Grid.
We identified the following basic Grid types actually used in Grids used by "real" users (i.e., not just experimental setups for very restricted community):

1. Grid middleware based. The classical representatives are Globus (including the pre-services version 3), Unicore, Akenti and the middleware developed as part of the EU DataGrid project. All the active middleware development projects are currently extended into the web/grid service oriented frameworks.

2. Resource management system based Grids. The classical example is Condor, with its ability to manage large pools of otherwise uncoordinated computing resources. However, other

resource management systems are increasingly used to provide coordinated management of very heterogeneous resources, making them a Grid. The PBS(Pro), SGE, LSF and recently the Community Scheduler are just examples of resource management systems that glue together distributed computing and storage resources.

3. Grid service (or WSRF) based Grids. These are just emerging, with the main examples of Globus and gLite from the EU EGEE project. These Grids could be also classified as middleware based, but as the WS (or WSRF) approach promises much higher interoperability, they deserve a distinction. Also, these Grids are "under construction" and only partially implemented yet.

   A specific group is Grids based on "pure" web services (i.e., no Grid related extensions on the web service level). One such Grid has been built within the GRIA project (`www.gria.org`). Also, the Open Middleware Infrastructure Initiative version 1 release is also based on the web service approach (`http://www.omii.ac.uk/`).

Hybrids of the first two kinds are the most frequent. They usually use some local resource management system to manage local resources and use middleware (almost exclusively Globus) to glue together such systems.
As noted in the previous section, the GridLab testbed is built on very heterogeneous resources, including parts of other Grids. In the following, we list some of the most important examples of GridLab testbed building blocks. The list is not exhaustive, only notable representatives of specific classes are presented:

- Individual resources, not parts of any other Grid:
  - Compaq AlphaServer at the University of Lecce
  - SGI ONYX and Origin in ZIB and Poznan
  - Hitachi SR8000 in Munich
- Resource management systems based Grids:
  - DAS and DAS-2 (Distributed ASCI Supercomputer), based on PBS
  - Hungarian ClusterGrid based on Condor
  - MetaCentrum based on PBSPro
- Hybrid systems:
  - Clusters from Datagrid testbed (PBS with Globus gateway)
  - TeraGrid based on PBS with Globus gateway
  - Korean K*Grid based on PBS with Globus

2 GridLab testbed

The GridLab testbed provides a stable and reliable Grid environment used for the development of new services and tools. It is a production-level Grid, where availability of resources and services is continuously monitored and managed. The testbed is described in previous Deliverables, esp. D5.3 *Production testbed*, and its actual state can be found on GridLab testbed web pages (`http://www.gridlab.org/WorkPackages/wp-5/testbed.html`).
This Grid is extremely heterogeneous. While all machines run some version of UNIX-like operating system (no MS Windows machine), following hardware platforms are its regular nodes:

- Intel (PIII, Xeon) based Linux clusters, managed by PBS, SGE, Condor
- Itanium2 based Linux machine
- Opteron based Linux machine
- SGI Origin and Onyx, running SGI Irix
- IBM Power4+ based machine, running IBM AIX
- PowerPC, 1 GHz G4 based, running Linux
- AlphaServer running Digital UNIX
- Hitachi SR8000 at the Leibniz Supercomputer Center in Munich.

  This is the most "exclusive" machine within the testbed, is its computing environment is very specific, does not include many components available on other machines (e.g., Java environment).

As has been noted above, none of these machines is fully dedicated for this Grid only and some are parts of local/national Grids.

2.1 Middleware

The GridLab testbed belongs to the middleware based Grids. Globus middleware (in its pre-service versions) was chosen as its basic building block. The prescribed Globus version (the testbed started with version 2.2) has to be installed on all the GridLab nodes. The Globus installation for Hitachi was the first of its kind and required strong collaboration between local Hitachi administrator and the testbed and Globus teams. The gateways were configured to use local batch (queuing) systems.
Despite this conceptually rather simple environment, several problems were identified (some of them rather surprising):

- Globus compatibility between versions 2.2, 2.4 and pre-ogsi parts of versions 3.0 and 3.2 was not as complete as expected. Small incompatibilities in `gsiftp` implementations (versions 2.2 and 2.4) and `gsi` implementation itself forced us to restrict Globus version on complete testbed to just one version. This complicated the testbed usability for development, as all the development teams had to use this version regardless of their preferences. Testbed upgrades were also synchronized, simultaneous coexistence of several versions has been impossible.
- Support for local batch systems (submission of jobs from Globus to local batch systems, monitoring of jobs) was problematic. Advantages of individual systems are in most cases very hard to use, while defined basic functionality is very limited. In case of PBS (the batch system used most often on GridLab testbed), practically all sites have to modify globus perl scripts submitting job to PBS according their specific (advance) setup.
- The lack of robustness. Job running in local batch system can be very easily lost by globus (e.g. in case of crash of globus jobmanager or front-end machine).
- Local (mis)configuration. This has been one of the most frequent reasons for site problems. Many different misconfiguration errors were encountered, ranging from incorrect operating

system and/or middleware installation, through bad setup of user environment, missing tools, incorrect library versions up to incorrect items in the configuration files.

However, these problems were rather easily found and corrected, esp. through extensive use of the Grid monitoring system. Rather unexpectedly, the most serious problems encountered were somehow associated with the dedicated nature of some resources/clusters. If the node has been previously used primary (or only) for a specific application, its transfer to more general (open) Grid environment has been very difficult and not always fully satisfiable. Settings, packages and tools required by GridLab, e.g., MPI, F90 compiler, firewall settings, specific environmental variables set for user accounts, but not regularly used previously were too often incorrectly or incompletely installed and configured. Identification of these errors, usually in a specific environment as dictated by the major application or mode of use, has always been a long and difficult process.

Also, these problems tended to have repetitious character. The open ports on firewalls were unexpectedly closed (because the local administrators forget to mark them as permanent and a reboot of firewall machine usually returned to the pre-Grid settings), environmental variables disappeared or were set incorrectly (to some default values), etc. The more dedicated local resource, the more problems were usually encountered.

- Support for MPI jobs was surprisingly problematic. More discussion on this problem can be found below.
- MPICH-G2 (MPICH implementation, which allows running MPI jobs across grid) support is hard—MPICH-G2 must be compiled with need of several "tricks" for globus packaging system, support for latest Globus version was in all cases delayed.

  The robustness of the MPICH-G2 was also bellow expectations, especially when combined with native MPI. Too many processes must correctly start on all machines in almost synchronous way—this requirement lead to too high failure rate, with very difficult error identification. Also, MPICH-G2 is almost useless without strong support from the resource management system—the coallocation of resources is strictly needed for correct synchronous start of the whole MPICH-G2 system.
- Stability of information system and reliability of stored data are discussed below.

2.1.1 GSI enabled access to machines

Besides Globus installation on all machines (front-end machines in case of clusters), gsi-enabled `ssh` was also used. This proved to be very useful, because such solution allowed to use only GSI for authentication—no user passwords were used, user was notified about his new account by open email, ... To avoid problems with local `ssh` setup, the gsi-enabled `ssh` run on non-standard port 2222.

2.2 Common configuration—`/etc/gridlab.conf`

Ability to build a Grid from non-dedicated (shared) components strongly limits extent of system-wide modifications to local system. Also, a mechanism that leaves a decision on placement of libraries, tools, and applications to the local administration has to be developed. The solution adopted within the GridLab testbed is based on just one "well known" item—the `/etc/gridlab.conf` configuration file. This file describes not only location of GridLab software, but also specific local setup of required software—libraries, compilers, development tools,

MPI etc. Complete set of defined variables is published on GridLab web pages, the current version is in Appendix.
This way, the GridLab testbed accepts and supports differences between local machine, operating system, middleware, and application setups.

2.3 Gridlab virtual organization

The whole GridLab testbed run only one virtual organization (VO). It has been maintained by the Grid Operation Center (GOC) in Brno and served mainly the authorization purposes. Developers from all parter institutions were allowed to join this VO directly through GOC, without need of any contact with other institutions. Gridlab Steering committee have to approve accounts for non-GridLab organizations. GOC asked sites to include new persons. The site autonomy is preserved, as sites has power to deny access for specific users when needed (e.g., by their usage policies).
GridLab VO is managed by software developed by Masaryk University (Perun [26]). Template grid-mapfile is published to LDAP and on a web and it's up to local administrators (notified by email) to update their local management systems. While this solution is very static and doesn't scale for large number of users, it allowed us to preserve site autonomy and not influence local management systems. And this approach served well even for large scale ad hoc Grids used for demonstration purposes—the scalability problem was reduced through just limited number of actual users. However, more scalable and automatic procedures must be developed, the VO management could become a bottleneck for a large-scale production systems.

2.4 Authentication

Because Globus middleware was used for integration, X.509 certificates are used for user authentication. GSI enabled access is the only guaranteed way in Gridlab testbed, no passwords and `ssh` keys are used. The only exception is access to CVS, where `ssh` keys are also used (regular and irregular CVS checkouts are part of the GridLab monitoring suite, so authentication to CVS is also an important Grid related issue).
In the area of Certification Authorities, we have leveraged on work provided by DataGrid and CrossGrid projects (currently transformed to EUGridPMA activity), where trustworthy and dependable CAs were established in almost all European countries. We have decided to accept all DataGrid and CrossGrid CAs, and also several US CAs—DOE, NCSA. A specific Gridlab CA was also established, but it is used only as temporary, backup solution, without any intention of establishing well known, world trusted CA. The GridLab CA serves as a "catch-all" CA for participants from countries where no EUGridPMA accepted CA is available.
MyProxy server was provided for Gridlab project—all users were allowed to use central MyProxy server. This server was also used by Resource Management System, Portals and testbed monitoring tools.

2.5 Parallel (MPI) jobs

Correct setup of Globus gateways to local batch systems emerged as one of the major problems. The support of parallel MPI job was the most dramatically influenced, but the problem complicated running other kinds of jobs, too. In general, Globus supports submission of both MPI and multiple (one program is started multiple times) jobs to several local batch systems (but not for all, e.g. Condor is not supported). However, correct setup of this gateway was found very tricky—nearly on all sites, specific modification of scripts, which are preparing scripts submitted later to local batch system were needed. When several MPI installations are available

on cluster, for example LAM and several MPICH versions, or MPI with support for specific networks—Myrinet, general version running on sockets, etc., only one version is chosen by local administrator and other implementations cannot be used. This information is not published anywhere. However, it is a crucial piece of information for ordinary users, which must use only this implementation when preparing (or recompiling) their application.
Moreover, the whole problem is complicated by the fact that some needed information (port range for open ports on firewall, path where dynamic libraries should be found, path to license files needed by compilers) is set only in environmental variables. These variables are by default not carried by globus gateway, local batch systems nor by `rsh`, which is used by MPI for startup of subtasks.
In GridLab, problem is addressed by strict check (and in case of need by modification) of all points, were environment can be destroyed and by configuration file `/etc/gridlab.conf`, where some of these variables can also be set.

2.6 Information systems

Reliable and up to date information about resources and their state is a necessary prerequisite for a functional Grid. The MDS, which is a part of the Globus system and was used as the primary service for GridLab testbed, was also not without problems.
The most important has been associated with GIIS, i.e. the "Grid of Grids" top level server. It is implemented as an OpenLDAP server, with very limited robustness support. This index server actively queries the GRIS servers (or even GIIS of the interconnected Grids), but is not protected against their misbehavior.

- Firewall problem. If a machine behind firewall registers into GIIS, it can completely hang up or drastically degrade its behavior. The actual manifestation depends on the firewall setup and the kind of packets not transfered.

- The "real" Grid always has some disconnected (crashed) nodes. Again, any query to such a machine will take rather long time (waiting for timeouts, that must not be too short to serve well even in case of overloaded nodes and networks).

In both cases, the situation is further complicated by the MDS implementation, where even a direct query to a specific machine leads to more queries on other registered machines. Each GIIS query thus take prohibitively long time, slowing scheduler, which relies on the information provided by the information service, to unacceptably low speed. The problem growss up with the size of the Grid, becoming unacceptable for large Grid of the Grids.
The solution, which we adopted for such situation, consists of introduction of a GIIS server with unlimited timeout (i.e. data never expire) combined with a more classical GIIS running in parallel and periodically querying all sites. This secondary GIIS provides data to the primary GIIS in a push mode. This way, most of the time the query to the primary GIIS will give a correct up to date answer, but it always take the same time (as all the information is provided locally).
More conceptual solution is a replacement of OpenLDAP MDS with a more robust system, like iGrid, also developed as part of the GridLab project. However, the iGrid is just only starting to be really tested in the GridLab testbed.
Another problem encountered with the information services lies in the data unreliability. All the "classical" grid information systems do not provide any validity checks on the data they publish. Information on clusters reporting thousands of processors, always empty queues or even negative number of processors is accepted by the information services and later used via

resource management system with all the negative consequences. This problem we solved using information from the infrastructure monitoring as an augmentation of information services, including consistency or 'reality" check.

2.7 Connectivity and firewalls

While the construction of a Grid requires a network connection between all nodes, the local security policies of individual nodes could block some or all network traffic. When site resources are behind a firewall, the local security policy directly influences the usability of such a site within a Grid. The local security rules are usually configured without wide area resource sharing in mind. Such a setup may be acceptable for a local activities, where all external traffic may be reduced and controlled, but becomes a serious problem in the case of distributed Grids.
Use of firewall to enforce network connectivity restrictions does not immediately rule out usability for grid applications. However, it may become a serious performance bottleneck—most firewalls are not capable to support traffic flow at the wire speed of high speed networks—and it also brings new administration problems. Some ports or port ranges must be opened, application must be informed about available open ports and must be adapted to adapt their network communication for these ports only.
Within the GridLab testbed, the firewall information is published in the information service. Grid infrastructure monitoring continuously checks correct firewall setup—without direct access to the firewalls themselves, this check is performed in a "black box" mode, i.e. failures are detected from the timeouts and/or node or site inaccessibility.
Worker nodes on clusters are sometime hidden behind a network address translator (NAT). If the NAT does not support external IP connectivity, it also prevents some types of applications. In some cases this problem can be overcome using a specific gateway—we used Mercury monitoring system, which supports forwarding of information from/to worker nodes through the front-end machine—but it seriously degrades performance and still does not provide a general solution.
The problem with private IP addresses behind NAT could disappear with the general use of the IPv6 protocol, but the use of firewalls to limit network connectivity is conceptually contradicting the Grid idea of resource sharing. The result will be always some kind of compromise, putting burden both on monitoring and management of Grids.

3 Open problems

We can summarize problems associated with building Grid of Grids to the following categories:

- Technical problems. While a substantial improvement can be seen in the last few years, the quality of the middleware implementation is still far from satisfactory. In a heterogeneous environment, some components were not thoroughly tested on some platforms or hardware/operating system combinations. Also, the dynamic character of any Grid, with node failures not exceptions but a regular events, is not yet fully reflected in the software. The middleware does not anticipate, only reacts on failures, so some failure modes were never tested and systems hang or crash.

- Security. The authentication through X.509 certificates requires in fact existence of a working PKI infrastructure, or at least a coordination and mutual trust at the level of Certification authorities. Connecting two Grids with no previous overlap in CAs may become a political problem (see also next item)—technically both Grids are fully compatible, but users are not able to authenticate to the "other" part of the full Grid.

For Globus, there is a problem with certificate chains. The public keys of all supported CAs must be copied to all machines, which is clearly an error prone and non-scalable requirement. For production grid, some solution with cross-signing CAs and bridging CA should be used.

- Policies. Currently, there is a lack of formal description of local policies—security, resource access and use, rules for getting accounts, ...—governing individual Grids. Without this formal description, it is impossible to automatically generate the "common denominator" policy for a Grid of Grids. Also, tools for such automatic policy generations are still an open research problem.

  Another open problem is rules for resource sharing. While the partners providing their Grids are usually willing to contribute, they expect a "fair" treatment, where no resources are abused nor underutilized. Some kind of "barter" or "clearing center" is a viable approach, but it requires access to accounting data, which are usually kept only locally and not distributed to other partners. This can bee seen as a first step towards full economic grid models.

4 Conclusion

Building a Grid is a complex task, still requiring a lot of manual work. The complexity even rises if the Grid is built bottom up, connecting existing heterogeneous resources, administered by different units, into a usable distributed system. And when these resources are Grids or their parts, the problems are almost intractable and must be solved via ad hoc approaches.
Probably the major problem lies in the lack of formal specification of a concrete Grid. Even use of the same middleware—e.g. the Globus system—does not imply that two such Grids will interoperate. Even if the same middleware version is used, there is no guarantee that local configurations will be compatible. On the contrary, our experience has shown that all sites add some "local" configuration details which makes all the installations unique and thus not easily interoperable. And these "tiny" changes are usually nowhere described—in fact, the local administrators often do not consider such "localization" as a harmful and something that must be somewhere explicitly described. The first step in building Grid of the Grids is thus search for common semantics, only after agreement on basic semantical categories the Grids could be glued together.
Grid information systems are another source of potential problems. Connecting information systems of previously independent grids usually opens new synchronization problems, leading to unexpected drastic slow downs or even complete hang ups of the whole information system. Again, semantics of individual fields must also be unified—even small differences may have huge consequences (e.g., the resource broker may be completely mislead).
Combination of resource brokers and schedulers is also a challenging undertaking. The understanding of complex behavior in presence of many resource brokers with different policies is incomplete and tuning such a Grid with many schedulers is a manual and tedious work.
While it is possible to build a Grid of Grids, it is very complex and difficult task. Too many things—from changes in the configurations through new service setup up to policy coordination—must be performed manually, with no or only little understanding of complex interaction between individual components. While better middleware implementation quality may help, as well as new tools for (semi)automatic processing, there are many open research problems that must be solved before building Grid of Grids could become an automatic and easy process. Some of the most promising areas for such research are discussed in the work.

A Variables defined in `/etc/gridlab.conf`

Variables defining installation root:

Variable	Default value	Description
HOSTNAME	–	FQDN
HEADNODE_HOSTNAME	–	FQDN of head node on clusters
GLOBUS_LOCATION	`/opt/globus`	Root of Globus installation tree
GLOBUS_FLAVOR	`gcc32dbg`	Globus flavor
GLOBUS_FLAVOR_THREADS	`gcc32dbgpthr`	Threaded Globus flavor
GLOBUS_INCLUDE		Include directory for non-threaded Globus flavor
GLOBUS_INCLUDE_THREADS		Include directory for threaded Globus flavor
GRIDLAB_LOCATION	`/opt/gridlab`	Root of GridLab installation tree
GRID_SECURITY	`/etc/grid-security`	`grid-security` directory, it contains host certificates, certificates of accepted CAs
GRID_MAPFILE	`$GRID_SECURITY/grid-mapfile`	Location of `grid-mapfile`
CERTIFICATION_AUTHORITY_DIR	`$GRID_SECURITY/certificates/`	Location of accepted CAs

Variables defining path to required or optional software. These variables are defined only if corresponding software is installed.

Variable	Description
F90_NATIVE	Pathname of the native F90 compiler
F90_KAI	Pathname of the Kai F90 compiler
F90_PGI	Pathname of the Portland Group F90 compiler
F90_INTEL	Pathname of the Intel F90 compiler
F90_GNU	Pathname of the GNU F90 compiler
CXX_NATIVE	Pathname of the native C++ compiler
CXX_KAI	Pathname of the Kai C++ compiler
CXX_PGI	Pathname of the Portland Group C++ compiler
CXX_INTEL	Pathname of the Intel C++ compiler
CXX_GCC	Pathname of the GNU C++ version 2.x compiler
CXX_GCC3	Pathname of the GNU C++ version 3.x compiler
C_NATIVE	Pathname of the native C compiler
C_KAI	Pathname of the Kai C compiler
C_PGI	Pathname of the Portland Group C compiler
C_INTEL	Pathname of the Intel C compiler
C_GCC	Pathname of the GNU C version 2.x compiler
C_GCC3	Pathname of the GNU C version 3.x compiler
JAVA_HOME	Path to the Java installation tree
GMAKE	Pathname of the GNU make
CVS	Pathname of the CVS
PERL	Pathname of the Perl (version >= 5.005)
HDF5_LOCATION	Root of HDF5 installation tree

Variables defining path to Gridlab software. These variables are defined only if corresponding software is installed.

Variable	Description
GRIDLAB_MONITORING_LOCATION	Root of Gridlab monitoring software (Mercury) installation tree
GRIDLAB_MDS_LOCATION	Root of Gridlab information service installation tree
GRIDLAB_IGRID_LOCATION	Root of iGrid installation tree
GRIDLAB_ADAPTIVE_LOCATION	Root of Gridlab adaptive software (Pythia) installation tree
CACTUS_LOCATION	Root of Cactus installation tree
CACTUS_DEV_LOCATION	Root of Cactus (development version) installation tree
GAT_LOCATION	Root of GAT installation tree
GAT_ADAPTOR_PATH	GAT Adaptor list
GAT_DEV_LOCATION	Root of GAT (development version) installation tree

Variables describing MPI installation are defined to be compatible with Cactus developers manual (`http://www.cactuscode.org/Guides/Stable/UsersGuide/UsersGuideHTML/node14.html`). The main difference is that all available MPI installations should be described. These variables are defined only if corresponding software is installed.

Variable	Description
MPI_DEFAULT	Default MPI, used by globus jobmanagers when MPI job is started. Can be one of `native`, `mpich`, `mpich-gm`, `lam`, `mpipro`.
MPI_NATIVE_DIR	Native MPI is available on this machine available in defined directory.
MPI_NATIVE_LIBS	If defined, linking with native MPI will use this libraries instead of "`-lmpi -lm`"
MPICH_DIR	Path to MPICH (`http://www-unix.mcs.anl.gov/mpi/mpich`) installation tree. If defined, corresponding MPICH_ARCH and MPICH_DEVICE variables should be defined too.
MPICH_ARCH	Machine architecture
MPICH_DEVICE	The device used by MPICH
MPICH_GM_DIR	Path MPICH-GM (`www.myri.com`) installation tree. If defined, corresponding MYRINET_DIR variable should be defined too.
MYRINET_DIR	Directory in which Myrinet libraries are installed.
LAM_DIR	Path to LAM (`http://www.lam-mpi.org/`) installation tree.
MPIPRO_DIR	Path to MPIPro (`http://www.mpi-softtech.com/`)installation tree.
MPICHG2_DIR	Path to MPICH-G2 (`http://www3.niu.edu/mpi/`) installation tree. If defined, corresponding MPICHG2_FLAVOR variable should be defined too.
MPICHG2_FLAVOR	Globus flavor used by MPICH-G2.

References

[1] Gabrielle Allen et al., Early Experiences with the EGrid Testbed. First IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2001), May 15-18, 2001, Brisbane, Australia.

[2] Globus - http://www.globus.org/

[3] Unicore - http://www.unicore.org/

[4] Avaki - www.avaki.com

[5] PBSPro - http://www.pbspro.com/

[6] OpenPBS - http://www.openpbs.org

[7] Condor - http://www.cs.wisc.edu/condor

[8] Sun Grid Engine - http://gridengine.sunsource.net/

[9] The Distributed ASCI Supercomputer 2 (DAS-2) - http://www.cs.vu.nl/das2/

[10] MetaCenter - http://meta.cesnet.cz

[11] The Hungarian ClusterGrid - http://www.clustergrid.niif.hu/

[12] K*Grid - http://testbed.gridcenter.or.kr/eng/project_01.htm

[13] The Southeastern Universities Research Association (SURA) - www.sura.org

[14] TeraGrid - http://www.teragrid.org/

[15] Datagrid testbed - http://marianne.in2p3.fr/

[16] EGEE testbed - http://egee-sa1.web.cern.ch/

[17] MyProxy - http://grid.ncsa.uiuc.edu/myproxy/

[18] GSI-Enabled OpenSSH - http://grid.ncsa.uiuc.edu/ssh/

[19] iGrid, the GridLab Information Service - http://www.gridlab.org/WorkPackages/wp-10/

[20] Mercury - http://www.gridlab.org/WorkPackages/wp-11/

[21] Gridlab testbed monitoring tool - http://www.gridlab.org/WorkPackages/wp-5

[22] The EU Grid PMA http://eugridpma.org/

[23] MPICH-G2 - http://www3.niu.edu/mpi/

[24] GRIA project - www.gria.org

[25] Open Middleware Infrastructure Initiative - http://www.omii.ac.uk/

[26] Aleš Křenek and Zora Sebestiánová. Perun – Fault-Tolerant Management of Grid Resources, Cracow Grid Workshop'04.