Deployment of gLite Middleware: An E-Science Grid Infrastructure

Mohmmad Hasanzadeh and Mohammad Reza Meybodi
Computer Engineering and Information Technology Department
Amirkabir University of Technology (Tehran Polytechnic) Tehran, Iran
{mdhassanzd, mmeybodi}@aut.ac.ir

Abstract— **In recent years, a new generation of distributed systems is evolving in the internet bed. Grid computing is one of these brand-new highly heterogeneous technologies which expanded into worldwide without any limitations on location expansion. The main scope of a Grid is to execute user's jobs by its available set of resources. However, a potential Grid needs to be scalable, fault tolerance and immune from network congestion. In order to deeply comprehend the key issues of Grid, in this paper we install and configure gLite (Lightweight middleware for Grid Computing) middleware. We also review the role of some basic gLite components and propose our customized grid architecture. The main idea of our research is to build a scalable platform for E-Science application from universities, organizations and industries computing power. We believe that these installation details and module regulation of gLite Grid would become beneficial for academic and industrial researchers who are employed in the design and implement of scalable Grids.**

Keywords- Grid computing; gLite Grid middleware; High Throughput Computing (HTC); E-Science.

I. INTRODUCTION

Grid computing [1], [2] is an evolving IT infrastructure from existing technologies such as: distributed computing, web services, internet cryptography and virtualization. The Grid [3] consists of a dynamic collection of Virtual organizations (VOs) [4] which provides users with resources. There are several motivations for using a Grid rather than other distributed systems such as: fault tolerance and reliability, resource sharing and balancing, parallel processing and Quality of Service (QoS).

The Grid consists of four basic layers: fabric, core middleware, user-level middleware and applications [5]. Grid middleware [6] is computer software which provides services for developers, users and applications of the Grid. The fabric layer of Grid accumulates a set of computational, storage and network resources. In order to efficiently use these resources, we need a middleware which facilitate and protect the resource providers. The gLite [7] is a middleware for building a Grid which is integrated a set of components designed to enable distributed computing, resource sharing along the internet. The fundamental components of gLite Grid are Computing Element (CE), Storage Element (SE) and Workload Manager System (WMS). Also, other components like User Interface (UI) have crucial influence in Grid operation. The gLite middleware [8] was produced by EGEE (Enabling Grids for E-Science in Europe) project [9]. The EGEE [10] aims to build a reliable European Grid infrastructure for e-Science.

This work is supported by Iran Telecommunication Research Center (ITRC) grant.

Recently, Grid computing research topics widely attracts researchers' attention. But there is a mere attention to Grid's functionality and deployment. A gLite Grid could consist of several Grid sites with heterogeneous node distribution. Assume there is a university with large number of computers and work stations which they are idle at least half-day. What if we could use this idle machinery power to build a high performance Grid and use it for e-Science projects like: virtual eye surgery [11], galaxy virtualization [12], High Energy Physic (HEP) and medical image processing [13]. Our goal is to give the perspective of building a Grid from a small cluster to a campus Grid for Iran's small institutes or universities. The gLite flexibility and its proper support encourage us to introduce it as this campus Grid infrastructure [14].

In this document we present the installation and configuration of gLite 3.2 Grid middleware [15]. The content of this paper is a handbook of adjustment of gLite Grid nodes. The user should refer to gLite Grid middleware official site [7] to download the updated engine version of each node. The gLite Grid 3.2 is supported by Scientific Linux 5 (SL5) operating system [16]. The gLite default package manager for SL5 is YUM (Yellow dog Updater, Modifier) repositories [15] and it is the recommended installation tool for each Grid node. After installation of a certain node type, the YAIM (YAIM Ain't an installation Manager) configuration tool [17] is needed to configure the corresponded node.

The rest of this paper is organized as follow: section 2 reviews the gLite Grid middleware architecture. Section 3 review our proposed campus Grid infrastructure and highlights installation guide of particular gLite Grid modules. Finally in section 4, we draw some concluding remarks and present an outlook on future works.

II. THE GLITE GRID MIDDLEWARE ARCHITECTURE

As we shown in figure 1, Grid [6], [18] consists of four key layers of operation: fabric, core middleware, user-level middleware and applications. The first layer is fabric layer which contains the Grid resources. The second and third layers of this architecture belong to the grid middleware layer. The middleware layer is divided into two levels: core Grid middleware and user-level Grid middleware. Core Grid middleware offers some elementary services of the Grid such as resource management, information system, security and QoS, while the user-level grid middleware presents a high level of abstraction for resource brokers [19] and Grid applications. The fourth level of this architecture is application layer.

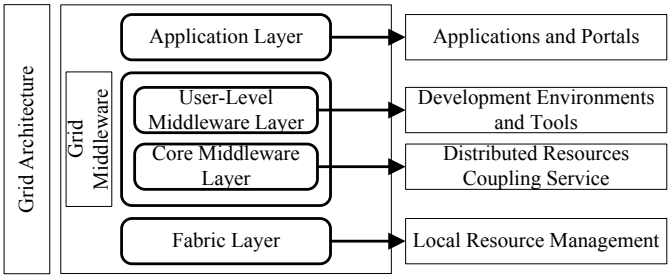

Figure 1. The Grid Middleware position in the Grid architecture.

The gLite [6] is a service oriented grid middleware which bring the required services for distributed computing management, storage resources, security, accounting and information services together. The gLite system [5] composes of some high level services which could verify and install on PCs, workstations, Clusters, etc. The structural view of configured gLite Grid modules is depicted in figure 2. It shows main modules of our proposed architecture which are listed as follow: Berkeley Database Information Index (BDII), ApelBox accounting service, batch system server, Worker Node (WN), CREAM Computing Element (CE), Virtual Organization Membership System (VOMS) and User Interface (UI).

III. Installation and Configuration of gLite Main Modules For E-Science Grid Facility

The main purpose of our work is to encourage small institutes, industrial corporations and prestigious universities to build their customized Grids upon their organization's vacant computing power without investing a penny. In the rest of this paper, we focus on campus Grid. As we shown in figure 3 there are lots of idle computing devices in the university area, including: the computer site of each college, laboratories servers, mainframe computers and high performance computers. These PCs, workstations and servers are idle at least half of the day. Our primary goal is to join these gratuitous computing machines via a Grid infrastructure and use them as a giant computing device for e-Science applications. As long as Grid is a scalable platform, these devices could plug and play whenever their users want. Also each college could take the role of one Grid site and all these sites will merge together as a global Grid site.

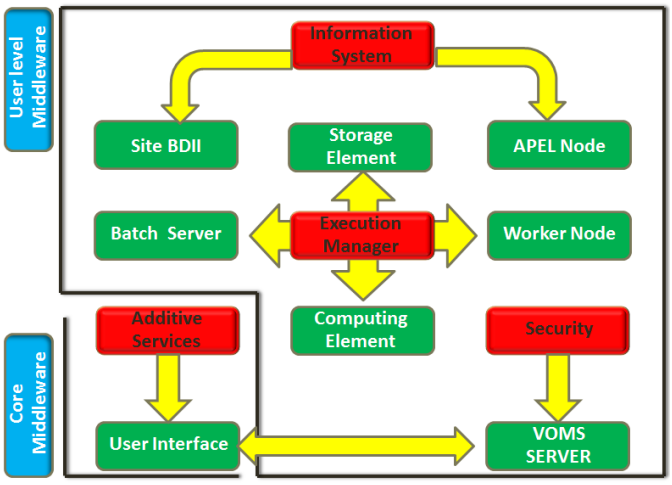

Figure 2. The gLite Grid modular architecture.

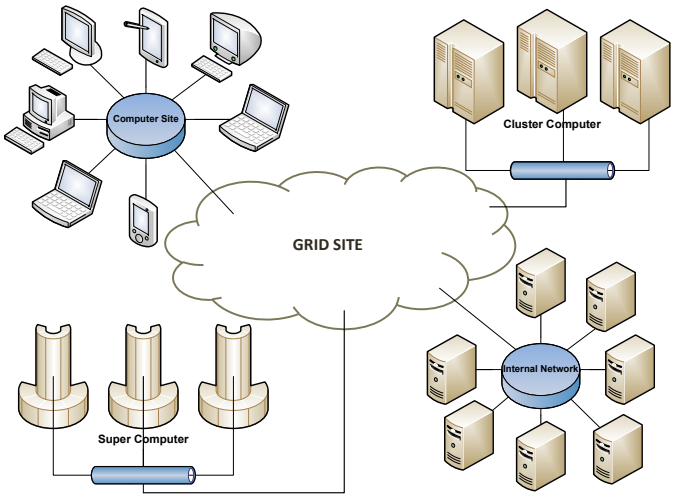

Figure 3. The topological view of campus Grid components.

A. *The gLite installation prerequisites*

The version of Implemented gLite Grid middleware of this paper is gLite 3.2. The user could retrieve packets and scripts of gLite 3.2 from [7]. The *Java* software is one of the important requirements of gLite, so be sure about its installation in each functioning node.

Host certificate is needed to install all nodes except for WN, BDII and UI. The Iran Grid Certificate Authority (IR GRID CA) [20] is the Certificate Authority (CA) which provides X.509 digital certificates to support Iranian Grid activities in e-Science. A valid certificate contains the public (hostcert.pem) and private (hostkey.pem) keys.

We should assure that the required packets were installed in the host machine. The following packets are crucial for all gLite nodes: *jpackage*, *c-ares*, *xerces-c*, *fetch-crl* packets. Also, in order to establish the connection between corresponded node and gLite repositories *yum.repos.d* should be configured. Finally the gLite CA certificate should install by *lcg-CA* command.

The *site-info.def* file [21] is the essential file of the gLite site that should be initialized for each node by site admin. Each gLite module has compulsory variables to redefine in this file. Having a proper communication between gLite modules, you should configure the Domain Name System (DNS) and define a Fully Qualified Domain Name (FQDN) for each node.

B. *Deploying Site Berkeley Database Information Index (BDII)*

The information system of gLite Grid runs as a hierarchy of SLADP (Stand-alone Lightweight Directory Access Protocol) daemons each aggregating the information of lower level. This information system consists of 3 levels: 1) Resource level which is run on SEs and CEs. 2) Site BDII which collects the information of the site service. 3) Top level BDII which collects all the site information. In our campus Grid the resources are all PCs, workstations, Servers, super computers of the university. Each college consider as a site BDII and the whole university act as a top level site BDII which interconnect the entire site BDII through the gLite Grid middleware.

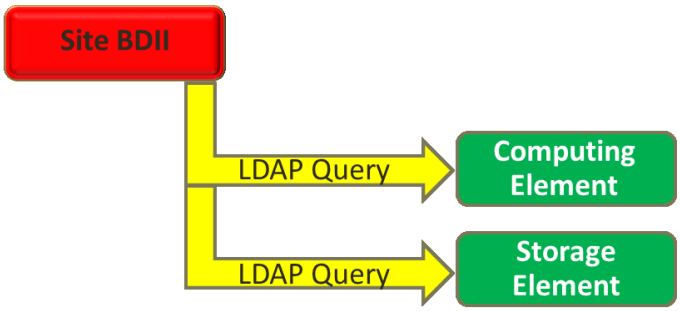

Figure 4. The schematic view of information collecting process of site BDII.

In order to be visible as a Grid, every Grid site should have one or more BDII site. The site BDII is the central information system of the Grid which collects all the information of site's gLite services. The core service of BDII is called BDII_top which accumulate information from all site BDIIs and supply users and resource brokers with this information.

Site BDII collects local resource information via LDAP protocol (see figure 4). For Grid, the standard LDAP port is set to 2170. Because of critical nature of the site BDII, it should be installed as an independent service, so other services malfunctions could not jeopardize the site's functionality.

Site BDII [22] contains two or more LDAP databases. Port forwarding is used to enable one database while the other one is updating. After installing BDII required repositories, the site BDII needed a lot of variables modification in *site-info.def* file, such as: the name of the site, location of site BDII, main email contact for site, etc. After adjusting the mandatory variables of *site-info.def* file, the site should configure by YAIM [17]. The YAIM placed all the site BDII resources information in *site-urls.conf* file. Now we can query site information from any node of Grid site which is equipped with LDAP protocol. The most important options of site BDII is placed in *site-url.def* file [21]. The Grid Information Provider (GIP) is a folder which contains static and dynamic information about every plugin and provider of gLite Grid site.

C. Deploying ApelBox Accounting Service

The gLite-APEL [23] is an accounting tool which provides a *MySQL* database for site participants. The ActiveMQ is the embedded transport mechanism of APEL. The central accounting database of APEL processed the accounting data of site to generate statistical summaries. The APEL structural view is sketched in figure 5. Note that in gLite Grid 3.1, MonBox provides the Rational-Grid Monitoring Architecture (R-GMA) and some basic accounting services, but in gLite Grid 3.2, MonBox and R-GMA are replaced with ApelBox and ActiveMQ, respectively.

The ApleBox prior to installation are *fetch-crl* and *MySQL* metapackage. Then the required repositories for ApelBox be downloaded and installed via YAIM [17]. There are some mandatory variables which should be edited in *site-info.def* file [21], such as: APEL host machine name, APEL database password, MySQL database password, CE machine, MON machine and site name. At the end by executing the *glite-apel-publisher* command the accounting information of gLite site will be issued.

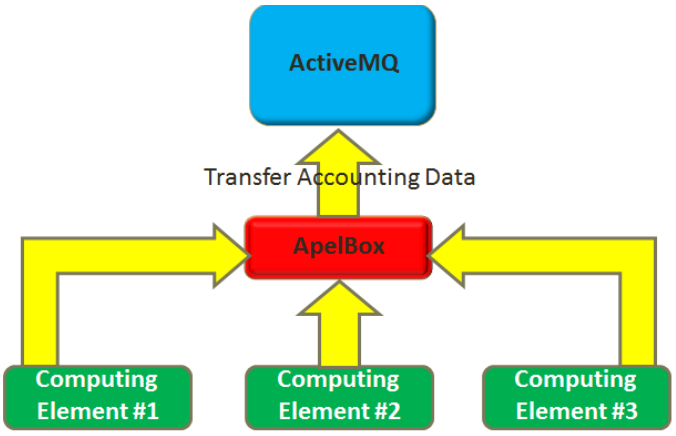

Figure 5. The structure of ApleBox accounting service

D. Deploying Batch Server System

After configuring Grid site and information system, we continue to establish the gLite batch system [24]. A batch system commonly includes a batch system server and bunch of worker nodes. The submitted jobs will be executed by batch system. Batch system server is the main component of local resource management system. Usually this system has the responsibility of job management and scheduling, consists of: 1) Job queues management. 2) Job routing between multiple queues. 3) Assigning Jobs to WNs. 4) Monitoring Jobs status.

The user's jobs usually submit through a batch system client like Portable Batch System (PBS), TORQUE or Maui. PBS is a scheduler and its principal task is to allocate available computing resources to users tasks. TORQUE resource manager which is produced from PBS project, has the functionality to handle batch jobs upon distributed computational resources. Maui is an intelligence scheduler which is the replacement of naïve FIFO scheduler. Maui could manage, monitor and troubleshoot the system more robust than the FIFO queue policy. In the proposed gLite Grid architecture the mentioned batch system client is installed on CREAM CE node. Having installed TORQUE and Maui packages from gLite repositories, they should be configured by YAIM tool [17]. The schematic view of Batch server system is shown in figure 6. The job submission flow start form the CE node, then batch server schedule and distribute them among its servitor WNs.

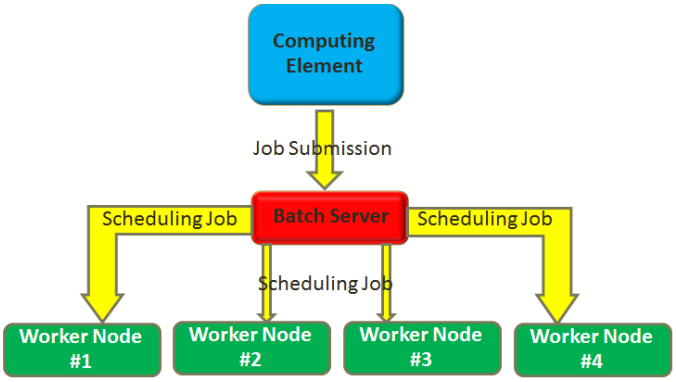

Figure 6. The job submission in batch system.

There are some compulsory variables in *site-info.def* file [21] which should be set: 1) The CE host machine address which enables the users to communicate with batch system. 2) The list of available job Queues. 3) The WN list which points the WNs addresses. 4) The groups and users which are authorized to submit job.

After running the Maui and TORQUE and also connecting at least one WN to the batch system we could use some additional tools like: 1) The *pbsnode* command to get the node's status. 2) The *queue* command to grasp the queues condition. 3) The *showq* command to show the jobs status in a specific queue. 4) The *showres* command to determine the reservation status. 5) The *qmgr* command to manage the queues. Also other commands like *checkjob, diagnose, tracejob* and *qsub* might be useful to manipulate jobs from batch server system or CREAM CE.

In order to transmit the accounting information of batch server to CE, batch system should run a Network File System (NFS) server. NFS provide a fast file sharing mechanism for the network. NFS mount a disk partition of a machine into another one. NFS establish the accounting flow of gLite Grid.

E. *Deploying Worker Node (WN)*

A Worker Node (WN) [25] is part of the local resource management. The submitted jobs of the batch system are scheduled via batch system server and allocate to WNs of the system. While the WN is a part of the batch system, the installed TORQUE client on it reports about the WN/job status to the batch server system. The TORQUE client software provides a command line to establish the information system queries. Figure 7 shows the bidirectional relation of batch server and WN.

Before running the WN the following metapackage should be installed on it: *xercex-c* and *gLite TORQUE_server*. The gLite WN is a set of clients which are used for executing the job. In gLite 3.2 the WN is equipped with Site Wide Area Testing (SWAT) packages. SWAT is a Grid monitoring tool which collects the adjustment information of all WNs. To set up the WN properly, TORQUE server and TORQUE client should install on the host machine.

Also, there are some important variables about Grid site and Virtual Organizations (VOs) which are supported by Grid site in *site-info.def* file [21], such as: site BDII host machine address, APEL host machine address, Storage Element (SE) list, etc. now we can deploy the desired configuration into the WN host machine by YAIM tool [17]. Finally, before running batch system software, we could test it by using *momctl* command through the Linux shell.

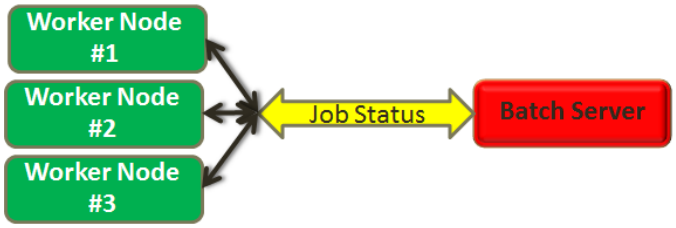

Figure 7. The relation between batch system server and WN.

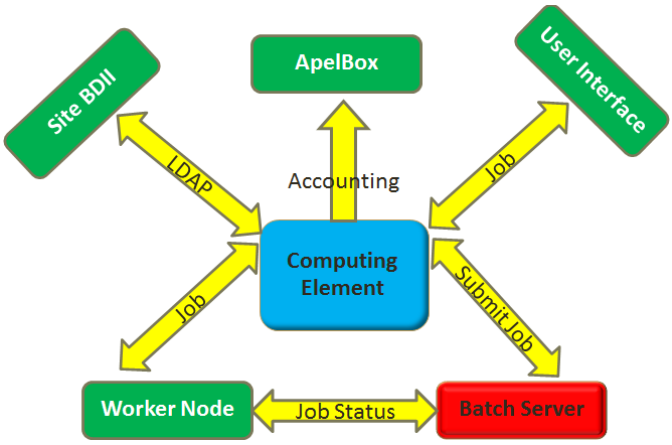

Figure 8. The architecture overview of Computing Element (CE).

F. *Deploying Computing Resource Execution And Management (CREAM) Computing Element (CE)*

After successfully installing the batch system, it's time to run CREAM CE [26] for Grid users. CE is a portable machine for transmitting the Grid's jobs to batch server. CE is responsible for information propagation of batch system to site BDII and also authentication of the users. CREAM CE receives the job submission and management requests. Figure 8 shows a stand-alone CE host machine in the campus Grid architecture. The user's jobs are delivered to the batch system through CE and then execute on WNs. After finishing the job, the job's output is copied from WN to CE host machine while the ApelBox take its accounting role for jobs. Similar to other middleware's software, a LDAP service maintain the connection between CE and site BDII for transmitting the computational resources information.

There are multiple ports in CE for establishing connection between internal services like site BDII and WMS. The BDII site uses LDAP, which listens to 2170 port number. CREAM CE web service listen to 8443 port number via *tomcat*. Finally for data transmission of *Gridftp,* the 2811 port number should be open.

The glite-CREAM package is composed of several lightweight services which administrate the work flow of Grid. The user could install CREAM CE via YAIM [17]. Also a TORQUE client should be installed on the CE node. Finally the designated Virtual Organization (VO) of the Grid be downloaded from the Grid CA and installed on CE host. The required variables of *site-info.def* [21] which should be redefined are as follow: CE hardware information, CE operating system information, Grid site resources, etc. finally there are two useful commands for CE: 1) The *qstat* command which connects user to batch server and shows the queues outputs. 2) The *pbsnodes* which shows the available WNs information.

Figure 9. The Users authenticate through the VOMS server.

G. Deploying Virtual Organization Memberp System (VOMS)

VOMS server [27] is the main repository of user's authentication and it provides facilities like: ordering users in hierarchical groups, tracing their roles, etc. The VOMS functionality is similar to Kerberos KDC protocol which lets the nodes to authorize themselves to other nodes of the network in a safe manner. The VOMS Admin is a web based application for managing the VOMS database and Virtual Organization membership service (see figure 9).

VOMS admin provides a web user interface for administrative chores and a Simple Object Access Protocol (SOAP) interface for clients. SOAP transmits the structured information of the network web services. The VOMS admin overall operation is available via SOAP interface. The administrator package consists of a SOAP client command line which utilizes for automated batch jobs or web interface. The VOMS server backbone could be MySQL or ORACLE database. In our proposed gLite Grid architecture, we use MySQL database as the VOMS database infrastructure.

Before deploying VOMS server we need to install *XML* package. The *XML* is needed to setup the TOMCAT5. TOMCAT enables the VOMS host to acts as an independent server. Hence our gLite backbone is based on MySQL database; we should download and install glite-VOMS_mysql metapackage from gLite repository [27]. There are some mandatory variables of *site-info.def* file [21] which should be configured before the node can work properly: VOMS host name, VOMS database host name, VOMS port number for listening to VO requests, etc. The glite-voms file is another necessary file for configuring VOMS server. Finally we could configure VOMS server by using YAIM tool [17].

The VOMS admin management is done through multiple tools: 1) The *voms-admin* command is used for constructing database schema, accounts options and database manipulations. 2) The *voms-admin-configure* command is used to install, upgrade or remove the VOMS server database. 3) The *mysql* command, could setup query through VOMS database. 4) The *VOMS admin web interface* provides an interface for administering the VOs. Also VOMS admin could use the created log files for VOMS server troubleshooting.

H. User Interface

When all the pervious steps are accomplished, now it's time to let the users reach the Grid resources and submit their jobs. The gLite user interface [28] is a set of APIs and libraries which users and applications could use them to access the computational and storage services of the gLite Grid. Figure 10 draws the user interactions with Grid through UI.

The following packages are some dependencies which should be resolved before installing UI module: *Xercex-c, libtar, Phyton, SOAPpy, gcc compiler*. Also host certificate should be installed in UI node. As long as we are not a CA, the files are downloaded from [20]. After installing the UI from yum repositories, some of the *site-info.def* variables [21] should be changed, such as: site BDII host name, Logging and Bookkeeping (LB) host name, supported VO list, etc. finally the UI node configured using YAIM tool [17].

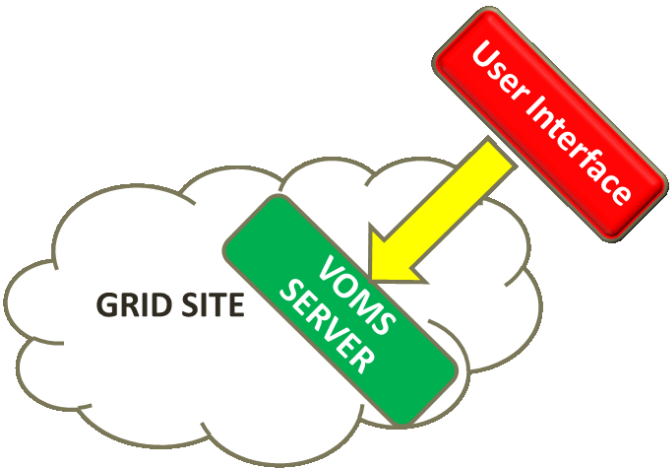

Figure 10. The User Interface: The main entry of Grid site

The following notifications may be beneficial to realize the previous steps of Grid deployment: First, the site BDII, batch system and CE are necessary to offer the computational resources to distant users. Second, the Grid site is integrated with Grid information system. Third, the Grid site is separated from UI.

There are two UI managers: the *lcg-infosites* and *lcg-info* tools. The *lcg-infosites* is a Perl script which runs some LDAP commands. This tool lets the users to exploit the Grid resource information. Also user could query the SEs and CEs via *lcg-info* tool. Figure 11 is a template script which shows how user establishes a proxy in UI and connects to CE.

```
adduser <your_name>
id <your_name>
scp -r <your_UI>:~/.globus/ <<+
root@<your_UI>:/home/<your_name>/
chown <your_name>: -R .globus/
voms-proxy-init --voms <vo_name>
glite-voms-proxy-info --all
glite-voms-proxy-destroy
```

Figure 11. Basic steps of configuring gLite User Interface (UI)

I. Job submission through CREAM Command Line (CLI)

Figure 12 is the process of job submission via CREAM CE. The description of each command is listed upon each command.

```
\\ USER LOGON
su - username
\\ WRITE THE JOB
> by the SL5 command line
\\ JOB SUBMISSION
qsub -q Q-name Job-name
\\ JOB STATUS
qstat job.sh.e <job-ID>
qstat -u username
\\ DELIVER JOB TO CLIENT BATCH SRV OF WN
qsub -I -Q Q_name
```

Figure 12. Job submission to CREAM

IV. Future Work

The authors are currently developing the Grid's theoretical concepts such as: scheduling algorithms [29] and resource discovery heuristics [30] by using Learning automata (LA) [31], [32]. Learning automaton is a stochastic learning machine operating on random environments. This Reinforcement Learning (RL) technique is most fitted to float and heterogeneous systems like Grid. By embedding learning automaton into Grid infrastructure, the system's performance will significantly improve.

Beside simulation of Grid behavior in some simulation tools such as: GridSim [5] and PalnetSim [33]. We want to extend our future works to implement on actual Grid middleware. So, we will develop and test our research in gLite Grid which is a realistic test bed for Grid computing problems.

V. Conclusion

The gLite middleware is a flexible middleware which could support and manage large amount of computing devices. Our idea is to build a massive computing facility through the campus scattered computer equipment. There is an agent in each node which reports the idle times of its host machine. The architecture is composed of three layers: The resource layer which is built from PCs, workstations and servers of the university. The middle layer is the Grid sub-sites which belong to each college. The final layer is the global Grid site which interconnects campus Grid and makes it easy for multi propose e-Science applications.

One challenge of the proposed model is the lack of the local CA to generate user's certificates. The next step of our work is to become CA and make the campus computing power available for students, indoor scientific researchers and other researchers from rest of the Iran's universities. Since our campus Grid configures through one site, we plan to extend it into multiple Grid sites and connect them as a universal Grid Computing center and establish a science portal for scientists.

Acknowledgment

The authors thank the Amirkabir University High Performance Computing Research Center (HPCRC) staff for providing valuable comments to improve Grid deployment procedure.

References

[1] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, "The physiology of the grid," in *Grid computing*, 2003, pp. 217–249.

[2] I. Foster, "The grid: A new infrastructure for 21st century science," in *Grid Computing*, 2003, pp. 51–63.

[3] I. Foster and C. Kesselman, *The grid: blueprint for a new computing infrastructure*. Morgan Kaufmann, 2004.

[4] B. Jacob, I. B. M. C. I. T. S. Organization, and S. B. O. (Firme), *Introduction to grid computing*. IBM, International Technical Support Organization, 2005.

[5] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and Computation: Practice and Experience*, vol. 14, no. 13-15, pp. 1175–1220, 2002.

[6] F. Magoulès, J. Pan, K. A. Tan, and A. Kumar, *Introduction to grid computing*, vol. 6. CRC, 2009.

[7] gLite, "Lightweight middleware for Grid Computing." [Online]. Available: http://glite.cern.ch/.

[8] M. Marzolla et al., "Open standards-based interoperability of job submission and management interfaces across the grid middleware platforms glite and unicore," in *e-Science and Grid Computing, IEEE International Conference on*, 2007, pp. 592–601.

[9] F. Gagliardi, B. Jones, F. Grey, M. E. Bégin, and M. Heikkurinen, "Building an infrastructure for scientific Grid computing: status and goals of the EGEE project," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 363, no. 1833, pp. 1729–1742, 2005.

[10] EGEE, "Enabling Grids for E-Science in Europe." [Online]. Available: http://www.eu-egee.org/.

[11] K. Bosa, W. Schreiner, M. Buchberger, and T. Kaltofen, "A Grid Software for Virtual Eye Surgery Based on Globus 4 and gLite," in *Parallel and Distributed Computing, 2007. ISPDC'07. Sixth International Symposium on*, 2007, pp. 23–23.

[12] I. Taylor, M. Shields, and I. Wang, "Distributed p2p computing within triana: A galaxy visualization test case," *International conference on Parallel and Distributed Processing Symposium*, 2003.

[13] S. D. Olabarriaga, T. Glatard, and P. T. de Boer, "A virtual laboratory for medical image analysis," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 14, no. 4, pp. 979–985, 2010.

[14] E. Laure et al., "Programming the Grid with gLite," *Computational Methods in Science and Technology*, vol. 12, no. 1, pp. 33–45, 2006.

[15] GenericInstallGuide320, "Generic Installation and Configuration Guide for gLite 3.2." [Online]. Available: https://twiki.cern.ch/twiki/bin/view/LCG/GenericInstallGuide320.

[16] SL, "Scientific Linux." [Online]. Available: https://www.scientificlinux.org/.

[17] YaimGuide400, "YAIM 4 guide for sysadmins." [Online]. Available: https://twiki.cern.ch/twiki/bin/view/LCG/YaimGuide400.

[18] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, "Grid services for distributed system integration," *Computer*, vol. 35, , pp. 37–46, 2002.

[19] D. Abramson, R. Buyya, and J. Giddy, "A computational economy for grid computing and its implementation in the Nimrod-G resource broker," *Future Generation Computer Systems*, vol. 18, no. 8, pp. 1061–1074, 2002.

[20] IR-GRID CA, "IRAN-GRID Certificate Authority." [Online]. Available: http://cagrid.ipm.ac.ir/.

[21] site-info, "site-info.def configuration variables." [Online]. Available: https://twiki.cern.ch/twiki/bin/view/LCG/Site-info_configuration_variables.

[22] glite-BDII_site, "Berkeley Database Information Index (BDII)." [Online]. Available: http://glite.cern.ch/glite-BDII_site/.

[23] glite-APEL, "APEL accounting software." [Online]. Available: http://glite.cern.ch/glite-APEL/.

[24] Batch sys., "Batch system server." [Online]. Available: http://glite.cern.ch/glite-TORQUE_client/.

[25] glite-WN, "Worker Node." [Online]. Available: http://glite.cern.ch/glite-WN/.

[26] glite-CREAM, "CREAM Computing Element (CE)." [Online]. Available: http://glite.cern.ch/glite-CREAM/.

[27] glite-VOMS_mysql, "gLite VOMS server." [Online]. Available: http://glite.web.cern.ch/glite/packages/R3.2/sl5_x86_64/deployment/glite-VOMS_mysql/glite-VOMS_mysql.asp.

[28] glite-UI, "gLite user Interface." [Online]. Available: http://glite.cern.ch/glite-UI/.

[29] S. Ghanbari and M. Meybodi, "Learning automata based algorithms for mapping of a class of independent tasks over highly heterogeneous grids," *Advances in Grid Computing-EGC 2005*, pp. 269–274, 2005.

[30] A. Sarhadi and M. R. Meybodi, "New algorithm for resource selection in economic grid with the aim of cost optimization using learning automata," *International Conference on Challenges in Environmental Science and Computer Engineering*, pp. 32–35,2010.

[31] K. S. Narendra and M. A. L. Thathachar, *Learning automata: an introduction*. Prentice-Hall, Inc., 1989.

[32] H. Beigy and M. Meybodi, "A mathematical framework for cellular learning automata," *Advances in Complex Systems*, vol. 7, no. 3-4, pp. 295–320, 2004.

[33] P. García, C. Pairot, R. Mondéjar, J. Pujol, H. Tejedor, and R. Rallo, "Planetsim: A new overlay network simulation framework," *Software Engineering and Middleware*, pp. 123–136, 2005.