EDAGRID@HCMUT: A CAMPUS GRID INFRASTRUCTURE FOR SERVICE-CENTRIC GRID APPLICATIONS

Nguyen Tuan Anh, Nguyen Cao Dat, Nguyen Quang Hung,Tran Ngoc Minh, Thoai Nam, Dang Tuan Nghia, Nguyen Thanh Son
University of Technology, VNU-HCM

1.INTRODUCTION

Grid computing has been a dominant theme of distributed computing research during the past few years. Many efforts are dedicated to designing and developing middleware that efficiently manage the complexity of the Grid environments such as Grid security [**Error! Reference source not found.**, **Error! Reference source not found.**], Grid information, data movement [**Error! Reference source not found.**, **Error! Reference source not found.**], service inter-operability [**Error! Reference source not found.**], etc. Among them, Globus Toolkit has been considered as one of the most important Grid pilot projects that provides basic Grid functionalities and architectures for integrating new user defined services.

Nevertheless, exploiting Grid power for HPC applications is still a big challenge. Most Grid middleware provides execution environments based on the resource-centric approach in which users have to manually select machines for their applications. Cross-site parallel executions of a single application, especially communication-intensive applications, need to be tuned carefully and manually. The service-centric approach has been proposed as a replacement of the traditional resource-centric approach to be more efficient in extracting Grid power for parallel applications. Instead of selecting resources for applications, users request "computing" services with given QoS requirements regardless the resource locations. This greatly facilitates users in dealing with the complexity of Grid environments. But such a paradigm shift also requires supports from both the Grid middleware and the programming environment.

EDAGrid project aims at building the campus Grid middleware which provides location-transparent computing/executing services with high level resource requirements. Based on GT4 as the local resource Grid middleware, EDAGrid develops various high level services such as resource discovery, resource reservation and information warehouse for service-centric applications. Users only need to describe capacities of resources they want for their applications and EDAGrid will automatically select resources that meet the user's need. Several applications on chip design optimizations and data mining will be built and deployed on the Grid middleware developed by the EDAGrid project.

EDAGrid is considered as one of the first Grid middleware projects in Vietnam. This is the preparation step for a national Grid infrastructure coined by our VN-Grid initiatives which aim at connecting campus Grids of universities and institutions to foster collaboration and resource sharing.

The paper will first present the VN-Grid initiatives in section 2. EDAGrid middleware, playing a very important role in the VN-Grid initiatives, will be discussed in section 3. Some conclusions and future directions are presented in section 4.

2.VN-GRID INITIATIVES

At presence, Grid computing is still a very new topic in Vietnam. The VN-Grid initiatives, which has been coined by HCMUT and several other universities, is one of the very first efforts of the Grid communities in Vietnam to speed up collaborations on Grid computing research to build the national Grid infrastructure. In Vietnam, the IT infrastructure, especially the telecommunication and the Internet bandwidth, is not so developed. So, in order to form a national Grid infrastructure, VN-Grid first suggests each institution to build their campus Grid as a "fat" Grid node. VN-Grid then connects these fat Grid nodes together. Technologies such as P2P can be used to guarantee the scalability and the fault tolerance of those fat Grid nodes.

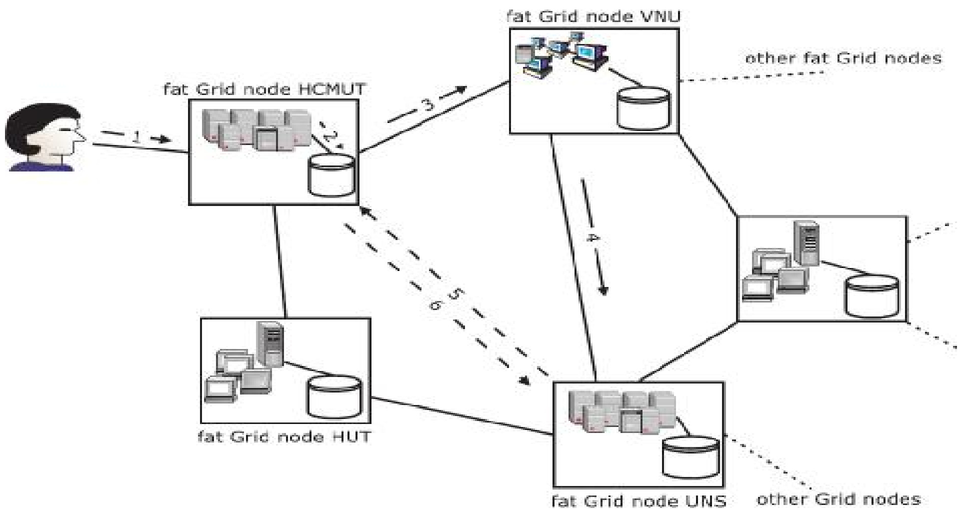


Figure 1: VN-Grid: a usage scenario

VN-Grid will provide an environment for sharing computing resources among institutions across Vietnam. The middleware developed by VN-Grid must greatly facilitate the exploitation of computing services for end-users. Users only need to describe their high level resource requirements and the middleware helps to execute user's parallel applications transparently. Figure 1 shows a usage scenario of the VN-Grid:

User requests a computing service to his institution by providing the QoS of his required service.

Upon receive a computing request the institution will look for resources locally. If a resource with the required service QoS is found, reservation is then made on that resource.

If no resource with the given QoS found locally in step 2, the request will be forwarded to the "neighbor" institutions to perform resource discovery. In the case a request consists of multiple resources, local qualified resources will be reserved and the rest of the request will be considered as a new request for forwarding.

Forwarding will be continued until a suitable resource is identified.

The discovery results will be returned to the original institution who initiated the request.

The original institution makes reservation on remote resources, then executes computing services on those resources and finally monitors the executions. Results will be automatically collected back to the user.

3.EDAGRID: A CAMPUS GRID DESIGN

VN-GRid initiatives give us a vision toward the national computing Grid infrastructure for Vietnam. The evolution of such an infrastructure should be through several stages in which the first stage should be building local Grid infrastructures or campus Grids. EDAGrid is one of such projects.

3.1.Overview

EDAGrid fits exactly in the VN-Grid initiatives but at a smaller scale. EDAGrid allows users to express their high level resource requirements such as the CPU performance, the memory or network bandwidth. Applications will be executed transparently and automatically in the authorized resources that satisfy user's requirements.

EDAGrid is to develop Grid middleware on top of Globus Toolkit 4.x for sharing resources within the university campus. As the network infrastructure within a university campus is rather fast and stable, EDAGrid uses the centralized model to manage resources. Each campus will have one or more servers on which we deploy campus-level services: resource discovery, resource information. These servers manage grid nodes-the computing elements of the EDAGrid environment. Each grid node is a workstation or a homogeneous cluster on which we deploy local services: reservation, information collection and Globus's WSGRAM (GT4).

3.2.Job execution

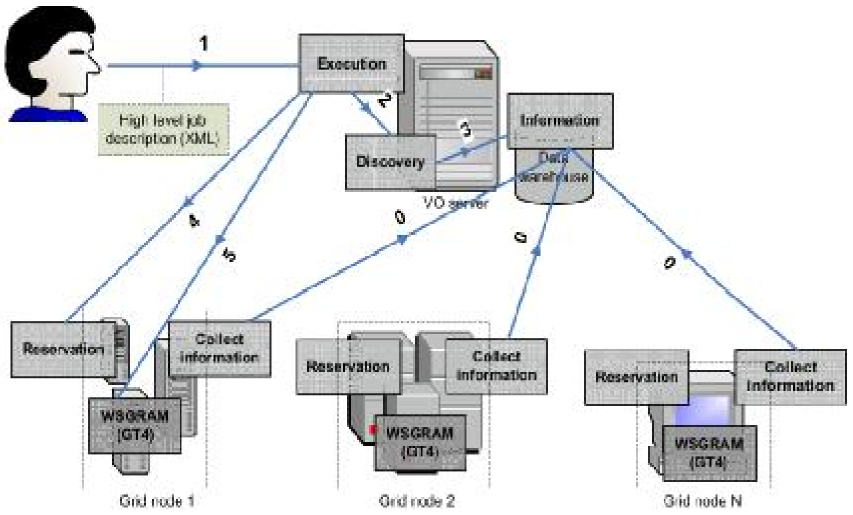


Figure 2. EDAGrid middleware components

The job execution of EDAGrid is shown in Fig. 2. Users will first describe their job requirements (section 3.3) in an XML format and submit this description to the Execution component (1). The Execution will first perform resource discovery by accessing the WSRF Discovery service (section 3.4) (2). In turn, Discovery needs to query qualified resources from the Information service (section 3.5) (3). List of resources will be matched against the user's requirements described in the job description, ranked based on the levels of satisfaction. Upon receiving available resources, the Execution makes reservation on the qualified grid node(s) through the Reservation service (section 3.6) (4) and finally launches the job with the given reservation on the corresponding Globus's WSGRAM service (5).

Each grid node runs a monitor which will periodically collect information of resources on its grid node and publish them to the information service on the campus server (0).

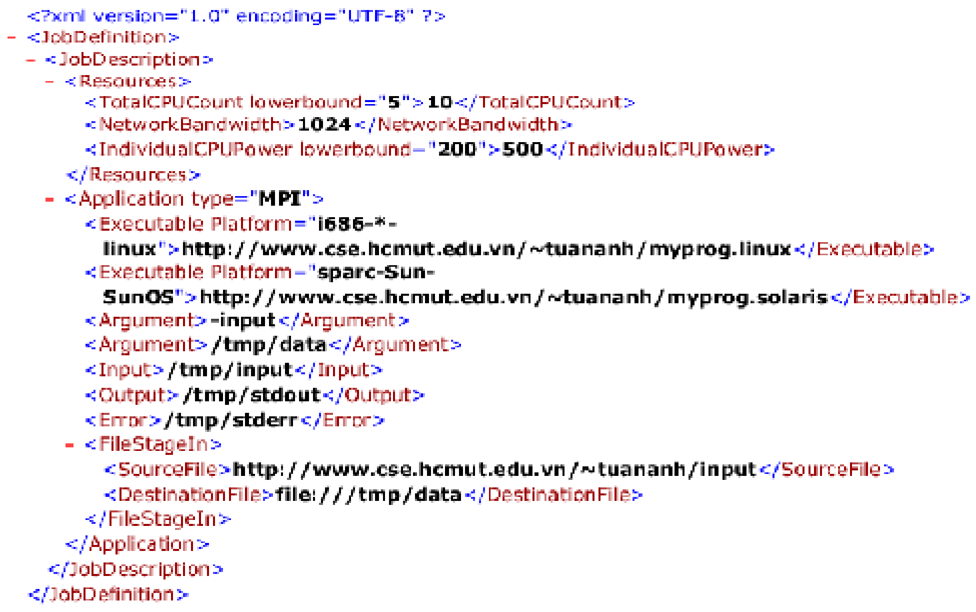
3.3.EDAGrid job description

The success of a Grid environment depends very much on how easy users can achieve the performance for their applications. EDAGrid approaches this by providing high-level services for requirement-driven and location-transparent parallel executions. EDAGrid's job description extends the Job Submission Description Language (JSDL) [**Error! Reference source not found.**] for additional resource types such as MFlops of each processor (Linpack benchmark), the network bandwidth among processes, number of CPUs of an SMP node, etc.

Users need to describe two types of information:

- Resource specifications: list all requirements of resources for the application. Resources can be for a single process such as CPU speed (MHz), CPU performance (MFlops), average CPU load, physical memory size, number of CPU/1 SMP node, software environments on each node, etc.; or it can be for inter-processes such as total number of CPUs allocated to the application, network bandwidth among processes, etc.

- Application specifications: define application arguments, inputs, outputs or file staging before/after running the application as well as platforms on which the application executables are supported.



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <JobDefinition>
  - <JobDescription>
    - <Resources>
        <TotalCPUCount lowerbound="5">10</TotalCPUCount>
        <NetworkBandwidth>1024</NetworkBandwidth>
        <IndividualCPUPower lowerbound="200">500</IndividualCPUPower>
      </Resources>
    - <Application type="MPI">
        <Executable Platform="i686-*-
          linux">http://www.cse.hcmut.edu.vn/~tuananh/myprog.linux</Executable>
        <Executable Platform="sparc-Sun-
          SunOS">http://www.cse.hcmut.edu.vn/~tuananh/myprog.solaris</Executable>
        <Argument>-input</Argument>
        <Argument>/tmp/data</Argument>
        <Input>/tmp/input</Input>
        <Output>/tmp/stdout</Output>
        <Error>/tmp/stderr</Error>
      - <FileStageIn>
          <SourceFile>http://www.cse.hcmut.edu.vn/~tuananh/input</SourceFile>
          <DestinationFile>file:///tmp/data</DestinationFile>
        </FileStageIn>
      </Application>
    </JobDescription>
  </JobDefinition>
```

Figure 3. Example of an EDAGrid job description

Figure 3 shows an example of an EDAGrid job description. Resource requirements are specified in the `Resources` block. The user requires 5-10 CPUs (`TotalCPUCount`) (10 is desirable, 5 is minimum acceptable); network bandwidth of at least 1024Mbit/s (`NetworkBandwidth`); Linpack benchmark value for each CPU is at least 200 Mflops although at least 500 Mflops CPU is desirable.

The second block of Fig. 3 tagged by `Application` describes the user's application specifications: it is an MPI application of which the executable is on the web and is available for two platforms: Linux-i686 ("*" mean any vendor) and Sun Sparc/Solaris (SunOS) (tag: `Executable`). The application takes two arguments: "-input /tmp/data" ("data" will be read by the application) (tag: `Argument`), the corresponding standard input, output and error consoles are taken from files "`input`", "`stdout`" and "`stderr`" in `/tmp` directory (tags: `Input`, `Output`, `Error`). Before executing the application, the system needs to move data stored on the web server at `http://www.cse.hcmut.edu.vn/~tuananh/input` to `/tmp/data` of the remote machine where the application will run (tag: `FileStageIn`).

EDAGrid provides two APIs for executing an application with job descriptions: a Java class API and a command line. The locations of application executions are totally transparent to users.

3.4.Resource discovery

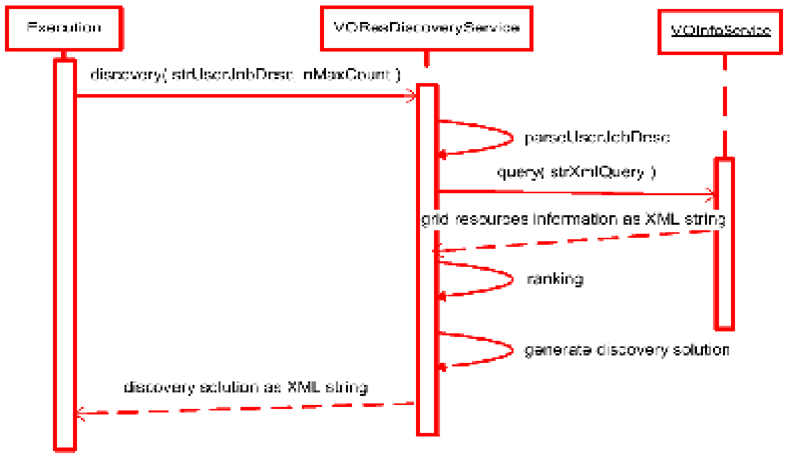

Execution
VOResDiscoveryService
VOInfoService
discovery(strUserJobDesc, nMaxCount)
parseUserJobDesc
query(strXmlQuery)
grid resources information as XML string
ranking
generate discovery solution
discovery solution as XML string

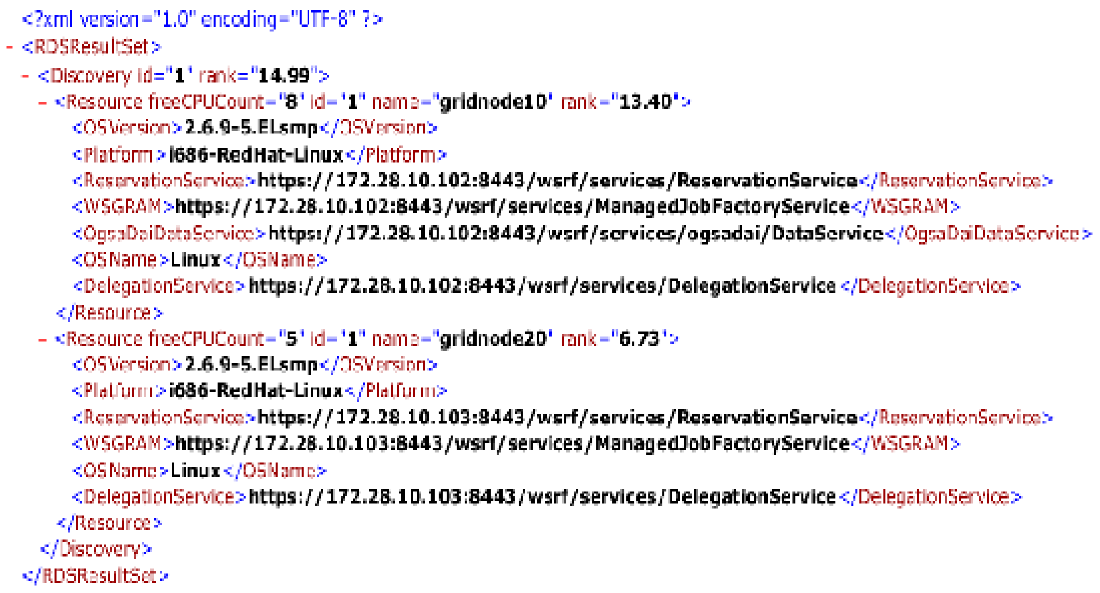
Figure 4. Discover resources in EDAGrid

The resource discovery service discovers resources that satisfy the user's job description, rank found resources based on level of satisfaction (availability of resources over user's requirements). The list of qualified resources is returned in an XML format. The EDAGrid's resource discovery service (RDS) is illustrated in Fig. 4 as follows:

- Step 1: The RDS receives a job description from the Execution.
- Step 2: The RDS transforms the job description into series of query requests.
- Step 3: The query requests will be sent to EDAGrid's Information service VOInfoService (section 3.5) to find appropriate resources.
- Step 4: Resources will be overall evaluated, synthesized and ranked against the user's job description.
- Step 5: Finally, a list of discovery solutions is given back to the Execution.

RResource discovery is a VO-level Grid service (WSRF) written in Java and deployed into the Globus's Java service container. We provide a Java API "`String discovery(String userJobDesc, int nMaxCount)`" to facilitate the use of RDS. The `discovery` API inputs two parameters: `userJobDesc` as the user's job description (XML string), and `nMaxCount` as the maximum discovery solutions which the client wants to receive from the RDS. List of qualified resources will be returned also as an XML string.

Figure 5 shows a resource discovery result. Each solution block is tagged by `Discovery` which contains a list of resources separated by the `Resource` tag.



```
<?xml version="1.0" encoding="UTF-8" ?>
- <RDSResultSet>
  - <Discovery id="1" rank="14.99">
    - <Resource freeCPUCount="8" id="1" name="gridnode10" rank="13.40">
        <OSVersion>2.6.9-5.ELsmp</OSVersion>
        <Platform>i686-RedHat-Linux</Platform>
        <ReservationService>https://172.28.10.102:8443/wsrf/services/ReservationService</ReservationService>
        <WSGRAM>https://172.28.10.102:8443/wsrf/services/ManagedJobFactoryService</WSGRAM>
        <OgsaDaiDataService>https://172.28.10.102:8443/wsrf/services/ogsadai/DataService</OgsaDaiDataService>
        <OSName>Linux</OSName>
        <DelegationService>https://172.28.10.102:8443/wsrf/services/DelegationService</DelegationService>
      </Resource>
    - <Resource freeCPUCount="5" id="1" name="gridnode20" rank="6.73">
        <OSVersion>2.6.9-5.ELsmp</OSVersion>
        <Platform>i686-RedHat-Linux</Platform>
        <ReservationService>https://172.28.10.103:8443/wsrf/services/ReservationService</ReservationService>
        <WSGRAM>https://172.28.10.103:8443/wsrf/services/ManagedJobFactoryService</WSGRAM>
        <OSName>Linux</OSName>
        <DelegationService>https://172.28.10.103:8443/wsrf/services/DelegationService</DelegationService>
      </Resource>
    </Discovery>
  </RDSResultSet>
```

Figure 5: Example of a resource discovery result

3.5.Information service

We decided to build our own information service instead of using the Globus MDS because of the performance issue. The EDAGrid information architecture is illustrated in Fig. 6, consisting of four elements:

The data warehouse to store all information about the compute resources of the campus Grid.

The Information service provide query and subscription functionalities.

The information client such as the Execution component which query information.

The information providers (compute grid nodes) and the reservation service subscribe their information to the information service.

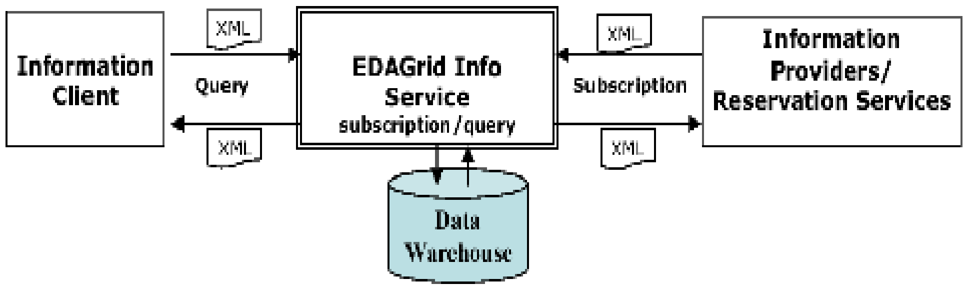


Figure 6.EDAGrid information service

3.5.1.Subscription/query service

The central component of the Information service is subscription/query service. It is WSRF-based service which collects monitoring information and makes this information available. We have implemented the subscription/query service in Globus Tool 4.0.

Monitoring information is a XML-formatted document which contains grid node (cluster/workstation) data such as host name, node count, individual node information (processor information, memory size, OS name and version, file system data, processor load data, external bandwidth, internal bandwidth), job queue and job description, etc. The subscription/query service receives information from reservation services (update/subscribe a reservation) and grid nodes.

Grid information is stored in the data warehouse of MySQL server.

3.5.2.Grid node information provider

To guarantee the functionality of EDAGrid requirement-driven job executions, all information about resources that users require within their job descriptions (section 3.3) must be available and up-to-date. An EDAGrid information provider is an external program running on each grid node (cluster or workstation) which is responsible for monitoring and collecting such information via some scripts. Dynamic information such as CPU loads, number of jobs on each grid node, etc. can be periodically published to the information service or it can be updated upon status changes. In addition to the internal node information, EDAGrid also collects network bandwidth information which will allow the resource discovery service to allocate more suitable resources for parallel applications.

We have implemented a simple information provider on Linux and Solaris operating systems.

3.6.Resource reservation

The resource reservation service runs on each grid node which plays a very important role in guaranteeing the availability of resources for cross-grid node resource co-allocations. This service is the Grid portal to the local job management system such as PBS [**Error! Reference source not found.**] or Supernode II [**Error! Reference source not found.**].

The reservation service is used by the EDAGrid Execution component right after the list of qualified resources has been discovered. The process of creating a reservation is illustrated in Fig. 7 as follows:

- The client (EDAGrid's Execution) sends a request together with the job description specifying which resources they want to reserve for the job to the Reservation service.
- The Reservation service will collect information from the VOInformation service to decide whether there are free resources to allocate to the job. If client request is satisfied, the Reservation service will create a reservation identifier and update the new information to the VOInformation service.
- The Reservation service makes a reservation on the local system.
- The Reservation service returns the reservation identifier to the client if success or an error message if failure.

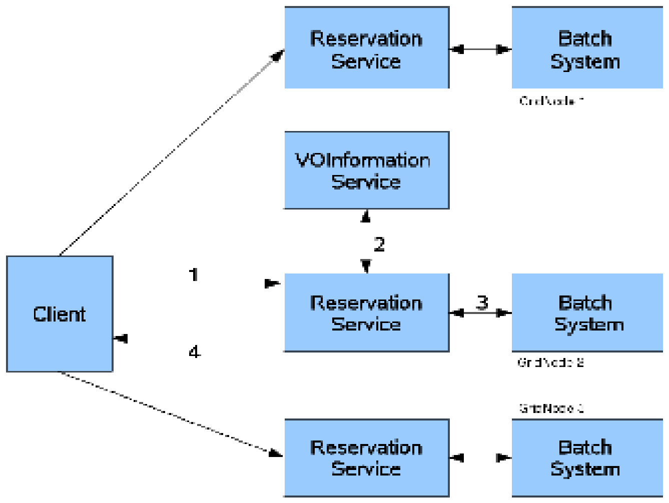


Figure 7. Process of creating a reservation

At present, our reservation service only supports processor reservation to PBS and SupernodeII clusters (developed by HCMUT).

4.CONCLUSION

EDAGrid focuses on providing a high-level resource requirement-driven execution environment for campus Grids that frees users from all complexities and issues of the Grid. Users only have to describe characteristics of resources such as the memory, the processing power and the network bandwidth they need from the execution environment and thank to EDAGrid services, their applications will be launched on appropriate resources transparently and automatically.

The EDAGrid's job description has been introduced to allow users to describe their jobs and resource requirements. EDAGrid is based on four principal elements: the Execution component, the resource discovery service, the information service and the reservation service. The Execution component is the interface between users and the Grid environment for executing jobs defined in the job description. The resource discovery service has been presented to discover resources satisfying the user's job description. The information service which collects information about resources and server other services has been discussed. The reservation service has been built to guarantee the availability of resources for applications. All EDAGrid services are WSRF-compliance, implemented using Java and deployed into the GT4 Java service container.The first prototype of the EDAGrid middleware is available for testing. We will build a EDAGrid testbed with Grid applications in the next few months.

EDAGrid is one of the very few Grid middleware projects in Vietnam at the moment. We consider this project as one important preparation step toward the national Grid infrastructure of Vietnam which has been coined through our VN-Grid initiatives.

REFERENCES

[1]. Jiageng-Li, D.C, *A scalable authorization approach for the Globus grid system.* Future Generation Computer Systems 21 (2), 191—301, (2005).

[2]. Welch, V., Siebenlist, F., Foster, I., Bresnahan, J., Czajkowski, K., Gawor, J., Kesselman, C., Meder, S., Pearlman, L., Tuecke, S, *Security for grid services. In Press, I., ed.*, Twelfth International Symposium on High Performance Distributed Computing (HPDC-12) (2003)

[3]. Allcock, B., Bester, J., Bresnahan, J., Chervenak, A., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D., Tuecke, S, *Secure, efficient data transport and replica management for high-performance data-intensive computing*. In: IEEE Mass Storage Conference. (2001)

[4]. Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., Tuecke, S, *The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets.* Journal of Network and Computer Applications (23), 187—200, (2001).

[5]. Foster, I., Kesselman, C., Nick, J., Tuecke, S, *Grid services for distributed system integration.* Computer 35 (6), (2002)

[6]. GGF: *Job Submission Description Language Specification.* http://www.gridforum.org/documents/GFD.56.pdf.

[7]. Bayucan, A., Lesiak, C., Mann, B., Henderson, R.L., Proett, T., Tweten, D, *OpenPBS: External Reference Specification. Release 1.1.12 ed*n, http://www-unix.mcs.anl.gov/openpbs/, (August 1998).

[8]. Nam, T., Toan, T.D., Tuong, T.V.N, *Resource management and scheduling on supernodeII*, In: Proceedings of School on Computational Sciences and Engineering: Theory and Applications, Ho Chi Minh city (March 2005)