

Cluster Abstraction: towards Uniform Resource Description and Access in Multicluster Grid

Maoyuan Xie, Zhifeng Yun, Zhou Lei, Gabrielle Allen

Center for Computation & Technology, Louisiana State University, Louisiana, USA
{mxie, zyun, zlei, gallen}@cct.lsu.edu

Abstract

Multicluster Grid has emerged as a major approach for people to pursue substantial computational capability improvement to support large-scale compute-intensive applications. However, it is challenging on how to efficiently manage application execution across the participating clusters, since the participating clusters are geographically distributed, heterogeneous, and self-administrative, and the network connection provided by the Internet is vulnerable in security and unpredictable in performance. The challenges include how to discover resource information, how to handle application workflow, how to allocate appropriate resources, how to make application execution reliable to endure system failures, etc. This paper proposes cluster abstraction to uniform describe and access resources in multicluster Grid. After investigating various cluster resource management systems, a reference description of cluster abstraction is presented to cover cluster computing characteristics, hiding individual cluster details for use. Following this concept, a deployment service is discussed. This deployment service is one of the essential components in ongoing resource management development for multicluster Grids.

1. Introduction

Cluster computing has entered the picture of high performance computing (HPC) for a long time. However, it is a well-known fact that the processing capability of any single cluster can no longer satisfy the growing compute demands of scientific and engineering applications. Thus, Grid computing [9] emerged to coordinate existing computing resources to meet such increasing computing demands. The Grid

consisting of multiple clusters, *i.e.*, multicluster Grid, is used to provide reliable HPC performance with cost efficiency for large-scale compute-intensive and/or data-intensive applications. In a multicluster Grid environment, each participating resource has the same architecture, *i.e.*, cluster. We call this situation as *architecture homogeneity*. Each participating cluster offers production-quality of services (QoS) with large computing capability. There exist a lot of compute intensive applications supported by cluster computing. Research on multicluster Grid is the natural extension of conventional cluster computing efforts.

The multicluster architecture is shown in Figure 1. The super-scheduler assigns tasks across the multicluster Grid to participating clusters. These tasks are considered as normal jobs to local scheduler systems. A local scheduler system allocates computing resources to assigned jobs according to its own policies. Most Grid researches focus on super-scheduler level, dealing with optimization of scheduling tasks across the Grid. Since Grid computing promises the collaboration of computing resources worldwide, application execution needs to respect local scheduling policies. After tasks (or jobs in the local resource management circumstance) are submitted to local clusters, local schedulers take over these tasks, placing them into local queues and waiting for resource allocation.

It is of significance to provide cluster abstraction and integration on multicluster environment, which greatly improves resource utilization with easy-to-use access. Cluster abstraction allows a user to view any underlying cluster as a generic computing unit for use, no matter how diverse underlying clusters in a multicluster environment are. On top of cluster abstraction, integrated services can be easily generated. Furthermore, a formal application programming interface (API) for multicluster can be achieved. By the API, users can easily take advantage of large-scale capability of multicluster Grid for their applications with high flexibility and portability.

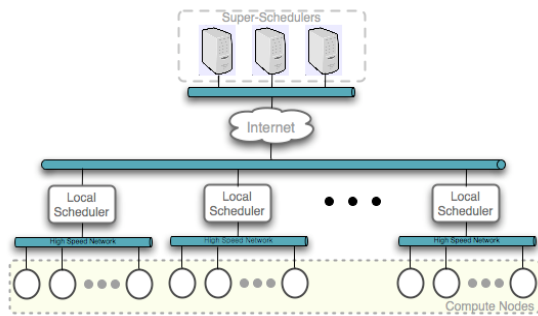


Figure 1. The common multicluster architecture

To abstract cluster efficiently, we need to answer following questions before bringing this concept into the picture. What is the underlying infrastructure? What information do we need? What technology can be used to realize the idea? How to access those information? How to operate on these virtual clusters? We describe our efforts to answer these questions in this paper. The remainder of this paper is organized as follows. The motivation is provided in Section 2. Section 3 illustrates cluster representation. High level interface is introduced in Section 4. The deployment service is demonstrated in Section 5. Section 6 concludes the paper.

2. Motivation

Abstraction and virtualization have become an important tool in computer system design for a long time. Despite the incredible complexity of computer systems, this technology has leveraged computer systems by virtualizing the separation levels of the hierarchies with well-defined interfaces. The simplifying abstractions hide lower-level implementation details from higher-level applications, therefore reducing the complexity of the development process. One good example is disk storage. Operating system abstracts hard disk physical addressing details, *i.e.*, sectors and tracks, so that to application software, hard disk appears as a set of files of variable sizes. Higher-level applications can operate on those files without any knowledge about the actual physical organization of the hard disk.

After its success in subsystems such as disks, the concept of abstraction is applied to the entire machines, introducing virtual machine [11]. Originally developed for large centralized computer systems, the concept of virtual machine consists of three aspects, CPU virtualization, memory virtualization and I/O virtualization. Virtual machines provide virtualizations of physical host machines, upon which a virtual machine monitor (hypervisor) is running. The virtual machine monitor is responsible for capturing and emulating instructions issued by the guest machines, and providing interfaces for operations on the VMs. Typically, clients can utilize virtual machines to create an execution environment with certain hardware or software configurations, and deploy it on any resource running hyper-visor. Based on virtual machines, the virtual workspace [12] uses those VM images reflecting a workspace's software requirements to represent an execution environment. But the virtual workspace

concept covers wider territory than the workspace consisting of virtual machines. Virtual workspace includes site-configured workspace, which is installed to support specific needs of specific communities such as TeraGrid [2]. Proposed approach for providing site-configured workspace is to obtain a priori agreement on specific configurations and propagate them, and provide access. Another component is virtual cluster workspaces, which can be constructed using the Cluster-on-Demand (COD) infrastructure [3]. Here, the concept of virtual cluster is totally different from the one used in this paper. The virtual cluster here is an on-demand integration of multiple VM images which may represent specialized nodes, i.e. compute or head nodes. The nodes of this virtual cluster have the same or similar configuration, but they are not physically belonging to the cluster. However, the virtual cluster used in this paper is some representation of physical clusters, similar to the VM images to the computational machines.

From the discussion above, it is easy to discover that none of the concepts focus on pure multicluster environment. In this way, all of them need to cover some unnecessary information and interfaces which are aiming at much more heterogeneous execution environment than pure multicluster environment. Since each cluster in multicluster environment is considerably uniform in architecture and powerful despite of the diversity of node configuration, we placed more focuses and efforts to abstract clusters which are the fundamental components of multicluster execution environment.

Cluster abstraction is a concept of describing clusters in the uniformed virtual cluster format and providing uniformed interfaces for accessing those resources. Figure 2 is a vivid interpretation of the

concept. Through uniformed high level interfaces, users and upper-level applications cannot see and operate further than virtual clusters in the virtual cluster pool (VCP). Under the virtual cluster pool is the complex and heterogeneous multicluster environment. In terms of various configurations.

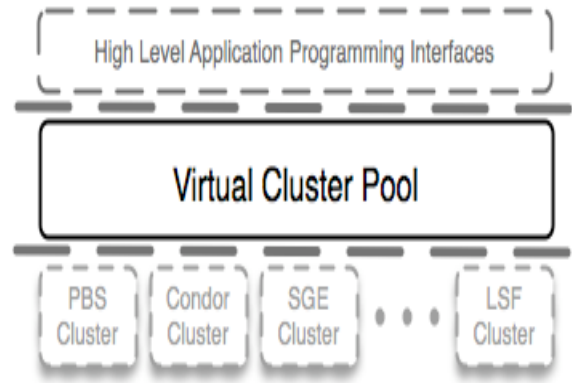


Figure 2. This figure shows the basic concept of the cluster abstraction. The virtual cluster pool here hides the details for the heterogeneous multicluster environment from high level services and users.

The understanding of the system and demands is crucial to efficiently abstract information. Despite the similar architecture, most configurations of clusters are different in some way, such as local resource management systems. Moreover, their status are different, i.e. workload, availability of compute nodes. Therefore, we need to know what information is of interest. Then, we need to know what technology to be used to abstract the information, and what interfaces should be provided.

3. Cluster description

3.1. Local resource management systems

Each cluster employs a resource management system, called Local resource management system (LRMS) in the Grid

circumstances. The LRMS's place the jobs in one or more queues with certain resource requirements, and make decision on the place and time for the job execution. Some of them offer check-pointing service, which assists the user to resume the job execution from certain point after job or machine failure occurs during the execution process. Widely adopted resource management systems include PBS, SGE, Condor, etc.

PBS, the Portable Batch System [17], is a batch job and computer system resource management software package. It accepts batch jobs, shell scripts and control attributes as input. Jobs are preserved, protected, and then executed, after which the output, the exit status and possible error messages will be delivered to the submitter. Four components are responsible for PBS, including commands, the job server, the job executor, and the job scheduler. PBS commands provide a platform for users and administrators to submit, monitor, modify, and delete jobs. PBS job server is essential to PBS, because basic batch services such as receiving, creating and modifying a batch job, protecting the job against potential system crashes, and executing the job, are fundamental functions provided by the job server. PBS job executor is the daemon which receives the job from the job server and actually places the job into execution. This executor is also responsible for the output return of user's jobs in response to the output return request from the job server. PBS job scheduler is a daemon which can be created by each individual local resource and communicates between each other and also with the job server and the job executors for job availability and system resource status. The scheduler basically contains information about the usage policy for each site and controls which job to be run, when and where to run the job. All these four components working together as a

whole system ensure the stability and reliability of PBS.

The Sun Grid Engine, SGE [18], allows users to submit computational tasks to the SGE system with transparent workload distribution. SGE accepts batch jobs, interactive jobs, and parallel jobs. Check-pointing mechanism is used in SGE with free user interference in check-pointing migration. Firstly, jobs are submitted from the *submit host* to the queue hosted by each *execution host*, where job attributes are reviewed and such decisions as whether the job may be migrated are made. The queues and the jobs are controlled by the *master host* which has the *master daemon* and the *Scheduler daemon* running. The *execution hosts* with the *Execution daemon* are responsible for the execution of jobs and the update of job status and workload to *Master daemon*. The *Communication daemon* is used for all communication among SGE components. Throughout the lifetime of the jobs, users can delete jobs and check job status; and the administrators can manage the queues and jobs.

The primary purpose of Condor [20] is to maximize the usage of workstations with less interference. The system also utilizes the check-pointing mechanism to guarantee execution of jobs. Since Condor is a batch system, it also provides queuing mechanism, scheduling policy, priority scheme, and resource classifications. Jobs are submitted to job queue, which is referred as a "Condor pool", and executed on distributed computational resources. Results and logs of the job execution are returned to the local submit machine. However, unlike traditional batch systems which can only operate with dedicated machines, Condor can also schedule jobs on non-dedicated machines.

Based upon these heterogeneous local batch systems and other resource management services, the concept of cluster

abstraction uses the information provided by them to generate virtual cluster images.

3.2. Static and dynamic information

As defined previously, multicluster environment consists of a group of clusters which agree the Grid policy and have their own local resource management tool. The configuration of clusters vary in several ways, such as number of computational nodes, number of CPUs for one node, the cluster operating system, available memory, and network connections. Due to many unpredictable reasons, the status of clusters may change in several aspects, *e.g.*, available nodes. Since all clusters need local resource management systems to manage computational tasks in them, jobs need to be scheduled according to LRMS scheduling policy. Besides static configuration information, two important dynamic information parameters are introduced to measure the performance of cluster: the number of running jobs and number of queuing jobs. Here, we dirty our hands by digging into most popular LRMSs for information they provided regarding to cluster performance.

PBS has interfaces to review the static configurations as well as dynamic status. Using these interfaces, information such as number of nodes, operating system, number of CPUs per node, size of memory, and scheduling policy which are referred as static configurations, can be easily gathered. Moreover, other information like status of each node, assigned memory, assigned CPUs, number of jobs in the queue, number of jobs running, number of jobs queuing, are available. These information are dynamic status information of the cluster.

The information which can be queried from Condor include operating system, hardware architecture, memory, state of the node (Claimed or Unclaimed), node activity (Idle, Owner or Busy), average load, activity time of the node, number of jobs running, number of jobs queuing, command for the jobs. The information can be grouped into two categories, first three as static configurations, and the rest of them as dynamic status.

SGE defines values of each node to include system architecture, number of processors, total memory, total swap space, total virtual memory, used memory, used swap, number of executing jobs, number of queuing and waiting jobs, etc.

For the convenience of higher-level application, the type of LRMS used in each cluster home directory of the user is considered as static information. When adding a new cluster into the pool, since the system has no pre-knowledge for this resource, it has to query blindly for the LRMS unless the user specify one. After the initial query, with the knowledge, access and operation to the virtual cluster will be much easier. Moreover, due to node unexpected crash, the number of available nodes, thus available CPUs, is changing, therefore can be classified as dynamic information. Since all computations will take some memory for processing, the size of available memory can be surely identified as dynamic information.

From the discussion above, it is easy to discover that these LRMS share the same static information and provide similar dynamic information. For the concern of higher-level application, the static information for the cluster should contain number of CPUs, operating system, total size of memory, number of nodes, the user's home directory, and the DNS name for the cluster. The dynamic information should include available nodes, available CPUs,

available memory, number of running jobs, number of queuing jobs.

3.3 XML representation

Dealing with these dynamics and heterogeneity is challenging for resource abstraction on the multicluster environment and is hindered by the lack of a consistent overview of available computational resources. We use XML Schema for uniformed resource description for any single cluster. The information for all available clusters is stored in one XML document, whose form is defined by the schema. All information entries are defined as simple elements in schema file. The “*static_info*” is a complex element which has all the elements representing static information. Another complex element called “*dynamic_info*” contains all the elements which represent dynamic information. These two complex elements are belonging to “*cluster*” element which is the root element.

The XML Schema for the cluster abstraction is provided in Figure 3. One sample XML description of a cluster available in Center for Computation and Technology at Louisiana State University is shown in Figure 4.

4. Uniform resource access

Since the abstraction hides complex details about the actual cluster hardware and software configurations and status from a user, the multicluster environment looks like a virtual cluster pool to the user. Despite the heterogeneity of different clusters, users can query information from the environment or configure their own execution environment (section 4.2) via simplified uniform interfaces. Due to different interfaces provided by different LRMSs, conventional

approach for utilizing the multicluster environment requires expertise knowledge on those LRMSs, which increases the difficulty for utilization. Some services from some LRMS may even be inapplicable for some other resources, for instance, Condor offers some interfaces for job execution history which is not applicable for PBS. The high level interfaces for the cluster abstraction concept should cover these diversities, and be versatile for all kinds of clusters, such that these interfaces should enable users (clients) to uniformly access individual resource or on a specific execution environment. These interfaces can be categorized into two groups according to their specific functions, **information related** and **operation related**.

4.1. Information interface

Since upper-level applications or users may need information on certain resources or certain jobs, the information related interfaces are divided into two categories, one for resource level and one for task level.

4.1.1. Resource level information Interface. The resource level information interfaces should be flexible enough to satisfy any query, from detailed information for one specific cluster to list of resources with demanding configurations. For instance, an execution management system wants to make decision on task allocation within a set of resources for which a specific user gets authorization. In this case, the execution management system needs information about these clusters. The information interface should be able to return a list of detailed information for each resource on the list. During this process, all that the execution management system knows is simply a list of clusters with DNS names only.

```

<!-- definition of simple elements -->
<xs:element name="DNS_name" type="xs:string"/>
<xs:element name="total_nodes" type="xs:positiveInteger"/>
<xs:element name="total_cpus" type="xs:positiveInteger"/>
<xs:element name="total_mem" type="xs:positiveInteger"/>
<xs:element name="available_nodes" type="xs:positiveInteger"/>
<xs:element name="available_cpus" type="xs:positiveInteger"/>
<xs:element name="available_mem" type="xs:positiveInteger"/>
<xs:element name="LRMS" type="xs:string"/>
<xs:element name="architecture" type="xs:string"/>
<xs:element name="running_jobs" type="xs:positiveInteger"/>
<xs:element name="queuing_jobs" type="xs:positiveInteger"/>
<xs:element name="home_dir" type="xs:string"/>

<!-- definition of complex elements -->
<xs:element name="static_info">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="DNS_name"/>
      <xs:element ref="total_nodes"/>
      <xs:element ref="total_cpus"/>
      <xs:element ref="total_mem"/>
      <xs:element ref="architecture"/>
      <xs:element ref="home_dir"/>
      <xs:element ref="LRMS"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="dynamic_info">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="available_nodes"/>
      <xs:element ref="available_cpus"/>
      <xs:element ref="available_mem"/>
      <xs:element ref="running_jobs"/>
      <xs:element ref="queuing_jobs"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="cluster">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="static_info"/>
      <xs:element ref="dynamic_info"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Figure 3. The pre-defined XML schema file, cluster.xsd

```

<cluster>
  <static_info>
    <DNS_name>helix2.cct.lsu.edu</DNS_name>
    <total_nodes>35</total_nodes>
    <total_cpus>70</total_cpus>
    <total_mem>72155440</total_mem>
    <architecture>linux</architecture>
    <LRMS>PBS</LRMS>
    <home_dir>/ibm_shark/home1/mxie</home_dir>
  </static_info>
  <dynamic_info>
    <available_nodes>32</available_nodes>
    <available_cpus>64</available_cpus>
    <available_mem>65970688</available_mem>
    <running_jobs>43</running_jobs>
    <queuing_jobs>764</queuing_jobs>
  </dynamic_info>
</cluster>

```

Figure 4. A sample resource entry in resource list, the cluster is named ,”Helix” from Center for Computation and Technology at Louisiana State University

4.1.2. Task level information interface

The task level information interface is responsible for any information requests about tasks on the virtual cluster pool. Requests include status of some specific jobs, or a set of jobs with same configuration, jobs belonging to a specific user, and jobs on a specific location or in some execution environment with certain configurations, etc. One good example is that, some user wants to know the status of all his jobs submitted to the multicluster environment before. In this case, without knowledge about the command for all the LRMSs on each cluster and without operating on each one individually, the status information should be returned to the user from the interface by simply specifying the execution environment and the user name.

4.2. Operation interface

Here, Operation Interface means a set of APIs responsible for specific operations on the virtual cluster pool, ranging from submitting tasks to managing customized execution environments. Before furthering our discussion on details, the term “*execution environment*” need to be clarified. The execution environment for multicluster Grid is a subset of clusters which satisfy certain hardware and software requirements from the user. It is defined in an environment file, which can be used for future operation on this environment. This environment file simplifies the process of specifying execution environment with multiple, complex constrains. More discussion about execution environment is covered in Section 4.3.

Due to different level of access permission, the operation interface contains two levels of interfaces: administrative operation interface and user operation interface.

4.2.1 Administrative operation interface

The administrative operation interfaces can provide cluster level or queue level administrations. Operations include adding new cluster to the virtual cluster pool or deleting virtual cluster from the pool, changing scheduling policy on specific cluster, managing user priorities, and other regular queue operations. The administrative regular queue operations can delete suspected user's jobs or jobs idling for an excessive long time, hold and release any job, reserve resources, etc. The administrative operation interface is effective only to those clusters which offer authorization as administrator to the user using this interface. This means that,

by default, all end users are considered as normal users and can use user operation interface. Only when the user has administrative permission to a cluster, he can perform administrative operations to that specific cluster.

4.2.2. User operation interface

Since an interface is only for normal users, all regular job operations, such as submitting, deleting, and holding. Releasing jobs can be only done to those which are owned by the users. According to the concept of cluster abstraction, the virtual cluster pool should be in user space, which means all users can manage their own virtual cluster pool by adding or deleting clusters. Users can customize their own execution environment by specifying requirements. After the specification, the service will select virtual clusters which satisfy the requirements and form an unique execution environment with an user-specified name. This name can be used to retrieve configurations for the defined environment in other interfaces.

To sum up, the administrative operation interface and the user operation interface share a subset of interfaces, with the only difference on the level of access permission. These interfaces are the regular job and pool operations.

4.3. Execution environment

The execution environment description contains sufficient information for a deployment service to create, manage, and perform regular task operation on this virtual environment. This information contains: 1. characteristic information specified at construction time, which can be used for future environment evaluation; 2.

description about participating computing resources.

Since we have virtualized the clusters by providing configuration and status, approaches to describe that participating computing resources range from the simple but inefficient one (e.g., a list of identifications for qualified resources) to the complex but powerful one (e.g., a collection of detailed information for each individual cluster).

In this paper, we believe that the second approach is more plausible in terms of efficient evaluation and deployment. The quantity of the statistic information defining the characteristics of the execution environment depends on the user specification at the time of initialization. Every user-defined execution environment has its own unique resource identification, which can be resolved to obtain information about this environment, such as its characteristics and detailed resource catalog. The unique name of the environment can also be used as task submission destinations for regular task level operation interface.

As mentioned above, all the information about the execution environment are stored in an environment file. The environment is represented as an arbitrary collection of entries that provide access information as well as other content. The characteristic information for the environment defines the common features for these resource entries, ranging from list of DNS names to arbitrary conditions on number of nodes. These definitions equip the deployment service ability to intelligently filter existing virtual cluster pool for qualified resources.

The deployment service should be able to make appropriate adjustment to the execution environment. After initialization of the environment, any event corresponding to the change of any configuration or status intriggers the reevaluation of the current

environments. This process filters the updated a virtual cluster pool for qualified resources, and update the information for the environment, which results in potential changes for the list of resource entries. Certain possible changes include removing such resource entry that fails in the process of reevaluation, or replacing such entry with qualified new entry. Reevaluation uses the characteristic information for standards. Therefore it makes no change to the property of the environment. Modifications to the characteristics of the environment can be done by using high level operation interfaces and intrigering reevaluation process. In conclusion, all execution environment modifications comprise three phases: obtaining updated characteristic information required for reevaluation; filtering virtual cluster pool with this update information; and updating the resource list with qualified data entries.

5. Deployment service

The deployment service has a set of components implementing the concept of cluster abstraction, which ensure the service to be user-space and cluster independent. The architecture is shown in Figure 5.

5.1 Service components

From Figure 5, we can identify all the components responsible for implementation and their relationships.

5.1.1. Information collector

Since the primary objective for cluster abstract is describing diverse participating clusters and providing uniform access to them, the information collector is able to communicate with all resources in the pool via their LRMSs.

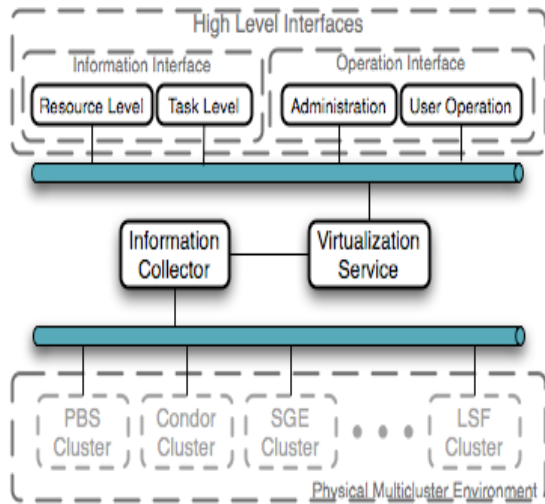


Figure 5. The service components responsible for the implementation of cluster abstraction.

The information collector broadcasts requests for required information to all available clusters in the pool, in another word authorized resources. These resources are identified by the client, using a list of DNS names. Since the static configurations are considered as constant for a small time period, after initial request, the information daemon updates this information for a large time interval, which is specified by administrator. For those dynamic statuses, they are changing every minute even in seconds. So, the information collector updates the dynamic information in a fairly small time interval. After the collection, the collector passes all information to the virtualization service, which creates virtual clusters in the virtual cluster pool.

5.1.2. Virtualization service

This service is responsible for the deployment of cluster virtualization. Based on the pre-defined XML schema, all information passed from the information collector are stored according to the format defined by the schema. The virtualization service also provides some basic operation interfaces, such as parsing.

The virtualization service has three components, a XML generator, a validator, and a parser. After the XML generator creates the XML file and fills all the information, the validator validates the file according to the schema. The parser is the critical part of the virtualization service, enabling access to the data from outside.

5.1.3. High level interface

The high level interfaces, which are also referred as application programmable interfaces, bridge the clients and operations on heterogeneous multicluster Grid environment. By using the information provided from virtual cluster pool, these interfaces ensure efficient and easy operations on the virtual clusters, i.e., task operations and resource operations. They also equipped users the capability of developing customized program logic, which made execution management, workflow management and so on much easier and more convenient.

5.2. Security

At deployment, excessive information exchanges between clusters and client machines can be expected, which are subject to authorizations and authentications before data transfer. Since the deployment service uses *SSH* and *Globus* for communication, one can use the security mechanism of these two toolkits for authentications. Once a client logs in, the service should allow this authorized client to operate without further authorization requests.

6. Conclusions and future work

By using XML schema technology, we represent physical clusters by virtual clusters which have all necessary information

adequate for performance evaluation. Moreover, all high level interfaces ensure easy and uniform access to the entire virtual cluster pool. Cluster abstraction successfully hides the heterogeneity of the multicluster Grid and simplifies the operations on these diverse resources. To users or higher-level applications, a set of virtual clusters with similar configurations and same interfaces exist in the virtual cluster pool. This means that, end users or higher-level applications can define their own virtual cluster pool with whatever clusters they have access permission to. Users can customize their own execution environment by specifying requirement for that environment. This execution environment contains a number of virtual clusters in the pool. After customizing, all executions regarding to this environment can be automatically submitted, executed and managed on those clusters within the environment.

Cluster abstraction is only one concept of our ongoing project for multicluster Grid execution management. The basic concept of execution management is called Dynamic Assignment with Task Container (DA-TC) [14]. A typical job submission includes submission to super-scheduler, submission from super-scheduler to local scheduler, execution on computational resources, and delivery of execution results and exit status. The DA-TC execution model tries to minimize the turnaround time by reducing the queuing time of any new job. Furthermore, many issues like co-scheduling, fault-tolerance need to be studied. For cluster abstraction, our immediate future work will make it compatible with other LRMSs. More APIs are expected to be added to make the system available for any operations on resources and jobs.

Acknowledgements

This research is sponsored by the U.S. Department of Energy (DOE) under Award Number DE-FG02-04ER46136 and the Board of Regents, State of Louisiana, under Contract No. DOE/LEQSF (2004-07).

References

- [1] M. Brune, A. Reinefeld, J. Varnholt, "A resource description environment for distributed computing systems," *Proc. of the 8th IEEE International Symp. on High-Performance Distributed Computing (HPDC'99)*, pp. 279-286, 1999.
- [2] C. Catlett, "The TeraGrid: A Primer," 2002.
- [3] J. Chase, L. Grit, D. Irwin, J. Moore, S. Sprenkle, "Dynamic Virtual Clusters in a Grid Site Manager," *Proc. of the 12th IEEE International Symp. on High-Performance Distributed Computing (HPDC-12)*, 2003.
- [4] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman, "Grid information services for distributed resource sharing," *Proc. of the 10th IEEE International Symp. on High-Performance Distributed Computing (HPDC-10'01)*, pp. 181-194, 2001.
- [5] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Tuecke, "A resource management architecture for metacomputing systems," *Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing*, pp. 62-82, 1998.
- [6] M. Feller, I. Foster, S. Martin, "GT4 GRAM: A Functionality and Performance Study."

- [7] M. Fisher, C. Kubicek, P. McKee, I. Mitrani, J. Palmer, R. Smith, "Dynamic Allocation of Servers in a Grid Hosting Environment," *Proc. of the 5th IEEE/ACM International Workshop on Grid Computing*, pp. 421-426, 2004.
- [8] I. Foster, C. Kesselman, "Globus: A toolkit-based Grid architecture," *The Grid: Blueprint for a New Computing Infrastructure*, pp. 259-278, 1999.
- [9] I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the grid: enabling scalable virtual organizations," *International J. Supercomputer Applications*, 15(3), 2001.
- [10] J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke, "Condor-G: A computation management agent for multi-institutional grids," *Proc. of the 10th IEEE International Symp. on High-Performance Distributed Computing (HPDC-10)*, August 2001.
- [11] R. Goldberg, "A survey of virtual machine research," *IEEE Computer*, 1974, 7(6), pp. 34-45.
- [12] K. Keahey, I. Foster, T. Freeman, X. Zhang, D. Galron, "Virtual workspaces in the Grid," *ANL/MCS-P1231-0205*, 2005.
- [13] T. Kosar, M. Livny, "Stork: making data placement a first class citizen in the Grid," *Proc. of 24th IEEE International Conference on Distributed Computing Systems (ICDCS 2004)*, March 2004.
- [14] Z. Lei, M. Xie, Z. Yun, G. Allen, N. Tzeng, "Efficient application execution management in multicluster Grids," *Submitted to: IEEE/ACM International Conference on Grid Computing (GRID-07)*, September 2007.
- [15] B. Lowekamp, B. Tierney, L. Cottrell, R. Hughes-Jones, T. Kielmann, M. Swany, "Enabling network measurement portability through a hierarchy of characteristics," *Proc. of the 4th International Workshop on Grid Computing (Grid2003)*, pp. 68-75, 2003.
- [16] LSF <http://www.platform.com/Products/Platform.LSF.Family>.
- [17] PBS <http://www.openpbs.org>.
- [18] SGE <http://gridengine.sunsource.net>.
- [19] M. Swany, R. Wolski, "Representing dynamic performance information in Grid environments with the network weather service," *Proc. of the 2nd IEEE/ACM International Symp. on Cluster Computing and the Grid (CCGrid'02)*, pp. 48-56, 2002.
- [20] D. Thain, T. Tannenbaum, M. Livny, "Condor and the Grid," *Grid Computing: Making the Global Infrastructure a Reality*, John Wiley, 2003, ISBN: 0-470-85319-0.