Provide Virtual Machine Information for Grid Computing

Article *in* IEEE Transactions on Systems Man and Cybernetics - Part A Systems and Humans · November 2010

DOI: 10.1109/TSMCA.2010.2052598 · Source: DBLP

5 authors, including:

Lizhe Wang
291 PUBLICATIONS **6,246** CITATIONS
SEE PROFILE

Gregor von Laszewski
Indiana University Bloomington
239 PUBLICATIONS **7,014** CITATIONS
SEE PROFILE

Dan Chen
Wuhan University
132 PUBLICATIONS **2,551** CITATIONS
SEE PROFILE

Jie Tao
Karlsruhe Institute of Technology
132 PUBLICATIONS **2,094** CITATIONS
SEE PROFILE

Provide Virtual Machine Information for Grid Computing

Lizhe Wang, *Member, IEEE*, Gregor von Laszewski, Marcel Kunze and Jie Tao

Abstract

Distributed virtual machines can help to build scalable, manageable and efficient Grid infrastructures. The work proposed in this paper focuses on employing virtual machines for Grid computing. In order to efficiently run Grid applications, resource information of virtual machines should be provided. The paper firstly discusses the system architecture of virtual machine pools and the process of information retrieval from virtual machines. Based on the characterization of the system model, the paper presents the work how to retrieve resource information from Xen/VMware virtual machines via VMware CIM SDK and lightweight Java agents. The resource information is integrated into Grid information service. The work is implemented in a test bed with Xen/VMware virtual machines and Globus Toolkit. With performance evaluation and discussion on real test bed, it is declared that the design and implementation of information service for virtual machine based Grid system are feasible, efficient, and scalable.

Index Terms

irtual machine; Grid computing; Information serviceirtual machine; Grid computing; Information serviceV

I. Introduction

Grid computing technology [1] offers exciting solutions for parallel and distributed computing. It can provide reliable, collaborative and secure access to remote computational resources as well as distributed data and scientific instruments. Although great advances have been made in the field of Grid computing, users still expect to meet some difficulties of employing Grid resources:

- QoS of resource provision and performance isolation
  Computational Grid is a highly dynamic environment in that resource capacities and access interfaces may change from time to time. Resources shared on computational Grids in general don't guarantee QoS (Quality of Service) of resource provision. Multiple Grid applications compete for resource usage when shared resources are employed.
- Customized runtime environment
  In general, Grid applications demand customized execution environment such as operating system, software packages and libraries, network configurations. Some requirements of these runtime configurations need administration privileges, which are normally impossible to acquire in a Grid environment.

A virtual machine is a computing platform that creates a virtualized layer between the computing hardware and the application. Employing virtual machines as computing environments for Grid applications can address challenges above. In general, Grid users may benefit from the virtualization techniques in following aspects:

- On demand creation and customization
  Users can create a customized virtual machine, which provides customized resource allocation for users, e.g., operating system, memory, storage, etc.
- Performance isolation
  Virtual machines can guarantee the performance for users and applications. Users of virtual machines could expect a dedicated computing environment, which is hard to find in multiple-user computing servers.
- Legacy software support
  Customized virtual machines which are compatible with legacy binary applications can be created. Users from specific application domains would find them very desirable and promising since some legacy libraries could be supported.
- Easy management
  Users in general should only access computing servers with restricted user privileges. It is thus difficult to process the work such as compilation, installation, configuration of desirable computing environment for users. Virtual machines on the contrary, could offer users with "root" access of the allocated virtual machine. Therefore application domains could manage the environments in their own interests.

Virtual machine based Grid systems are characterized by some special features, which bring research challenges for deploying, monitoring and operating the system:

- Site autonomy
  In the virtual machine based Grid system, the hosting resources, which run a Virtual Machine Monitors (VMM) and support multiple virtual machines, are commonly owned and controlled by different institutes or organizations at different sites. Users may expect to meet different resource management policies during the creation and manipulation of virtual machines.
- Hierarchy
  A virtual machine based Grid system is hierarchical in nature. It contains several levels, virtual machine level, hosting resource level and the user access point, i.e., Grid portal.

Manuscript received xxx xx, 2008; revised xxx xx, 2008.

Lizhe Wang and Gregor von Laszewski are with Service Oriented Cyberinfrastruture Lab, Rochester Institute of Technology. Marcel Kunze and Jie Tao are with the Steinbuch Centre for Computing, Research Center Karlsruhe, Germany.

- Heterogeneity
  A virtual machine based Grid system includes heterogeneous hosting resources, virtual machine technologies (e.g., Xen [2], VMWare [3]) as well as programming interfaces.
- Large scale distribution
  Computer centers and data centers frequently employ virtual machines and build Grid infrastructures across geographically distributed sites.

In a virtual machine based Grid system, some specific requirements demand attentions for the information service:

- Efficient delivery of resource information from virtual machines to clients in the hierarchical Grid environment;
- Information services should be scalable and robust with regard to dynamic startup/shutdown of virtual machines;
- The information collector which runs inside in the virtual machine should be lightweight, portable and manageable;
- Information from popular VMM such as Xen and VMware which is defined by the CIM (Common Information Model) schema [4] should be translated to higher level Grid services and user access, where information is defined in GLUE schema [5].

In this paper, we present our work on information service for Grid infrastructures which consist of distributed virtual machines. The system is hierarchical and includes following components:

- Information collector
  The information collector runs in the virtual machines and functions to get resource information of virtual machines.
- Information translator and provider
  The information provider collects resource information from multiple information collectors inside one computing center or manage domain, and provides to high level Grid services of various Virtual Organizations (VOs).
- High level Grid information service
  In the VO level, Grid information service or other services, such as Globus WebMDS [6], could demand virtual machine resource information which is provided by multiple information providers.

This paper presents our work on information service for virtual machine based Grid system. Our contributions include:

- define a hierarchical virtual machine based infrastructure;
- build a scalable and efficient information service that can provide virtual machine information to Grid system;
- build a prototype which translates CIM schema based virtual machine information to Grid information with GLUE schema.

The rest of the paper is organized as follows: related work is investigated in Section II. Section III gives an overview on the design and implementation of the information service. Section IV, Section V, Section VI detail the design and the implementation of components in the information service: the information collector, the information translator, the information provider and the higher level Grid services. In Section VII test results are presented and discussed. Section VIII concludes the paper and points out future work.

II. Related work

Since several years the Grid computing research community shows interest for virtual machines and virtual environments. The typical Virtual Machine Monitor (VMM) or hypervisor setup includes Xen VMM [2], VMware server/ESX server [3], and User Mode Linux [7]. The Globus alliance recently implemented the concept of virtual workspace [8] which allows a Grid client to define an environment in terms of its requirements, manage it, and then deploy the environment on the Grid. The implementation is based on Globus Toolkit 4 (GT4) [9] and it only supports Xen VMM. Some other research work also focuses on deploying computing systems or test beds with virtual machines, for example, virtualization of batch queueing system [10], GridBuilder [11], using virtual machine as Grid gateway [12], multi-site MPI platform with Xen virtual machine [13], migration of virtual machines in MAN/WAN [14].

Other researchers try to build virtualized middleware for clusters and distributed systems. Xen Grid Engine [15] follows an approach to create dynamic virtual cluster partitions using para-virtualization techniques. The work presented in [16] builds virtual clusters and virtualized distributed infrastructures. The In-VIGO [17] project aims to build virtualization middleware for computational Grids. In-VIGO provides a distributed environment where multiple application instances can coexist in virtual or physical resources so that clients are not aware of the complexity inherent to Grid computing. Cluster-on-Demand (COD) [18], [19] from Duke Univ. aims to separate cluster usage from cluster management by implementing a virtual cluster with virtual machines. Amazon Elastic Compute Cloud (EC2) [20] is a Web service that provides resizable compute capacity in the cloud, which is enabled by Xen virtual machines.

Various advances have been made in field of virtual network. Violin [16] employs user-level communication indirection between the virtual machines and the underlying infrastructure. The In-VIGO system implements Wide-Area Overlays of Virtual Workstations (WOWs) by creating virtual IP networks on top of P2P overlays. While the Violin and the In-VIGO implementations are based on user-level overlay networks, VNET [21] of Virtuoso [22] system is realized in both user-level and kernel level: host kernel-level devices are created to tunnel network traffic.

Another important topic is the performance analysis of virtual machines or virtual environments. The Xen group has published the performance evaluation and comparison between several popular VMMs concerning the performance overhead in different scenarios [23]. Other research efforts refer to virtual machine based systems, i.e., performance of para- and paene- virtualized systems [24], performance enhancement of SMP clusters with virtual machines [25].

The Grid Information Service (GIS) is a core component in the Grid software infrastructure. In a Grid environment, the description, discovery, and monitoring of resources (e.g. hardware, software, data, instruments) is very challenging due to the diversity, large numbers, transient membership, dynamic behavior, and geographical distribution of the entities in which a user might be interested. Consequently, Grid Information Service (GIS) is a vital part of any Grid software infrastructure, providing a fundamental mechanism for discovery and monitoring, and hence for planning and adapting application behavior [26]. There are some existing examples of GIS: the Monitoring and Discovery System (MDS) [27] in the Globus Toolkit [9] is one of popular implementations. The MDS is an Information Service for information management in computational Grids. It provides interfaces for information management like generating, aggregating and accessing of information data from Grid resources.

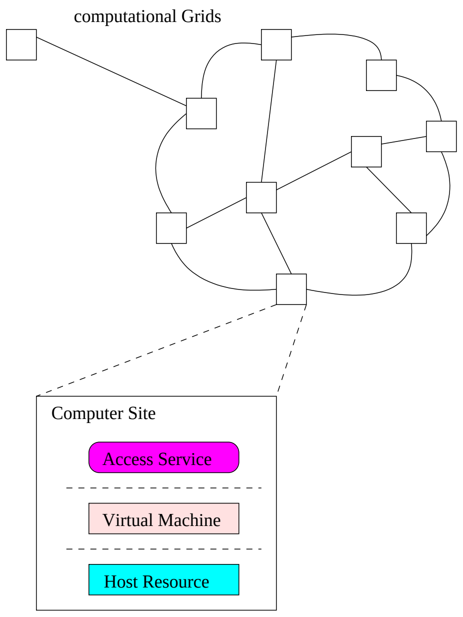


Fig. 1. Target system model

III. SYSTEM ARCHITECTURE OF VIRTUAL MACHINE BASED GRID SYSTEM

We propose a system model to describe a distributed, hierarchical, heterogeneous virtual machine based Grid system, which contains distributed **Computer Site**s interconnected by networking (see also Figure 1).

Each Computer Site consists following levels logically:

- The Compute Site provides an **access service** which allows remote users to access resources of the computer center. The access service could be offered by existing Grid middleware, a portal, Web services, or any functionalities that support remote steering. Our information service is developed and integrated in this level.
- In the middle level exist **virtual machines** that are backed by host resources. Information service operates virtual machines in this level.
- Fabric level contains various **host resources** or servers, which are installed with virtual machine VMMs. Host resources offer multiple virtual machines. Back ends of information service are implemented in this level.

The information service consists of an information collector in the virtual machine, the client of information collector, and information translator and an information provider in the access point, and the aggregated Globus index service. The information collector which runs inside a virtual machine is used to retrieve resource information for the information provider. Information collector clients get results from information collectors, the information translators change CIM [4] information to GLUE [5] information and information providers for Globus MDS (Monitoring and Discovery System) [27] organize the resource information from information collectors, which are aggregated into the Globus index service. The WebMDS [6] can be configured as a graphical user interface based on the Globus index service (see also Fig. 2).

IV. INFORMATION COLLECTOR

The information collector is a lightweight software, which resides inside a virtual machine and collects resource information. In the system, two types of information collectors have been implemented: CIMOM agent for VMware ESX server and light-weight Java agent.

A. *CIMOM agent for VMware ESX server*

VMware ESX server is a commercial virtualization product of VMware Inc. Common Information Model (CIM) [4] is defined as an international standard by the Distributed Management Task Force (DMTF) [28]. The VMware ESX server together with CIM SDK provide a CIM-compliant object model for virtual machines and their related storage devices. Fig. 3 shows a typical configuration environment of VMware ESX server. The virtual machine contains a virtual disk that resides as a virtual disk file on a storage area network.

The SMI-S (Storage Management Initiative Specification) schema for the sample VMware ESX server environment is shown using UML in Fig. 4. `ESXComputerSystem` is the kernel object of the system. It associates `VM`, `VirtualDisk` and `FC HBA&LUN`[1] with `HostedDependency`, `HostedStoragePool` and `SystemDevice` relationships respectively. `VM` is associated with `VirtualDisk` in `ArchiConnection` relationship. The latter is associated with `FC HBA&LUN` in `ConcreteComponent` relationship.

The pegasus CIMOM (CIM Object Manager) [29] is deployed on the VMware ESX server. The information provider at the access point works as CIM client and communicates with pegasus CIMOM to retrieve information from virtual machines and their associated storage. The information provider complies with SMI-S profile [30] and transports CIM XML over TCP/IP to pegasus CIMOM (see also Fig. 5).

[1]Fibre Channel Host Bus Adaptor & Logical Unit Number

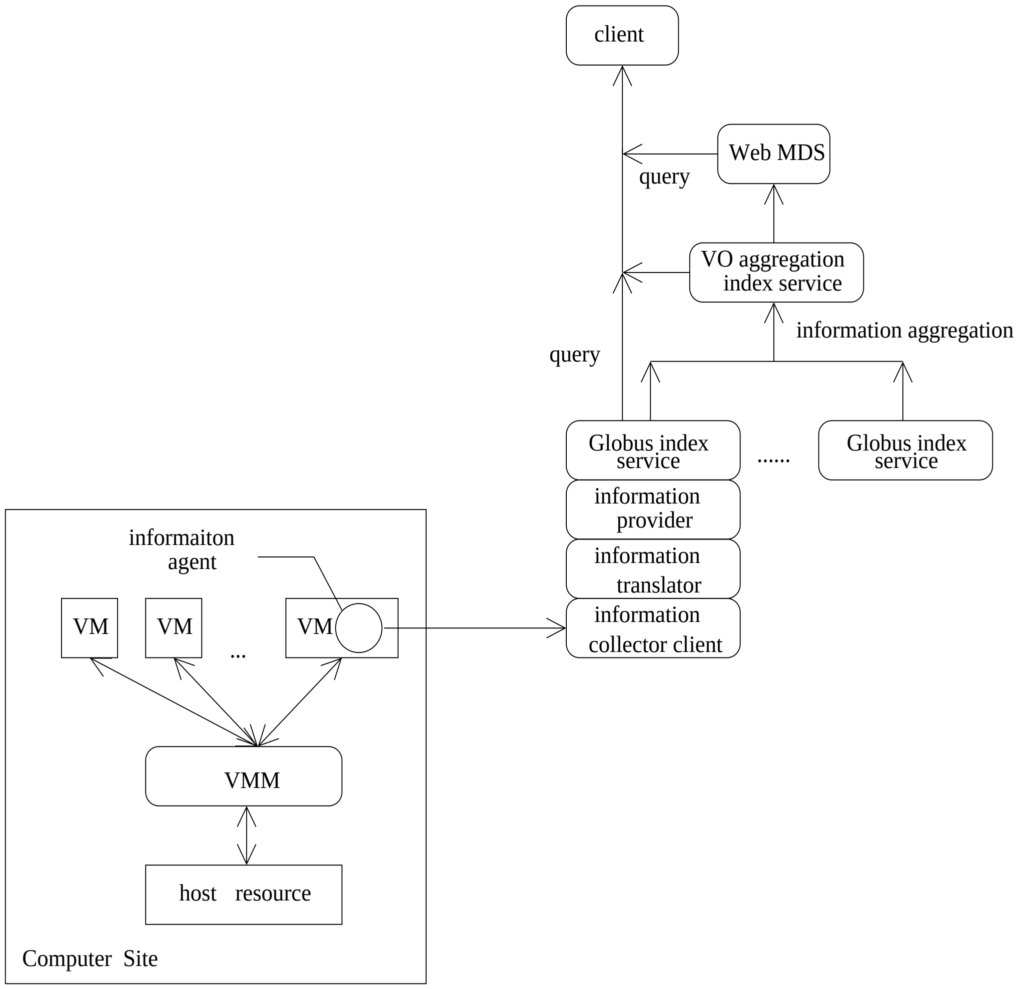


Fig. 2. Overview of the information service

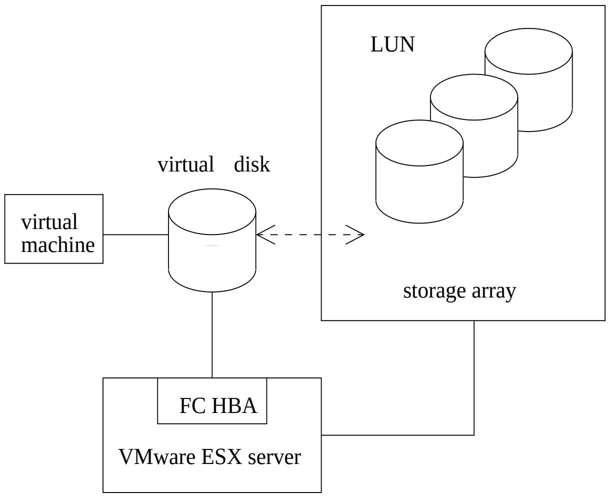


Fig. 3. Sample environment of VMware ESX server

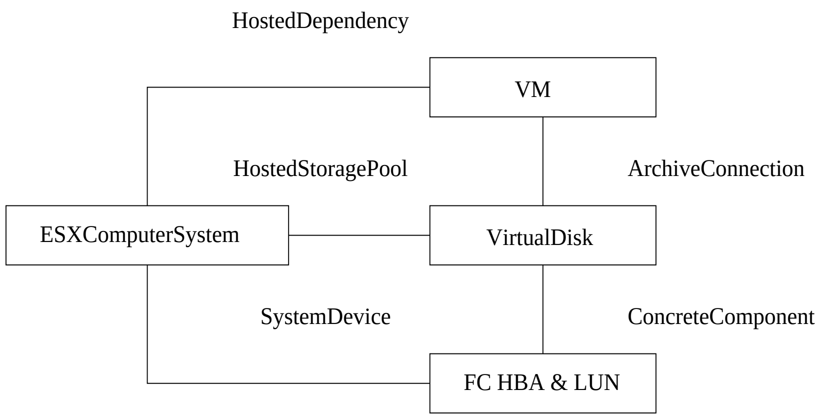


Fig. 4. CIM schema for VMware ESX server

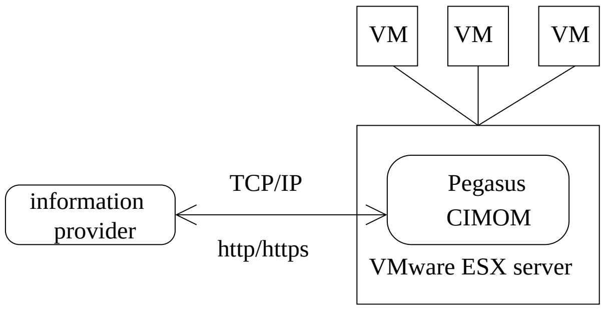


Fig. 5. CIMON agent for VMware ESX server

We implement the client side codes which communicate with pegasus CIMOM server on the basis of VMware ESX server APIs. The communication between the client and pegasus CIMOM server is based on HTTP/HTTPs or TCP/IP protocols. The client side codes, together with information translator and information provider, reside in the access point of the site.

B. Lightweight Java agent for VMware server and Xen VMM

VMware server is a free virtualization product of VMware VMM without CIM SDK support. The Xen CIM project is providing an implementation of the preliminary virtualization and resource allocation models, and it is currently being defined by the DMTF System Virtualization, Partitioning, and Clustering Working Group (SVPC WG). As the CIM SDK or programming support is not available for Xen VMM and VMware server we implemented a lightweight Java agent as information collector for Xen VMM and VMware server.

The information agent is a Java class and runs as a daemon in the virtual machine. Clients can:

- query the information agent from command line,
- obtain the resource information from the information agent via Socket message communication, and
- subscribe to the information agent and update the information periodically.

The information agent contains the following components (see also Fig. 6):

- Information Server
  The server of information agent receives inputs from clients, makes queries to information engine, and returns resource information to clients. In the implementation, information agent creates a thread in the information server to process the query when new query comes.
- Information Engine
  The engine of the information agent is responsible of processing the queries forwarded by the information server. The information engine firstly queries on the information item cache in the memory. Each information item has a time stamp and timeout value. Information item values expire when time passes by more than the timeout value. When the information in the cache is missed or expired, the engine then runs information sensors to get the latest information.
- Information Sensor
  The information sensor is devoted to retrieve some specific resource information. For example, memory sensor provides memory information such as free size, total size and swap size. In the implementation, the information agents are perl scripts to get memory, CPU, OS information. It is also possible to get the application runtime information with user-defined sensors.
- Information Item Cache
  Each information sensor can provide several information items. For instance, CPU sensor delivers information items such as CPU type, CPU number and CPU load. The information item cache is a block of memory allocated to store values of information items. Each time after the information sensors are executed, information item cache is updated.

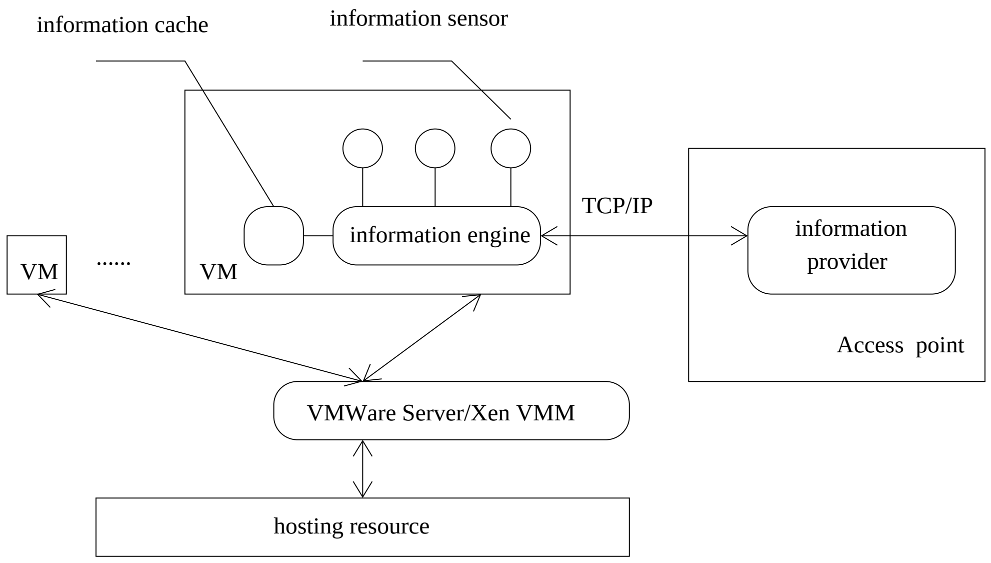


Fig. 6. Lightweight Java agent for VMware server and Xen VMM

The communication between the information agent and information provider, which is the client of the information agent, in the access point is implemented using TCP/IP socket communication. The information agent periodically updates the information to the information provider at the access point. The information provider can also query information agents to obtain the latest information.

Information server is constructed when the information agent is started. The server reads two configuration files:

- Information sensor/item configuration file
  The server reads the configuration and builds a table in memory, which maps information items to information sensors. Based on the table, information server can map the query to item/sensor pair and the related information, e.g., information source and time out for the item.
- Agent configuration file
  Information server also finds agent configurations in the configuration file, e.g., the port number it listens for query.

On receiving the query, the information server starts up a thread to precess the query. It checks the syntax of the query and locates the related information item/sensor for the query. The information server starts the information engine to retrieve the value for the query. In the information engine, an information item cache is maintained in memory. If the information engine cannot locate the information item in the cache or the value in the cache is expired, the engine executes the corresponding sensors and retrieves the information. After the execution, the information sensor also updates other related information items in the cache. After it acquires the resource information from the cache or sensors, the information engine forwards the values to the information server, which finally returns them to the client.

V. Information Translator

A. CIM schema for virtual machine information

The CIM (Common Information Model) schema [4], which is a standard created by DMTF (Distributed Management Task Force), provides a common definition of management information for systems, networks, applications and services. CIM is a conceptual information model for describing computing and business entities in enterprize environments. The fundamental goals of CIM are common definitions that enable vendors to exchange semantically rich management information between wide varieties of systems.

The VMware CIM SDK provides a CIM interface for developers to build management applications. With the VMware CIM SDK, developers can use CIM-compliant applications to explore the virtual machines on ESX Server, along with associated storage resources.

B. Grid information schema and information service

Various types of resources which are shared on computational Grids should be described in a precise and systematic manner. The Grid resources are thus able to be discovered for subsequent management or use.

The GLUE (Grid Laboratory Uniform Environment) schema [5] represents an abstract model for Grid resources and mappings to concrete schemas that can be used by information services within Grids. The underlying idea of the schema therefore is to provide an information model that can be used to exchange pieces of information among different knowledge domains and virtual organizations [31]. The GLUE schema is widely used in production Grid such as EGEE [32], OSG [33] and TeraGrid [34].

The GLUE schema is defined in UML diagrams (1.3 version) or XML (1.2 version). The GLUE schema defines so called core entities, such as `Site`, `Service`, `ComputingElement` and `StorageElement`. The relations between core entities are represented in concept level, such as objects and properties.

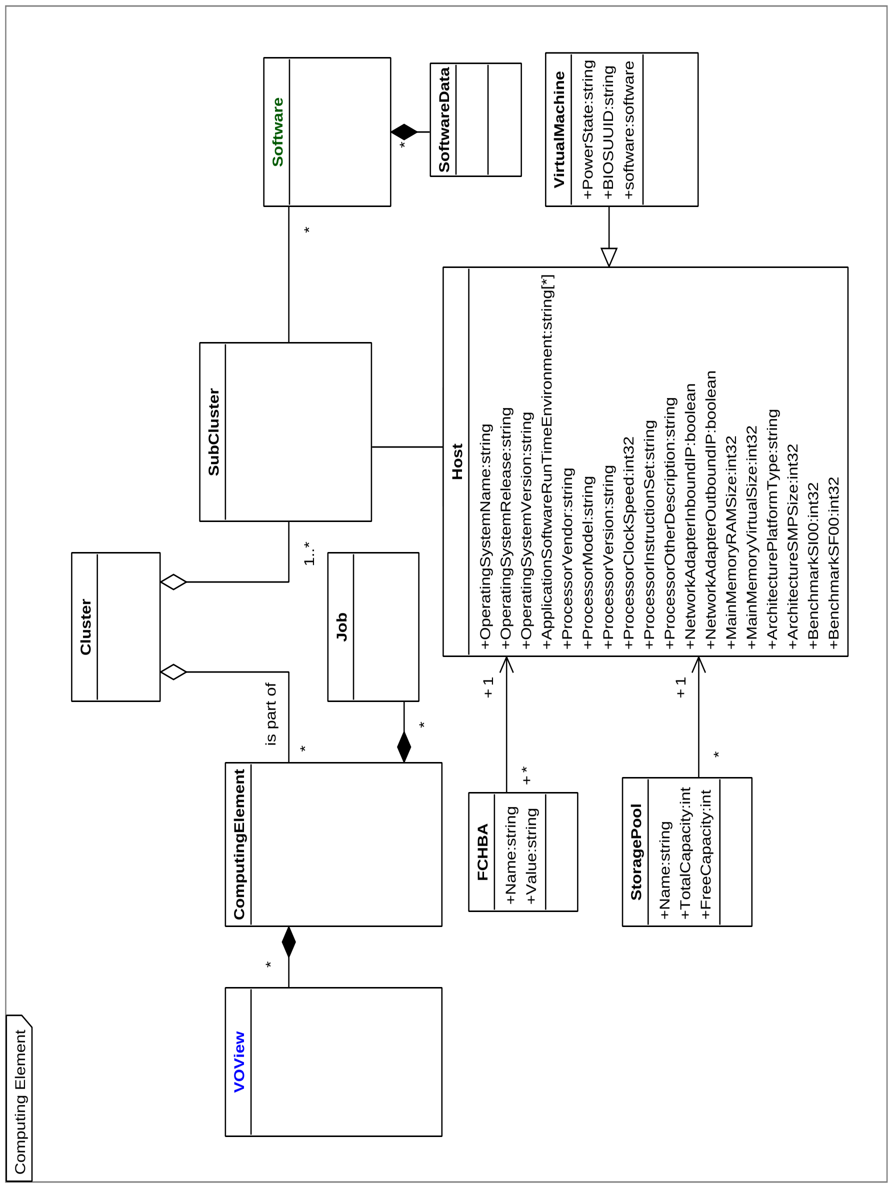


Fig. 7. Extended GLUE schema

An Grid information service provides information about a Grid infrastructure that consists of a wide variety of Grid resources. Grid information is thereafter used for various Grid operations, such as resource discovery/monitoring/accounting and job submission/execution.

In our production Grid environment, the Globus index service is used for the Grid level information service. The Grid index service can collect information and publish the information to clients. The Grid index service can also register to each other in a hierarchical fashion in order to aggregate data at several levels. The `Aggregator Framework` of Globus index service is used to build services that collect and aggregate data. For example, we build an information provider from virtual machines and provide information to the `Aggregator Framework` [35].

C. *Translation of CIM information to GLUE information*

The virtual machine information of VMware is organized with CIM schema. In production computational Grids, resource information is defined by GLUE schema. The information provider therefore needs to translate the CIM information into GLUE information, then marshes the data into GLUE XML file.

There are mainly two technical problems for translating CIM information to GLUE information:

- The GLUE schema does not define the virtual machine concept and its related objects.
- CIM schema and GLUE schema make their own representation at different levels. CIM schema is represented with various Java/C++ objects, and GLUE schema is defined in XML and UML entities. It is thus impossible to make a direct mapping of CIM schema to GLUE schema.

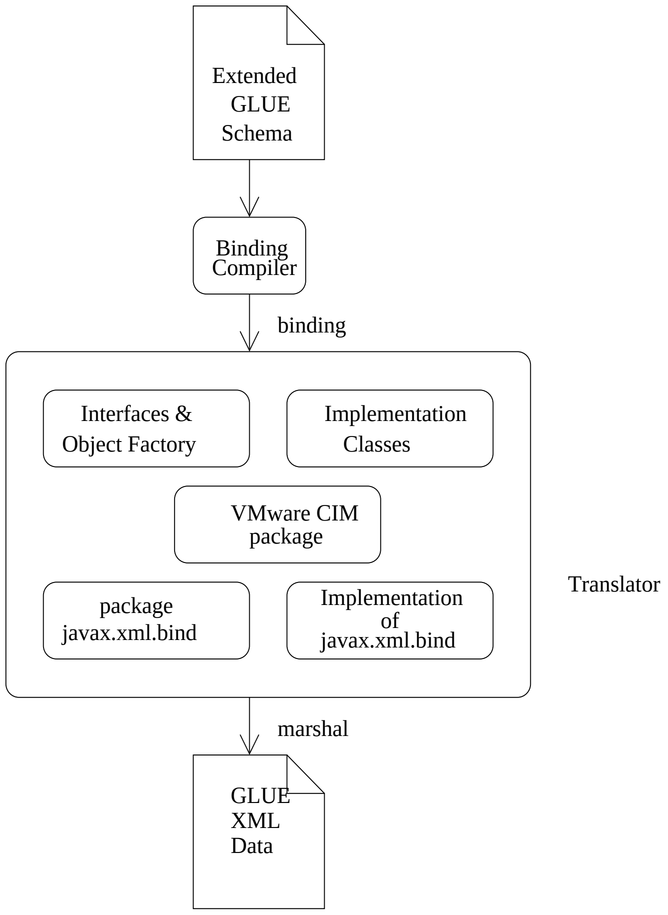


Fig. 8. Translation from CIM schema to GLUE schema

We solve the above problems as follows:

- The GLUE schema has been extended accordingly to support virtual machine concept. A **VirtualMachine** class is created by inheriting **Host** class in the GLUE schema. Attributes and operations are extended in **VirtualMachine** classes to represent virtual machine concepts. Each **Host** is associated with multiple **FCHBA** classes and **StoragePool** classes. Figure 7 shows the extended GLUE schema.
- The GLUE schema is represented in various high level formats, such as XML schema and UML class diagram. The CIM schema, on the other hand, provides an object oriented format. The CIM schema shipped by VMware is represented in Java classes. Thus there is no direct mapping from CIM schema to GLUE schema. To solve this problem, extended GLUE schema generates a set of bing Java classes with JAXB binding compiler. The Java Architecture for XML Binding (JAXB) [36] provides a fast and convenient way to bind an XML schema to Java representations. With the JAXB compiler and binding tools, Java binding classes could be automatically generated from GLUE schema. Now the GLUE binding Java classes can map to CIM Java classes by invoking operations of CIM Java classes.

VI. Information provider and Globus Toolkit Aggregation Framework

The Globus Toolkit `Aggregator Framework` [35] is a software framework used in Globus Toolkit to build services that collect and aggregate data. Grid higher level services, i.e. the Globus Index service, built on the `Aggregator Framework`, are sometimes called `Aggregator services`. They collect information via `Aggregator Source`s. An `Aggregator Source` is a Java class that implements an interface to collect XML-formatted data. In our implementation we use the `Execution Aggregator Source`, which executes an administrator-supplied program, the information provider, to collect the information.

The information provider is an executable that runs on the access point. The information provider acts as the client of information collector and gets resource information from virtual machines. It then runs the translator to translate the CIM based resource information to Grid resource information in GLUE schema.

Some configurations are needed for Globus index service to get information from `Execution Aggregator source`, for example, mapping the information provider in the deployment file of the Globus index service and configure the schedule to run the information provider. Therefore GLUE XML data could be automatically generated and provided to Globus `Aggregator Framework`. The Globus index service could provide resource information from virtual machines to higher Grid services, for example, Globus WebMDS [6] or be accessed from the Globus Toolkit command line.

VII. Performance evaluation and discussion

A. Motivation

To evaluate the efficiency of the information service, we use the **throughput** as performance metric, which is defined as: the average number of requests processed by the information service within a time unit, for example, one second.

We are also interested in the performance issues with concerns on scalability of information service: how does the performance of the information service scale with the number of the users? To evaluate the scalability of the information service, we use **response time** as the performance metric, which is defined as: the time between the instant when a client sends query to the information service and the instant when the the client gets the reply from the information service. We test various response times with different number of users.

B. Test bed

The work is carried on the SCC (Steinbuch Centre for Computing) [37] productional test bed shown in Figure 9, Table I and Table II. The test bed contains two parts: computer servers in IWR/FZK (EGEE project tier-1 site) and a compute cluster in RZ/UKA (EGEE project tier-3 site). Computer servers in IWR/FZK are installed with various VMMs, such as Xen server, VMware server and VMware ESX server. The compute cluster in RZ/UKA consists 1 head node and 12 worker nodes, all of which are installed Xen server. The head node backs 8 virtual machines, which offer various cluster/Grid functionalities, e.g., file server, monitor server and Grid portal. Each worker node provides one virtual machine, which is used to execute Grid jobs. There is a batch schedular in the cluster to schedule local jobs on worker nodes.

TABLE I
IWR/FZK TEST BED DESCRIPTION

Resource name	**Resource Type**	**Resource description**
Blade9/Blade10	VMware ESX server IBM eServer Bladecenter HS20	2 × Intel® Xeon™ CPU 3.00 GHz, 5.08 GB RAM
Blade11/Blade12	Xen server IBM eServer Bladecenter LS120	2 × AMD Opteron™ Processor 250, 5.08 GB RAM
CVS2	VMware Infrastructure 3	multiple Bladecenters
Test1	Xen Enterprise Server	2 × Intel® Xeon™ CPU 2.4 GHz, 3.0 GB RAM
LZ01	Linux Server	Intel® Celeron™ CPU 2.00 GHz, 1.0 GB RAM
LZ02		Intel® Pentium™M processor 1.5 GHz, 512 MB RAM
LZ03		AMD Athlon™ 64 Processor 3500+, 2 GB RAM
VM	Virtual machine	Worker node

TABLE II
TEST CLUSTER AT RZ/UKA

Resource name	**Resource Type**	**Resource description**
Test cluster	Compute cluster	Maui batch scheduler, ic0 = head node, ic0n01∼ic012 = worker node
clu01∼clu012	Virtual machine	Worker nodes
cluctl		Grid index service
lcgse		Storage Element
lcgwms		Resource Broker
lcgmon		Grid Monitoring Server
lcgce		Computing Element
ganlia		Ganglia Cluster Monitoring Server
lcgbdii		BDII Server
cms1		Portal

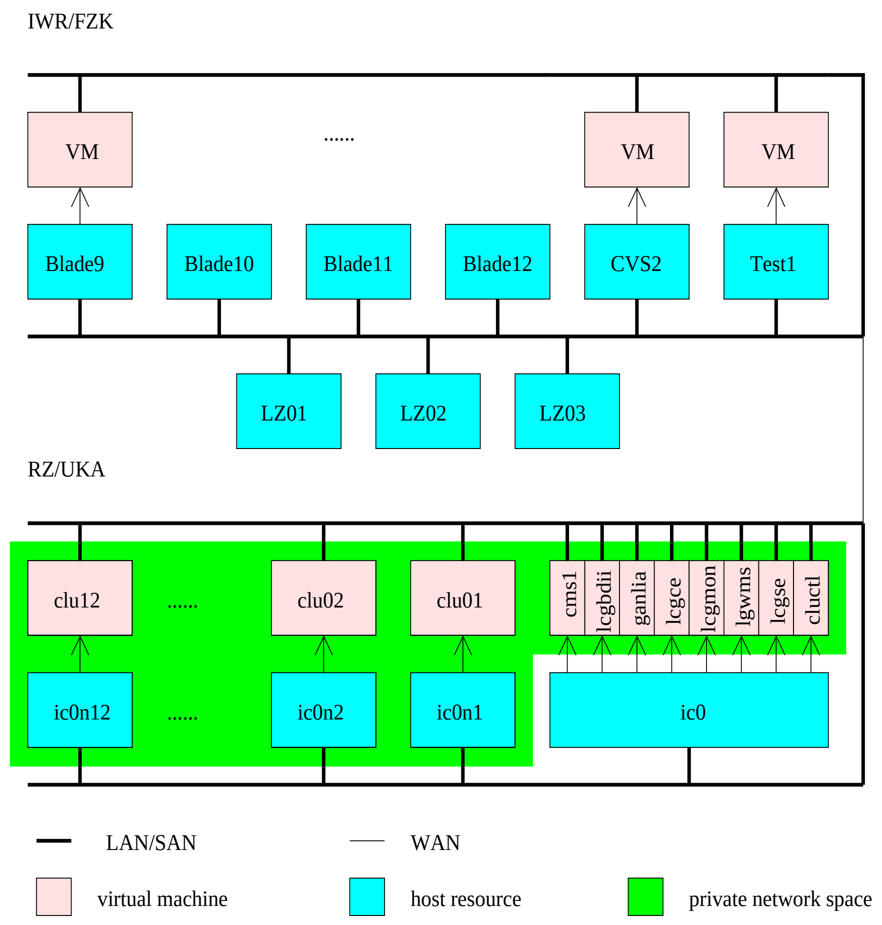
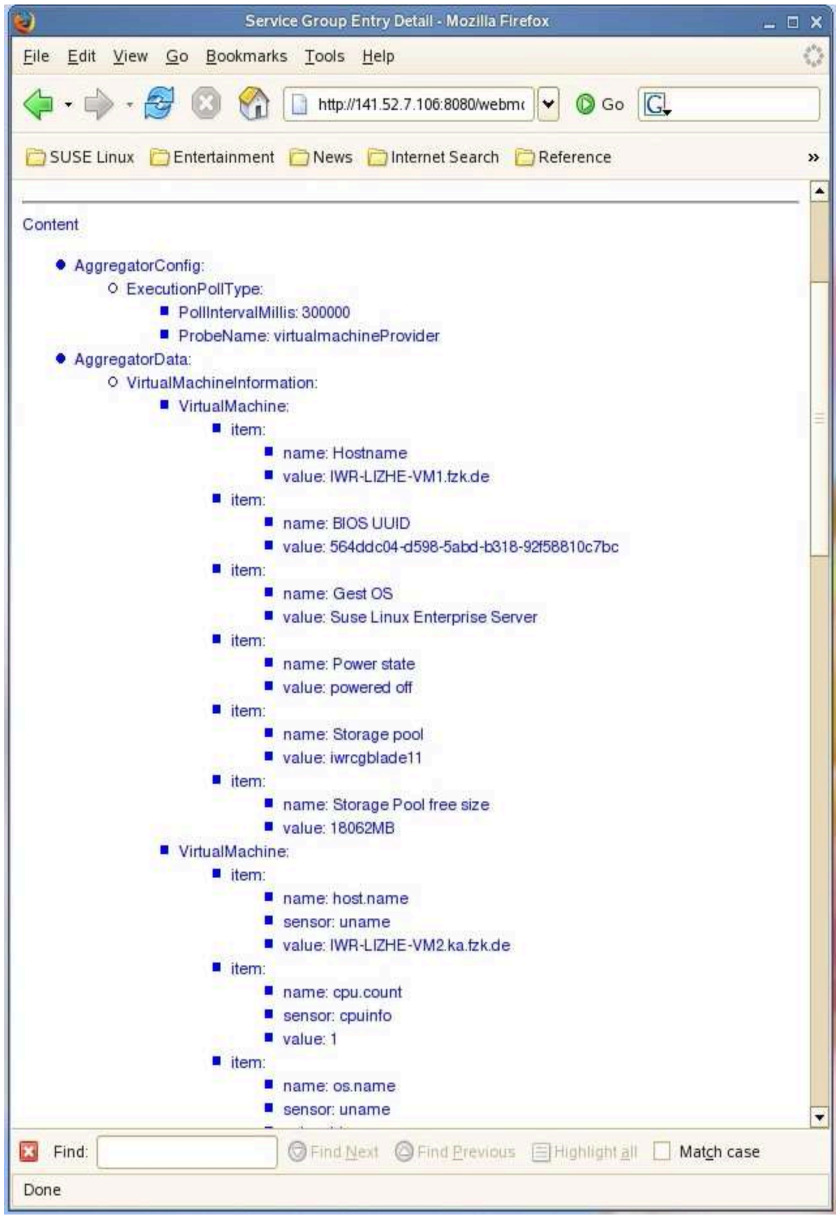


Fig. 9. Test bed


Service Group Entry Detail - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://141.52.7.106:8080/webmc
Go
SUSE Linux Entertainment News Internet Search Reference
Find: Find Next Find Previous Highlight all Match case
Done

Content

- AggregatorConfig:
  - ExecutionPollType:
    - PollIntervalMillis: 300000
    - ProbeName: virtualmachineProvider
- AggregatorData:
  - VirtualMachineInformation:
    - VirtualMachine:
      - item:
        - name: Hostname
        - value: IWR-LIZHE-VM1.fzk.de
      - item:
        - name: BIOS UUID
        - value: 564ddc04-d598-5abd-b318-92f58810c7bc
      - item:
        - name: Gest OS
        - value: Suse Linux Enterprise Server
      - item:
        - name: Power state
        - value: powered off
      - item:
        - name: Storage pool
        - value: iwrcgblade11
      - item:
        - name: Storage Pool free size
        - value: 18062MB
    - VirtualMachine:
      - item:
        - name: host.name
        - sensor: uname
        - value: IWR-LIZHE-VM2.ka.fzk.de
      - item:
        - name: cpu.count
        - sensor: cpuinfo
        - value: 1
      - item:
        - name: os.name
        - sensor: uname

Fig. 10. Resource information from virtual machines

C. Test setup

We deploy lightweight Java agents in all virtual machines of the test bed. **LZ01** and **cluct1** are employed as head nodes and installed with Globus index service, which aggregate virtual machine information from IWR/FZK and RC/UKA respectively. The WebMDS is deployed at **LZ01**.

We simulate concurrent users as follows: when the Globus index service at **LZ01** starts and retrieves resource information from virtual machines at IWR/FZK, virtual machines at RZ/UKA work as clients and run query commands of Globus index service. When the Globus index service at **cluct1**, virtual machines at IWR/FZK work as clients and run query commands of Globus index service.

D. Test results and discussion

1) Usage of information service: The access point runs an information provider to retrieve information from virtual machines via lightweight Java agent and VMware ESX server SDK respectively. The information provider thus furnishes the organized information to the back end of Globus index service. Users can retrieve the information with client programs of Globus index service or browse the information via WebMDS. Fig. 10 shows the resource information retrieved from virtual machine pool with WebMDS on **LZ01**. These results justify the prototype implemented on the test bed.

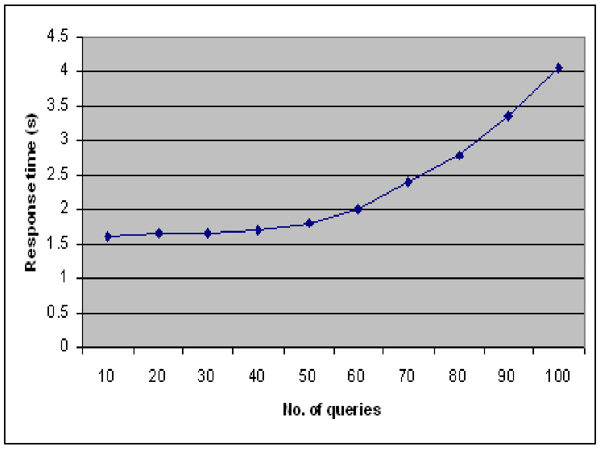


Fig. 11. Response time of information service

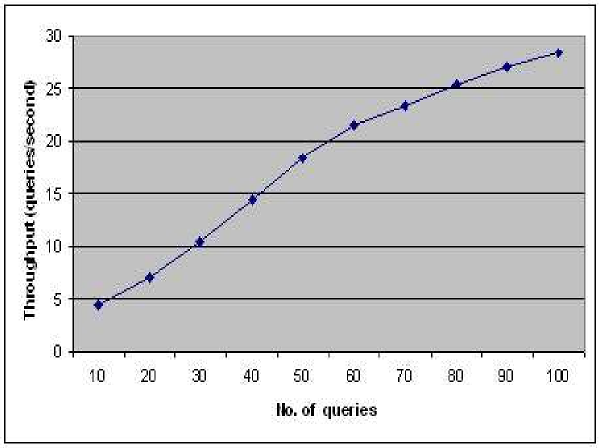


Fig. 12. Throughput of information service

2) Scalability with number of queries: Figure 11 shows the response time with number of concurrent queries. It could be concluded from Figure 11 that when fewer concurrent queries come to the information service, e.g., less than 60 queries, the response time is less 2 seconds. In this context, the information service does not reach its maximum processing ability. When concurrent queries are more than 60, the response time increases quickly. In this scenario the maximum processing ability of information service is accessed and the coming queries are queued in the information server

3) Efficiency of information service: Figure 12 shows the throughput of information service with number of concurrent queries. We can see when concurrent query number is less than 60, the throughput of information service increases in a linear style. The concurrent query number is more than 60, the throughput of information service increases less slowly and it is expected to reach its maximum when around more than concurrent 100 queries come.

The explanation for above phenomena shown in Section VII-D.2 and Section VII-D.3 could be identified as follows:

- Local data cache of information agent plays an important role. When concurrent query number is less than 60, most of queries could be returned by querying on local data cache. When concurrent query number is more than 60, more queried information items are missed from the local data cache.
- Network traffic is another impact factor the performance of information service. We believe that when When concurrent query number is more than 60, the network queue of server side is full and the information service reaches its maximum processing ability.

4) Discussion of the design pattern: The information service is designed and implemented in a hierarchical and distributed pattern. The low level is lightweight Java agent: the executable is only 4K bytes and consumes 1.64 MB memory at runtime. The lightweight Java agent also give a open interface of information providers, who can implement their own java agent and provide customized resource information with personalized information schema.

VIII. Conclusion and future work

Virtual machines are widely accepted in computer centers to support various applications. This paper implements an information service of virtual machine pools in a computer center for Grid computing. The information service can monitor virtual machines backed by popular VMMs, such as Xen VMM, VMware server and VMware ESX server. Our work distinguishes itself from related work in that:

- The information service is implemented in lightweight: CIMON agent and Java agent are developed to fetch the resource information of distributed virtual machines instead of installing Grid middleware in virtual machines.
- The information service is implemented in a hierarchical flavor. The lower level of Information agent gives opportunities to information users to deliver customized resource information;
- The information service is satisfactory considering performance metrics: throughput and response time.

Later work will developed in large scaled test bed, for example D-Grid [38] test bed. The implementation of information of the information service should match to the horizontal D-Grid information infrastructure, which includes multiple Grid middleware: Globus Toolkit [9], Unicore [39] and gLite [40].

REFERENCES

[1] I. Foster, C. Kesselman, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Technical report, Open Grid Service Infrastructure Work Group, Global Grid Forum, June 2002.
[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization . In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP)*, pages 164–177, New York, USA, Oct. 2003. ACM Press.
[3] VMware virtualization products. http://www.vmware.com/, access on Jan. 2008.
[4] Common Information Model (CIM). http://www.dmtf.org/standards/cim/, access on Jan. 2008.
[5] Open Grid Forum. Glue schema v1.3, Jan 2006.
[6] WebMDS. http://www.globus.org/toolkit/docs/4.0/info/webmds/, access on Jan. 2008.
[7] User Mode Linux. http://user-mode-linux.sourceforge.net/, access on Jan. 2008.
[8] K. Keahey, I. Foster, T. Freeman, X. Zhang, and D. Galron. Virtual Workspaces in the Grid. *Proceedings 11th International Euro-Par Conference, LNCS*, 3648:421–431, 2005.
[9] Globus Toolkit. http://www.globus.org/toolkit/, access on Jan. 2008.
[10] V. Buege, Y. Kemp, M. Kunze, O. Oberst, and G. Quast. Virtualizing a Batch Queueing System at a University Grid Center. *Proceeding of Workshop on XEN in HPC Cluster and Grid Computing Environments (XHPC), LNCS*, 4331:Italy, 397-406 2006.
[11] S. Childs, B. Coghlan, and J. McCandless. GridBuilder: A Tool for Creating Virtual Grid Testbeds. In *Proceedings of 2nd IEEE Conference on eScience and Grid computing (e-Science)*, pages 77–77, Amsterdam, Netherlands, Dec. 2006. IEEE Computer Society.
[12] S. Childs, B. Coghlan, D. O'Callaghan, G. Quigley, and J. Walsh. A Single-computer Grid Gateway Using Virtual Machines. In *Proceedings of the 19th International Conference on Advanced Information Networking and Applications*, pages 310 –315, Washington, DC, USA, 2005. IEEE Computer Society.
[13] M. Tatezono, N. Maruyama, and S.Matsuoka. Making Wide-Area, Multi-Site MPI Feasible Using Xen VM. *Proceeding of Workshop on XEN in HPC Cluster and Grid Computing Environments (XHPC), LNCS*, 4331:387–396, 2006.
[14] F. Travostino, P. Daspit, L. Gommans, C. Jog, C. de Laat, J. Mambretti, I. Monga, B. van Oudenaarde, S. Raghunath, and P. Wang. Seamless Live Migration of Virtual Machines over the MAN/WAN. *Future Generations Computer Systems*, 22:901–907, 2006.
[15] N. Fallenbeck, H. J. Picht, M. Smith, and B. Freisleben. Xen and the art of cluster scheduling. In *Proc. of 1st International Workshop on Virtualization Technology in Distributed Computing*, USA, Nov. 2006. IEEE Computer Society.
[16] P. Ruth, X. Jiang, D. Xu, and S. Goasguen. Towards Virtual Distributed Environments in a Shared Infrustructure. *IEEE Computer*, 38(5):63–69, 2005.
[17] S. Adabala, V. Chadha, P. Chawla, R. Figueiredo, J. Fortes, I. Krsul, A. Matsunaga, M. Tsugawa, J. Zhang, M. Zhao, L. Zhu, and X. Zhu. From Virtualized Resources to Virtual Computing Grids: the In-VIGO System. *Future Generation Computing Systems*, 21(6):896–909, June 2005.
[18] D. E. Irwin, J. S. Chase, L. E. Grit, A. R. Yumerefendi, D. Becker, and K. Yocum. Sharing networked resources with brokered leases. In *Proceedings of the 2006 USENIX Annual Technical Conference*, pages 199–212, 2006.
[19] J. S. Chase, D. E. Irwin, L. E. Grit, J. D. Moore, and S. Sprenkle. Dynamic virtual clusters in a grid site manager. In *Proceedings of the 12th International Symposium on High Performance Distributed Computing*, pages 90–103, 2003.
[20] Amazon Elastic Compute Cloud [URL]. http://aws.amazon.com/ec2, access on Nov. 2007.
[21] A. I. Sundararaj and P. A. Dinda. Towards virtual networks for virtual machine grid computing. In *Virtual Machine Research and Technology Symposium*, pages 177–190, 2004.
[22] A. Shoykhet, J. Lange, and P. Dinda. Virtuoso: A system for virtual machine marketplaces. Technical Report NWU-CS-04-39, Northwest University, July 2004.
[23] University of Cambridge Computer Laboratory. Performance comparison of vmms. available from http://www.cl.cam.ac.uk/research/srg/netos/xen/performance.html/, access on Jan. 2008.
[24] S. Soltesz, M. E. Fiuczynski, L. Peterson, M. McCabe, and J. Matthews. Virtual Doppelganger: On the Performance, Isolation, and Scalability of Para- and Paene- Virtualized Systems. available from http://www.cs.princeton.edu/∽mef/research/paenevirtualization.pdf, Nov. 2005.
[25] P. Strazdins, R. Alexander, and D. Barr. Performance Enhancement of SMP Clusters with Multiple Network Interfaces Using Virtualization. *Proceeding of Workshop on XEN in HPC Cluster and Grid Computing Environments (XHPC), LNCS*, 4331:452–463, 2006.
[26] W. Jie, W. Cai, L. Wang, and R. Procter. A secure information service for monitoring large scale grids. *Parallel Computing*, 33(7-8):572–591, 2007.
[27] MDS. http://www.globus.org/toolkit/mds, access on Jan. 2008.
[28] Dsitributed Management Task Force. http://www.dmtf.org/, access on Jan. 2008.
[29] Pegasus CIMOM. http://www.openpegasus.org/, access on Jan. 2008.
[30] Storage Management Initiative Specification. http://www.snia.org/smi/tech_activities/smi_spec_pr/spec/, access on Jan. 2008.
[31] Open Grid Forum. Experiences from interoperation scenarios in production grids, Feb. 2007.
[32] Enabling Grids for E-sciencE (EGEE) Project. http://www.eu-egee.org/, access on Jan. 2008.
[33] Open Science Grid (OSG) Project. http://www.opensciencegrid.org/, access on Jan. 2008.
[34] Teragrid. http://www.teragrid.org/, access on Jan. 2008.
[35] Globus Aggregator Framework. http://www.globus.org/toolkit/docs/4.0/info/key-index.html/, access on Jan. 2008.
[36] Java Architecture for XML Binding (JAXB). http://java.sun.com/developer/technicalarticles/webservices/jaxb/, access on Jan. 2008.
[37] Steinbuch Centre for Computing. http://www.rz.uni-karlsruhe.de/rz/scc/, access on Nov. 2007.
[38] D-Grid Project. http://www.d-grid.org/, access on Jan. 2008.
[39] Unicore Project. http://www.unicore.org/, access on Jan. 2008.
[40] gLite Project. http://glite.web.cern.ch/glite/, access on Jan. 2008.