Provide Virtual Distributed Environments for Grid Computing on Demand

Lizhe Wang[†], Gregor von Laszewski[¶], Marcel Kunze[‡], Jie Tao[‡], Jai Dayal[†]

† *Service Oriented Cyberinfrastructure Lab, Rochester Institute of Technology, Rochester, NY 14623*

¶*Pervasive Technology Institute, Indiana University at Bloomington, Bloomington, IN 47408*

‡ *Steinbuch Center for Computing, Research Center Karlsruhe, Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany*

Abstract

Grid users always expect to meet some challenges to employ Grid resources, such as customized computing environment and QoS support. In this paper, we propose a new methodology for Grid computing – to use virtual machines as computing resources and provide Virtual Distributed Environments (VDE) for Grid users. It is declared that employing virtual environment for Grid computing can bring various advantages, for instance, computing environment customization, QoS guarantee and easy management. A light weight Grid middleware, Grid Virtualization Engine, is developed accordingly to provide functions of building virtual environment for Grids. We also present a typical use case, on-demand build a virtual e-Science infrastructure to justify the methodology.

Key words: Virtual Environment, Virtual Machine, Grid Computing

1 Introduction

1.1 Identify current issues of Grid computing

Current Grid users employ Grid resources by submitting jobs to remote computers. Grid middleware, operating systems, and software packages & libraries compose together the computing environments for Grid users. It is can be concluded that user computing environments are directly plugged into the Grid resources. Great burdens are afforded to Grid resource providers and Grid middleware:

- Every time a job is submitted, user authentication and authorization has to be carried out. To reach single sign-on, Grid middleware has to provide complex policies and schemes for security control and user delegation.
- Resource management becomes a bottleneck for the Grid middleware. Huge efforts have been to put into manage various types of resources to fulfill different kinds of application domains.

A lot of work on security control and resource management, which is application specific, has to be mapped to heterogeneous Grid resources. For example, a workflow application [3] for sure requires different resource management functions from a parameter sweep application [14]. An community-centric Grid application [9] certainly has a security control scheme distinct with a data-centric application [10]. These functions, although should be decided and deployed in the user level, are implemented however inside Grid middleware.

Grid users can expect various disadvantages with the above methodology:

- Users expect customized computing environments with special software and hardware configuration or the abilities to configure such environments. However multiple Grid users share the same resources, it is therefore hard to balance different users' requirements. Even a Grid user monopolize the resource, it is generally impossible to offer the user with administrative privileges to configure the resource.
- Grid users normally require performance guarantees for executing their applications, e.g., CPU bandwidth, memory allocation. In the traditional multi-programmed computing model, multiple users share the same resource with local users of Grid site. Grid users have to suffer from the performance fluctuation when executing jobs on Grid resources.

We therefore identify the reasons that bring above embarrassments:

- Grid middleware provides too many functionalities, most of which should be moved to and implemented in the Grid user level. We recognize that Grid middleware is only responsible of providing basic functionalities of resource provision, information provision and security control.
- The user computing environments are directly interposed into the operating system of Grid resources. Therefore a number of interfaces between Grid resources and users' environments give into birth. As the user computing environment is attached to the operating system of Grid resource, customization and performance isolation demand more work.
- Users harness Grid resources with a fine granularity: "job" or "process". Therefore, every operation on this fine granularity would invoke a set of functions of Grid middleware. Furthermore, it is hard to provide customized environment and guarantee performance with fine granularity.

In general, Grid users can benefit from the virtual machines in the following aspects [7]:

- Performance isolation
  Virtual machines can guarantee the performance for users and applications. Applications executed in virtual machines will not find the performance perturbation, which is invoked by competition of concurrent processes on traditional multi-programmed machines.
- On-demand creation and customization
  Users can create and customize a virtual machine, which can provide desired resource allocation for users, e.g., operating system, memory, storage, etc.
- Legacy system support
  As virtual machine can be on-demand created, virtual machine thus can support entire legacy environments, such as hardware, operating system, and software libraries.
- Administration privileges
  Users of virtual machines can gain the “root” privilege because each user of the hosting resources are allocated with a virtual machine. This alleviates the task of system administrator and gives the flexibility of application users.
- Resource control
  One virtual machine can be allocated to one user or one application, it is thus easy to account and control the resource usage.

Grid computing research community recently shows interests in virtual machines and virtual computing environments. Some research work focuses on deploying computing systems or testbeds with virtual machines, for example, virtualization in a batch system [2], GridBuilder [4], virtual machine based Grid gateway [5], Xen Grid Engine [6], and OpenNebula [19]. Above systems are implemented in a cluster scale or a LAN scale, while our work of Grid Virtualization Engine is implemented in large scale distributed Grids.

Globus virtual workspace and Nimbus [20], [8] provide a set of Globus Toolkit services for virtual machine provision and managemnet. The implementation is based on Globus Toolkit version 4 and it only supports Xen VMM. We build our virtual workflow system with standard Web service technologies, such as XML, SOAP and HTTP, and it can support both Xen and VMware virtual machine. Therefore the virtual workflow system can enjoy various advantages of Web services framework, for example, scalability, interoperability, legacy application support, and underlying platform independence.

Amazon Elastic Compute Cloud (Amazon EC2) [15] is a Web service that provides resizable compute capacity with virtual machines. It is designed to make web-scale computing easier for developers. Eucalyptus [16] from UCSB is an open-source software infrastructure for implementing "cloud computing" on clusters. Amazon EC2 and Eucalyptus employ Web service technologies and provide virtual machine operations in a large scale distributed environments. However, they do not aim to work on existing Grid infrastructures and application level systems, such as Grid workflow. Furthermore, There are still no report of successful large scale scientific applications, e.g. high energy physics, deployed on these systems. Our implementation of Grid Virtualization works on existing Grid infrastructures and adopts current Grid computing model.

In this paper, we present a paradigm which employs virtual machines as computing resources for Grid workflow applications. Furthermore, we propose building Virtual Distributed Environments (VDE) with multiple virtual machines, thus provide users a desired user computing environments. The rest of this paper is organized as follows. Section 2 proposes the philosophy of virtual environment for Grid computing. Section 3 presents a middleware which enables virtual environments for Grid computing and Section 4 discusses our experiments for testing the GVE performance. It follows Section 5, which discusses a sample Virtual Distributed Environment – the virtual e-Science infrastructure. Section 6 concludes the paper.

2 Philosophy of Virtual Environment for Grid Computing

2.1 Build multiple VDEs on shared Grid infrastructures

We firstly define the term Virtual Distributed Environment (VDE) (see also Figure 1) as follows:

- a VDE contains multiple virtual machines, which can be linked by normal networking or virtual networking;
- a VDE is installed with some network protocols or middleware for distributed system, which manage the distributed virtual machines in the user level;
- the administrator, who creates the VDE, is responsible of managing the VDE, such as job scheduling, user authentication, and data movement.

This section therefore proposes a new philosophy for Grid usage based on the VDE concept:

- Virtual machines are used as computing resources for Grid applications.

Users can on-demand build and operate virtual machines, then a Virtual Distributed Environment (VDE), which contains multiple virtual machines.
- Users submit jobs to VMs or VDEs, which are provided by remote Grid resources. Users can furthermore submit pre-configured VMs or VDEs to remote Grid resources for execution.
- The VMs and VDEs on Grid infrastructures should be managed by a light weight middleware, which only offers basic functionalities such as resource supply, information provision and security control. We implement a prototype of the lightweight middleware: Grid Virtualization Engine (GVE).

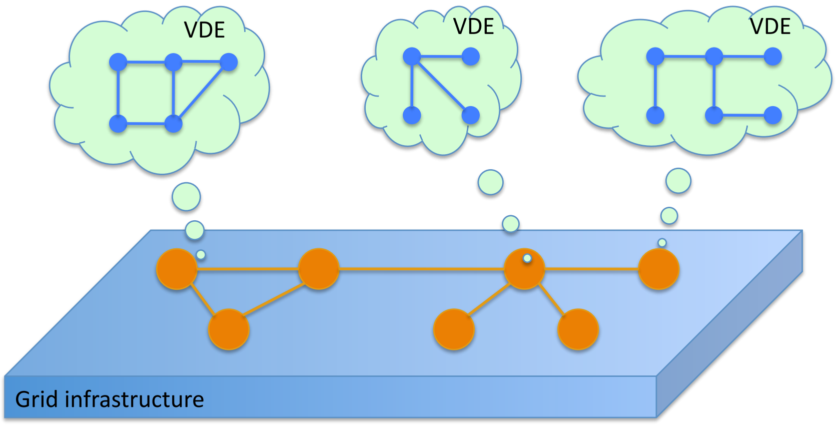

Fig. 1. Provide multiple VDEs on a shared Grid infrastructure

The new methodology of Grid usage can help solve the issues faced by Grid communities:

- Resource management now is in the coarse granularity: "virtual machine" instead of the fine granularity: "Grid job". Resource management becomes easier with coarser granularity. Users develop application specific or user community specific resource management system inside the VDE.
- Security control is based per VDE, or Virtual Organization (VO). The VO administrator authenticates itself and builds its VDE. Users thus use the VDE with the security policies defined by the VO, and they do not need to contact with GVE and Grid resources.
- Virtual machines as computing resources can offer performance guarantee and provide customized computing environments for users.

2.2 How does the virtualization occur?

Common underlying ideas of resource virtualization fall into two categories:

- Resource multiplexing
  A middleware is posed between resources and users, and it produces an illusion for each user that he monopolizes the resource without awareness of

the existence of other users. Normally a translator is needed to map user level logics to resource level stuffs. Examples can be found in Virtual Private Network (VPN), Virtual Machine (VM), and hierarchical memory management in modern operating systems. It can be summarized as "**mapping multiple users to one resource: M users → 1 resource**".
- Resource consolidation
  Resource consolidation is generally used for discrete resource management, to hide low level technical details and provide an easy-used, uniformed interface to users. Various instances can be enumerated in modern parallel/distributed computing systems, e.g. a cluster operating system. It can be summarized as "**mapping one user to multiple resources: 1 user → M resources**".

The methodology of building multiple VDEs on a shared Grid infrastructure paves a further step and combines the ideas of **resource multiplexing** and **resource consolidation** together. Access distributed Grid resources via Grid middleware can be deemed as **resource consolidation**. To build multiple VDEs on shared Grid infrastructure via Grid Virtualization can be considered as **resource multiplexing**. Therefore a novel usage methodology of Grid computing is proposed: **resource consolidation + resource multiplexing**, in short: "**M users → M resources**".

3 Grid Virtualization Engine: a Light Weight Grid middleware

3.1 Overview

To build multiple VDEs on Grid infrastructures, a light weight middleware, Grid Virtualization Engine (GVE) is designed and implemented in the is this work. The GVE aims to provide users with following functions:

- users can remotely create, operate and configure virtual machines;
- users can create and customize a VDE with multiple virtual machines on wide area networking

In detail, the GVE offers following functions:

- VM Requirement
  · Create a new VM, and
  · Require an existing VM;
- VM Operation
  · Start/Shutdown/suspend a VM,
  · Clone a VM,

  · Run a script in a VM,
  · Copy files from/to a VM, and
  · VM configuration by run scripts inside VMs.
- VDE construction by on-demand creation of multiple pre-configured VMs.

The GVE implements standard Web service interface that provides various functionalities.

3.2 Target system model

This section defines the Grid system model, which contains distributed sites interconnected by networking.

Each site consists following levels logically:

- The **computer site** provides an access service which allows remote users to access resources of the computer center. The access service can be offered by existing Grid middleware, a portal, Web services, or any functionalities that support remote steering. Grid Virtualization Engine is developed and integrated in this level.
- In the middle level exist **virtual machine**s that are backed by host resources. These virtual machines form VDEs. Grid Virtualization Engine operates virtual machines in this level.
- The fabric level contains various **host resource**s or servers, which are installed with virtual machine hypervisors. Host resources offer multiple virtual machines. Operations of Grid Virtualization Engine is implemented in this level with aids of VMM APIs and SDKs.

3.3 Implementation

The GVE is a software layer on distributed host resources and it offers on-demand provision of virtual machines, virtual networks, and VDEs. Virtualization Service is implemented in distributed and hierarchical favors with standard Web service. Current implementation of the Virtualization Service can work on popular VMMs, Xen center and VMware ESX server. The GVE (see also Figure 2) contains following components: Site Service and the Virtualization Agent (VA) .

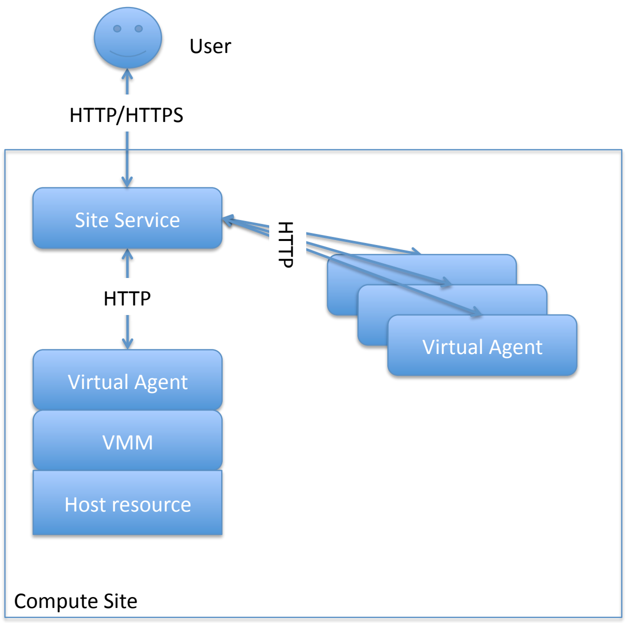

Fig. 2. Overview of Grid Virtualization Engine

3.3.1 Site Service

The Site Service resides on the access point of a computer center and supports the functions of VM or VDE provision from the host resources inside the computer center. A Site Service controls multiple underlying host resources by communicating with the Virtualization Agents.

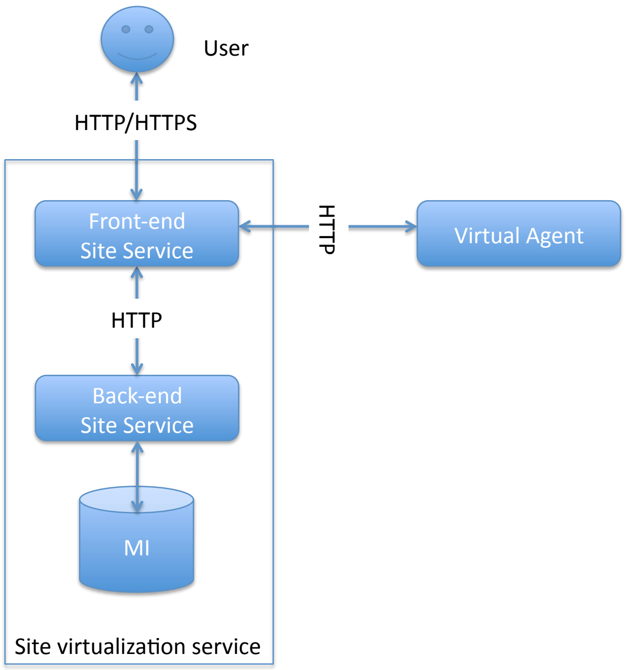

Fig. 3. Site Service

The Site Service consists two components:

- Front-end Site Service
  The Front-end Site Service is responsible of the business logic of the Site Service. It accepts requirements from users and contacts the underlying Virtualization Agents for virtual machine operation.
- Back-end Site Service
  The Back-end Site Service needs to access the Management Information Database for virtual machines manipulation. The Back-end Service thus is built to help the Front-end Site Service to access the Management Information Database.

The Management Information (MI) Database stores the following information:

- management policies for users to access the virtual machines inside the computer center.
- current virtual machine allocation for users, e.g., virtual machine ID and duration,
- virtual machine information, such as virtual machine profiles and states,
- the underlying Virtualization Agents which have registered themselves on the Front-end Service.

3.3.2 Virtualization Agent

On each host resource exists a Virtualization Agent (VA), which gets commands from the Site Service and operates on virtual machines.

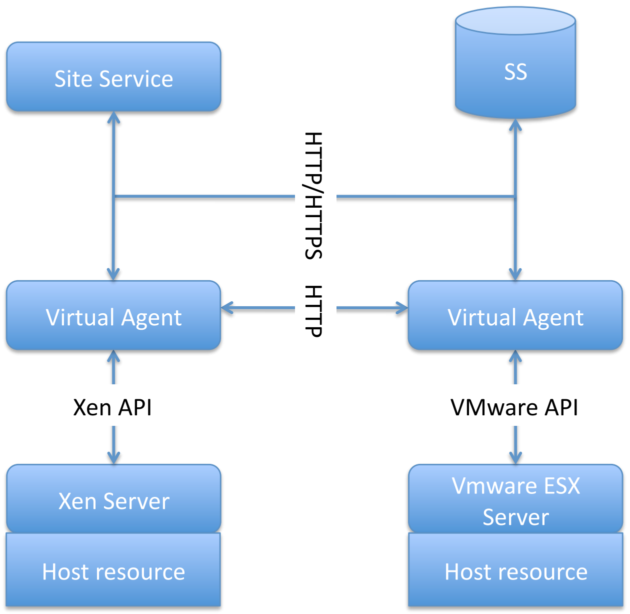

Fig. 4. Virtualization Agent

The Virtualization Agent is a Web service which runs on host resources. It receives operation commands from the Site Service and talks with the specific VMM, e.g., VMware ESX server, and Xen server, which are installed on host resources. The Virtualization Agent is VMM dependent. In other words, for each type of VMM, a correspondent Virtualization Agent should be implemented. In the thesis, two types of Virtualization Agents are implemented for VMware ESX server and Xen server respectively. Software Stack (SS) database is required when Virtualization Agent demands the VMM to create new virtual machines. The database of Software Stack contains following information:

- virtual machine disk images used when virtual machine is created, and
- various of software packages to build a pre-configured virtual machines.

3.4 Discussion

The GVE distinguishes itself from related work [1,11,12] in that:

- The GVE is designed and implemented in modularity. System components are wrapped with standard Web service interfaces. The modular design philosophy brings advantages such as scalability, availability and interoperability to the system.
- The GVE is designed and implemented in the hierarchical flavor. The higher level service provides general interface, which is VM technology independent; the low level service handles VM specific implementations. The hierarchical design pattern makes the system more scalable to incorporate new VM technologies.

4 Experiments and discussion

This section introduces our test experiment to test the Grid Virtualization Engine. The first experiment is to test how much overhead that the GVE introduces for a virtual machine operation, for example, start a virtual machine. We measure the time form when a user issues a command to to start a virtual machine to the time when the virtual machine instance is available for use.

We use the Condor virtual machine images from NSF funded Grid Appliance (http://www.grid-appliance.org/). The experimented virtual machine image is configured with 100 GB hard disk and 512 MB RAM. Figure 5 shows the time for virtual machine instance startup at various scenarios: start one instance of virtual machine both locally and with the GVE, simultaneously start 2, 4 and 8 instances of virtual machine with GVE respectively.

We can see that the GVE does not introduce much overhead in term of virtual machine startup. In Figure 5 when 8 tiny Linux virtual machine instances are started simultaneously, the max overhead of starting time is around 17%.

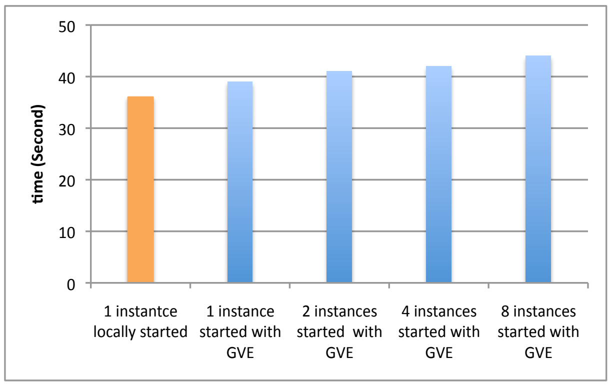

Fig. 5. GVE instance start overhead

Another experiment measures the communication overhead between virtual machines. The GVE does not provide virtual network solutions. The virtual machine that is managed by GVE uses native virtual network solutions provided by Xen server or VMware ESX server. Normally a virtual machine uses a virtual network interface and is assigned with an IP address. We run a MPI ping-pong program between virtual machines to test the network performance. To make a comparison, MPI ping-pong program is executed on real machines. The VMM used in this experiment is VMware ESX server 3.5. In Figure 6, we can see that the throughputs between virtual machines can reach around 90% of those between real machines when message sizes are big enough.

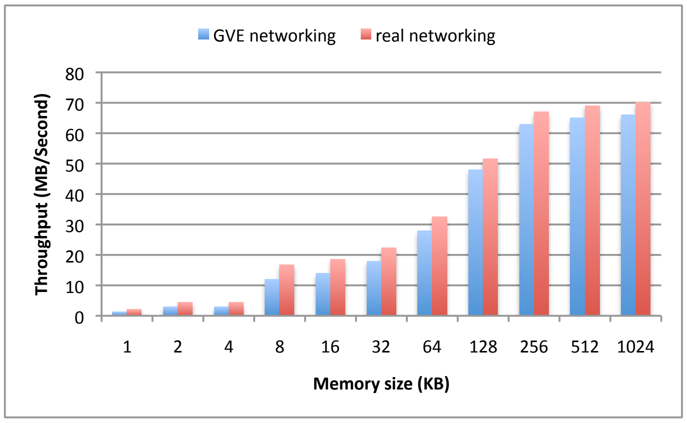

Fig. 6. GVE network performance

5 Build a Virtual e-Science Infrastructure at runtime: a Sample Use Scenario

GridSAM [13] is a standard Job Submission and Monitoring Web Service that provides a common interface to a variety of DRMs (Distributed Resource Managements), which is developed with widely accepted and standardized Web Service specifications and related technologies. ActiveBPEL engine [18] is a modeling, monitoring and execution environment for scientific workflows based on the Business Process Execution Language (BPEL) [17]. A typical e-Science infrastructure [18] that involves BPEL (both the BPEL script and BPEL runtime) and GridSAM [13] is as this:

- use BPEL Designer to design a BPEL process that interacts with GridSAM's job submission service port and job monitoring service port; produce the deployment archive by BPEL Designer at the end of the modeling;
- deploy the process onto ActiveBPEL, which is hosted in OMII Server container; from the BPEL Designer construct the request message that triggers the BPEL process;
- once got started, ActiveBPEL submits a pre-defined job in JSDL to GridSAM; GridSAM translates the JSDL script to whatever works for the underneath resource manager and sends the job to the underlying Grid computers;
- ActiveBPEL polls the job status through GridSAM's monitoring interface until the job is completed eventually.

5.1 System integration with Grid Virtualization Service

Virtual machines are employed as computing resources for workflow execution. The ActiveBPEL engine dynamically invokes the Grid Virtualization Service to request virtual machines with GridSAM pre-installation, then organize the application in workflow and submit the workflow to virtual machines via GridSAM. The integrated workflow system includes following components:

- Workflow service
  The workflow service contains a Web service as interface, which can be invoked by workflow client. The ActiveBPEL acts as workflow engine. It invokes Grid Virtualization Service to request virtual machines with GridSAM installation, then executes workflow jobs on virtual machines via GridSAM interface.
- Proxy service
  A proxy service is implemented as an interface between ActiveBPEL engine and GridSAM service.
- GridSAM service

GridSAM job submission and monitoring services are installed in the virtual machines, via which ActiveBPEL engine submits jobs to virtual machines.

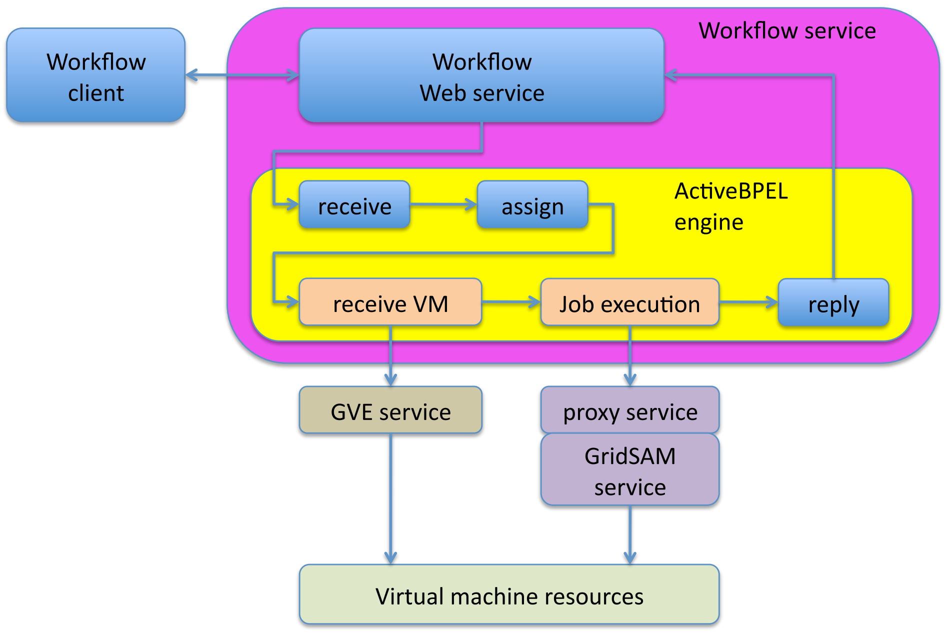

Fig. 7. Service composition of workflow system

5.2 How ActiveBPEL dynamically invokes GVE service

The ActiveBPEL engine invokes the GVE Service to get virtual machines. In order to invoke a Web service (GVE Service), the ActiveBPEL engine proceeds following steps:

- define *partnerLinkType*
  The GVE service should firstly be declared in the BPEL workflow by defining a *partnerLinkType* in the workflow Web service description file. The declaration of *partnerLinkType* describes the GVE Service to be invoked, e.g., *name*, *namespace*, and *portType*.
- define *PartnerLink*
  *PartnerLink*s describe the roles that a process or a Web service plays and the data it manipulates.
- Invoke the GVE Service operations
  The ActiveBPEL engine can therefore parse the document in other to discover all the operation provided by the GVE Service Web service with *partnerLinkType* and *PartnerLink* declarations. The input and output variables called *requestVirtualMachine* and *requestVirtualMachineResponse* are declared in the part $< bpel : variable >$ of the BPEL document. The operation of *requestVirtualMachine* is invoked by using a $< bpel : invoke >$ activity; the *partnerLink* attribute indicates which Web service is addressed.

The *portType* attribute specifies the *portType* that contains the operation invoked. The operation attribute contains de name of the invoked operation.

5.3 GridSAM proxy service: proxy for ActiveBPEL to call GridSAM service

The ActiveBPEL engine cannot dynamically invoke GridSAM Web service in that:

- As virtual machines are requested dynamically, GridSAM Web services that are installed on virtual machines only can be identified at runtime. When the ActiveBPEL engine organizes a workflow, the endpoints of Web services are not yet returned because GVE Service is not yet invoked by the ActiveBPEL engine.
- The GridSAM Web service is secured using WS-Security, which is unfortunately not supported by current release of the ActiveBPEL engine.

A proxy service is developed in the this work to overcome the above challenges as follows:

- The access to GridSAM proxy Web service is not secured using WS-security, therefore allowing non-security interaction with the ActiveBPEL engine.
- The ActiveBPEL engine does not directly invoke the GridSAM Web service. The ActiveBPEL engine invokes Grid Virtualization service then gets a set of endpoints of GridSAM Web services at runtime. It then invokes the proxy service and passes the endpoint of GridSAM Web services to be invoked as parameters. The proxy service thereafter invokes the GridSAM Web service.

6 Conclusion

Grid community currently faces some challenges like, customized and guaranteed computing environment provision, and burdensome tasks of resource management. The reasons are identified, for example, overloaded functionalities of Grid concepts and Grid middleware, and current Grid use methodology.

This paper presents the methodology of building virtual environments for Grid computing. Instead of direct providing Grid resources for Grid jobs, computational Grids provide customized virtual computing environments for Grid job execution. We reifier this methodology by developing a light weight Grid middleware, Grid Virtualization Engine, which supports virtual environments for Grid users. A typical use case: dynamically build a virtual e-Science infrastructure is discussed in the paper to justify the methodology.

During the period of project development, the concept of "Infrastructure as a Service (IaaS)" for Cloud computing is proposed. The idea of IaaS can implemented by providing virtual machines as computing resources, thereafter building customized computing environments for Cloud computing. Our work can be integrated into the Cloud computing context and on-demand provides computing infrastructures for users.

References

[1] S. Adabala, V. Chadha, P. Chawla, R. Figueiredo, J. Fortes, I. Krsul, A. Matsunaga, M. Tsugawa, J. Zhang, Mi. Zhao, L. Zhu, and X. Zhu. From virtualized resources to virtual computing Grids: the In-VIGO system. *Future Generation Comp. Syst.*, 21(6):896–909, 2005.

[2] 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.

[3] J. Cao, S. A. Jarvis, S. Saini, and G. R. Nudd. GridFlow: workflow management for Grid computing. In *Proceedings of the 3rd IEEE International Symposium on Cluster Computing and the Grid*, pages 198–205, 2003.

[4] 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.

[5] 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.

[6] 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.

[7] R. J. O. Figueiredo, P. A. Dinda, and J. A. B. Fortes. A case for Grid computing on virtual machine. In *Proceedings of the 23rd International Conference on Distributed Computing Systems*, pages 550–559, 2003.

[8] K. Keahey, I. Foster, T. Freeman, and X. Zhang. Virtual workspaces: achieving quality of service and quality of life in the Grid. *Scientific Programming*, 13(4):265–275, 2005.

[9] M. Mowafi, N. Jiang, T. Caudell, W. Shu, and M. Wu. Real-time transmission of stereo images over the access Grid. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 633–636, 2002.

[10] M. Russell, G. Allen, G. Daues, I. Foster, E.Seidel, J. Novotny, J. Shalf, and G. v. Laszewski. Wrong label. the astrophysics simulation collaboratory: a science portal enabling community software development. *Cluster Computing*, 5(3):297–304, 2002.

[11] P. Ruth, X. Jiang, D. Xu, and S. Goasguen. Virtual distributed environments in a shared infrastructure. *IEEE Computer*, 38(5):63–69, 2005.

[12] A. Shoykhet, J. Lange, and P. Dinda. Virtuoso: a system for virtual machine marketplaces. Technical Report NWU-CS-04-39, Northwest University, July 2004.

[13] GridSAM: Grid Job Submission and Monitoring Web Service [URL]. http://gridsam.sourceforge.net/, access on Nov. 2007.

[14] W. Sudholt, K. Baldridge, D. Abramson, C. Enticott, S. Garic, C. Kondric, and D. Nguyen. Application of grid computing to parameter sweeps and optimizations in molecular modeling. *Future Generation Comp. Syst.*, 21(1):27–35, 2005.

[15] Amazon Elastic Compute Cloud [URL]. http://aws.amazon.com/ec2/, access on Nov. 2008.

[16] Eucalyptus Project [URL]. http://eucalyptus.cs.ucsb.edu/, access on Sep. 2008.

[17] OASIS
Web Services Business Process Execution Language [URL]. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel/, access on Nov. 2008.

[18] OMII-BPEL [URL]. http://sse.cs.ucl.ac.uk/omii-bpel/, access on Nov. 2008.

[19] OpenNEbula Project [URL]. http://www.opennebula.org/.

[20] Globus virtual workspace interface guide [URL]. http://workspace.globus.org/vm/tp1.3/interfaces/index.html/, access on Nov. 2008.