Virtual Data System on Distributed Virtual Machines in Computational Grids

Lizhe Wang* and Gregor von Laszewski
Pervasive Institute of Technology, Indiana University, U.S.
E-mail: {Lizhe.Wang, Laszewski}@gmail.com
*Corresponding author

Jie Tao and Marcel Kunze
Steinbuch Centre for Computing, Karlsruhe Institute of Technology, Germany
E-mail: {Jie.Tao, Marcel.Kunze}@kit.edu

Abstract: This paper presents the work of building a Grid workflow system on distributed virtual machines. A Grid Virtualization Engine (GVE) is implemented to manage virtual machines as computing resources for Grid applications. The Virtual Data System (VDS) functions as a Grid workflow engine. This paper designs and implements the Virtual Data System on distributed virtual machines, which are enabled by the Grid Virtualization Engine. Various interfaces between GVE and VDS have been defined and implemented. A high energy physics application, the Compact Muon Solenoid (CMS) benchmark, is tested on a test bed equipped with the developed technologies. System discussion, test results and performance evaluation justify the paradigm of building VDS on distributed virtual machines and the prototype of our design and implementation.

Keywords: Grid computing; high energy physics application; Grid workflow system; virtual machine

1 Introduction

Grid computing technology [1] offers effective solutions for parallel and distributed applications. It can provide reliable, collaborative and secure access to remote computational, data and scientific instrument resources. Distinguished from conventional parallel and distributed computing, Grid computing focuses on resource sharing among geographically distributed sites and the development of innovative, high-performance applications. A computational Grid can present users with pervasive and inexpensive access to a wide variety of resources.

A Grid workflow system [2] can help to model complex scientific and business applications and execute them on computational Grids. The general scenario could be specified as follows: an inexperienced Grid user specifies the complex application in term of workflow with high level specification languages. A workflow management system converts the abstract workflow to concrete Grid jobs and automatically allocates resources for the workflow. Then the Grid user can interact with applications and achieve results with various methods, e.g. graphical user interfaces or Grid portals.

Although great advances have been made in the field of Grid computing, users are still expected to meet some difficulties of employing Grid resources. The most important research issue of Grid computing Qualities of Service (QoS) provision. Computational Grid is a highly dynamic environment because resource capacities and access interfaces may change from time to time. Furthermore, remote Grid users and local resource consumers compete for Grid resources like CPU, memory, etc. Therefore resources shared on computational Grids in general cannot guarantee QoS of resource provision. An example of QoS provision at the Steinbuch Centre for Computing (SCC) of the Research Center of Karlsruhe (FZK) is shown as follows. The researchers there organized a Grid computing center (GridKa) for a large number of high energy physics applications like the ATLAS experiment for the Large Hadron Collider (LHC) at CERN (European Organization for Nuclear Research) as well as the BaBar experiment from the Stanford Linear Accelerator Center (SLAC). These high energy physics jobs are executed with QoS requirements, e.g., data transfer for these jobs should be at least 10MB/s between computer centers, and the job turn-around time could not exceed 12 hours. These jobs demand QoS guaranteed environments which are isolated from other applications ensuring that their performance would remain unaffected.

Another important research focus of Grid computing is to provide customized runtime environment for Grid applications. In general Grid applications demand customized execution environments such as operating system, software packages & libraries, and network configurations. It is painful, and sometimes impossible, for Grid resources to provide different runtime environments for multiple Grid applications. Some requirements of these runtime configurations require administration privileges. It is thus unfeasible for normal Grid users to configure their own runtime environments on Grid resources. For example, a Compact Muon Solenoid (CMS) experiment job from CERN needs, for example, a Scientific Linux operating system and a set of CMS software libraries for execution. To build such a software environment for a CMS job costs more than 4 hours for an expert. A computer center has to establish a staff group to build and maintain various computing environments for numerous high energy physics applications.

Virtualization [3] is the process of presenting a logical grouping or subset of computing resources so that they can be accessed in abstract ways that give benefits over the original configuration. Virtualization technology can bring various advantages such as resource customization, performance guarantee and isolation, and easy resource management. Employing virtual machines as computing environments for Grid applications therefore might be a promising solution to address these challenges:

- Virtual machines are allocated with specified resources required by Grid applications, for example, CPU bandwidth, memory size and storage capacity. Thus QoS is guaranteed since virtual machine resources are dedicated to Grid applications during the execution.
- Various pre-configured virtual machine templates could be prepared for different Grid applications. When a Grid application arrives at a compute center, a virtual machine could be created from certain virtual machine template and started on demand. Therefore Grid applications could always find desired execution environments if customized virtual machines are prepared. Furthermore Grid users could also configure the allocated virtual machines to build desired computing environments.

This paper presents a new computing paradigm which employs virtual machines as computing resources for Grid workflow applications. A Grid Virtualization Engine (GVE) deploys virtual machines for Grid workflow. Virtual Data System (VDS) is employed as a workflow engine to organize scientific applications. This paper develops various interfaces between the VDS and the GVE, for instance, Site Catalog and Transform Catalog. The contribution of this paper is shown as follows. This paper proposes a new computing paradigm: building virtual machines and virtual distributed environments as application computing environments. The model for a virtual machine based Grid computing system is defined in this paper. The GVE presented in this paper is a scalable, hierarchical distributed middleware which enables the new computing paradigm. Various interfaces between the Grid workflow system (the Virtual Data System) and distributed virtual machines are defined and implemented in this paper. This paper also delivers application experience by performance valuation with a Compact Muon Solenoid (CMS) benchmark.

The rest of this paper is structured as follows. Section 2 introduces the background and related work. Section 3 formally defines the research issue for this paper and presents our computing paradigm. Section 4 presents the methodology of employing virtual machines as computing resources for the Grid workflow system and overviews the system architecture. Section 5 defines the system model of a virtual machine based Grid computing infrastructure. Section 6 discusses the design and implementation of the GVE. Section 7 introduces the VDS. Section 8 provides the work of building a Grid workflow system, namely the VDS, on distributed virtual machines. A discussion on the design and implementation from a system viewpoint is also presented. In Section 9, a high energy physics application, the Compact Muon Solenoid (CMS) benchmark, is employed to evaluate our implementation. Section 10 concludes the paper and outlooks future work.

2 Background and Related work

A Virtual Machine (VM) is a software artifact that executes other software in the same manner as the machine for which the software is developed and executed [4]. The software that supports multiple virtual machines on the same resource is termed as Virtual Machine Monitor (VMM) or hypervisor, for example, Xen [5] and VMware ESX server [6].

In general, users can benefit from the virtualization techniques in the following aspects. For example, users can create a customized virtual machine on demand, which can provide customized resource allocation for users, e.g., operating system, memory, storage, etc. Virtual machines can also guarantee application performance for users because virtual machines are allocated with dedicated resources. Users can create virtual machines which are compatible with legacy binary codes. The virtual machines even can demand no re-compilation and no dynamic re-linking for the legacy software. Users of hosting resources can gain the *root* privileges provided that they are allocated with virtual machines to execute applications. This alleviates the task of system administrator and gives the flexibility of application users.

Parallel 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 [7], GridBuilder [8], virtual machine based Grid gateway [9], Xen Grid Engine [10], and OpenNebula [11]. 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 [12, 13] 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) [14] is a Web service that provides resizable compute capacity with virtual machines. It is designed to make web-scale computing easier for developers. Eucalyptus [15] 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. Current Globus Toolkit and Web services cannot manage distributed virtual machines. Our implementation of GVE works on existing Grid infrastructures and adopts current Grid computing model.

Our previous work [16] has demonstrated the initial design of the Grid Virtualization Engine. In this paper, we provide a completed presentation on the Grid Virtualization Engine, including the philosophy, design pattern and system discussion. The performance of the CMS application on distributed virtual machines is firstly discussed in [17]. This paper includes a more detailed discussion on the CMS application's performance, which is enabled by the GVE. In conclusion, in this paper our work of VDS on distributed virtual machines enables a new computing philosophy on computational Grids:

- Users build a Grid system on distributed virtual machines on demand, which can offer users desired computing resources, software configuration, and user access policies.

- On the virtual machine based Grid system, users can furthermore deploy various Grid middleware and application level distributed system, for example, Globus Toolkit and Virtual Data System.

The new computing philosophy decouples Grid infrastructures and user applications, thus renders users great flexibility, guaranteed QoS and system manageability.

3 Research Problem Definition

Normally a Grid computing system consists of three parts: Grid resource, Grid middleware and Grid applications. Grid middleware is installed on Grid resources which present a Grid infrastructure. Grid application runtime environments are directly plugged into operating systems of computing resources. This use scenario thus produces a number of interfaces between Grid resources and Grid user environments, for example, process invocation, security control, information exchange. Therefore user computing environment customization and QoS provision are difficult to implement.

This paper proposes a new paradigm for Grid usage. Virtual machines are used as computing resources for Grid applications. Users can build and operate on virtual machines, thereafter a virtual distributed environment on demand, which contains multiple virtual machines. Users submit jobs to virtual machines or virtual distributed environments, which are provided by Grid infrastructures. The virtual machines and virtual distributed environments on Grid infrastructures are supported and managed by a lightweight middleware, which only offers basic functionalities such as resource supply. A prototype of the lightweight middleware – the GVE is implemented.

Provided that Grid applications are allocated with dedicated virtual machines or virtual distributed environments, QoS could be guaranteed, and customized computing environments can be found for Grid applications. This computing paradigm makes Grid resource layer separated from middleware level and user level. Users could build desired middleware and user computing environments on distributed virtual machines on demand (in this paper, Globus Toolkit and VDS are examples).

To build a virtual machine based Grid computing system, it is required to solve the research issues, such as definition of a system model for virtual machine based Grid computing, development of a middleware (in our implementation, GVE) to manage distributed virtual machines, and organization of Grid applications in a user level software framework. In this study the Virtual Data System (VDS) is used.

4 System Overview

The virtual machine based Grid workflow system contains the following sub-systems (Figure 1): Grid infrastructure, GVE and VDS on virtual machine based Grid computing systems. The proposed system offers an integrated system solution to on-demand deploy virtual machine based VDS system on Grid infrastructures. Figure 1 overviews the proposed system as follows. Grid users want to build a customized computing environment provided by Grid infrastructure. They therefore require customized virtual machines from the GVE. On recept of requirements from Grid users, the GVE manipulates virtual machines from Grid infrastructures, for example, creates and customizes

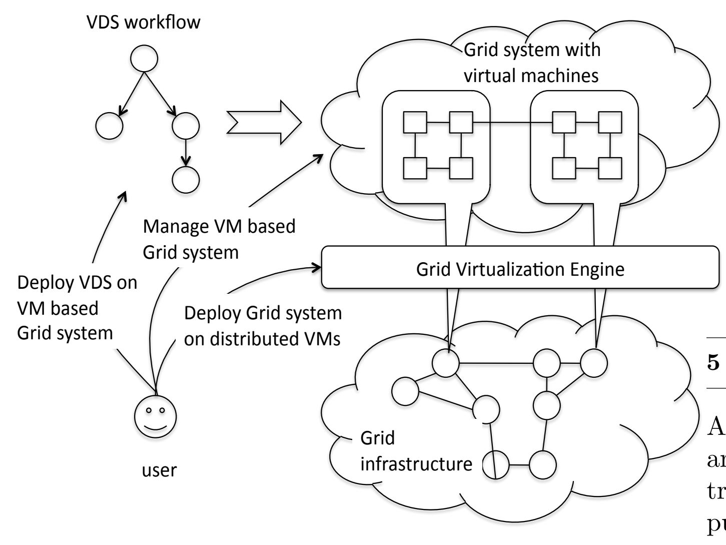
VDS workflow
Grid system with virtual machines
Manage VM based Grid system
Deploy VDS on VM based Grid system
Grid Virtualization Engine
Deploy Grid system on distributed VMs
Grid infrastructure
user

Figure 1: Overview of VDS on distributed virtual machines

virtual machines by software installation and user environment configuration. Grid users then are returned with a set of desired virtual machines. Grid users therefore organize their applications with the VDS workflow and run the application on the virtualized environment.

The Grid infrastructure contains distributed compute centers linked by high performance networking. The compute center offers numerous computer servers, which can back multiple virtual machines. Section 5 discusses the system model of a virtual machine based Grid infrastructure. The Grid infrastructure offers multiple virtual machines for Grid users.

The GVE is a software layer between various virtualization implementations, computing centers and Grid users. Users can require and employ virtual machines via an access interface of the GVE. The GVE talks with underlying computing centers and VMMs for virtual machine operations. Computer centers could also provide virtual machines to form Grid infrastructures. The employed virtualization technology may differ from one computing center to another (i.e., VMWare and XEN technologies). The GVE thus provides a standard and uniform access to virtual computing resources. Section 6 presents details of the design and implementation of the Grid Virtualization Engine. Users require virtual machines to build a Grid computing system via the help of the GVE.

The VDS [18] is a Grid workflow system. It provides a set of tools for expressing, executing, and tracking the results of workflows. Section 7 introduces details of the VDS. A Grid application is organized in a VDS workflow and executed on distributed virtual machines provided by Grid infrastructure and enabled by the GVE.

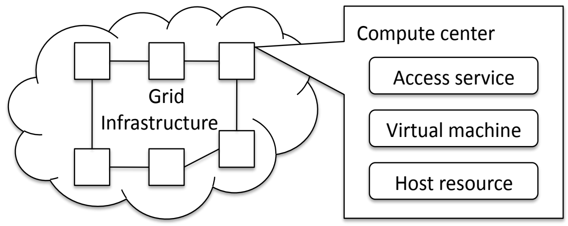

Figure 2: Virtual machine based Grid system model

5 Virtual machine based Grid infrastructure

A system model is proposed to describe a distributed, hierarchical, heterogeneous virtual machine based Grid infrastructure which contains distributed Computer Sites (Computer Centers) interconnected by networking (see also Figure 2).

Each Computer Site logically consists of the three levels. 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 functionality that supports remote steering. The GVE service is developed and integrated at this level. The middle level includes **virtual machine**s that are backed by host resources. These virtual machines form virtual distributed environmetns. The GVE service operates virtual machines at this level. The fabric level contains various **host resources** or servers, which are installed with virtual machine hypervisors. Host resources offer multiple virtual machines.

6 Grid Virtualization Engine

The GVE is designed in distributed and hierarchical favors with standard Web service interfaces. Current implementation of the GVE can work on popular VMMs, for instance, Xen server and VMware ESX server. The GVE is designed to work on the target Grid system model which is defined above. As shown in Figure 3, a GVE contains the following components: GVE Site service, GVE Agent Service, and Virtual Machine Disk Image Database. The GVE Site Service resides on the access point of the Computer Site and it operates GVE Agents on host resources to manipulate virtual machines. The GVE offers the functions for example, virtual machine requirement: create/clone a virtual machine, and virtual machine management: start/shutdown/suspend/migrate a virtual machine.

6.1 GVE Site Service

The GVE Site Service resides on the access point of a compute center. It manages host resources inside the center by communicating with the GVE Agent Services that run on the host resources. Users who want to build a Grid in-

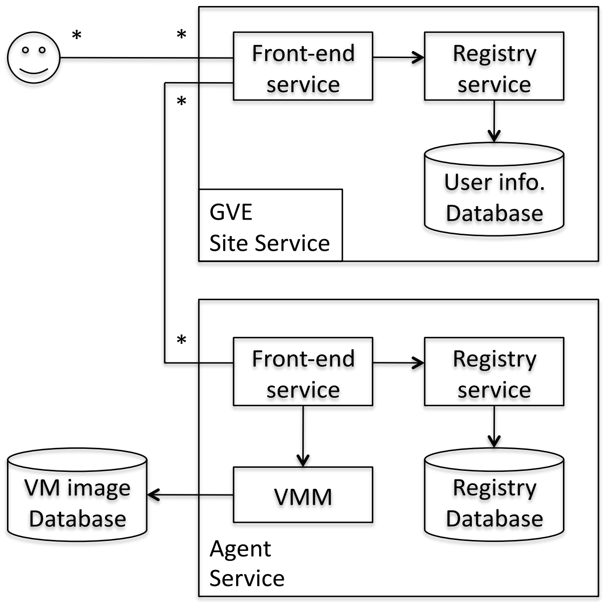

Figure 3: Grid Virtualization Engine Architecture

frastructure with distributed virtual machines can access multiple GVE Site Services for virtual machine manipulations. During the deployment process or at runtime, an administrator must explicitly define which GVE agents are connected to the GVE Site Service.

The GVE Site Service includes three main components: the GVE Front-end Service, the User Information Service and the User Information Database.

The GVE Front-end Service is responsible for the business logic of the GVE Site Service. It decides the GVE Agent Service to which the virtual machine request should be sent to. The GVE Front-end Service also defines policies for resource allocation. For the latter purpose, it needs the User Information Service which provides access to the data stored in the User Information Database.

The User Information Database records management policies for user access to the virtual machines inside the computing center. For example, the GVE checks whether the resource quota of the user has been reached before new resources are allocated. The User Information Database also stores the information about all GVE Agent Services on which the GVE Site Service is registered. The User Information Database is accessed via the User Information Service.

6.2 GVE Agent Service

The GVE Agent Service is a Web Service which runs on the host resource. It contains a GVE Front-end Service and a Registry Service on the back-end. The Front-end Service receives operation commands from a GVE Site Service and talks to the VMM on the underlying host resource. Thus the GVE Agent Service is virtualization technology dependent. In other words, for each type of VMM, a corresponding GVE Agent Service should be implemented.

All tasks related to virtual machine management from a GVE Site Service are delegated to Agent Services. A GVE Agent Service knows how to call the management functions of the underlying VMMs. All GVE Agent Services implement uniform Web Service interfaces for invocations by GVE Site Services.

The Registry Database stores various information, for example, states of virtual machines and the GVE Site information for identification. The Registry Database is accessed via the Registry Service.

6.3 Virtual Machine Image Database

The Virtual Machine Disk Database contains virtual machine disk image templates that are cloned to create new virtual machines. This Database is VMM dependent and should be developed accordingly to the VMM data models.

7 Virtual Data System

7.1 Introduction to VDS

The VDS is a Grid workflow system and it provides a set of tools for expressing, executing, and tracking the results of workflows [18]. The VDS provides a means to describe a desired data product and to produce it in the Grid environment. The VDS provides a catalog to describe a set of application programs (*transformations*) and then tracks all the data files (*derivations*) produced by executing those applications.

Figure 4 overviews a VDS. Users define a workflow with the Virtual Data Language (VDL) in a VDL text (VDLt) file. A system tool *vdlt2vdlx* converts the VDLt file into a VDL XML (VDLx) file. A Virtual Data Catalog (VDC) contains virtual data definitions. The VDC can be changed by the tools of *insertvdc* and *updatevdc* from VDLx files. Then the abstract DAG (aDAG) file, which describes the task and data in the workflow with abstract definitions, is produced by the tool *gencdax*. A concrete DAG (cDAG) file defines the workflow with real executables & data files and can be submitted to Grid resources for execution.

In order to convert the abstract DAG files to concrete DAG files by the VDS tool *gencdag*, the following information is required: the Site catalog, the Transform Catalog and the Replica Catalog. A Site Catalog (SC) records Compute Site profiles on the Grids, such as site name, *Globus gatekeeper*, and *gridftp* server. With the information from the Site Catalog, *gencdag* selects proper sites to execute the workflow. A Transform Catalog (TC) stores the mapping of logical application names to physical executables. The tool of *gencdag* translates logical application names in aDAG to real executables in cDAG. A Replica Catalog (RC) stores the information that maps logical data files to physical data files. The tool of *gencdag* converts the logical data files in aDAG to physical data files in the cDAG. When a concrete workflow is generated, it is then

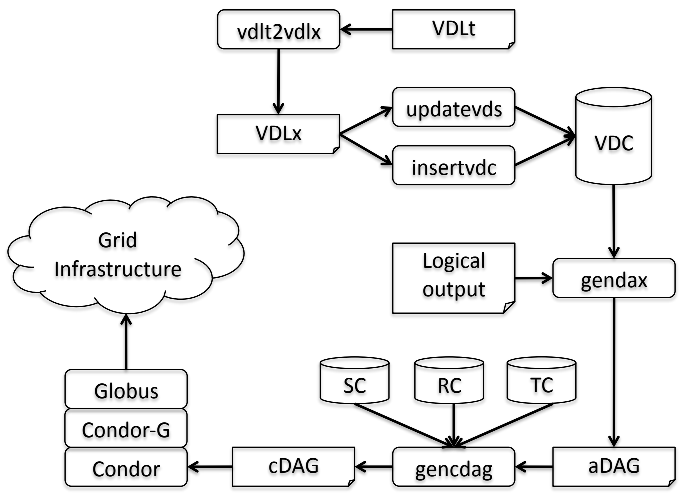

Figure 4: Architecture of Virtual Data System

submitted to computational Grids via Condor-G [19] and Globus Toolkit [20].

7.2 Grid Model for Running VDS

The Grid system where a VDS workflow is executed contains the several elements with different functionalities (Figure 5): a Catalog Host, a Submit Host, and a Compute Site. The Catalog Host (CH) contains VDC, VDS packages and DAG directories, where the VDS workflow and VDC are prepared and manipulated. The Submit Host (SH) is installed with Globus Toolkit, personal Condor and Condor-G, via which a cDAG is submitted to Grids. In our implementation, the Catalog Host is merged to the same computer as the Submit Host. The computational Grid is composed by multiple Compute Sites (CS). In the Compute Site, Globus gatekeeper is installed in a head node and can accept VDS workflows. The Compute Site contains multiple work nodes (WN).

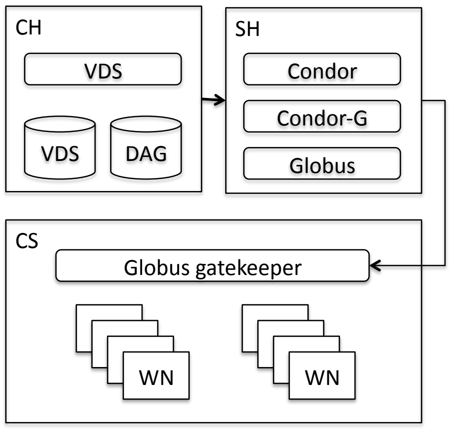

Figure 5: Grid model for running VDS

8 VDS on Distributed Virtual Machines

This Section presents how to build a VDS on distributed virtual machines. There are mainly two ways to do this. Subsection A discusses how to automatically build a VDS on distributed virtual machines from scratch with GVE. This process of building a VDS contains several stages by automatically running the installation scripts via calling the GVE Web service function *runscriptinVM* in virtual machines. Another method to build VDS on distributed virtual machines is to clone virtual machines from existing virtual machine templates, which is described in Subsection B.

8.1 Build a VDS on distributed virtual machines from scratch

8.1.1 Build a Virtual Cluster for Compute Site

To create a virtual cluster, it is required to configure a head node and several worker nodes. On the head node, Globus Toolkit and Condor master software module are installed. On the worker nodes, Condor worker modules are installed. Globus Toolkit gatekeeper in head node is used to communicate with clients at Submit Hosts, for example, by getting jobs and returning job results. Condor is used in the virtual cluster as a local batch scheduler to schedule jobs across multiple work nodes. The following processes are required to build the virtual cluster:

To configure a head node a virtual machine with the Linux operating system is required. An XML file in GLUE schema [21] is used to describe the virtual machine. Users can invoke GVE Web service function *requestVM*, which takes an XML file as a parameter and returns a virtual machine to users.

Now with a virtual machine on hand, users can install Globus Toolkit and Condor master module to build a cluster head node.

Before the installation, it is demanded to stage the required software, Globus Toolkit and Condor, from the ftp server to the virtual machine by calling GVE Web service function *copyfiletoVM*.

The installation is carried automatically by invoking GVE function *RunScriptInVM*. Some configuration work could not be automatically processed and human interaction is still required. For example, the Globus Toolkit *gatekeepter* configuration is not only a technical work, but also a management task. The German national Grid project – D-Grid requires Globus Toolkit *gatekeepter* certificates signed by Grid Center of Karlsruhe (GriKa). The process of requesting and signing a certificate is a manual work with security identification.

A condor cluster could be built automatically by running the GVE Web service function *RunScriptInVM*.

8.1.2 Build a Submit Host

On the Submit Host, users need a Replica Location Service (RLS) [22] to map abstract data files to concrete data files.

A personal Condor, which includes DAGMan and Condor-G, is used to manage workflows and submit the workflows to Grids. Globus Toolkit is also needed for job execution on Grids. The following must be accomplished to build a Submit Host.

The Submit Host needs a personal Condor with Condor-G to manage workflows and Globus Toolkit to submit workflows to computational Grids. Similar scripts for building Compute Site are reused.

8.1.3 Start the Grid Platform

Daemons of various software should be started to run Grid and cluster services. For example: *Globus gatekeeper*, *Condor master*, and the RLS service should be started. Starting daemons could be finished by running scripts inside virtual machines.

8.1.4 Build VDS on the Virtual Grid Infrastructure

The VDS packages are installed in the Submit Host with installation scripts. Several important interfaces should be automatically configured so that VDS can run on the virtual Grid infrastructure, for example, the Site Catalog, the Transform Catalog and the Replica Catalog. The Site Catalog defines the compute site profiles. As the Grid platform is installed and configured with some pre-defined installation scripts, Site Catalog therefore could be automatically generated in the process of Grid construction. Then the VDS tool *genpoolconfig* is executed to convert the Site Catalog from text format to XML format. The Transform Catalog and the Replica Catalog are used to record application executables and data. They are configured when an application is organized in VDS.

8.2 Build VDS from Virtual Machine Image Database

Another method to build a VDS system on virtual machines is to clone virtual machines from various virtual machine templates. The virtual machine image database is implemented on OSFarm [23]. OSFarm is an application to generate virtual machine images and virtual appliances. OSFarm provides various methods for the users to connect to the OSFarm Server, for example, a Web portal and the *wget* invocation.

Several typical virtual machine templates are built on the virtual machine image database, for example, virtual cluster head node, virtual cluster worker node, and submit hosts. When GVE gets requirement to build a VDS on distributed virtual machines, it search the database and clones virtual machines from templates. These virtual machines are then organized as a VDS and returned to users.

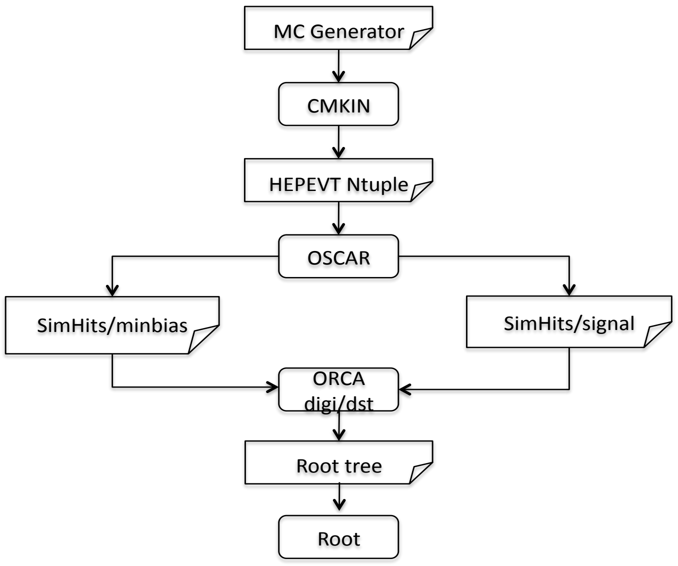

Figure 6: CMS workflow

8.3 Discussion on System Design and Implementation

The virtual machine based Grid system designed and implemented in this paper enjoys scalability, flexility and efficiency. The GVE makes itself distinguished from related work [24, 25, 26] in several aspects. The GVE is designed and implemented in modularity and the system components are wrapped with standard Web service interfaces. The modular design flavor 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, Virtualization Site Service, provides a general interface, which is virtual machine technology independent. The lower level service, Virtualization Agent, handles virtual machine specific implementations. The hierarchical design pattern makes the system more scalable to incorporate new virtual machine technologies.

This paper implements Grid applications with a VDS workflow system and develops various interfaces between the GVE and the VDS. The Grid middleware and application software are installed and configured in the distributed virtual machines. This computing paradigm detaches the resource provisions and application environments. It other words, our work does not reply on any specific Grid middleware and application environments. Users can build distributed computing environments on demand by invoking the GVE services to build a set of pre-configured virtual machines. Another example of our work is to build an e-Science infrastructure on distributed virtual machines [27] on demand.

The performance to build VDS on distributed virtual machines depends on the implementation. If users create a VDS from scratch by invoking GVE Web services, it would take up to 2 - 3 hours to build Globus and install VDS. However, this process is almost automatic and could be left as background task. If users build VDS by cloning virtual machines from virtual machine templates, it costs 2 – 10 minutes to build a VDS on distributed virtual machines.

9 Performance evaluation with CMS Benchmark

9.1 CMS Computing Software

The Compact Muon Solenoid (CMS) is one the biggest high energy physics experiments in the Large Hadron Collider (LHC) at CERN. It uses a general-purpose detector to investigate a wide range of physics, which includes the search for the Higgs boson, extra dimensions, and particles that could make up dark matter. The CMS software contains CMKIN, OSCAR, ORCA, HEPEVT, Pythia and ROOT module. The CMS benchmark could be used here as a test case is CMS OO (namely OSCAR and ORCA) [28]. In the CMS OO, Monte-Carlo events are generated and stored in CMKIN, the CMS detector is described and its interaction with particles are simulated in OSCAR, and simulated events are digitized in ORCA. To make full performance evaluation of virtual machines, this paper also includes several ROOT [28] based tests and the Monte Carlo generator Pythia [28] in the benchmark.

The CMS application is complex Grid application, which includes sophisticated software models. The compilation and deployment of CMS normally require intensive workload (e.g., several hours' work) and pre-knowledge of operating system & high energy physics. Therefore to put the CMS application into virtual machines can ease the onerous task of CMS deployment for users. This section shows that CMS application can be on-demand deployed on distributed virtual machines. We also compare the performance of CMS application on virtual machines and on real machines. Performance discussion shows that the distributed virtual machine can guarantee the QoS for CMS applications.

9.2 Test Bed

This work is carried on the SCC (Steinbuch Centre for Computing) test bed. The test bed contains two parts: computer servers in IWR/FZK and a compute cluster in RZ/UKA. Computer servers at the Institute for Scientific Computing (IWR) at Research Center of Karlsruhe (FZK) are installed with various VMMs, such as a Xen server, VMware server and VMware ESX server. The compute cluster in the Computing Center (RZ) at the University Karlsruhe (UKA) consists of 1 head node and 12 worker nodes, all of which are installed with Xen servers.

9.3 Generate CMS Workflow

The abstract workflow is organized as shown in Figure 6. CMKIN reads *datacards* from the MC generator and generates events in the *HEPEVT Ntuple*. OSCAR simulates high energy physics events and generates *pool data*: *Pool Simhits/minbias* and *Pool SimHits/signal*. The ORCA digitizes and reconstructs the events that are simulated by OSCAR. The ORCA outputs *ROOT Tree*, which can be further employed for ROOT benchmarking.

This work organizes virtual machines that contain CMS software packages inside. All software packages register themselves in a *TC* file. The input/output data of software packages are registered in the RLS server. The VDS *gencdag* then converts the abstract workflow to the concrete workflow with the above information.

9.4 Test organization and results

The CMS benchmark is executed on virtual machines with 64-Bit processors (virtual AMD Opteron Processor 250), which support both 32-Bit operating system and 64-Bit operating system. The CMS benchmark is a legacy application that demands various software libraries. Therefore, we are interested in migrating pre-compiled 32-Bit CMS benchmark to 32-Bit and 64-Bit operating systems. The test is organized in the following modes [29, 30]:

- Legacy mode: A 64-Bit architecture is installed with a 32-Bit operating system and a 32-Bit application is executed.

- Compatibility mode: A 64-Bit architecture is installed with a 64-Bit operating system and a 32-Bit applications are executed.

- Full 64-Bit mode: A 64-Bit architecture is installed with a 64-Bit operating system. Applications recompiled with 64-Bit libraries are executed.

In the test, we run CMS benchmarks in legacy mode, compatibility mode and full 64-bit mode to evaluate the performance of distributed virtual machines. OSCAR, ORCA packages are only available for legacy mode and compatibility mode. Pythia and ROOT are available for legacy mode, compatibility mode and full 64-bit mode.

Table 1 shows the performance of CMS workflow in term of *Event/Second*.

Table 1: CMS OO benchmark on virtual AMD opteron processor 250

CMS benchmark	**Legacy mode**	**Compatibility mode**	**Full 64-bit mode**
OSCAR	0.0880	0.0897	–
ORCA-Digi	0.135	0.119	–
ORCA-Dst	0.974	0.931	–
ROOT	945.0	864.2	1227.4
Pythia	105.30	103.96	121.38

9.5 Performance evaluation

Performance evaluation of CMS benchmark on virtual machines is performed by comparing the test results on virtual machines (Table 1) with those on real machines. Test

Table 2: CMS benchmark on AMD opteron processor 250

CMS benchmark	Legacy mode	Compatibility mode	Full 64-bit mode
OSCAR	0.1186	0.1154	–
ORCA-digi	0.1562	0.1300	–
ORCA-DST	–	1.039	–
ROOT stress	957.6	898.3	1402.6
Pythia	121.3	120.8	146.9

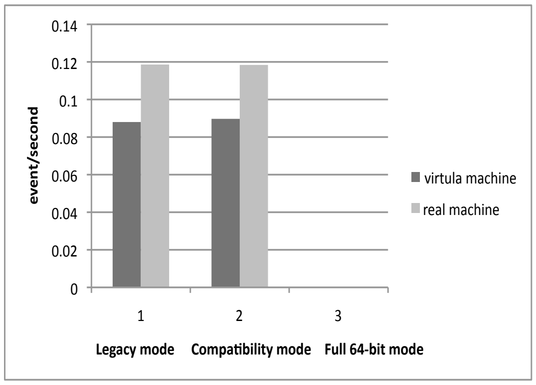

Figure 7: OSCAR test results

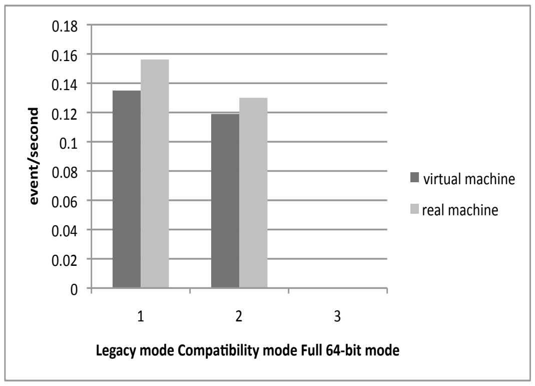

Figure 8: ORCA-digi test results

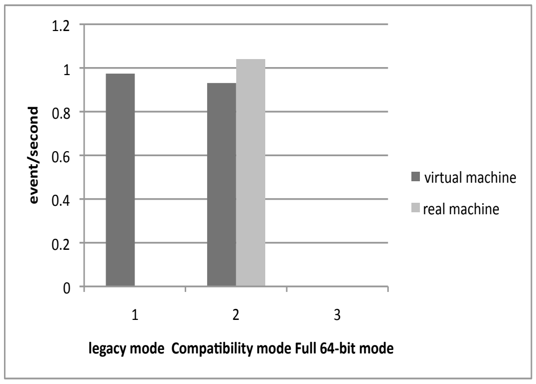

Figure 9: ORCA-dst test results

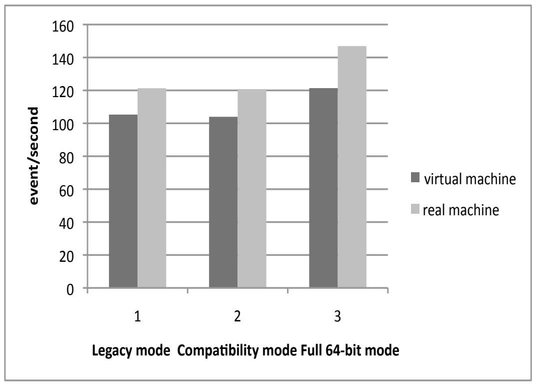

Figure 10: Pythia test results

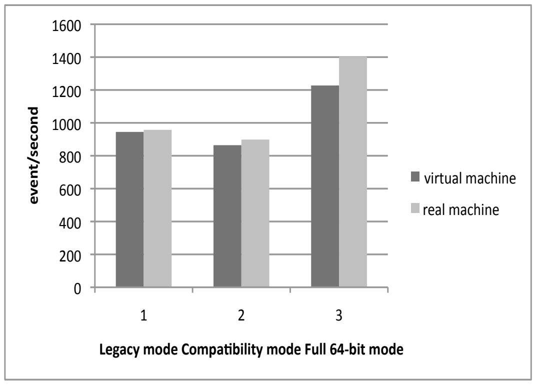

Figure 11: ROOT test results

results on real machines are reported in CMS internal reports [29, 30]. Table 2 shows the results of running CMS benchmark on real AMD opteron processor 250.

Figure 7 – Figure 11 show the comparisons of OSCAR, ORCA-digi, ORCA-dst, ROOT and Pythia test results between virtual machines and real machines. In these tests, virtual machines can attain 70%-95% performance of real machines. ROOT benchmark achieves good performance on virtual machines, since ROOT stress is an I/O intensive application. Compared with ROOT benchmark, results from OSCAR tests are unsatisfactory as OSCAR is a computationally intensive application. Comparison on different modes is also tested. It could be concluded that legacy mode and compatibility mode achieve almost the same performance. Full 64-bit mode can achieve the best performance among the three modes. It is rather obvious because in the full 64-bit mode, 64-bit applications receive access to the full physical memory range and also allowed access to the new General Purpose Registers (GPRs) as well as the expanded GPRs in 64-bit processors.

10 Conclusion and future work

Grid computing is an exciting achievement of computing technology. It focuses on distributed resource sharing, huge data processing and innovative problem solving.

Workflow systems, originally coming from business process automation, are introduced into Grid computing to alleviate Grid users' burden of running jobs, especially for complex applications. Despite great progresses have been made in Grid computing, QoS provision and customized computing environment for Grid applications remain challenging research issues.

This paper presents our work of building a VDS on distributed virtual machines to solve the forementioned research challenges. This paper proposes a computing paradigm – to employ virtual machines as computing resources and decouple Grid middleware & user computing environments from the Grid resource layer. This computing paradigm can bring various advantages such as QoS resource provision and deployment of customized computing environment.

An integrated system is presented in this paper, which employs a VDS as a workflow engine and runs Grid applications on distributed virtual machines enabled by the GVE. The whole system is designed and implemented in modularity. The interfaces between different GVE services and VDS are clearly defined and implemented in flexible flavors. This test provides the experience and performance evaluation of executing high performance engineering application (the CMS benchmark) on distributed virtual machines.

The GVE will be further developed and deployed in the large scale test bed, for example, a pan-Germany D-Grid [31] test bed and TeraGrid [32] for various Grid applications. It is also planed to develop the GVE with general user access interfaces and APIs. Various user level computing environments could be integrated to the GVE easily.

Acknowledgment

Work conducted by Gregor von Laszewski is supported (in part) by NSF CMMI 0540076 and NSF SDCI NMI 0721656.

REFERENCES

[1] I. Foster and C. Kesselman. *The Grid: blueprint for a new computing infrastructure.* Morgan Kaufmann, 1998.

[2] L. Wang, W. Jie, and H. Zhu. *State of the arts: workflow system for Grid computing*, chapter in Grid technologies: emerging from distributed architectures to virtual organizations. WIT press, M. Bekakos and A. A. Gravvanis and H. R. Arabnia edition, 2006.

[3] L. Seawright and R. MacKinnon. VM/370 – a study of multiplicity and usefulness. *IBM Systems Journal*, 18(1):4–17, 1979.

[4] J. Smith and R. Nair. *Virtual machines: versatile platforms for systems and processes.* The Morgan Kaufmann, 2003.

[5] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. L. 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*, pages 164–177, New York, U. S. A., Oct. 2003.

[6] VMware virtualization technology [URL]. http://www.vmware.com.

[7] 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.

[8] S. Childs, B. Coghlan, and J. McCandless. Grid-Builder: a tool for creating virtual grid testbeds. In *Proceedings of 2nd IEEE Conference on eScience and Grid computing (e-Science)*, pages 77–83, Amsterdam, Netherlands, Dec. 2006. IEEE Computer Society.

[9] 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.

[10] 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.

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

[12] Globus virtual workspace interface guide [URL]. http://workspace.globus.org/vm/.

[13] 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.

[14] Amazon Elastic Compute Cloud [URL]. http://aws.amazon.com/ec2/.

[15] Eucalyptus Project [URL]. http://eucalyptus.cs.ucsb.edu/.

[16] Emeric Kwemou, Lizhe Wang, Jie Tao, Marcel Kunze, David Kramer, and Wolfgang Karl. Grid virtualization engine: Providing virtual resources for grid infrastructure. In *9th Workshop on Parallel Systems and Algorithms (PASA)*, pages 27–36, 2008.

[17] L. Wang, M. Kunze, and J. Tao. Performance evaluation of virtual machine-based grid workflow system. *Concurrency and Computation: Practice and Experience*, 20(15):1759–1771, 2008.

[18] Virtual Data System [URL]. http://vds.uchicago.edu/.

[19] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke. Condor-G: a computation management agent for multi-institutional Grids. *Cluster Computing*, 5(3):237–246, 2002.

[20] Globus Toolkit [URL]. http://www.globus.org/toolkit/.

[21] Glue Schema V1.3 [URL]. http://glueschema.forge.cnaf.infn.it/spec/v13/.

[22] A. L. Chervenak, N. Palavalli, S. Bharathi, C. Kesselman, and R. Schwartzkopf. Performance and scalability of a replica location service. In *Proceedings of the 13th International Symposium on High-Performance Distributed Computing*, pages 182–191, 2004.

[23] OS Farm project [URL]. http://cern.ch/osfarm/.

[24] 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.

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

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

[27] L. Wang, M. Kunze, E. Kwemou, J. Tao, D. Kramer, and W. Karl. On demand build a virtual e-Science workflow. In *Proceedings of the 3rd International Conference on Grid and Pervasive Computing*, pages 93–98, 2008.

[28] CMS software [URL]. http://cms.cern.ch/software/cms/.

[29] H. Wenzel. Benchmarking AMD Opteron (AMD64) and Interl (EM64T) systems. Technical report, CMS Internal Note, CMS-IN 2005/012, CMS Software and Computing Group, 2005.

[30] H. Wenzel and S. Langley. Benchmarking dual-core AMD Opteron (AMD64) systems. Technical report, CMS Internal Note, CMS-IN 2005/030, CMS Software and Computing Group, 2005.

[31] Germany national Grid initiative [URL]. http://www.dgrid.de/.

[32] TeraGrid project [URL]. http://www.teragrid.org/.