Job execution in virtualized runtime environments in grid

To cite this article: Lev Shamardin et al 2010 J. Phys.: Conf. Ser. 219 062065

View the article online for updates and enhancements.
Job execution in virtualized runtime environments in grid

Lev Shamardin, Andrey Demichev, Ilya Gorbunov, Slava Ilyin and Alexander Kryukov
DTHEP, SINP MSU, Leninskie Gory 1, GSP-1, Moscow, 119991, Russia
E-mail: shamardin@theory.sinp.msu.ru

Abstract.
Grid systems are used for calculations and data processing in various applied areas such as biomedicine, nanotechnology and materials science, cosmophysics and high energy physics as well as in a number of industrial and commercial areas. Traditional method of execution of jobs in grid is running jobs directly on the cluster nodes. This puts restrictions on the choice of the operational environment to the operating system of the node and also does not allow to enforce resource sharing policies or jobs isolation nor guarantee minimal level of available system resources. We propose a new approach to running jobs on the cluster nodes when each grid job runs in its own virtual environment. This allows to use different operating systems for different jobs on the same nodes in cluster, provides better isolation between running jobs and allows to enforce resource sharing policies. The implementation of the proposed approach was made in the framework of gLite middleware of the EGEE/WLCG project and was successfully tested in SINP MSU. The implementation is transparent for the grid user and allows to submit binaries compiled for various operating systems using exactly the same gLite interface. Virtual machine images with the standard gLite worker node software and sample MS Windows execution environment were created.

1. Introduction
The grid systems are widely used in scientific communities for processing of large amounts of data. LHC Computing Grid [1] for example is built to process the huge amounts of data from the physical experiments carried out on the Large Hadron Collider in CERN. Traditional grids are usually built on top of batch clusters, and the jobs submitted to such grids are executed directly on the cluster nodes. This puts some restrictions both to the jobs and to the grid architecture. Because the job is executed on the cluster node, it must be designed for execution on the cluster node operating system. This also means that operating systems used on grid cluster nodes must be binary compatible, if not identical. For example, in LCG all production sites have nodes that are binary compatible with Scientific Linux 4. Such limitations restrict the area of application and adoption of grids, because there are many software applications written for newer and older versions of Linux, and there is a big number of applications written for other operating systems including MS Windows.

In our work we try to extend the limits of applications which can be run in grid by using virtualization technologies to execute virtual machines (VM) on cluster nodes with user applications running inside. Such approach gives a number of benefits compared with regular
jobs running on cluster nodes:

- Better isolation of jobs. If each job is running in its own VM than it cannot interact with other jobs running on the same system. It cannot accidentally modify files of another job, even if it has the same credentials, it cannot kill other jobs' subprocesses.
- Enforced resource sharing policies. Each job is strictly limited to the system resources allocated for the VM. A job cannot consume more memory or CPU resources than allocated, and a job has a guarantee that the resources allocated for the job will be exclusively available during the whole job lifetime.
- Different OSes. Each job running in its own VM can use a different OS, it is not restricted to host Linux version and even MS Windows can be used as a guest OS.

We created a Virtual Execution Environments (VEE) system which allows to run virtualized jobs on top of the gLite middleware.

From the beginning we intended to use our system mainly with the LCG grid infrastructure and the implementation is compatible with gLite [2] middleware. Wide adoption of this software will be possible only if it is easy to install, requires minimal intrusion to gLite and does not break regular gLite functions. So the basic principles we used were:

- Minimal modification of the gLite CE and gLite WN middleware.
- Minimal impact on the regular functions of the gLite CE and WN.
- No modifications to the informational system and the core services (like Workload Management System (WMS)).
- No modifications to the gLite UI software.

There are some other researches which use virtualization combined with gLite software. There is a project CernVM [3] which provides configurable OS images with preinstalled gLite and physics software, these images were used for some tests in our research. There are projects like [4] which use virtual nodes to build the gLite infrastructure on demand. There are also researches which utilize per-job virutalization on worker nodes. For example in [5] authors use Xen virtualization to run worker node images with older versions of ScientificLinux OS on nodes with newer versions of the OS to hide the incompatibilities of older software and new hardware. In their newer work [6] presented at this conference they use the approach similar to ours to inject the grid jobs to a prepared VM image, but they focus primarily on running regular worker node images and use virtualization to overcome the limitations of the OS and other components versions introduced by the gLite and physics software.

We focus on utilizing gLite infrastructure for running jobs which were designed for operating systems completely foreign for the gLite grid, like jobs targeted for Microsoft Windows OS. A user can create a gLite job using standard job description syntax and provide a windows executable, and this job can be submitted and executed on Computing Elements which have our software and corresponding OS images installed. And the same Computing Elements and Worker Nodes can be still used as a part of a regular gLite infrastructure.

We also allow execution of user-constructed virtual machine images as regular grid jobs, which can allow a “grid cloud”-like mode of operation for the existing gLite infrastructure.

2. System overview

We use Xen [7] as the virtualization hypervisor. Xen must be installed on all gLite Worker Nodes (WNs) connected to a gLite Computing Element (CE) which will be used to run virtualized jobs.

The CEs running VEE software publish the list of the available operating system images as regular GlueHostApplicationSoftwareRunTimeEnvironment tags, so they can be easily selected with the standard Requirements clause in the JDL [8] description.
When a job with VEE requirements in JDL arrives at the CE the system starts a virtual machine on one of the WNs and runs a status monitor job as a regular batch job. This job is responsible for the interaction with the batch system. It is accounted as a regular grid job, and if a request to terminate this job arrives from the grid to the local batch system, it stops the virtual machine running the real job.

The monitor job is also responsible for transferring the input and output sandboxes of the job to and from the VM image. The sandboxes are first transferred to the WN using standard gLite mechanisms and then the monitor job copies the input sandbox to the virtual machine image before execution, and copies the output sandbox back after the execution.

2.1. User work flow
To submit a job using the VEE system a user must create a JDL file which specifies the desired virtual machine as a regular requirement for the software runtime environment. An example of the JDL for the job to be executed under Windows XP operating system may be found in Appendix A.

Currently there is an implementation limitation which requires a user to include the JDL file itself as a part of the input sandbox of the job.

3. Implementation details
Our system is implemented for the CEs using Torque [9] batch system. The VEE system consists several scripts, some of which are installed on the CE, and some on the WNs connected to this CE.

Information about the available runtime environments is published as regular software runtime environment tags using the regular gLite CE info provider in the file usually located at /opt/edg/var/info/host.

3.1. Computing Element
The CE package of the VEE system overrides the regular lcg-sam-jobwrapper package and replaces the standard /opt/lcg/libexec/jobwrapper with our version. It does all the functions of the original version, but it additionally modifies the submitted job before its execution if it is detected to be submitted through the gLite WMS system.

All jobs arriving on the CE through the gLite WMS system are actually special bootstrap wrapper scripts. The jobwrapper-vee script detects such scripts and inserts the call to itself in place of actual job execution, preserving the original executable name and options. The modified job then is started on the WN through the regular batch job work flow.

3.2. Worker Node
The worker node requires more complex modifications. First of all, all WNs must have CPUs capable of hardware virtualization support (Intel VT [10] or AMD-V [11]) and have Xen with hardware virtualization support installed.

A pool of blank virtual machine images is created on each worker node. These images are used as templates for actual VM images with submitted jobs. The set of these VM templates must match the set advertised by the CE.

For each of the images a configuration file must be created. This file specifies the path inside the VM where it expects the job and its input sandbox to be located, and the path where the output sandbox files are placed. Each VM’s guest OS must be configured in a way that it automatically starts the job from this predefined location right after the system boot, redirecting standard IO streams as required, and shuts down automatically when the job execution is finished.
Execution of part of VEE scripts using `sudo` must be enabled for all of the user pool accounts. When the modified job script arrives on the WN through the batch system it checks its sandbox for a job JDL file and if found checks whether the job requests the VEE runtime environment tag. If these conditions are not satisfied, the job is started directly on the WN.

If the VEE wrapper script detects that the job does request a VEE environment tag, then it does the following:

(i) Creates a new VM image using the required OS template image.
(ii) Mounts the created image, copies all input sandbox into the path inside the VM defined in the configuration and then unmounts the image.
(iii) Starts the VM using Xen.
(iv) Sleeps in the background during the VM execution waiting either for its termination or for the signals from the batch to terminate the job.
(v) If the signal to terminate the job from the batch is received it destroys the running VM and removes the VM image.
(vi) Upon shutdown of the VM it mounts the VM image again and transfers the output sandbox files to the job working directory on the WN. The VM image is disposed after that.

The wrapper script is running as a batch job while the VM is actually working (Figure 1), this script is treated by the batch system as a regular job, so it gets accounted, the WN CPU is published as occupied, etc.

### 3.3. Guest OS template image

Our system requires specially prepared guest OS images. When the guest OS image is to be started on the WN all files from the job input sandbox are copied to the VM image. The OS must have appropriate boot configuration to start the job automatically from the predefined location when it is booted, to handle the redirection of the job IO streams and to shut down the OS when the job is terminated (Figure 2).

We have prepared guest OS template images with MS Windows, Fedora Linux and slightly modified standard gLite WN as examples.

### 3.4. Non-standard operation modes

Above we described the regular way of VEE operation. But there are two extra modes of VEE operation which also provide attractive features. First, the VEE also publishes a special runtime
environment with a tag ‘VEE-UserVM’. The jobs with such requirements have special treatment. The executable name of such jobs is considered to be a gLite LFN of the VM image. Instead of copying the sandboxes and using a template VM image, the user image is copied to the WN using lcg-cp and then started under Xen. The user has the responsibility to shutdown his VM when appropriate, otherwise it will be destroyed when the batch system decides to kill the job due to time limits for example. The user is also responsible for getting the results from the VM using its own methods. The VM image is disposed as usual after the execution.

The second experimental operation mode wraps all incoming gLite jobs into the VEE-SL4gLiteWN image regardless of the jobs requirements. The VEE-SL4gLiteWN contains a standard blank Scientific Linux 4 gLite Worker Node, except it automatically starts the jobs copied inside it instead of acting as a batch node. This allows to execute any of the gLite jobs coming to the CE in a separate virtual machine.

4. Conclusion
The proposed software can be used to extend the functionality of the existing grid infrastructures based on gLite middleware. It allows to create the environments for execution of jobs designed for other operating systems, such as MS Windows, other Linux versions simplifying grid utilization in such areas as nanotechnology, thermonuclear power, biomedicine, geoscience, etc. The suggested approach provides a substantial extension of the class of applied problems that can be solved with the aid of global grid infrastructures. This is an important step to further development of grids and their wide commercial applications.

The proposed technology is unobtrusive. The required modifications to a production gLite CE are minimal and do not interrupt the regular operation of the CE. No modifications in the core services are required.

Acknowledgments
This work was partially financially supported by The Federal Agency for Science and Innovation (contract No. 02.514.11.4072), RFBR (grant No. 07-07-12023) and Scientific School HIII–1456.2008.2.

Appendix A. JDL file examples
The following JDL file may be used for a simple job to be executed under MS Windows XP operating system:

```plaintext
Executable = "test.exe";
StdOutput = "testJob.out";
StdError = "testJob.err";
InputSandbox = {"test.exe", "job.jdl"};
OutputSandbox = {"testJob.out", "testJob.err"};
Requirements = Member("VEE-WindowsXP",
                       other.GlueHostApplicationSoftwareRunTimeEnvironment);
```

The following JDL file may be used for a simple job to be executed under a virtualized instance of the gLite WN:

```plaintext
Executable = "test";
StdOutput = "testJob.out";
StdError = "testJob.err";
InputSandbox = {"test", "job.jdl"};
OutputSandbox = {"testJob.out", "testJob.err"};
Requirements = Member("VEE-SL4gLiteWN",
                       other.GlueHostApplicationSoftwareRunTimeEnvironment);
```
References

[1] Lamanna M 2004 *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **543** 1-6 proceedings of the IXth International Workshop on Advanced Computing and Analysis Techniques in Physics Research

[2] gLite: Lightweight Middleware for Grid Computing URL [http://glite.web.cern.ch/glite/](http://glite.web.cern.ch/glite/)

[3] Buncic P, Aguado Sanchez C, Bloomer J, Franco L, Klemer S and Mato P 2008 *Proceedings of XII Advanced Computing and Analysis Techniques in Physics Research* (Proceedings of Science vol PoS(ACAT08)012)

[4] Harutyunyan A, Buncic P, Freeman T and Keahey K In these proceedings

[5] Büge V, Kemp Y, Kunze M, Oberst O and Quast G 2006 *Lecture Notes In Computer Science* **4331** 397–406 URL [http://www.springerlink.com/content/t36h2pg2207703u8](http://www.springerlink.com/content/t36h2pg2207703u8)

[6] Büge V, Haas A, Hessling H, Kemp Y, Kunze M, Oberst O, Quast G and Synge O 2009 *Journal of Physics: Conference Series* 17th International Conference on Computing in High Energy and Nuclear Physics (CHEP09) IOP Publishing Journal of Physics: Conference Series 219 (2010) 062065 doi:10.1088/1742-6596/219/6/062065

[7] Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I and Warfield A 2003 *Proceedings of the 19th ACM Symposium on Operating Systems Principles*

[8] Job description language attributes specification URL [https://edms.cern.ch/document/555796/1](https://edms.cern.ch/document/555796/1)

[9] TORQUE Resource Manager URL [http://www.clusterresources.com/products/torque-resource-manager.php](http://www.clusterresources.com/products/torque-resource-manager.php)

[10] Neiger G, Santoni A, Leung F, Rodgers D and Uhlig R 2006 *Intel Technology Journal* 10 ISSN 1535-864X URL [http://www.intel.com/technology/itj/2006/v10i3/1-hardware/1-abstract.htm](http://www.intel.com/technology/itj/2006/v10i3/1-hardware/1-abstract.htm)

[11] AMD 2005 *AMD64 Virtualization Codenamed “Pacifica” Technology: Secure Virtual Machine Architecture Reference Manual*