Dynamic virtualization of AliEn grid jobs using the Vmbatch system

B. Kileng, H. Helstrup, K.F. Hetland
for the ALICE collaboration
Høgskolen i Bergen, Postboks 7030, 5020 Bergen
Bjarte.Kileng@hib.no, Havard.Helstrup@hib.no, Kristin.Hetland@hib.no

October 27, 2016

Abstract

The Vmbatch system is shown to be a robust and reliable system for running batch jobs inside virtual machines. The system has been developed as a lightweight tool to establish and clean up virtual machines for CernVM processing of ALICE grid jobs. It can work with a stock guest image and interfaces with the Torque batch system.

With the use of virtualization, the system can create a homogeneous execution environment for grid jobs that can be expanded dynamically upon availability of generic computing resources.

1 Introduction

Virtualization has increasingly been used as a technique to obtain a common interface to differing underlying computing architectures. The ALICE [1] experiment uses its own middleware AliEn [2] to coordinate grid based analysis of the several PB of data recorded from particle collisions produced by the LHC accelerator at CERN. AliEn interfaces directly to underlying batch systems at computing elements (CEs), but is also set up to forward jobs to other grid middleware systems like WLCG [3] and OSG [4].

AliEn grid operation now uses CernVM-FS [5] as distribution method for experiment software. It is straightforward to use a virtual setup using CernVM [6] as an AliEn computing element. The performance degradation of using virtual machines has been proven to be acceptable [7].

The number of ALICE jobs running at a given site can vary due to several parameters (system load, availability and profile of ALICE jobs, quota considerations etc.). To obtain an efficient use of underlying resources, it is desirable to have a dynamic management of virtual machines in the computing cluster. The Vmbatch system has been developed as a lightweight tool to establish and clean up virtual machines for CernVM processing of ALICE grid jobs.

This paper was presented at the NIK-2016 conference; see http://www.nik.no/.

2 Virtual resources for grid jobs

Grid computing resources are loosely coupled and generally heterogeneous, both in terms of hardware and software. A more homogeneous execution environment can be obtained by running grid jobs inside virtual machines. This can also improve security by better protecting the host from malicious jobs. With one virtual machine for each job, another possible benefit is less interference between jobs that run on the same host.

Virtual machines allow for grid jobs to use CernVM. CernVM is a Virtual Software Appliance designed for the CERN LHC experiments. It consists of a read-only image of approximately 20MB containing a Linux kernel, a network file system client and the union file system [8] AUFS [9]. The network file system used by CernVM, CernVM-FS, is a read-only network file system that uses HTTP as underlying transport protocol. It contains the the major parts of the CernVM OS but also CERN experiment software.

When CernVM boots for the first time, an empty disk image should be attached to the guest. This disk will be used as an AUFS read-write layer above CernVM-FS [10]. Whenever a file is read from CernVM-FS, a copy is stored in the local read-write layer, the CernVM-FS cache. CernVM can also run in a Docker container. CernVM with Docker can use Parrot [11] to access the CernVM-FS, but the use of Parrot requires that CernVM is run as a privileged container. This will give the container certain root capabilities on the host. Another option is to mount the CernVM-FS on the host, i.e. the host must be adapted to use the CernVM Docker image.

A few systems exist that can use cloud resources for grid jobs, i.e. Cloud Scheduler [12], AliEn with Openstack [13], Elastiq [14] and Nimbus [15]. These systems require a full private cloud infrastructure, or access to a commercial cloud infrastructure. Virtual worker nodes for the grid system will be created on demand in the cloud.

Running batch jobs inside virtual machines is possible by some systems, e.g. HTCondor [16]. They all require that the guest image must adapted for the system, prior to running the first job. Some systems also allow for batch jobs to be run inside Docker containers, e.g. Torque with Moab [17], OpenLava [18], HTCondor [16]. As explained above, the CernVM Docker solution has issues.

3 Vmbatch specifications

The aim of this work is to present the Vmbatch system as a useful virtualisation tool adapted to ALICE needs. Testing has been focused on the Vmbatch-specific parts of the processing, i.e. startup and completion of jobs. The overhead produced by Vmbatch is shown to be small compared to the average running time of an ALICE grid job, and the testing presented demonstrate robustness to handle realistic exception scenarios in a grid production environment.

The Vmbatch system is designed to fulfill the following requirements:

- No prior adaption of guest images for the system should be needed. If a guest can run the job, then the system should be able to use the guest image.
- The system should be able to run AliEn grid jobs by the use of a stock, unmodified CernVM image or with standard OS images (e.g. CentOS 6).

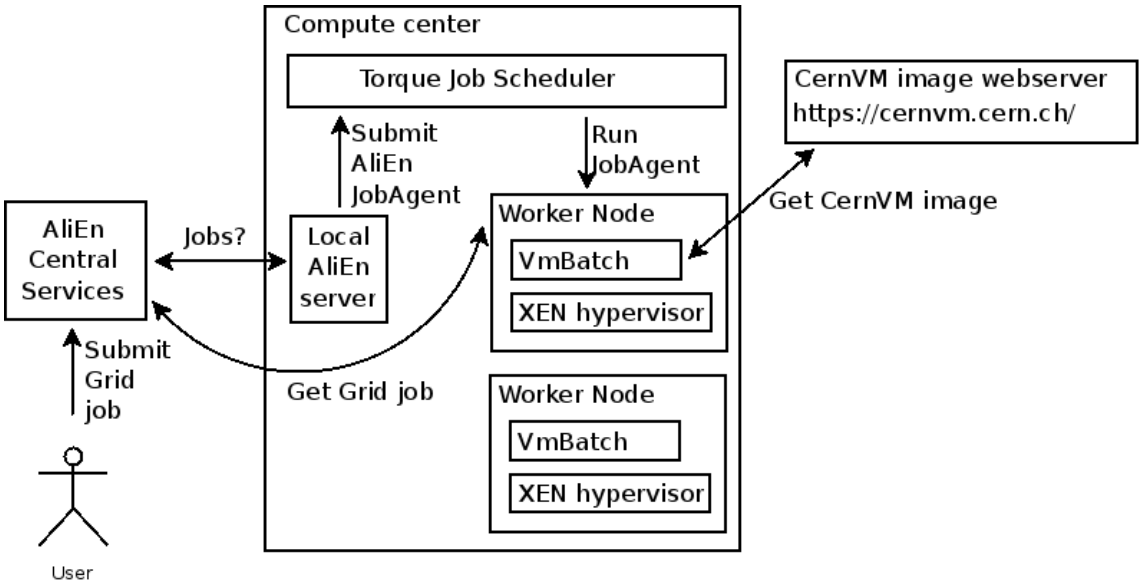

Figure 1: Job management in Alien using VmBatch on worker nodes.

- When the system has been set up on a host, the host system should need no adaption to run specific jobs or use specific guest images.
- The system should not have a negative impact on the security of the host system.

Vmbatch is designed to run as a part of the job management by the AliEn grid middleware. A diagram of components is shown in figure 1. When a user submits an AliEn job, the job is put into a central job queue at the AliEn central services. The central services will match the job with a compute element (CE). Each AliEn CE will have a local AliEn server that regularly asks the central services if there are waiting jobs. If a job matches the CE, the local AliEn server will submit a pilot job, a Job Agent, into the local batch system. The scheduler of the batch system will then match a worker node with the pilot job. The pilot job will set up the AliEn Grid environment on the worker node, and then ask the central services for the Grid job. Running without Vmbatch, the worker node must be set up in advance with a Grid shared filesystem, the CernVM-FS.

If the worker node is set up with Vmbatch, Vmbatch will prepare a disk image to run the job. The disk image must have the necessary software for the job, but no prior adaption for the Vmbatch system is necessary, neither for the CE nor for the worker node. The CernVM image already includes all software to run the AliEn Job Agent, and Vmbatch can work with a stock CernVM image. Vmbatch can download the CernVM image from the CernVM webserver. A CernVM virtual machine will have access to the Grid shared filesystem, and no AliEn software is needed on the worker node. Vmbatch will run one virtual machine for each Job Agent, and remove and clean up the system when the Job Agent is done.

4 The Vmbatch system

Vmbatch is a system for running batch jobs inside virtual machines. It is far less complex than a full private cloud infrastructure. The batch job is injected into the virtual machine and the same image can be used with different batch job.

Vmbatch currently supports Torque as job scheduler and Xen3 and Xen4 as hypervisors with Redhat type hosts and guests. With Xen3, Vmbatch has been tested on SLC5[1] hosts to run SLC5 and CernVM guests. With Xen4, Vmbatch has been tested on CentOS6 hosts to run CentOS6 and CernVM guests. Vmbatch uses Libvirt [19] to manage the Xen guests.

When Vmbatch receives a job, it will create and contextualize a new virtual machine to run the job. Each job will run inside its own virtual machine, and when the job finishes, the virtual machine will be removed by the system.

Vmbatch has been developed by the authors of this paper (source code available at [20]) based on the Vibatch system [21]. Vmbatch uses Torque prologue and epilogue scripts to establish and clean up the virtual machine. The prologue script is run as root by Torque when a job arrives at host, and the epilogue is run as root by Torque when the job has finished.

A script, remoteshell, gives Torque access to a running guest. The default setup returns a SSH connection into the guest. Vmbatch will run each guest in a Vmbatch job slot, each characterized by a guest name and a guest MAC address. If Vmbatch can not find a free slot, the default behaviour is to abort the job, but Vmbatch can also ask the Torque server to resubmit the job.

Unless configured read-only, a disk image can not be shared between simultaneously running virtual machines. Vmbatch can therefore use the guest images as templates, and create new images for each job. The template images are left unchanged, and all jobs that use the same guest images will run in identical environments.

Vmbatch allows the user to specify what guest images to use when submitting the job, as job resources. Images can be located on the worker node, or be downloaded from the net. Vmbatch can create the empty disk that CernVM will use for the CernVM-FS cache. The disk cache can be created and initialized independently for each job, or by a pilot job. When the pilot job finishes, the disk cache is released and available for later jobs.

Torque will abort a prologue script that runs for more than 5 minutes, and the Torque prologue script can also time out if copying large images. The new images created from guest templates should therefore preferably be copy-on-write images. From version 4 of Xen, Vmbatch can create copy-on-write images from raw disk images, and from VHD, QCOW and QCOW2 disk images.

The Vmbatch contextualization of the guest can include the following operations:

- Inject the job into the guest
- Create the user that should run the job
- Configure the guest network
- Install and remove packages (programs)
- Unpack tar archives
- Enable, disable, start and stop init services
- Configure NFS

[1]SLC = Scientific Linux CERN, a recompiled and extended version of RedHat distributions

- Configure or disable SELinux

Vmbatch requires that the home directory of the job submitter on the host must be available as a NFS share inside the guest. This share is used for storing job output, but also for transferring guest parameters (e.g. IP number of the guest) to the host.

For the contextualization process, Vmbatch can use three different approaches as outlined below.

DISK contextualization

The most powerful approach is the DISK contextualization approach. Following this approach, Vmbatch will configure the guest disk images prior to starting the virtual machine. From Xen version 4, no prior adaption of the image is required. The DISK contextualization approach currently does not support LVM disks.

CDROM contextualization

Vmbatch can create a CDROM with contextualization scripts and parameters, and attach the CDROM to the guest. When using this approach, the virtual machine must be configured in advance to mount the CDROM, and to run two Vmbatch scripts, *prologe.sh* and *epiloge.sh* that is put on the CDROM. The CernVM Virtual Software Appliance includes the functionality required by Vmbatch and Vmbatch can therefore work with a stock CernVM image.

ROOTSSH contextualization

The ROOTSSH contextualization approach requires that Vmbatch specific init scripts are copied to the guest before using the guest with Vmbatch. Contextualization parameters are transferred to the guest on the kernel boot line and the job is copied into the guest by the use of a SSH connection as root.

The ROOTSSH contextualization approach is most easily adapted to new Linux distributions. It is also the most vulnerable approach, since the Torque prologue script must wait for the guest to be up and running. A guest that takes long time to boot can cause the prologue script to time out.

5 Vmbatch for ALICE jobs

Vmbatch has successfully run jobs from (our testbed) AliEn grid. For Vmbatch to be a viable solution, the software must be beneficial both from a grid user and grid site administrator perspective. Job throughput and system reliability is essential. It is also important that the system is easy to install and maintain, with a small footprint on the host systems.

The goal of this study has been to check the reliability and stability of the solution, but the use of virtualization does have implications for many aspects important for grid users and grid site administrators. By the use of the CernVM virtual machine, no grid specific configurations or grid software need to be available on the worker nodes. This simplifies the task of adding worker nodes to the AliEn grid, and non-dedicated computers can easily be added to the grid at times of high load. Vmbatch will turn workers into multipurpose batch system nodes. Since programs and libraries for the jobs only need to exist inside the guests, the same worker node can run jobs from different grid systems and different environments.

Neither security, nor resource utilization is the aim of this study, but some general remarks are made. The hypervisor will require some memory, and disk images can require much disk space. The use of disk image formats like QCOW2 or VHD does significantly reduce the required disk space. The hypervisor can be tuned for the resource requirements of the job, or to avoid individual jobs to starve the worker node. Virtual machines will improve the isolation of jobs from the host, and this should improve security. The use of virtualization is also a means for putting hard limits on the resources available to jobs.

The use of virtualization will impact the performance of the grid hosts, and the grid system as a whole. The overall consequences has yet to be measured. Virtualization will require some extra resources, but at the same time, elastic computing can allow for opportunistic use of other available resources for grid jobs.

6 Setup for testing robustness

Robustness refers to the correct operation of software in the presence of invalid inputs or stressful environments [22]. Example scenarios are abnormal terminations, system freezes, or lost events. Input to Vmbatch are batch jobs and job parameters. Vmbatch can not prevent erroneous input to cause a job to fail, but the system itself should not crash, and subsequent correct jobs should run without problems. Failure of Vmbatch to run jobs can be caused by problems in any of the depending systems, e.g. Torque, NFS, SSH, operating system or hypervisor services. Our aim is to make Vmbatch as reliable and robust as job execution directly on host with Torque. We will therefore focus on Vmbatch itself and the hypervisor services. An attempt for robustness requirements can include that Vmbatch should withstand failing jobs, without causing errors going beyond the failing jobs itself.

The potentially problematic phases of job execution with Vmbatch are in the setup of the batch job for Vmbatch, setup of the guest, the startup of the guest, and in the cleaning phase after job execution. In these phases, the job is not executing its code, and can not fail, but wrong job parameters, an erroneous guest image and also job termination signals can cause Vmbatch to fail. Vmbatch can also fail due to time out. Torque will abort the Vmbatch prologue script if the startup phase execution lasts more than 5 minutes.

The robustness requirements when testing Vmbatch is therefore chosen as:

- A deleted batch job should not cause concurrent or subsequent jobs to fail.
- A batch job that causes Vmbatch to time out or fail should not cause concurrent or subsequent jobs to fail.

The setup for testing the first robustness requirement includes two scripts. The first of these will submit jobs to Torque and try to keep the job queue filled at all times. The next script will arbitrarily kill jobs. Robustness is then checked by investigating the output of the not-killed jobs. These jobs should run without problems. The setup for testing the second robustness requirement is based on white-box testing. The code was studied in order to find problematic spots in the code. Code was inserted that could cause Vmbatch to either abort or time out.

The most problematic spots in the code are dealing with shared resources, i.e. guest images, devices, guest names, MAC addresses and IP numbers. The shared resources are handled in critical regions that are protected by locks. In order to

handle a crash inside critical regions, locks are assigned a time out, and proper cleaning has to be executed if a lock is taken by force. A setup was created to test the three following scenarios for all the problematic points in the code:

- Vmbatch is asked to crash at one of the problematic spots. Then a normal job is submitted.
- A job is marked as rerunnable, and Vmbatch is asked to time out at a problematic spot. The rerun of the same job is run as normal.
- A job is marked as rerunnable, and Vmbatch is asked to time out at a problematic spot. Also all reruns of the same job will cause Vmbatch to time out. The next new job is run as normal.

Robustness was checked by investigating the output of the normal jobs. They should run without any problems.

7 Setup for testing reliability

Reliability describes the ability of a system to provide a continuous service. Worker nodes for grid jobs are expected to be running continuously for longer periods of time without manual interventions. For Vmbatch to be a viable solution, it must provide a reliable service for grid jobs.

Reliability can be tested by submitting a lot of different jobs to Torque, using different guest images and different job parameters. Since job execution occurs inside a hypervisor, the consequences of a problematic batch job is expected to be less severe compared to a batch job that run directly on the host. In this study, we have therefore only tested rather simple batch jobs. A more thorough test should involve real grid jobs run by Job Agents in a grid environment with heavy load. Our focus for the reliability tests in this study has been to test the Vmbatch modes that we expect will be the most commonly used.

The following setup was used for testing reliability:

- One Torque server and one worker node
- The worker node uses Xen as hypervisor
- Jobs are submitted in order to keep 5 jobs in the queue at all times
- Torque is configured to run two jobs concurrently on the worker node
- A total of 20 000 jobs are submitted.

The test will cause Vmbatch to start and stop 20 000 guests, one for each job. Most of the time, the worker node will have two concurrent guests running. All the Vmbatch contextualization approaches were tested, i.e. DISK, CDROM and ROOTSSH. The tests have been run on a CentOS6 host, using both CentOS6 and CernVM guests.

8 Results and discussion

Each run of a batch job will cause Vmbatch to start a new virtual machine and remove the virtual machine when the job has finished. Proper functioning of Vmbatch requires that the epilogue script is run after each batch job and can release the Vmbatch slot of the job. If no free slot can be found, Vmbatch will abort the job, or ask the Torque server to resubmit the job.

In most tests, Torque has run the epilogue script as intended when the batch job finishes, and the job slot has been released. A few cases were observed with ROOTSSH contextualization where Torque started to run the epilogue script of a killed job before the prologue script had finished. Vmbatch is handling the job slot inside a critical region, and the Torque problem did not cause Vmbatch to fail.

There have been a few cases where the epilogue script did not run, but all these cases have been hardware related. When the host system recovered, a manual cleaning both of the Vmbatch and Torque environment was necessary. Vmbatch is supplied with a script that will do a proper cleaning of the Vmbatch system. If the script is run by the host init system, very little further manual maintenance is required.

Robustness

The first robustness requirement states that a deleted job should not cause concurrent or subsequent jobs to fail. The behaviour of Vmbatch when batch jobs are deleted was tested for all the Vmbatch contextualization approaches, i.e. DISK, CDROM and ROOTSSH. In each test, 2000 jobs were submitted, and jobs were deleted arbitrarily. With the chosen parameters, 40 to 50% of the jobs were deleted.

Several batch jobs failed because Vmbatch could not find an available Vmbatch job slot. This revealed a problem in Torque, which was cured by introducing a delay of 4 seconds before deleting a job, to ensure consistent Torque job status when the jobs are deleted.

The second robustness requirement states that a job that causes Vmbatch to time out or fail should not cause concurrent or subsequent jobs to fail. When Vmbatch aborts due to missing resources, mis-configurations etc., Vmbatch is set up to do a proper cleaning. If the prologue or epilogue script times out, the script will be aborted, and resources might not be released. All operations that can take some time to finish are therefore protected by a time limit. If the Vmbatch code runs as expected, the prologue and epilogue scripts can not time out, and aborts should not cause the system to fail.

Tasks and operations that are not supposed to fail are not protected in the Vmbatch code. The behaviour when Vmbatch fails was tested by aborting the prologue and remoteshell scripts abruptly, without letting the script release its locks and resources. The next job was run as normal, and its behaviour was studied. All critical regions were successfully tested, for all three Vmbatch contextualization approaches.

Vmbatch does not put time limits on tasks that are supposed to run fast. Timeouts in the prologue and epilogue scripts can therefore occur if tasks run much slower than expected, and the added time while waiting for locks and tasks add up to more than 5 minutes. When a prologue or epilogue script gets a time out, the script will not release its locks and resources. Timeout behaviour was tested with two different scenarios. In the first scenario, a job times out in the prologue phase

and is rerun by Torque. The rerun of the same job does not time out, and should run as normal. The remoteshell script gets its time from the job wall time, and is not subject to the 5 minutes limit. All critical regions in the prologue script were successfully tested, for all the three Vmbatch contextualization approaches.

In the second scenario, a job times out in the prologue phase and is resubmitted by Torque. Also all reruns of the same job times out. The next new job does not time out. The default Torque scheduler, *pbs_ sched*, does not work well in this scenario. A test was performed with *pbs_ sched* and a job wall time of 72 hours. After 116 hours, and 1279 attempts, *pbs_ sched* still tried to run the job. The *maui* scheduler will try to run the job a couple of times, and then put the job back in the queue, and the Vmbatch job slot is released. From then on, *maui* will retry the job approximately every hour, and eventually abort the job. Using the *maui* scheduler, timeout in all the critical regions was tested, for all three Vmbatch contextualization approaches. All the jobs that did not time out succeeded. Vmbatch can be configured to abort a job if exceeding a set number of runs on the same host. The prologue timeout test was redone with the default Torque scheduler, *pbs_ sched*, with Vmbatch set to abort a job after 4 failed attempts on the same host. In this setup all the jobs that did not time out succeeded.

Reliability

A viable solution must be able to run for longer periods without manual intervention. The reliability of the Vmbatch system appeared to be the most difficult to achieve. When running the system for several days, and after successfully running thousands jobs, the system would fail. Several problems were discovered in the initial attempts to create a reliable system.

The long running tests caused Libvirt to exhaust the host memory. Each guest, after being removed, increased the Libvirt memory consumption. For a CernVM guest, the memory increase was approximately 20 MByte. When Libvirt exhausts the host memory, it will be killed by the kernel (OOM killer). This can create a Xen Zombie. The Xen Zombie is invisible to Libvirt, but take up the assigned resources.

When Libvirt is killed by the kernel, it can not always be restarted, and the host system must be rebooted. The existence of Xen Zombies can cause the host to ignore a normal reboot, and a forced host reboot might be necessary. A few cases were observed where the SSH connection into the guest failed, and a few guests failed to mount the NFS share with the home of the job submitter.

Vmbatch has been made reliable by considering all the observed problems. Vmbatch will monitor memory consumption of the hypervisor services, and restart the services if above a set threshold. The hypervisor services are also restarted if they do not successfully respond to Vmbatch calls. When creating a new guest, Vmbatch always first removes Xen Zombies, if found. Vmbatch also allows several attempts when restarting the hypervisor services, mounting NFS shares and connecting to guests with SSH.

The improved version of Vmbatch successfully passed all the tests for reliability, and produced no error messages. The total run times for each of the tests can be found in tables 1 and 2. The tables also include tests where Vmbatch downloads guest images from the net, and creates the disk for the CernVM-FS cache. These tests also run without problems.

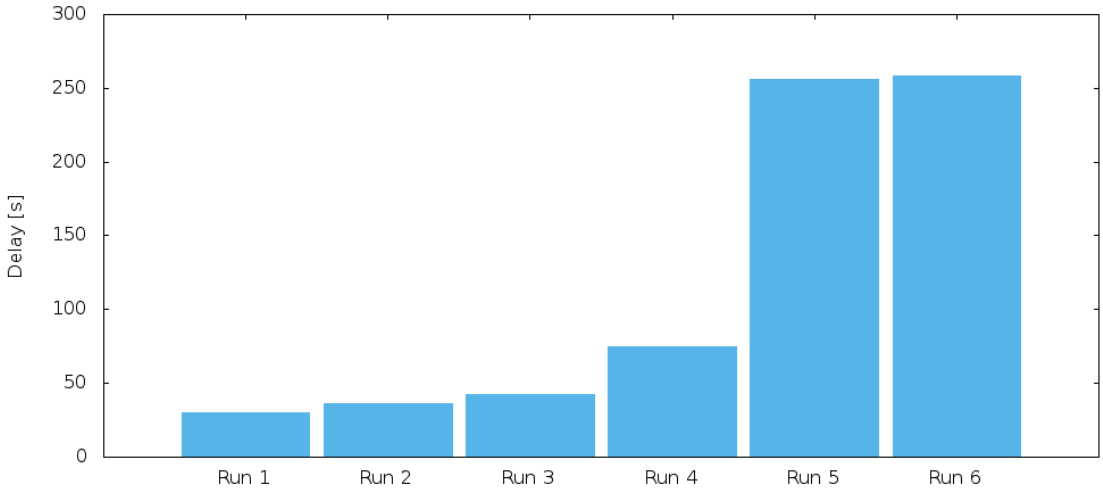

Figure 2: Average job delays for different Vmbatch configurations. See text and tables for details.

Table 1: Vmbatch contextualization approaches

Run	Vmbatch contextualization	Number of jobs	Run time
1	CDROM	20 000	140h, 10min
2	DISK	20 000	167h, 26min
3	ROOTSSH	20 000	152h, 43min

Job run times

An earlier work [7] has shown that the run time of grid jobs with Xen is close to native running on the computer. The use of CernVM does bring a penalty [23], much of which can be attributed to the use of the CernVM-FS network file system. The robustness tests of this study have produced numbers for job delays that can influence the decision whether or not to use the system.

The Vmbatch initialization of the guest, and the guest startup time does introduce a delay before the worker can start to execute a batch job. Figure 2 shows the average delays for different setups and tables 1 and 2 outline the details for each of the test runs. The job delay is measured as the difference between two timestamps. The first is set when Vmbatch sees a job for the first time, and the second is set when the job starts executing inside the virtual machine.

Runs 1 to 3 test the three different Vmbatch contextualization approaches. They all use a CentOS6 guest and 20 000 jobs. Runs 4 to 6 use a CernVM image when running jobs. In runs 4 and 5, the CernVM image is downloaded to the host before running jobs through Vmbatch. In run 6, the CernVM image is downloaded by Vmbatch for each batch job independently. In run 4, the CernVM-FS cache is created prior to running the first job, whereas in runs 5 and 6, the CernVM-FS cache is created by Vmbatch for each job independently.

With the AliEn grid system, the batch job that is sent to and run by the batch system is a job wrapper, the Job Agent. The Job Agent communicates with the grid system and retrieves grid jobs. This job wrapper typically runs for 48 hours. Each

Table 2: CernVM configurations used for robustness tests

Run	CernVM preparation	CernVM-FS cache	Jobs	Run time
4	Downloaded in advance	Prepared in advance	20 000	249h, 53min
5	Downloaded in advance	Prepared for job	2 969	119h, 23min
6	Downloaded for job	Prepared for job	619	24h, 16min

ALICE grid job typically runs for 1-2 hours. As long as there is a Job Agent running, the use of virtualization does not introduce a delay before the start of a grid job. For the site administrator, the delays shown in figure 2 are therefore insignificant. If using a CentOS6 image, the use of Vmbatch should not deteriorate the throughput of grid jobs. Also with CernVM, the delays are insignificant, but as shown in [23], the use of an empty CernVM-FS disk cache will increase the job run time. As the Job Agent is running more grid jobs, more grid files will be found in the cache and the grid job run times should improve.

For the user who submits a grid job that is run by a new Job Agent, the job delay introduced by Vmbatch might be visible, but since grid jobs typically have long running times, the delay should not be important. The aging effect might also be observed by grid users who sees that similar grid jobs can have different run times depending on the age of the Job Agent (as more files get cached during the life time of the agent).

9 Conclusions

The Vmbatch system is proven to be a robust and reliable lightweight tool for running batch jobs inside virtual machines. Vmbatch can work with a stock CentOS6 or CernVM guest image without prior adaption, and can also download the CernVM image from the net. The CernVM-FS cache can be prepared by the system for each job independently, or by a pilot job for all later jobs.

Vmbatch has workarounds for problems discovered in the hypervisor services and the Torque batch system. The system is easy to install, and it requires very little manual maintenance, as the system is able to recover from problematic jobs and hypervisor failures.

References

[1] K. Aamodt et al, *The ALICE experiment at the CERN LHC*, 2008 JINST 3 S08002

[2] S. Bagnasco et al.,*AliEn: ALICE environment on the GRID* Journal of Physics: Conference Series 119 (2008) 062012

[3] I. Bird, *Computing for the Large Hadron Collider*, Annual Review of Nuclear and Particle Science, Vol. 61: 99-118, November 2011

[4] M. Altunay et al., *A science driven production cyberinfrastructure: the OpenScience Grid*, Journal of Grid Computing, Volume 9, Number 2, 201-218, 2011

[5] J. Blomer, C. Aguado-Sánchez, P. Buncic, A. Harutyunyan *Distributing LHC application software and conditions databases using the CernVM file system*, Journal of Physics: Conference Series 331 (2011) 042003

[6] P. Buncic, C. Aguado Sanchez, J. Blomer, L Franco, A. Harutyunian, P. Mato, Y. Yao, *CernVM - a virtual software appliance for LHC applications*, Journal of Physics: Conference Series 219 (2010) 042003

[7] B. Kileng, B. Wagner: *Dynamic virtualization tools for running ALICE grid jobs on the Nordic ARC grid*, I: NIK 2013. Akademika forlag 2013 ISBN 978-82-321-0365-2. 158-161

[8] C.P. Wright J. Dave, P. Gupta, H. Krishnan, D.P. Quigley, E. Zadok, M.N. Zubair, *Versatility and Unix Semantics in Namespace Unification* 2004, Tech. Rep. FSL-04-01b Stony Brook University

[9] http://aufs.sourceforge.net

[10] J Blomer, D Berzano, P Buncic, I Charalampidis, G Ganis, G Lestaris, R Meusel and V Nicolaou, *Micro-CernVM: slashing the cost of building and deploying virtual machines*, J. Phys.: Conf. Ser. **513** (2014)

[11] http://ccl.cse.nd.edu/software/parrot/

[12] http://cloudscheduler.org/

[13] M Storetvedt, *Applying Cloud Technologies for Grid Analysis*, Master thesis at Bergen University College, 2016.

[14] https://github.com/dberzano/elastiq, D Berzano, J Blomer, P Buncic, I Charalampidis, G Ganis, G Lestaris and R Meusel *PROOF as a Service on the Cloud: a Virtual Analysis Facility based on the CernVM ecosystem*, Journal of Physics: Conference Series 513 (2014) 032007

[15] http://www.nimbusproject.org/

[16] https://research.cs.wisc.edu/htcondor/

[17] http://www.adaptivecomputing.com/

[18] http://www.openlava.org/

[19] http://libvirt.org/

[20] http://eple.hib.no/svn/vmbatch/tags/latest/

[21] https://ekptrac.physik.uni-karlsruhe.de/trac/BatchVirt/wiki/ViBatch

[22] IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology, 1990.

[23] B. Wagner, B. Kileng: *Exploring virtualisation tools with a new virtualisation provisioning method to test dynamic grid environments for ALICE grid jobs over ARC grid middleware*, J. Phys.: Conf. Ser. **513** (2014)