Dynamic virtualization tools for running ALICE grid jobs on the Nordic ARC grid

B Kileng[1], B Wagner[2], for the ALICE Collaboration

[1] Høgskolen i Bergen, [2] Universitetet i Bergen

Bjarte.Kileng@hib.no, Boris.Wagner@uni.no

September 16, 2013

Abstract

This paper examines different virtualization technologies with the aim of running ALICE grid jobs on the ARC middleware which is used as internal middleware for the Nordic Tier-1 centre. The Vmbatch system, which allows dynamic management of virtual worker nodes, is presented. First tests confirm that management of virtual nodes cause minimal overhead for a typical ALICE simulation job.

1 Introduction

Grid computing forms the backbone of the processing infrastructure for storage and analysis of data measured by the LHC collider at CERN. The experiments generate order of tens of Petabytes per year. ALICE [1] has developed the AliEn [2] middleware platform, which is also applied by ALICE as a common interface to a range of underlying middleware systems within the Worldwide LHC Computing Grid project (WLCG) [3].

The Nordic countries have set up a distributed centre using the grid middleware ARC (Advanced Resource Connector) [4] as internal middleware. The distributed storage element is based on dCache [5], which has been extended also to operate over wide area networks.

The AliEn system does not have a transparent interface to the ARC job management, so the internal centres in NDGF appear as separate entities in the ALICE grid. This project is an approach to create an interface, allowing also the NDGF Computing Element (CE) to appear as a single entity.

2 Virtual Machines with AliEn

One of the big challenges of grid computing is the provision of software packages to heterogeneous computer systems. To approach this issue a virtual machine can be provided with the necessary operating system and application stack, including all needed libraries.

This paper was presented at the NIK-2013 conference; see http://www.nik.no/.

To evaluate different virtualization technologies a test-bed system called Vmbatch was developed. This system takes some code from an existing system, ViBatch [7]. Vmbatch uses Xen as virtualization method. It is based on the libvirt library and can be extended to support other hypervisors.

The system uses Torque as the cluster resource management system [8], from which it uses its prologue and epilogue scripts to start and stop the guest. A shared NFS file system is used for data exchange between a host and its guest systems. This can be eliminated in future versions by using the existing staging mechanism in Torque. There is no site specific configuration to be done to the guest systems. The guests must be able to mount the shared NFS file system. The system also has requirements concerning the guest disk layout.

The Vmbatch system

Separate queues in Torque should be assigned to the Vmbatch system. A job from any of these queues will cause Vmbatch to start a virtual machine on the worker node running the job. The job will be run inside this newly created guest system by using an SSH connection into the guest. Jobs belonging to different Torque queues can use different sets of disk templates and different system configurations.

The site should provide a shared NFS file system with the home directory of the submitter. The Vmbatch installation must be accessible on all worker nodes and should preferrably be installed on the NFS share. Each worker node needs a local Vmbatch directory for storing disk images, logs and lock files. The worker node must run a Xen kernel and should provide a network via libvirt to its guests. The host system can enable network address translation (NAT) between the host and its guest, but a bridged network will work as well.

Qemu Copy On Write disk images (qcow) [9] are used to provide a fast method for copying large disk images, as only the difference from the existing backing image must actually be stored. The bootloader used by Xen on Scientific Linux 5 does not handle the qcow format. The recommended disk setup, therefore, uses at least two disk templates. One disk template should be a raw disk image configured read-only in Xen. This image should contain only the boot partition which must be configured read-only at the guest.

When a job arrives, a Torque prologue script will create new disk images based on the template images. The prologue script will then prepare the new disk images for use by Vmbatch. This step will set up Vmbatch specific system services inside the guest and configure the network of the guest. The Vmbatch specific system services will create the submitting user, mount the NFS share and create a specific file on the NFS share as a signal to the prologue script that the guest has started. When the guest is up and running, the prologue script will exit.

The execution shell of the job will connect with the guest and run the job inside the guest. When the job is finished a Torque epilogue script will destroy the guest, delete the disk images and clean up lock files.

Test run

For the test jobs we are using a standard ALICE benchmark test program which is CPU intensive and without much i/o.

3 Discussion of Results

The test jobs were run using different virtualization technologies. The results are shown in figure 1. They are all are produced on the same computer with a dual core CPU which is virtualization compatible. Scientific Linux 5 (SL5) and 6 (SL6) were chosen as the host/native and guest operating systems. In each run, the same operating system is used for the guest and its host.

Scientific Linux 6 does not ship with a Xen kernel which explains the missing column in panel (b).

From figure 1 we see that the overhead of virtualization is small. As expected, paravirtualized Xen provided results close to the native tests.

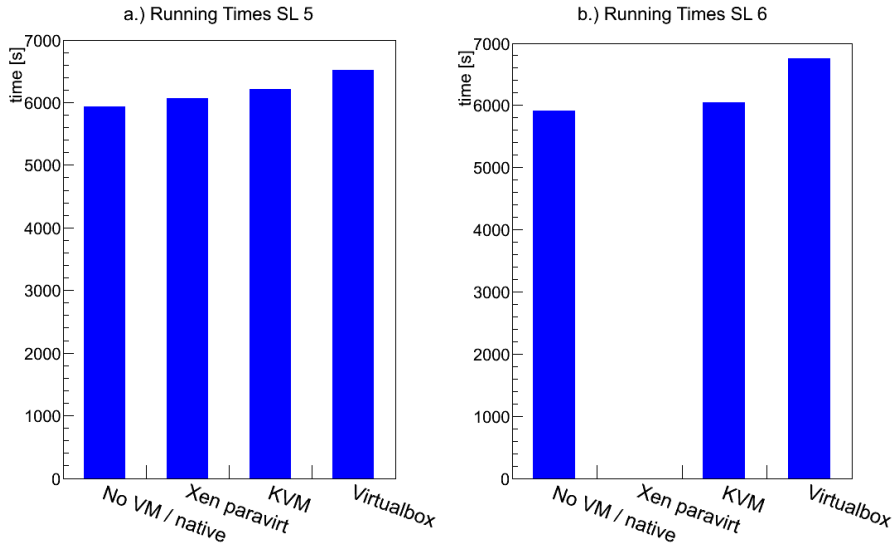

Figure 1: Execution times of ALICE benchmark test program on different virtualization technologies

With the new Vmbatch system in place, performance measurements using simple unix tools like time and date, were done for the native case and with Xen. The computer running the Vmbatch system is an older and slower computer than the computer used to produce the numbers in figure 1.

Performance

The primary performance measure in our case is the running time of the same testjob with and without virtualization. Also interesting to see is the influence of the creation and destruction of the Vmbatch environment on the running time.

Figure 2 shows the averaged times from several testruns in a native and a virtualized environment.

Each run measures the times for each of the three phases of the job. In figure 2 the column *Xen Prologue* measures how long it takes to setup the virtual environment. The time used to run the job in the virtual environment is shown in the column *Xen Testrun*. The column *Xen Epilogue* shows how long it takes to clean up the environment after job completion.

For comparison, the leftmost column shows how long time it take to run the job in the nonvirtual environment.

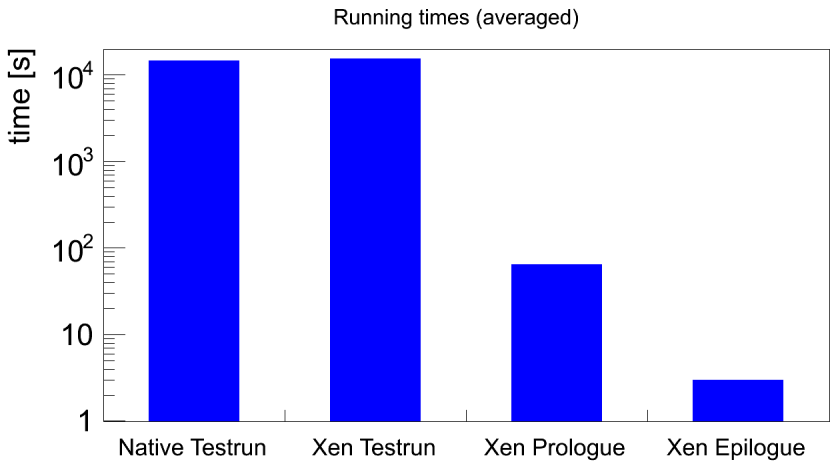

Figure 2: Summary of runtimes for the ALICE benchmark test program

The total run time of a job is clearly dominated by the job running time. The efficient creation and destruction of the virtual environment in less than ≈ 70 seconds and has a negligible influence on realistic run times of grid jobs. The overhead and longer running time of 6.8% of the Xen run is within the expected range.

4 Conclusion and Outlook

This first test of the Vmbatch system shows that it could be a lightweight, performant virtualization provisioning system. It is highly adjustable, but is often usable out of the box without much tweaking.

With the test-bed setup in place, more fine grained performance analysis can be performed, including disk and network I/O measurements. A realistic use case has been proven successful.

References

[1] K. Aamodt et al., 2008 JINST **3** S08002

[2] S. Bagnasco et al., J. Phys.: Conf. Ser. **119** (2008) 062012

[3] I. Bird, Annual Review of Nuclear and Particle Science, Vol. 61: 99-118, November 2011

[4] M. Ellert et al., Future Generation Computer Systems 23 (2007) 219 2013240

[5] G. Behrmann et al, 2008 J. Phys.: Conf. Ser. **119** 062014

[6] Subversion repository `http://eple.hib.no/svn/vmbatch/tags/latest/`

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

[8] G. Staples *Torque resource manager*, SC '06 Proceedings of the 2006 ACM/IEEE conference on Supercomputing, 2006

[9] `http://www.linux-kvm.org/page/Main_Page`, seen June 2013