Use of containerisation as an alternative to full virtualisation in grid environments.

Robin Long
Physics Department, Lancaster University, Lancaster, UK, LA1 4YB
E-mail: r.long@lancs.ac.uk

Abstract. Virtualisation is a key tool on the Grid. It can be used to provide varying work environments or as part of a cloud infrastructure. Virtualisation itself carries certain overheads that decrease the performance of the system through requiring extra resources to virtualise the software and hardware stack, and CPU-cycles wasted instantiating or destroying virtual machines for each job. With the rise and improvements in containerisation, where only the software stack is kept separate and no hardware or kernel virtualisation is used, there is scope for speed improvements and efficiency increases over standard virtualisation. We compare containerisation and virtualisation, including a comparison against bare-metal machines as a benchmark.

1. Introduction
Cloud technology comes in many forms including: “Software as a Service”, whereby users are provided with a pool of computers on which they can run pre-installed software; “Platform as a Service”, in which a computing platform, where software can be developed, compiled and run, is provided; and “Infrastructure as a Service”, the most basic setup where users are provided with a piece of hardware or a hypervisor on which Operating Systems can be installed or Virtual Machines can be deployed.

The Grid already provides cloud computing in the form of “Software as a Service” and “Platform as a Service”. There is a desire to make more efficient and opportunistic use of resources as the needs place on the Grid grow. An example of such resource use comes from the High Level Trigger (HLT) on the ATLAS experiment. HLT is a massive computing farm dedicated to filtering events from ATLAS (or other experiments) triggers. In periods of no data taking, this farm is sat idle for anywhere from a few hours to years (as has happened for the long shut down whilst the LHC was being upgraded). Being able to rapidly provision new VMs onto this hardware that can be converted into a Grid compute resource, and then re-provisioning it with new HLT VMs so that it could act as a Trigger farm again would provide a very efficient use of resources.

The hypervisor, which controls the Virtual Machines, takes up a non insignificant amount of resource on the physical node. Containerisation, whilst being more limited in the operating systems it can “containerise”, has a much lower overhead. This paper therefore examines the ability of Docker[4] to provide a containerised environment and compares the performance of this with a KVM Virtual Machine[5].

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI. Published under licence by IOP Publishing Ltd
Figure 1. A graphical representation of how Bare Metal, Virtualisation (KVM), and Containerisation (Docker) works. To run a service on a baremetal machine we need the kernel, userland (libraries that interact with the kernel), and the software we want to run. When we create a virtual machine, we need to virtualise all of this. Using containers, we only need to virtualise the userland and software as the container's userland interacts with the host kernel.

2. Virtualisation
Virtualisation has been widely adopted by business to enable multiple operating systems, often of differing types (e.g. Linux / Windows), to leverage a single piece of hardware. This has led to large scale reduction of hardware both in terms of servers but also in the connecting infrastructure needed to support such hardware; monitors, keyboards, racks, air-conditioning to name but a few. The ability to run many virtual instances on a single host is attractive in business to save money through hardware savings in software purchases. The rapid provisioning of machines due to the removal of the need to buy a physical machine for each operating system instance is also very attractive.

KVM is one of the leading OpenSource virtualisation solutions; the hypervisor comes in the form of a service installed on a standard linux installation. This allows for the virtualisation of hardware, and the freedom to use any operating system as the guest.

3. Containerisation
Traditional virtualisation and paravirtualisation requires the whole operating system and hardware to be virtualised. Using containers, individual applications can be separated from each other and “virtual machines” can be created without the overhead of traditional setups.

Containers allow the running of individual applications and their dependencies. These, in the case of docker, then run on top of the docker services. Only the userland and applications are virtualised which allows significant overhead savings. This is detailed in Figure 1. As only userland and applications are virtualised, an operating system that requires a different kernel type such as MS Windows or GNU Hurd cannot be containerised on a Linux host.

For this analysis we use Docker which is the leading containerisation solution on Linux.

4. The testing environment
Our setup consists of two compute nodes running Scientific Linux 6. These are of identical specification, each having 16 core Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz CPUs, with 64 GB RAM and 1TB Western Digital Sata Hard drive.

Configuring the compute nodes to act as a KVM hyper-visor was relatively trivial. The standard KVM packages were installed from the repositories. A single virtual machine utilising all available resources was created using virt-manager. A virtual drive was created as part of the host to hold the datasets.

Configuring the compute node to act as a Docker host took a little more work. As it was non trivial to mount CVMFS inside the container, CVMFS was mounted on the host machine and...
then added as a volume to the container by using the “-v” flag on the command line. A single container was created which once again utilised all available resources.

The docker container used was an official Centos 6 container with the necessary grid software installed on top. An external volume called “/data” was used to save analysis output into, and the container was ran interactively. The command line options are listed below where “longr/centos6-chep” is the Centos 6 container with grid software installed.

```
docker -D run -v /data:/data -v /afs:/afs -v /cvmfs/atlas.cern.ch:/cvmfs/atlas.cern.ch -i -t longr/centos6-chep /bin/bash
```

5. Analysis
We wanted to compare virtualisation and containerisation, and how both of these compare with baremetal performance. To do this we used standard tools designed to benchmark critical parts of a system, namely CPU, network and disk access. The tools used were:

- iperf-2.0.5
- bonnie++-1.96
- hep-spec06 (v1.0)

These tools were ran three times, on two different machines with each of the three setups.

5.1. HEP-SPEC
HEP-SPEC06[1] is the HEP-wide benchmark for measuring CPU performance and is based on the SPEC CPU2006 benchmark suite. The HEP-SPEC tests were run three times for each configuration, where more than one guest is running on the host, the scores are summed together to give a total HEPSPEC for the machine.

Figure 2(a) shows that the management overhead due to the hyper-visor is clear for the virtual machine where a significant drop in HEPSPEC score is seen. Containerisation shows a small drop of 1-2 HEPSPECs, but this is within the noise for Bare Metal.
Figure 3. Performance of Bonnie’s write (green), re-write (blue) and read (orange) tests for the various machine configurations.

The lower loss in HEPSPECs for containerisation is consistent with the much lower overhead outlined in Figure 1.

5.2. Bonnie++

Disk access speed is very important in a High End Computing environment. We used Bonnie++[3] to test the disk access speeds. Bonnie++ was set to write to a file four times the size of the RAM available to each VM.

Figure 3 shows a significant drop in all areas (write, re-write, and read) for the full virtualisation, whilst no significant drop is seen in any performance for containerisation when compared to bare metal. This is expected as containerisation does not virtualise the disk, it only writes to a single file on the local disk which requires virtualisation of the file system only.

5.3. IPerf

The separation of CPU from storage disk on grid sites means that network bandwidth is very important. We tested this on the physical and virtual machines using IPerf[2]. The physical machines contained Intel 80003ES2LAN Gigabit ethernet cards. As these test machines are re-purposed production machines, and have not been moved from our production room, all tests were done on a non-dedicated network. Figure 2(b) shows that bandwidth is not obviously affected by virtualisation, changes in value fluctuate a lot between configurations and seem to be due to contention on the network infrastructure rather than to do with the virtualisation process.

6. Conclusion

Virtualisation is shown to have significant but not insurmountable overheads. In contrast, overheads are almost non-existant for containerisation. The most significant overhead in virtualisation comes from virtualising the disk, and this is clear from the Bonnie++ results. Whilst no significant overheads are seen in containerisation, what has not been tested is the cumulative affect when multiple containers are running on the same system.
Containerisation certainly shows promise as an alternative to full virtualisation, although it must be noted that certain cases of virtualisation cannot be containerised, such as those where a different type of OS is required such as Windows on a Linux host.

References
[1] HEP-SPEC06 Benchmark (Accessed 19/05/2015); https://w3.hepix.org/benchmarks/doku.php
[2] IPerf - TCP and UDP bandwidth performance measurement tool (Accessed 19/05/2015); http://code.google.com/p/iperf/
[3] Bonnie++ homepage (Accessed 19/05/2015); http://www.coker.com.au/bonnie++/
[4] Docker (Accessed 19/05/2015); https://www.docker.com/
[5] KVM, Kernel-based Virtual Machine http://www.linux-kvm.org/page/Main_Page