ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-TUDPP04

DATA ACQUISITION AND VIRTUALISATION OF THE CLARA CONTROLS SYSTEM*

R. F. Clarke †, G. Cox, D. M. Hancock, P. W. Heath, B. G. Martlew, A. Oates, P. H. Owens, W. Smith, J. T. G. Wilson, S. Kinder, N. Knowles, STFC Daresbury Laboratory, Warrington, UK

Abstract

STFC Daresbury Laboratory is developing a novel Free Electron Laser (FEL) test facility focused on the generation of ultra-short photon pulses of coherent light with high levels of stability and synchronisation. The first phase of the Compact Linear Accelerator for Research and Applications, CLARA was successfully used to produce electron beam used for experiments during 2018 to 2019. It is planned that CLARA will continue to be exploited for user experiments during development. The next user period will start early in 2020. Work on the EPICS Archiver Appliance, Virtual Accelerator and a mid-level interface were successfully trialled during the last run.

The last run showed new requirements that CLARA will place on the controls system. There was heavy use of data archiving, the request to duplicate the Virtual Accelerator for CLARA simulations and rapid expansion of EPICS soft IOCs for a mid-level interface. This motivated a review of the control system's current infrastructure for the future of CLARA.

Clara's current control system has been tuned for production and is very stable with most services running for years without issue. If virtualisation is to be embedded into this production system it must be stable, recover automatically in the event of power outages and be easy to diagnose.

This paper will discuss how virtualisation of certain services can harden the current control systems against failure. It will discuss how it provides flexible services for users and sandbox environments for developers. It will discuss various tests and challenges in integrating virtualisation into the current control system. Finally, it will discuss how it will act as a backbone to novel projects such as the Accelerator Physics group's machine learning program.

INTRODUCTION

The controls system is designed around a common framework provided by EPICS [1]. EPICS provides a common interface to disparate subsystems via process variables (PVs). PVs generally form the backbone to GUIs for operation of the machine in the control room. With CLARA this landscape is changing.

Due to the complexity, timing constraints and numerous subsystems, CLARA [2] will require a mid-layer level of automation between the experimental and machine level controls.

Traditionally, scripts would be developed by users on the client side to steer EPICS at this level. This is hard to maintain and expand and as these custom scripts become more numerous. They also open the possibility of race conditions if one script is trying to drive PVs in use by another. Errors like this are notoriously hard to trace.

The controls group is working in collaboration with the accelerator physics group to develop a managed layer of abstraction between high level physics and machine operations. This abstraction has been called the mid-level interface. Solutions include adopting a common naming by using an abstract device defined at the physics level with simple states and expanding that to control all the different and related subsystems attached to it. Another solution is to identify and move all critical client-side scripts into the EPICS Input Output Controllers, (IOCs). IOCs developed as part of the mid-level interface exists only to drive EPICS PVs and in this way group together IOCs that drive hardware. The IOCs developed for physics use are called interface IOCs. Interface IOCs can become complicated as they they make heavy use of state machines to drive all the operations and manage the the hardware IOCs to get them into the state expected by the physics operation.

Interface IOCs are used to automate the RF gun conditioning system and deliver a common interface to drive different YAG screens and magnet IOCs controllers over the whole CLARA. Interface IOCs are considerably more stable and easier to implement than client-side scripts.

IOCs for hardware control are currently hosted on dedicated servers. As the mid-level interface starts to accumulate interface IOCs for different use cases, and as they are not limited by hardware, their numbers could rapidly expand and consume servers and rack-space. Virtualising these software interfaces will allow the controls group to contain their proliferation to a couple of hosts and offer the accelerator physics group the flexibility of designing the mid-level interface with minimal constraints.

This trend of integration at a physics level to the machine operation will require more than simple PV data being passed between the Accelerator Physics group (AP) and the controls system. EPICS Version 7 [3] is being investigated as a solution to add structured data to PVs. Controls data will be fed to, and feed from, the AP group's data acquisition and simulations databases. The development of PVs to provide physics engineering units for magnets PVs is a case for EPICS V7. These units are calculated in EPICS from magnet fitting curves provided by AP group but depend on the state of the machine. These are set in soft IOCs monitoring real magnets. The output of these PV are used to update the AP group's database, a set of YAML configuration files [4] that are then used for simulations. Using EPICS V3 PVs for this operation shows that PVs would be better grouped into

* STFC Daresbury Laboratory
† rory.clarke@stfc.ac.uk

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-TUDPP04

more logical objects. EPICS V7 will provide more structured objects for exactly these type of operations. The use of structured objects will increase the complexity of interfaces between IOCs and client code. Virtualised of IOCs and the controls networking would allow the off-line construction of the CLARA controls system where these different interface could be tested.

Large scale data acquisition is another trend that will change the nature of the controls system. During the last run the Archiver Appliance [5] was heavily used in CLARA and had also been adopted by another facility, the Vertical Test Facility [6] to generate reports. The data acquisition ability of the archiver and its easy to use web interface are proving popular with users. In the ALICE facility [7], the archiver collected a few gigabytes of data over a fourteen year period. At CLARA, with an out of the box setup, in one run, 4 Terabytes of data were collected. The Archiver Appliance is easily outperforming expectations; even with waveforms at 100Hz and saving to a NAS drive the archiver has remained stable.

The archiver has been used for various calibration and experimental studies and the data quality has proven to be robust under analysis. The AP group have agreed that the archiver will now become the backbone of its machine learning program by gathering raw machine data. For this to work the controls group needs to be able to provide custom archivers on a case by case basis. As servers, this would take too long to set up to be flexible during operations. As a virtual machine, archivers could be provided as an on-demand service.

Virtualisation, containers, cluster computing and distributed file systems are some of the technologies that have been examined in relation to the controls system. It will allow us to provide redundant and flexible services on the scale needed for CLARA's development.

THE CURRENT LAYOUT

The current control system architecture is shown in Figure. 1. A central server is a key component that delivers a copy of the compiled IOCs and general file system over a NFS. Its filesystem, but not the whole operating system, is mirrored via rsync to a slave server. It also acts as a build server and provides several services needed for the control system to operate such as DHCP and TFTP.

Rebooting the server or network interruptions can leave the NFS mounts in a stale state. Often this is only noticed when the IOC cannot be restarted. The main server has grown and its services need to be distributed to harden the control system. This is not a trivial task on a server that is currently used in several roles. If it were put into a state where it would not reboot the whole system would break.

The group used this server to develop and maintain software for several projects concurrently on this server. The disks were mounted under a RAID array and there is a site backups of the server. This year, the backup system failed. The RAID array broke down over a period of time before shutting down which corrupted the disk array. Unfortunately, rsync copied the corruption to the slave server and the slaves server's operating system was quite far behind the master. The off site backups were inaccessible during this period. If it were not for investigations into virtualisation that were taking place at that time it could have taken weeks to recover. As it was, an experimental virtual copy of the server's operating system could be made. This allowed the server to be brought online. Within a few days of testing the filesystem was recovered from the RAID array and rebuilt. Currently the server is running as a KVM virtual machine mounted on a virtual disk, a file, that is being duplicated every other day. Now that it is virtualised then next stage will be to break the server into several VMs running distinct services.

VIRTUALISING EPICS IOCS

EPICS IOCs are started and managed via procserv [1]. IOCs and procserv have standard port numbers that clients expect to find during operations. A server with a single IP address can not have multiple applications running on the same port number. This means that the CLARA runs IOCs on dedicated servers.

Having an IOC for limited hardware residing on its own server makes sense when setup cost is compared to ease of use during operations. The mid-level interface IOCs are not regulated by hardware and may become numerous as different groups are invented for different physics processes. This would rapidly consume servers and rack space as previously mentioned.

The first attempts at designing the interface avoided using new IOCs and merged records and state machines into existing hardware IOCs. This is a poor idea as it is not clear which hardware IOCs should be used and an unnecessary dependency is created between the interface and specific hardware.

This dependency caused issues during operations. The interface tended to be modified for specific experimental changes. As the interface is embedded into a hardware IOC, the reboots were affecting the hardware set-point that would then require the hardware IOC to be modified.

The next idea was to have a single server with one IOC for all interface operations. With Channel Access gateway [1], many IOCs could run on a single server if they had different port numbers. This is undesirable from an operations point of view as non-standard port numbers are added. At Daresbury, being mainly a research facility, operations are quite often fluid. A magnet may be moved or some other equipment added during a shift. Where time is at a premium, non standard port numbers could cause confusion. It is expected that procserv is on port 7001 for example. It would also suffer from the same issue seen with embedding the interface into a hardware IOC; unrelated interfaces would reboot as this single server's IOC is restarted.

An answer is to use virtualised IOCs. Virtualised systems can use a virtual bridge, a software network switch. It is simple to have several virtual IOCs running with their own

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-TUDPP04

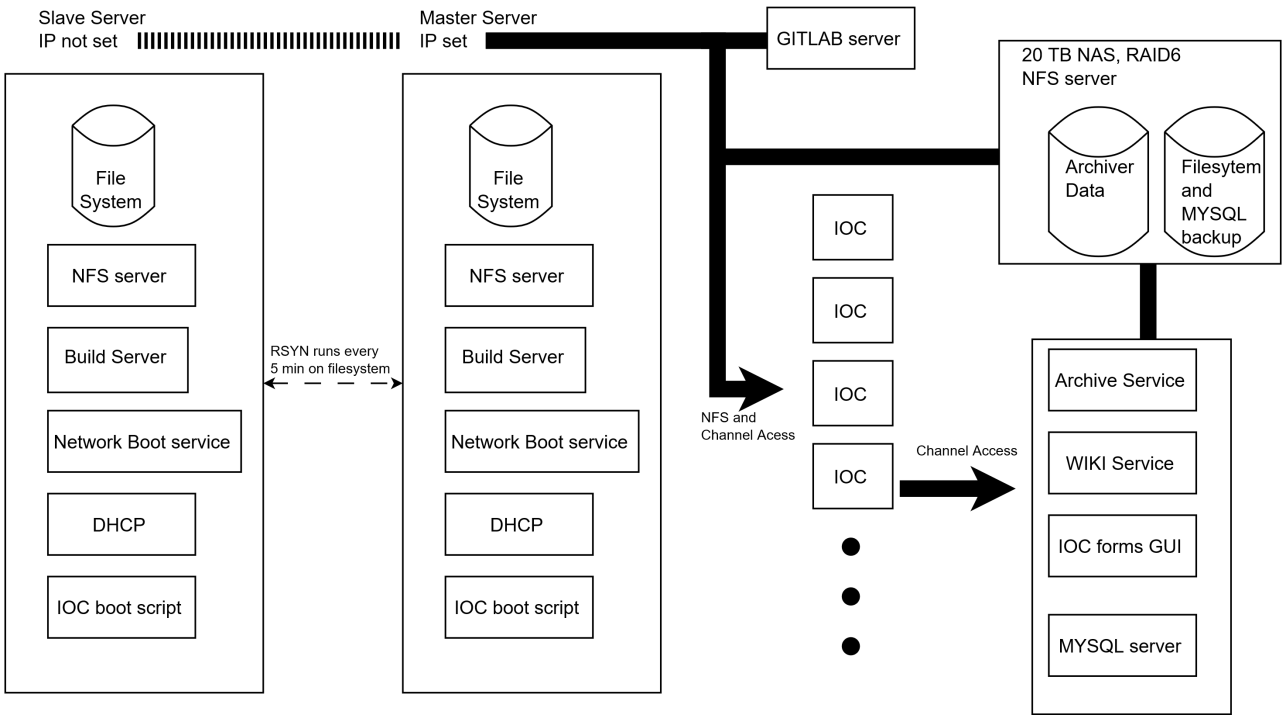

Figure 1: Current control system infrastructure.

IP address and standard EPICS ports on one server with one network connection. Channel Access gateway can also be virtualised and connected to this system. The limit is the network bandwidth but with 1Gbit/Sec, with several interface IOCs running on one server, peak network throughput was seen at just a few percent. From the controls point of view, everything looks the same as individual servers; users can SSH into IOCs or use procserv over telnet as if the IOC is running on a dedicated server. The 'ticks' on Figure 2 show replicated 'VM A' in this state.

There are several advantages to this method. The IOC's can be duplicated with one connected to the network and the other disconnected. During development, the test IOC can be updated without affecting operations. The test and live IOC's network connection can then be switch with the live IOC now offline but still running. The operation can be quickly reversed if issues are seen.

VIRTUAL MACHINES AND CONTAINERS

KVM

There are many virtualisation solutions on the market. Our control system will depend on two types of technology rooted in the modern Linux kernel; the Kernel Virtual Machine, (KVM), and Linux Containers, (LXC).

KVM can hosts a number of native operating systems. Each virtual machine has private virtualised hardware; a network card, disk, graphics adapter, etc. For example Windows OS, Mac OS, even 32 bit operating system can run unmodified in an isolated environment on a native 64 bit host.

Without KVM the virtualisation library has to simulate the hardware which leads to heavy CPU use. KVM works with hardware enhanced virtualisation contained in modern CPUs. This is similar to a 'type 1 hypervisor' such as Xen [8] providing close to native server performance.

KVM still needs to reserve an amount of RAM on the host for each virtual machine. There are methods to distribute the RAM over VMs. Depending on the load, the sum of memory of the VMs can even exceed the host's as most VMs rarely run at full capacity. Still, each VM will use RAM measured in Gigabytes just to hold their operating system. With a good setup and plenty of RAM it was possible to comfortably run about 15 straight Centos7 VMs on a 32 Gig host.

LXC

Linux containers work differently, they do not use CPU virtualisation for speed but use the Linux kernel itself. The Linux kernel is the process through which all interactions with the machine take place. All applications under Linux have a unique process id that is connected by a tree of processes to the root process. In a container, the kernel splits this root process into self contained namespaces. Processes are attached to that namespace tree and multiple machines run in unique namespaces. Each container attaches to its own copy of a root filesystem. Hardware like networks cards and USBs are the same as the host's but now separated in the kernel.

As the container is using host's kernel, it can only run the host kernel and and any kernel modules to drive hardware have to be loaded in the host kernel first. The container must

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-TUDPP04

not alter certain processes otherwise it will directly affect the host. Containers are supplied with default levels of security that need to be tuned, for example, NFS mounts will not work out of the box with LXC.

A Linux Container does not boot the same way a server or KVM would. Booting starts by setting up a connections between the hardware, or virtualised hardware in the case of KVM, and the Linux Kernel. A container attaches directly to hardware via the host's kernel and that part of the boot stage is skipped. Even operating system level services needed by the container are the same as the host's but split into namespaces. All a container will need to do is bringing up its own copy of applications not running on the host. This step is very fast and requires a tiny amount of memory as the whole operating system did not need to be duplicated in RAM.

When an IOC is set running in a container that uses 17MB of RAM the container uses about 18MB on the host. Containers only take up the amount of RAM their unique processes require. Where there may be 20 KVM instances that need a full OS mapped to RAM but run one or two low RAM applications. There can be hundreds of LXC instances. This makes LXC ideal for IOCs, small databases, build servers, etc.

We have picked LXC over another popular container solution Docker [9] due to the design philosophy. Docker containers are only designed to run a single process, say a MySQL server, they are not intended for a full operating system with procserv, telnet, users logins and SSH. LXC is designed for emulating a server and this is what our users would expect to see.

KVM and LXC Deployment

KVM has been used to duplicate existing servers. This is an exact copy of the server and brings the original machine up in a virtual environment.

Copies of the servers can be cloned and services moved about in a safe, isolated environment. Using snapshots, modifications allows changes to be rolled back to working copies. A host running multiple services was broken into different VMs providing single services using this method. The real host could not be modified as it was no longer certain how its services were interlinked and it was in constant use. In a clone of the server the unexpected dependencies were found before deploying VMs to production and removing the physical host.

LXC's efficient use of memory has been selected for deploying IOCs. Most IOCs have a footprint of only a few megabytes so LXC can host many IOCs and small services like the group's Wiki and a MySQL database.

Both KVM and LXC have the ability to replicate and migrate. Replication keeps one copy of the VM on a different host that can be started if the other host stops, see Figure 2, the ticked VMs B and C are running on Host Main. VM A is under development on Host Main so was started on Host Clone. When development is finished VM A can be switched back to Host Main. If Host Main were to fail, the replicated copied on Host Clone can be started.

Migration is a similar process where running VMs can be moved off of one host to another. This is useful if a host needs to be maintained or swapped. KVM can be used to 'live migrate' a running server with no downtime. A copy of the memory is sent over a secure connection so the machine is always 'on'. This feature has been used extensively during development to avoid disruption to users.

With the above considerations, and using a distributed file system discussed later, the current control system seen in Figure 1 could be replaced by the virtualised system seen in Figure 2.

Native, KVM and LXC Performance

Comparisons of CPU and RAM speeds were made for KVM and LXC against a native host, see Table 1. The CPU tests were made by performing matrix operations for five minutes. The memory tests were made by writing and reading from 1G of RAM for five minuets. These tests are part of the *stress-ng*[10] package. The tests in both cases are are within a few percent of the native host.

Table 1: Comparison of LXC and KVM vs. Native Machine

	Native	LXC	KVM
CPU	100%	99.0%	97.5%
Memory	100%	100%	92.0%

VIRTUAL DISKS

A physical drive attached to a VM are simply files and this opens interesting possibilities for server backup and recovery. To the VM or container, the file will mount as if it were a real SCSI drive. It can be formatted and partitioned. These files are stored on a common file system accessible to all the hosts via NFS. Care must be taken to only mount only one of these virtual drive as read-write to one VM. Mounting a virtual drive read-write to multiple VMs will corrupt it.

With native servers, backup of the host OS involved taking a server offline and using Clonezilla [11] to copy the drive over the network. This could take a lot of time and the group ended up with several incompatible versions for the Clonezilla disks. With a virtual drive, backup is simply a case of copying a file and this can be done live.

Two types of virtual drive are used in the system, raw image and QCOW2 [12]. A raw image file is as big as the drive and a simple binary image. QCOW2 is more sophisticated than a raw image, the drive image may be a Terabyte but the file will only take up the size of files stored within it. Also, a snapshot can be taken where only a copy of changes are made. This is much smaller then a full backup if only a few files change between snapshots. This is called Copy on Write.

QCOW2 is used for the KVM servers. KVM will have copies of the full operating system so there are a lot of small files that do not change often. The biggest changes are when

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-TUDPP04

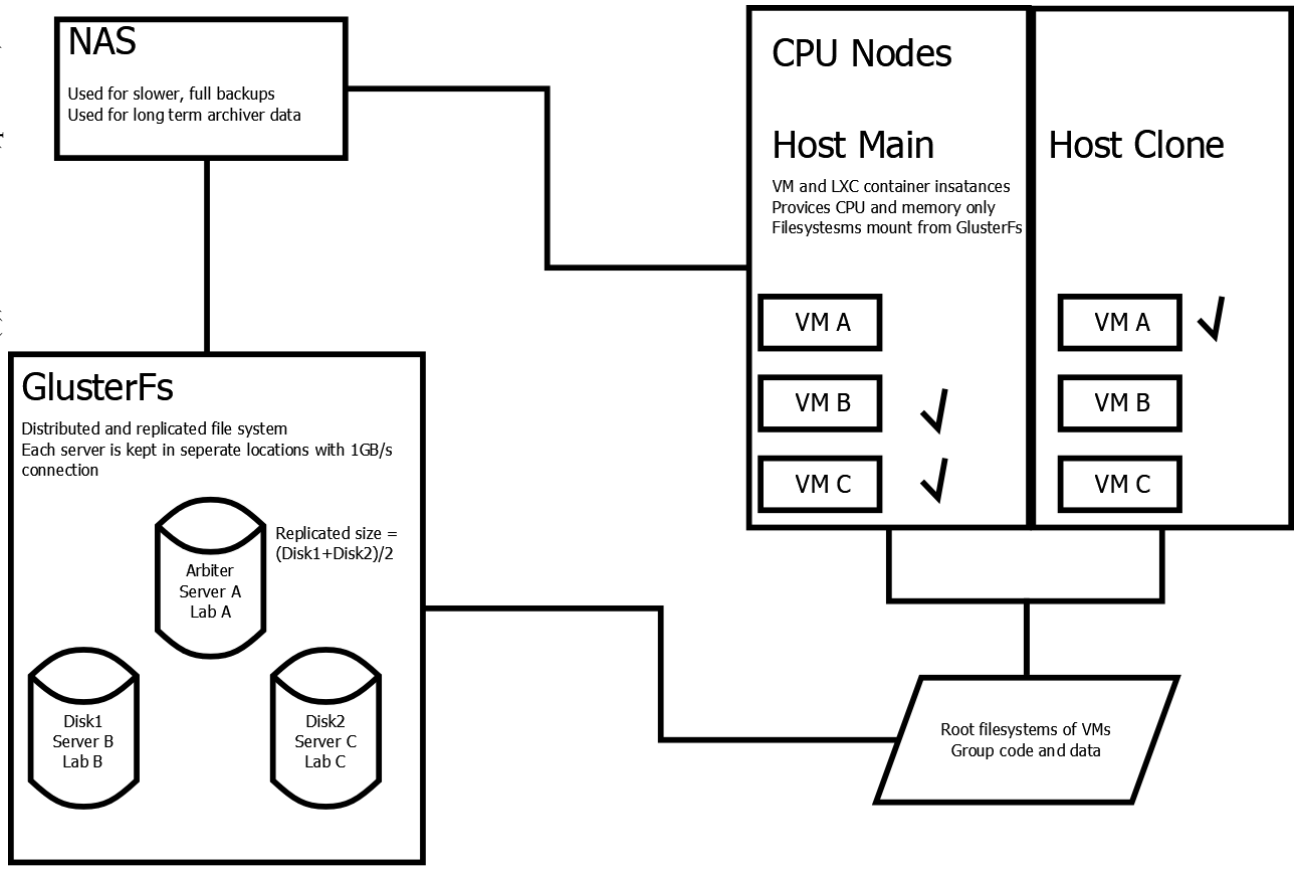

Figure 2: A virtualised control system layout as a replacement of Figure 1.

installing new software or making updates. Snapshots help here as changes can quickly be rolled back. On a native server this process can be tricky to undo and if an update fails the system could fail to boot.

Raw images are used for LXC as that was only option under the current framework. It not much of as issue as the disks are kept as small as possible. Generally, they are only a few gigabytes and are used to run IOCs. IOCs operating systems are are generally just copies of each other with different IP addresses. Scripts have been developed that can set up LXC IOCs automatically.

NFS AND DISTRIBUTED FILE SYSTEMS

The disks for the virtualised machines are stored on a 20TB Network Storage server with RAID6 and accessed via NFS. This also contains the data store for archiving. The groups software is kept on another server mounted using NFS.

The set-up has proven stable over the years but NFS can sometimes enter a stale state where the connection breaks and the IOCs carries on running. This problem is only noticed when the IOC is restarted.

NFS mounts have been tuned over time across clients. Mount setting can differ but no performance problems have been seen on the current system. The heaviest load is from users connected directly to the file system as read-write using it to compile software. The network's maximum theoretical rate is 128MB/s for a Gigabit connection and NFS is seen to peak at about 90MB/s which seems reasonable.

Using VMs that require read/write access will present our current NFS as a single point of failure. Configuration quirks may cause a problem with continuous read-write access needed by VMs.

One solution to NFS as a single point of failure is to use a distributed file system. This replicates files across several servers, in effect, the servers become a network connected redundant storage. If one server disconnects files will feed from another server seamlessly. The servers can be located on different parts of the site to protect against physical failure such as power cuts or even a fire in the rack.

Distributed storage systems are made from simple Linux servers with disks attached. There are a lot of solutions to choose from. Our requirements are for a system that can be easily expanded, that has redundant storage and can make snapshots of the whole file system. There were two solutions investigated; Ceph [13] and Gluster [14]. At the time of testing Ceph exceeded our network requirements so Gluster was investigated.

The minimum amount of Gluster servers needed for safe replication is three, see Figure 2. Any less than this can lead to a situation where if one server goes down, closely followed by another it becomes impossible to synchronise

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-TUDPP04

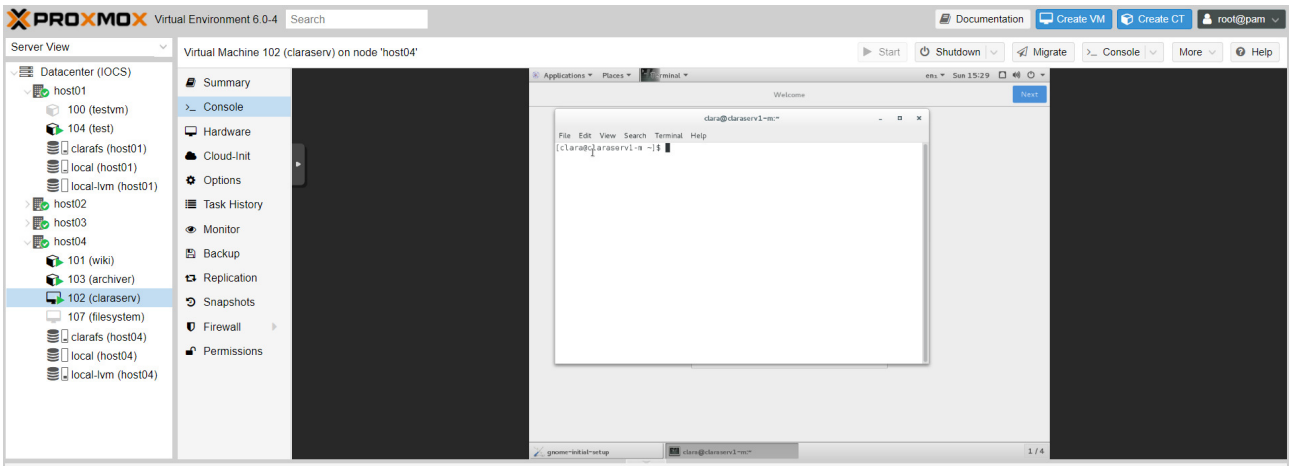

Figure 3: The Proxmox web GUI showing a console to a virtual machine. On the left all the active nodes and file-systems health can be seem. Active nodes are marked with a green tick. This visual type of management is much faster to use in a production environment.

the files when they are restarted. This is called 'split-brain'. By having an odd number of replication servers this situation is recoverable. [15].

Read and write comparisons between NFS and Gluster were made using the linux tool 'dd' to write 1GB of random data to the disk. 'dd' was used with the 'direct' option to avoid memory cache and data was copied in blocks of 1kB and 10MB. To further ensure genuine disk I/O and that we are not measuring not memory cache, each test started with: `sync; echo 3 > /proc/sys/vm/drop-caches`. The results are shown in Table 2.

Table 2: Peak rates of NFS and Gluster Reading and Writing 1G of Data in 1kB and 10MB Blocks

Block 1K	Read MB/s	Write MB/s
NFS	91	78
Gluster	70-80	7
Block 10M	Read MB/s	Write
NFS	111	110
Gluster	96	16

In live tests, Gluster was considerably slower than a normal NFS mount for compiling, see table 3. It did not seem to be a network load issue so more investigation is needed. If Gluster cannot be improved the next idea is to test Ceph. Ceph is being used at CERN to store Petabyte levels of data [16]. If both solutions prove too complex to set up for the control system then another option is distribute an NFS connection between two servers using Corosync [17] and Pacemaker [18].

Table 3: NFS and Gluster Compile Time Test

NFS	Gluster
3 second	45 seconds

MANAGING THE HOSTS WITH PROXMOX GUI

The initial set-up was difficult to use. The interface was all through the command line and there were different commands needed for KVM and LXC for similar operations such as starting, stopping and migrating VMs. The only way to manage this was to use scripts. The scripting then started to become a very time consuming itself.

Proxmox [19] provides a user friendly interface to manage a lot of the operations that we were trying to script in-house. It is open source, free to use but with a subscription model for support and enterprise level libraries. So far this has not been needed. It is installed on a native server from a USB drive and set up with an IP. The server is now one of the Proxmox nodes. Several nodes can be set up in this way and accessed through a web based GUI. Nodes can be clustered together so that any web GUI will show the state of all the other nodes.

Proxmox provided an easy interface for setting up a common NFS drive to host VMs and containers. It managed the backup and snapshots of the drive images and offers and automated backup based on dates and time settings. A nice feature was to limit how many backups should be kept and this way a rolling four weekly, six month backup procedure can be set up.

The Web GUI makes several operations a lot easier such as setting up network bridges and building VMs or containers. It makes monitoring disks, network and CPU loads easier with web charts. It also helps with management tasks such as starting and connecting devices to a software bridge.

Two features that saves a lot of time were the a web based terminal and the migration function. The web based terminal will connect even if SSH is not running or the VM had no IP address, see Figure 3. the second feature was the ease of migrating VMs and containers with the Proxmox GUI compared to using scripts.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-TUDPP04

FURTHER WORK

More work will need to be done to see if Gluster, Ceph or a dynamic NFS setup is best for a distributed file system. IOCs are built in LXC with a series of scripts, these need to be tested to see if they are user friendly.

The Accelerator Physics Group are requesting web interfaces for the virtual accelerator and the archiver appliance. Also, within the group we currently keep track of IOC and other services via a small database and a web GUI. This could be integrated into the IOC container build process.

CONCLUSION

Virtualisation has been achieved at production level in the CLARA control system using a combination of Kernel-based Virtual Machine, (KVM), and Linux containers, (LXC). KVM has been used to replace and replicate physical servers with heavy workloads such as file systems and build servers and simplifies operations needed to clone, upgrade and maintain servers.

LXC, with a small memory footprint in proportion to the applications it serves is useful to host small services such as IOCs and databases. It will also be able to host the IOCs used for the Mid Level Interface which are likely to proliferate as CLARA grows.

Both KVM and LXC have been shown to use CPU and memory within only a few percent difference to a native server.

Building a replicated file system to remove the single point of failure presented by NFS mounts using a Gluster file instead of the current network storage still requires investigation and tuning before it is production quality.

Management of the system becomes involved as more hosts or virtualised machines are added. The Proxmox GUI has proved useful as a simple development tool that users can understand

These systems will harden the controls network to failure, simplify disaster recovery and will allow us to supply novel new services such as “Archiving-on-Demand”. The Archiver Appliance proved remarkably successful during the last exploitation period but required extensive set up on individual servers. Virtualising the archiver will allow easy management of data storage, CPU and memory resources. It will allow isolation of critical archiver services and the possibility for users to rapidly "spin-up" custom archivers.

ACKNOWLEDGEMENTS

With thanks to the STFC Laboratory, Warrington, UK

REFERENCES

[1] EPICS, http://www.aps.anl.gov/epics/
[2] J. Clarke *et al.*, “The conceptual design of CLARA, a novel FEL test facility for ultra-short pulse generation”, pp. 496–501, in Proc. FEL’13, New York, NY, USA, Aug. 2013, paper WEPSO04.
[3] EPICS V7, https://epics.anl.gov/base/R7-0/
[4] YAML markup language, https://yaml.org/
[5] Archiver Appliance, https://slacmshankar.github.io/epicsarchiver_docs/
[6] Vertical Test Facility, https://stfc.ukri.org/news-events-and-publications/whats-happening/daresbury-hits-new-heights-with-vertical-test-facility/
[7] F. Jackson *et al.* , “The Status of the ALICE Accelerator R&D Facility at STFC Daresbury Laboratory”, in *Proc. 2nd Int. Particle Accelerator Conf. (IPAC’11)* , San Sebastian, Spain, Sep. 2011, paper TUODA03, pp. 934–936.
[8] Xenproject, https://xenproject.org
[9] Docker, https://www.docker.com/
[10] stress-ng project, https://wiki.ubuntu.com/Kernel/Reference/stress-ng
[11] Clonezilla, http://clonezilla.org
[12] KVM Qcow2, https://www.linux-kvm.org/page/Qcow2
[13] CEPH file system, https://ceph.io/
[14] Gluster file system, https://www.gluster.org/
[15] Three servers and split brain, https://access.redhat.com/documentation/en-us/red_hat_gluster_storage/3/html/administration_guide/sect-managing_split-brain
[16] D. van der Ster and A. Wiebalck, “Building an organic block storage service at CERN with ceph”, *Journal of Physics: Conference Series*, vol. 513, no. 4, p. 042 047, Jun. 2014. DOI: 10.1088/1742-6596/513/4/042047.
[17] Corosync, http://corosync.github.io/corosync/
[18] Pacemaker, https://wiki.clusterlabs.org/wiki/Pacemaker
[19] Proxmox, https://www.proxmox.com