SANDIA REPORT

SAND2017-10668
Unlimited Release
Printed October 2017

A Reference Architecture for Emulytics™ Clusters

John Floren, Jerry Friesen, Craig Ulmer, and Stephen T. Jones

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology and Engineering Solutions of Sandia LLC.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2017-10668
Unlimited Release
Printed October 2017

A Reference Architecture for Emulytics™ Clusters

John Floren
Cyber Systems Assessments
MS 0932
P.O. Box 5800
Albuquerque, NM 87185-9999
jfflore@sandia.gov

Jerry Friesen
Scalable Modeling and Analysis
MS 9152
P.O. Box 969
Livermore, CA 94551-0969
jafries@sandia.gov

Craig Ulmer
Scalable Modeling and Analysis
MS 9152
P.O. Box 969
Livermore, CA 94551-0969
cdulmer@sandia.gov

Stephen T. Jones
Cyber Initiatives
MS 0621
P.O. Box 5800
Albuquerque, NM 87185-9999
stjones@sandia.gov

Abstract

In this document we describe a reference architecture developed for Emulytics^{TM} clusters at Sandia National Laboratories. Taking into consideration the constraints of our Emulytics software and the requirements for integration with the larger computing facilities at Sandia, we developed a cluster platform suitable for use by Sandia's several Emulytics toolsets and also useful for more general large-scale computing tasks.

Acknowledgment

This work was made possible by funding from the Mission Area Alignment Initiative (MAAI). The funding was used to organize workshops, perform research, and run small-scale experiments with the overall goal of determining a useful reference standard for an Emulytics^{TM} cluster.

Contents

1 Introduction 9
2 Existing Platforms 10
2.1 minimega 10
2.2 Firewheel 11
2.3 Commonality 12
3 Reference Architecture 13
3.1 Physical Layout 13
3.2 Network 13
3.3 Nodes 15
3.4 Power Distribution 15
4 Software Configuration 16
4.1 Head Node Configuration 16
4.2 Compute Node Software 18
4.3 Switch Configuration 18
5 Future Work 20
6 Conclusion 21

1 Introduction

For nearly a decade Sandia has experimented with running large numbers of virtual machines in virtual network topologies in order to perform experiments for cyber security research, training, and testing. This experimentation has given rise to a collection of tools and practices known as Emulytics™. Although early experiments were performed on desktop and laptop computers, Emulytics quickly transitioned to HPC-style clusters in the interest of greater scale.

As Emulytics has matured as a field, the number of people performing experiments has increased. This has put a heavy load on the relatively ad-hoc clusters currently used for Emulytics, which started out supporting a small number of closely-communicating researchers and were strained by the technical and organizational challenges of supporting a wider community. Sandia identified a need for a corporate-managed Emulytics cluster resource. In specifying the requirements for that cluster, we have taken the opportunity to identify key requirements and features for Emulytics clusters in general.

The aim of this document is to provide information on what we consider an exemplar Emulytics cluster, one well-suited to a variety of Emulytics tasks. We intend it to act as a set of guidelines for organizations planning their own Emulytics cluster builds, to help them avoid pitfalls and challenges Sandia has already solved.

2 Existing Platforms

Sandia has been building ad-hoc clusters for Emulytics for nearly a decade now. These clusters have generally been small affairs, assembled by their users and customized for the particular software they will run. In developing a standardized architecture, we had to consider the needs of Sandia's two primary Emulytics toolsets (minimega and Firewheel) and develop something suitable for both.

2.1 minimega

minimega is an Emulytics tool developed with rapid deployment in mind. The initial work on minimega took place on a cluster whose nodes had no hard disks at all–they netbooted from the head node, ran from a ramdisk, and were wiped clean by a simple reboot. With that in mind, minimega was designed to be a single standalone binary. Deploying minimega is a matter of copying the binary to a node and starting it; there are no configuration files.

Clusters built specifically for use with minimega tend to use the igor reservation system. Distributed with minimega, igor allows users to reserve nodes and specify a kernel+ramdisk pair for those nodes to boot. This allows users to boot environments customized to their particular experiments.

minimega uses VLAN tagging to define virtual networks in an experiment. This could lead to conflicts if two users on the same cluster attempt to use the same VLANs. Initially, this problem was solved informally: users would communicate with each other which VLANs they were using. As more people came to use minimega, the custom cluster reservation tool was modified such that it enabled QinQ encapsulation for each reservation, preventing users from interfering with one another.

Summarized, minimega's preferred environment includes:

- Net-booted nodes with no persistent OS installation (this allows the cluster to be easily 'reset' to a known-working state).
- A hard disk per node, mounted as scratch space.
- A very fast experiment-plane network with a core switch capable of QinQ encapsulation.
- A service network providing net-booting and SSH access (to avoid interfering with experiment plane).
- A reservation system which allows users to specify their own kernel+ramdisk to be booted.

2.2 Firewheel

Firewheel is an Emulytics framework designed to allow a researcher to design and implement experimental environments consisting of virtual machines, virtual networks, control systems, and physical system simulations. Firewheel was designed to scale from small experiments containing just a few virtual machines and a simple, flat network to large experiments that include tens of thousands of diverse virtual machine configurations and complicated network topologies. Firewheel can run on systems ranging from a single laptop up to commodity clusters or other HPC systems with hundreds of high performance nodes.

The key features that Firewheel demands from a computer cluster to run large, demanding experiments can be split into hardware configuration requirements and software management requirements.

Firewheel runs best on clusters with recent and generously configured nodes. As of 2017, multi-socket, Intel Xeon blades with 256 GB or more of RAM are typical. Moderate amounts of local, fast SSD storage to cache virtual machine images and spool experimental results are useful. Firewheel prefers two separate networks to which all nodes connect: one fast experimental network of 10 Gbps Ethernet or above and a potentially slower control network used for orchestration purposes. Firewheel can use VLAN and related cluster switch-based techniques to partition experimental network segments, but more typically uses GRE tunnels and in-experiment software switches for this purpose, which reduces the coordination burden on clusters shared among multiple research groups and avoids VLAN exhaustion for large experiments.

Firewheel requires enough privilege to create and manage virtual machines. In the default configuration that uses the Linux kernel virtual machine (KVM) that means read/write access to the /dev/kvm special file. Firewheel also needs privilege to create and manage Linux bridges, GRE tunnels, and virtual NICs or tap devices.

Firewheel developers and experimenters have been using Canonical's MAAS cluster management system successfully for several years. It allows any authorized user to allocate a number of unallocated nodes, control node power state, and specify a base operating system image. MAAS automatically grants remote ssh access for allocated nodes to the sub-cluster owner. MAAS authentication and authorization integrates transparently with Sandia's LDAP database allowing users to authenticate with their normal Sandia Kerberos credentials. We typically grant root privileges on allocated nodes to the sub-cluster owner for the duration of the allocation. When a series of experiments is complete, the sub-cluster is easily released and made available to other users.

While Firewheel is somewhat large and depends on several open source packages and services, it has an easy-to-use installer that makes deploying Firewheel across a cluster straightforward. Using MAAS, allocating and deploying a sub-cluster and installing Firewheel takes about 30 minutes. The current version of Firewheel runs well on recent Ubuntu long term support releases.

2.3 Commonality

In examining the requirements and preferences of minimega and Firewheel, we found sufficient common ground to design a generic platform. Both tend to prefer nodes that prioritize RAM over CPU power and provide some quantity of fast SSD storage for caching and temporary storage. Both prefer to have at least two networks, with experiment traffic transiting the fastest possible network and kept separate from all other traffic. Both use similar techniques for encapsulating experimental traffic.

Firewheel's use of Canonical's MAAS cluster provisioning system differs from minimega's history of netbooting images using DHCP+PXE+TFTP. Many of Sandia's existing HPC systems use Cobbler to provision and manage nodes, and there was a desire to integrate the new cluster with the Cobbler infrastructure already in place. Discussion with the minimega and Firewheel teams indicated that both should be capable of working with Cobbler rather than their previous systems.

3 Reference Architecture

This section describes the reference architecture for an Emulytics cluster as developed by a working group of Sandia SMEs and released as an RFQ in the spring of 2017. It attempts to provide reasoning for the choices made, and suggest alternatives that may be more cost-effective if desired.

3.1 Physical Layout

The reference architecture specifies a recommended layout of equipment within the physical racks of the cluster. Broadly, it specifies a single core network unit (CNU) rack surrounded by scalable compute unit (SCU) racks on either side.

The CNU rack contains the cluster's head node and the core switches for the three cluster networks. Each SCU rack contains a boot network switch and a management switch at the top of the rack, connected to the core switches in the CNU.

3.2 Network

The reference architecture includes three primary networks: experiment, boot, and management. All inter-rack cabling is routed over the tops of the racks for ease of access (vs. under-floor routing).

Experiment network

The function of the experiment network is to carry traffic from the Emulytics experiments running on the cluster. This will typically be VLAN-tagged or GRE-tunneled traffic, which should be encapsulated with 802.1ad (QinQ) at the switch level to prevent experiments from unintentionally interfering with one another. To guarantee full throughput from any node to any other node regardless of traffic on the rest of the cluster, the experiment network should use a single core switch rather than a leaf-spine topology.

The experiment network should be a high-speed Ethernet network (40GbE, 50GbE, or 100GbE, with 100GbE being specified in the reference design), with a single non-blocking centralized core switch. Because Sandia has existing Emulytics tools to manage Arista hardware, the Arista 7500 series devices are considered ideal, but other devices may be suitable as well. The requirements for the core switch are:

- A software defined networking (SDN) programmable control plane.
- A separate Ethernet connection for remote management.

- Sufficient available network ports to support all cluster nodes plus additional (16 recommended) ports to connect existing systems or hardware-in-the-loop.
- Management and power redundancy.
- Support for 802.1ad (QinQ) tunneling with support for forwarding on VLAN+MAC, enabling duplicate MACs to exist on a single switch.
- A scriptable interface.

Boot network

The boot network is used for network-booting the nodes in the cluster and to provide ssh access to the nodes for users. Because the nodes frequently boot ramdisk images of several hundred megabytes, the boot network should be reasonably fast, although it does not need to be as fast as the experiment network. We recommend 10GbE as a good compromise between speed and price. 1GbE is functional but will lead to slower boot time for nodes, especially for a very large reservation.

The reference design of the boot network is a 40GbE managed core switch installed in the CNU rack, connected to top-of-rack 10GbE switches in each SCU. Each 10GbE top-of-rack switch should be connected to the boot network's core switch using four 10GbE links. Although this configuration does not guarantee a full 10Gb available throughput from every node to every other node, it represents a workable compromise between cost and performance.

Management network

The management network is used to manage hardware within the cluster. It is connected to:

- Every node's service processor (IPMI)
- The management interfaces on all network switches
- The management interfaces on PDUs

The management network allows IPMI control of the nodes, which can be used to cycle power, check temperature sensors, and in some cases access the node's console. Generally, users won't connect directly to endpoints on the management network; they will instead run commands on the head node to e.g. power-cycle a block of nodes.

Throughput is not essential for the management network; 1GbE is sufficient. The reference design specifies a managed 1GbE core switch installed in the CNU rack, connected to top-of-rack 1GbE switches in each SCU rack using two 1GbE links per switch.

3.3 Nodes

The cluster nodes consist of a single head node and a large number of compute nodes. The head node can be the same hardware as the compute nodes, but unlike the compute nodes it will boot its operating system from non-volatile storage.

The compute nodes for an Emulytics cluster should be optimized for memory, as this is typically the limiting factor of an Emulytics experiment. We recommend no less than 256 GB per node; 512 GB per node would be better. The reference design calls for 512 GB of DDR4 2400MHz memory per node.

The processor is also important, but less so than memory–any dual Xeon system should be sufficient, with number of cores being more important than clock rate although we recommend a minimum clock rate of 2.1GHz. All processors must support Intel VT-x with Extended Page Tables (EPT).

In order to support all three networks, the reference design requires that compute nodes contain at least two PCIe expansion slots, populated with a PCIe x16 100Gb Ethernet card for the experiment network and a PCIe 10Gb Ethernet card for the boot network.

Although compute nodes should typically boot from the network, it is useful to provide some physical storage on each node. The reference design specifies that each compute node should contain one 400-1000GB solid state disk (SSD). Solid state disks are recommended due to their significantly higher read and write throughputs compared to traditional spinning-platter hard disks; however, if cost is a concern or if a greater volume of storage is needed, traditional hard disks may be employed.

Nodes should have redundant power supplies for additional reliability.

Compute nodes should support remote configuration and management using IPMI. This allows things like remote rebooting, sensor monitoring, and console access. Although it is typically possible to power-cycle a node by cyling power at the PDU level, rebooting via IPMI is far ‘healthier’ for the hardware.

3.4 Power Distribution

Each rack should contain sufficient network-managed power distribution units (PDUs) to provide dual-circuit redundant power to each installed component. The use of redundant power allows each node to be connected to multiple power circuits; this allows continued operation despite the loss of any individual power circuit, PDU, or node power supply.

The reference design specifies that the PDUs will connect to three-phase 208V 60A input circuits via 10 foot whips to overhead power outlets. The whips should be outfitted with 4-pin male (3P+G) plugs.

4 Software Configuration

The software configuration of the cluster is composed of several parts: the software on the head node, the software on the compute nodes, and the configurations of the various other hardware components (switches, PDUs, etc.).

4.1 Head Node Configuration

The function of the head node is to provide users access to the cluster, to act as a staging area for experiments, and to manage reservations. It is the last task which requires special configuration.

Emulytics cluster head nodes should run a Linux OS, preferably Debian or RedHat.

DHCP and Netbooting

The head node manages IP addresses for all devices in the cluster and provides netboot services for the compute nodes. It should serve DHCP on all cluster networks (experiment, boot, and management) and TFTP on the boot and management networks–we recommend including TFTP on the management network because several switches and PDUs use TFTP to install updates. `dnsmasq` is convenient for its ability to serve both TFTP and DHCP.

We recommend defining sufficiently large subnets for each network, then mapping IP addresses to hostnames in /etc/hosts. Compute node hostnames should consist of a prefix followed by a number; in Sandia's "Country Club Cluster", nodes are numbered `ccc1`, `ccc2`, etc. `dnsmasq` configuration files can then map MAC addresses to hostnames for the purposes of serving DHCP.

Configuring PXE booting is outside the scope of this document, but in short the DHCP server should be configured to offer the `PXELINUX` boot image via TFTP. We typically root the TFTP server under `/tftpboot` but the specific location is not important. Configure a default kernel and ramdisk for compute nodes to boot; other configurations will be managed by the reservation tool.

Rather than configuring DHCP and PXE booting manually, a provisioning tool like Cobbler can provide a convenient centralized method for configuring IP addresses and netbooting. Many of Sandia's clusters use Cobbler, including the cluster specified in this RFQ.

Reservations: Considerations & igor

If an Emulytics cluster is to have more than one or two users, it needs a way for users to reserve nodes for their own exclusive use. Because emails and other manual forms of record-keeping tend to fail in an environment with many users, we strongly recommend the use of some form of automatic reservation system.

There are a few special considerations for making reservations on an Emulytics cluster. Because there are multiple Emulytics frameworks in use, it is not necessarily feasible to run every experiment on the same underlying OS installation. For example, Firewheel manages networking differently than minimega, and having both installed at the same time could cause conflicts. For this reason we prefer to let the user provide a Linux kernel and ramdisk to boot on their reserved nodes, a capability not present in standard HPC job schedulers.

Emulytics jobs may require some level of user interaction, so it is important that a user knows when his or her reservation will begin. The most common scheduling method for HPC reservation systems is to place jobs in a queue and begin running as soon as resources are available; this does not guarantee a specific start time and thus makes it a challenge for the user to plan ahead. An Emulytics cluster should be precisely scheduled, with each job having a defined beginning and end time known from the moment the reservation is created.

`igor` is a cluster reservation tool developed by Sandia for managing Emulytics clusters. It integrates with either a "standard" PXELINUX netboot installation or with Cobbler to provision cluster nodes. Users create a reservation by specifying how many nodes they need, how long they need them, and what kernel+ramdisk pair the nodes should boot.

Every reservation is assigned a time slot at the moment of creation. Unless the user specifies otherwise, `igor` will find the first available time slot in which the desired number of nodes are available for the desired duration. The user may also specify that the reservation should begin at or after a certain time–for instance, to reserve some nodes for a live demonstration next Monday at noon. Once a reservation has been created, its time slot will never change, even if another reservation is deleted and thus frees up an earlier time; if the user wishes to move his reservation to the now-available earlier time, he or she can delete the old reservation and create a new one.

When a reservation's time slot begins, `igor` will automatically modify the netboot configurations for the reserved nodes to boot the user-provided kernel+ramdisk. Optionally, `igor` can also then power-cycle the reserved nodes to make them automatically boot into the new image; otherwise, the user must power-cycle the nodes themselves (this provision allows a 'grace period' if the previous user's reservation should accidentally run long). When the reservation ends, `igor` returns the nodes to a default configuration and optionally reboots them.

`igor` also performs network segmentation on compatible core switches. As mentioned in the section on hardware, the use of 802.1ad (QinQ) encapsulation can allow multiple experiments on the same experiment plane to use the same set of VLANs and MAC addresses without interfering with each other. On Arista 7500-series core switches, `igor` will configure the experiment network such that each reservation's nodes are in their own outer QinQ VLAN.

`igor` is distributed with the minimega suite of tools; instructions for configuration are found on the minimega website at `http://minimega.org/`.

4.2 Compute Node Software

As described in the previous section, compute nodes should boot over the network using kernel+ramdisk pairs provided by the user when making a reservation. In general, these ramdisk images will only contain very minimal software: just what is required to run the user's Emulytics experiment. We have found it convenient to build custom images which use a shell script in place of the traditional Linux init system; this allows for a faster boot and a high level of control over how the node is configured.

vmbetter

`vmbetter` is a tool distributed with the minimega suite which can automate the construction of bootable ramdisk images for Emulytics clusters. To build an image, the user simply provides a configuration file listing any Debian packages they want installed on the image and any post-installation commands they way to run; optionally, they can place files in a subdirectory to be copied to the image (we typically add at least a /init file). When `vmbetter` is run, it uses `debootstrap` to install the desired packages, then uses `chroot` to issue any post-installation commands within the newly-created image.

4.3 Switch Configuration

Switch configurations are complex and, of course, vary from device to device. Broadly, the most important thing is to configure all switch ports connected to compute nodes to trunk all VLANs. Listing 1 shows some selected configuration settings used in production on a Sandia Emulytics cluster's Arista 7500-series core switch.

Listing 1. Basic Arista Configuration Excerpt

```
transceiver qsfp default-mode 4x10G
!
platform sand forwarding mode Arad
!
spanning-tree mode mstp
!
vlan 2-4001
!
interface Ethernet3/1/1
   speed forced 40gfull
   switchport mode trunk
!
interface Ethernet3/1/2
   switchport mode trunk
```

```
!
interface Ethernet3/1/3
   switchport mode trunk
!
interface Ethernet3/1/4
   switchport mode trunk
!
```

One important capability of Emulytics is hardware-in-the-loop, in which a piece of physical hardware is bridged directly into the Emulytics experiment. This is typically accomplished by connecting the hardware to the core switch and configuring those ports as "access" ports into appropriate VLANs used by the experiment. Listing 2 demonstrates how a network firewall with two interfaces might be bridged into an experiment. In this case, VLAN 501 is the 'Internet' interface for the firewall, and VLAN 500 is the 'LAN' interface.

Listing 2. Hardware-in-the-loop Configuration

```
interface Ethernet10/43
   description firewall-external
   switchport access vlan 501
!
interface Ethernet10/44
   description firewall-internal
   switchport access vlan 500
```

5 Future Work

The reference architecture described in the previous sections will be instantiated as Sandia's first-ever corporate Emulytics™ cluster. The move from small clusters hosting small, close-knit groups of researchers will doubtless bring new challenges to the fore. Aside from these unforeseen challenges, we expect to perform additional work and refinement in the following areas:

- Emulytics job scheduling. The current tooling around Emulytics experiments allows the user to reserve nodes and boot a particular image on the compute nodes, with the expectation that the user will launch their experiment manually after that. We intend to add capabilities to automatically launch the experiment itself and gather results when it has finished running. This will be made more challenging by the multiple Emulytics toolkits in use, but we believe a common interface can be developed to improve automation.
- Multi-use clusters. We intend to use the new corporate cluster not just for Emulytics experiments but for a variety of computational tasks. To this end, we will continue to adapt tools such as `igor` to more easily allow sections of the cluster to be used for non-Emulytics tasks.
- Compute node boot images. Although the `vmbetter` tool has proved useful on existing Emulytics clusters, we expect that the larger community around the new cluster will have needs which it does not address. Building bootable images tends to be viewed as an arcane art in many communities; we hope improved tools could remove some of the mystery and difficulty.
- Improved management and instrumentation. Modern computer systems with IPMI-connected thermal sensors, switches with detailed packet statistics, power distribution units with per-outlet current monitoring, and similar features means clusters built from commodity hardware can now gather detailed information about the operating environment from top to bottom. Future developments should strive to gather that information and present it in the most useful way, whether through combinations of existing tools or by developing custom dashboards.

As Emulytics tools grow and develop, it is important to pay attention to the decades of work in HPC systems. While not all HPC concepts and tools are applicable to Emulytics, we should carefully survey all existing 'wheels' before deciding to invent a new one.

6 Conclusion

The use of large scale emulated environments for cyber security work and other applications continues to grow. We hope that by providing a description of Sandia's Emulytics^{TM} reference architecture (soon to be implemented in a corporate-managed cluster), we may help others avoid pitfalls in designing their own clusters. Eventually, this field may coalesce around a set of hardware architectures, software, and practices in the same way the HPC community has; this can only happen by sharing information in documents such as this one.

DISTRIBUTION:

1	MS 9105	Jeff Boote, 8766 (electronic copy)
1	MS 0899	Technical Library, 10756 (electronic copy)


Sandia National Laboratories