Scaling techniques for cyber emulation

John Floren
11 Aug 2017

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. SAND2017-8709 C

Introduction

One of the biggest reasons for using Emulytics toolsets for experimentation is that they can handle large-scale virtual environments easily. In this talk, we discuss some of the techniques we have used to achieve scale with minimega:

- Oversubscription
- KSM
- Hugepages
- Containers
- Smart scheduling
- Network considerations
- Buying more nodes

Scaling

In creating an Emulytics environment, we aim for:

- Density: we want as many VMs as possible in our limited hardware
- Realism: we want VMs to behave like the real-world systems we're emulating

Defining the environment involves striking a balance between getting enough VMs per host to accomplish your goal without packing them so tightly nothing can actually get done.

Resources

Our VMs compete for host resources:

- CPU: Not usually a bottleneck; the average desktop computer's CPU is usually almost completely idle
- Memory: Most common bottleneck. Modern software takes up a lot of memory.
- Network: Can be a problem if you're doing test and evaluation ("can this webserver really handle 10Gbps traffic?")

If we wanted to be extra careful, we'd allocate VMs something like this:

- 32 CPU cores on host = no more than 32 1-CPU VMs
- 100 GB of available memory = no more than 50 VMs with 2GB RAM each
- 10Gbps Ethernet card = no more than 10 VMs with 1 gig interfaces

Oversubscription

Oversubscription is core to Emulytics, but you have to be smart about it:

- Allocating 2GB RAM to a VM doesn't mean you immediately 'lose' 2GB on the host; as the guest allocates memory, the VM's memory use grows **up to** 2GB.
- Desktops spend a lot of time idle = allocate more desktop VMs than physical cores
- Some machines don't create much network traffic. Pack 1000 of them on a host.
- An extremely busy core switch may need a dedicated physical host to actually hit 10Gbps.
- Experiments often consist of 'important' VMs and 'background' VMs. The important VMs typically need more resources. The background VMs may do little more than generate occasional HTTP traffic. Take this into consideration when oversubscribing.

General rule: start cautious and scale from there.

KSM

Kernel Same-page Merging is a capability in the Linux kernel which can improve memory density. It scans through all allocated memory pages on the system and merges any duplicate pages into a single copy-on-write page.

- Especially effective if all VMs on the host booted from the same image.
- Adds significant CPU load initially, with on-going maintenance cycles. Use when memory is more important than CPU cycles.
- Can take several minutes to complete the initial scan
- Works best for nodes that are either mostly quiescent or are all running the same software.
- When the VMs start, they'll consume the 'normal' amount of memory, then slowly drop as KSM works
- So you launch incrementally smaller batches of VMs until memory is mostly full--you need to leave some buffer in case some VMs need it.

KSM

In a simple experiment, 500 VMs were booted from a Debian disk image. Without KSM, the VMs consumed 200MB of memory per VM. With KSM enabled, they consumed only 80 MB/VM.

Note that if you go in and start **different** programs on each VM (e.g. Chrome on one, Firefox on another, MS Word on a third, etc.), memory use will balloon and could cause an Out-Of-Memory condition. Use KSM when you know what programs VMs will run and are confident they won't suddenly exceed available memory.

minimega allows you to turn KSM on and off with the `optimize ksm [true,false]` command.

Hugepages

Default page size on AMD64 is 4KB. Linux offers the ability to run certain programs with 2MB pages.

Using hugepages for VM processes has several advantages:

- Smaller page tables saves memory
- Reduces TLB misses
- Fewer page allocations

minimega can run VMs with 2MB pages using the `optimize hugepages <path>` command, where '<path>' is the path to a mounted hugetablesfs.

Containers

Full-weight VMs are not always necessary. When possible, we deploy lightweight Linux containers in our experiments for greater density. Containers offer some compelling advantages:

- Smaller memory footprint (just processes running sandboxed on host kernel)
- Lower CPU use
- Very fast network (no virtual drivers)
- Easier to move files in/out of running containers

Containers also have some disadvantages:

- Always homogenous (same kernel as host, Linux only)
- No iptables for containers
- Container breakout considered easier than VM breakout--avoid running untrusted code

Smart scheduling

VM scheduling is the process by which the orchestration system distributes VMs to physical hosts. Although automatic scheduling may be possible, it is sometimes useful to add some human intervention for better performance/scale:

- Schedule 'like' VMs on the same host. This helps KSM, and if you know the VMs will be largely quiescent, they can be denser
- Schedule 'like' network endpoints on the same host. When possible, put all hosts in a subnet on the same physical host so traffic doesn't hit the physical network.

We've started adding more advanced scheduling to minimega.

- Scheduler code can place VMs based on each physical node's available memory, CPU load, or the number of virtual network interfaces already in place.
- This allows a crude prioritization of memory, CPU, or networking.
- minimega also allows the user to specify that a VM must land on one node, or that a given node cannot have more than N VMs.

Networking options

minimega provides three ways to move virtual network traffic over the cluster's physical network. Each option has tradeoffs between performance, realism, and ease of use.

- VLAN tagging. Each virtual net is assigned a VLAN tag and trunked over the physical net that way. Fast (line-rate).
- GRE tunnels. Slower, but supports more virtual networks. Requires explicit tunnel creation.
- VXLAN tunnels. Faster than GRE, supports large numbers of virtual networks. Requires explicit tunnel creation.

VLAN tagging is fast and easy to set up, and makes hardware-in-the-loop easy, but it means no VLAN tagging **inside** the experiment. GRE and VXLAN tunnels require more configuration but can allow a more flexible virtual environment or enable federation of clusters over the Internet.

More nodes

Sometimes the only answer is to throw more nodes at the problem. The orchestration tooling needs to be able to scale smoothly as the number of nodes increases.

We launched 1,000 KVMs across 1, 2, 4, 8, 16, and 32 nodes, then repeated the experiment with containers. The wall-clock times to boot (in seconds) were:

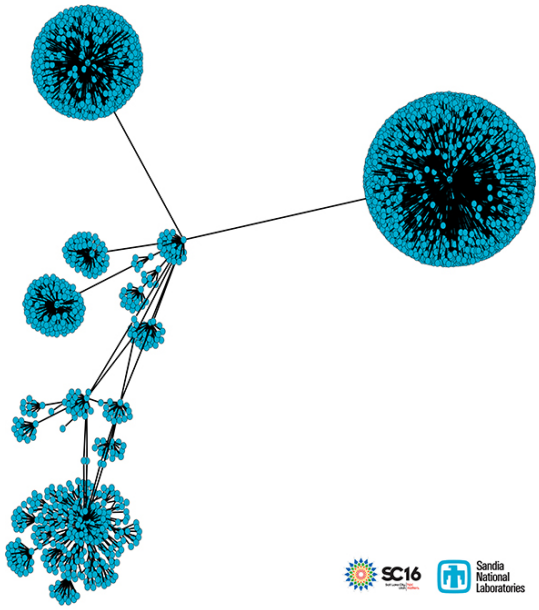
Nodes	KVM	Container
1	681 (681)	459 (459)
2	367 (734)	259 (518)
4	186 (744)	122 (488)
8	92 (736)	58 (464)
16	46 (736)	28 (448)
32	24 (768)	13 (416)

We also tried launching 1,000 VMs **per node** and found smooth scaling:

Nodes	KVM	Container
1	823	658
2	949	836
4	946	838
8	935	842
16	945	825
32	923	768

Putting it together

Here's the network at SC 2016:



Putting it together

We used our tooling to generate a model from packet capture, router configs, and other data sources. Model contains about 9500 endpoints, including laptops and mobile devices.

The model can then be tweaked based on what we want to accomplish.

We could boot every endpoint as a container, but use real Cisco/Juniper/Brocade router images to experiment with topology and router configurations.

We could use minimobile to boot Android devices and emulate the physical locations of the WiFi infrastructure while leaving the other endpoints as containers.

We could 'zoom in' on one subnet, instantiating it with KVM VMs booting full-size Linux and Windows VMs and using containers in the other subnets to generate traffic and provide a milieu.

With enough nodes (perhaps a dozen), we could simply instantiate every VM as a full-size Linux, Windows, or Android device.

Audience participation time

Are you using the techniques we described?

Are you using any other techniques to achieve large-scale environments?

Any questions / comments?

Thank you

John Floren
11 Aug 2017

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.