Migrating the Ciena SIM Environment into the Cloud - (Applying LTTng tracing towards performance analysis)

Ecole Polytechnique - Progress Report Meeting - December 2016

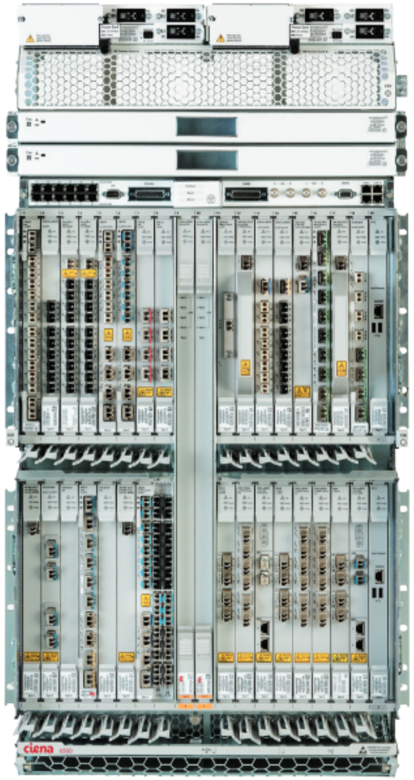
Octavian Stelescu

December 2016

Agenda

Ciena is moving our 6500 SIM into the cloud

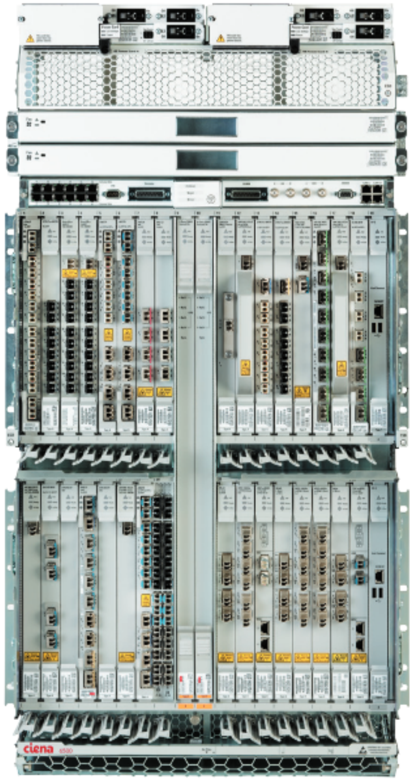
Where are you taking me?



At Ciena we are taking a direction to push some areas of our operations into the cloud including our 6500 product simulator.

- Moving the simulation infrastructure is driven by many factors:
  - Cost reduction, space savings, flexibility, speed to create virtual workstations for designers
- Leveraging cloud virtualization technologies is beneficial to improving Ciena's efficiency in multiple areas:
  - Design, testing, automation
  - Customer facing projects (Ciena Emulation Cloud): customers testing rest APIs in the cloud to manage our equipment
  - Hibernating large network configurations. Networks which can take ~ 1 hour to start could be recovered on the fly by waking up a VM
- There is a significant challenge involved in bringing the simulator into the cloud and striving to achieve near bare metal performance

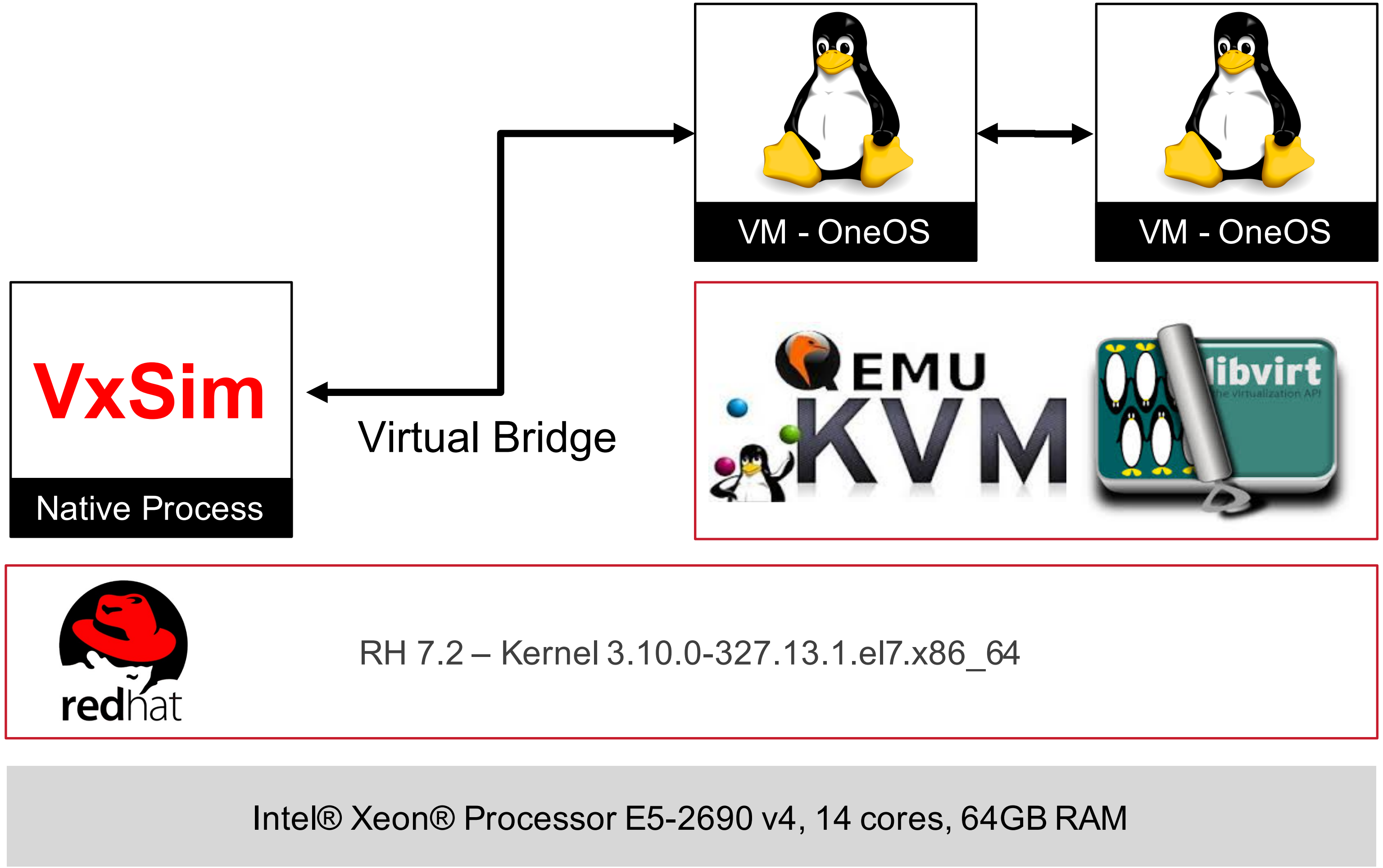
Where do we use our sim?



- Testing our software without the need for target physical hardware
- Creating virtual networks for certain interworking functions
- Bounds Checking on the compiled software syntax
- Simulator back end for our nodal manager / network manager (BluePlanet)

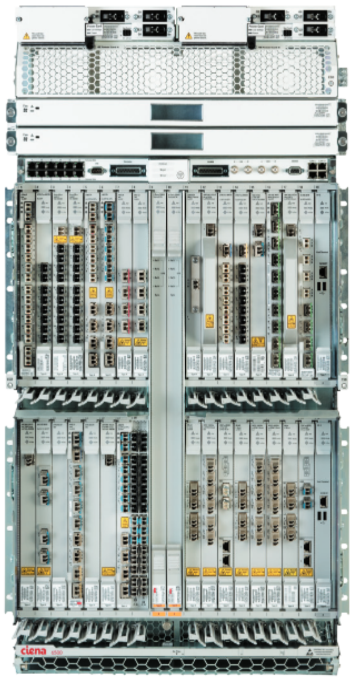
Ciena 6500 Product Simulator

- **The basic architecture of our sim:**
  - Running RHEL 6.3 – 7.2 for our host operating system
  - VxSim is used for our shelf processor as well as some of our line cards
  - Our packet optical cards which originate from different Ciena products appear as qemu-kvm Linux based virtual machines
  - These are tied together through our python SIM script (GSIM), Linux virtual interfaces, as well as our own software which forwards packets between the VxSim and the OneOS world

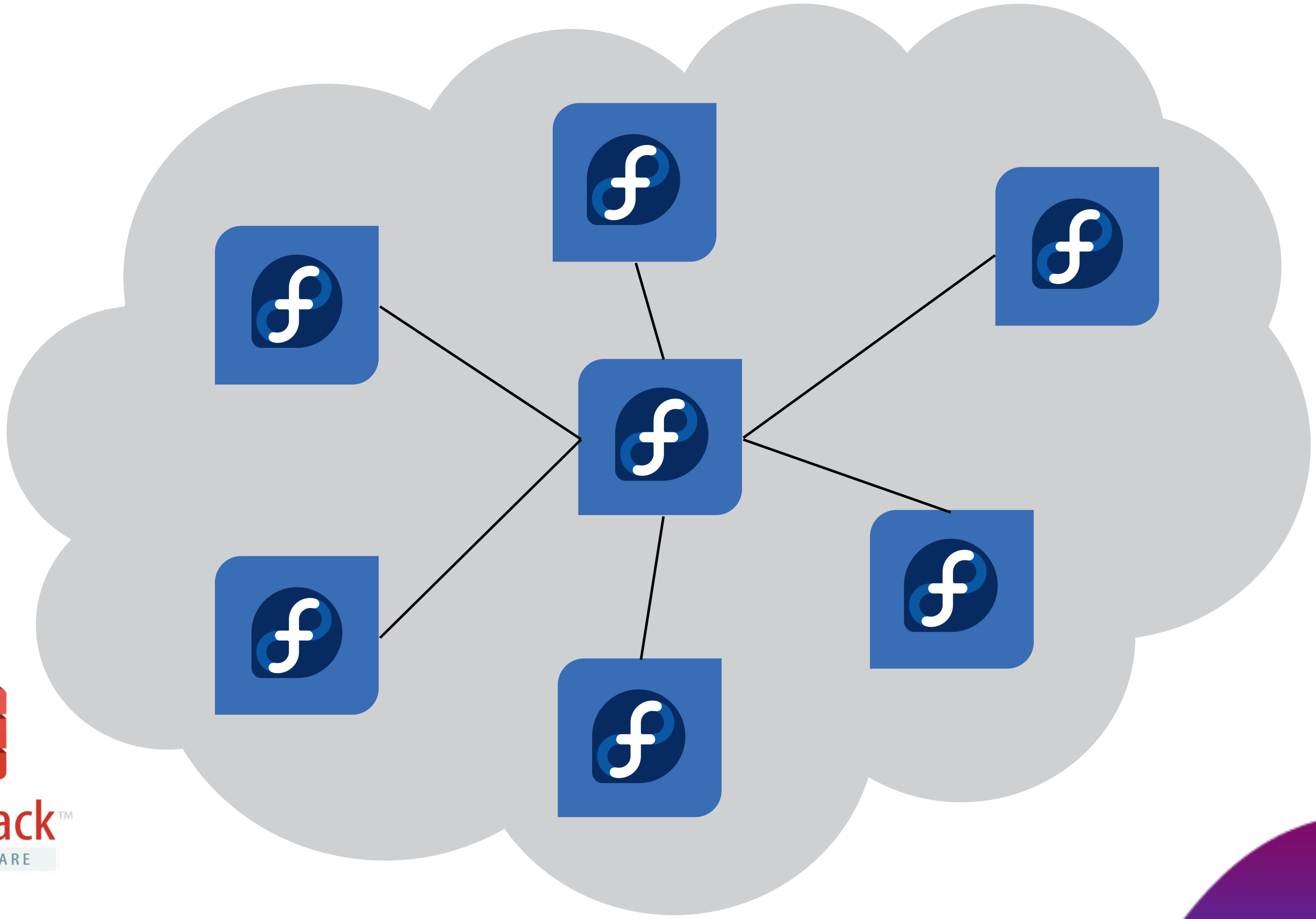


Ciena 6500 Product Simulator

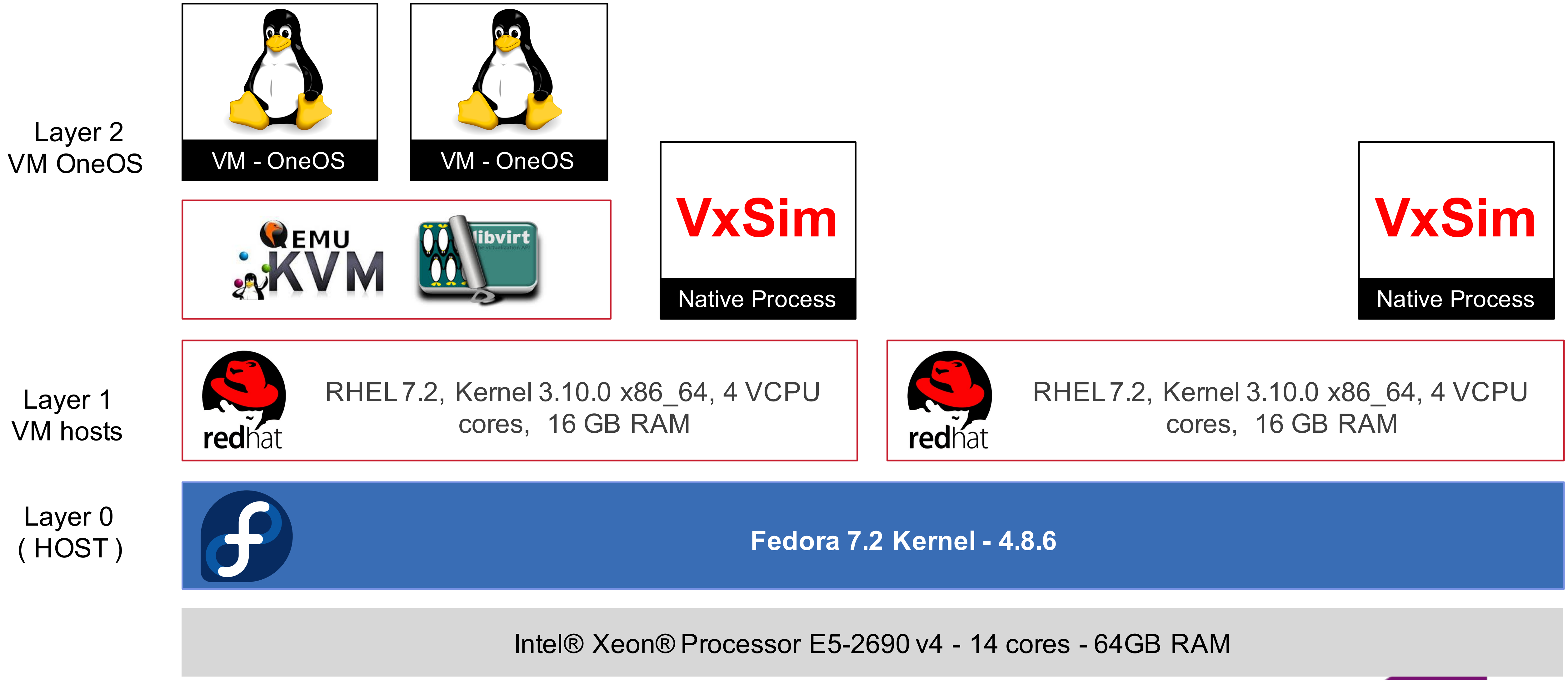
- The Ciena cloud is built on top of openstack
  - At a low level we have collection of Fedora Cloud workstations
  - On each Fedora Cloud workstation we can have multiple RHEL 7.2 VMs to host our SIM



openstack™
CLOUD SOFTWARE



Ciena 6500 Product Simulator on single node in the cloud



TOP running on a designer workstation

- A single OneOS VM uses 10 X CPU than our VxSim based cards
  - In this example we are simulating two nodes (2 VMs and one VxSim process per node) for a bare bones shelf
  - ome_sp2_vx_appl is the shelf processor
  - qemu-kvm VMs are packet optical cards

```
op - 15:23:22 up 4 days,  6:37,  4 users,  load average: 0.36, 0.52, 0.52
asks: 641 total,   1 running, 640 sleeping,   0 stopped,   0 zombie
Cpu(s):  3.6 us,  1.2 sy,  0.0 ni, 95.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
iB Mem : 16156908 total,  4792896 free,  6458544 used,  4905468 buff/cache
iB Swap:  4194300 total,  4194300 free,        0 used.  9344184 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
16496 root      20   0  616556 556396 337616 S  20.6  3.4   0:08.63 gdb
17124 qemu      20   0 1542248 1.042g  11088 S   9.6  6.8   1:53.49 qemu-kvm
24656 qemu      20   0 1541220 0.981g  11076 S   9.3  6.4   0:56.87 qemu-kvm
24855 qemu      20   0 1540200 0.985g  11076 S   9.3  6.4   0:40.88 qemu-kvm
17276 qemu      20   0 1541224 1.043g  11080 S   8.3  6.8   1:30.95 qemu-kvm
17257 root      20   0  950252 313528  31932 S   0.7  1.9   0:17.60 ome_sp2_vx_appl
24601 root      20   0  950252 313608  32012 S   0.7  1.9   0:14.63 ome_sp2_vx_appl
25060 root      20   0   68040    752    588 S   0.7  0.0   0:02.51 vxbridge
 3267 root      20   0  248260 184888  13208 S   0.3  1.1   5:16.97 splunkd
10011 ostelesc  20   0 1624180 242608  45140 S   0.3  1.5   0:37.22 gnome-shell
24167 root      20   0  616556 555984 337616 S   0.3  3.4   0:06.19 gdb
    1 root      20   0  192340   7308   2404 S   0.0  0.0   0:31.86 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.01 kthreadd
```

OneOS

VxSim

TOP running on VM workstation in the cloud

- Moving our sim into a designer workstation in the cloud we see a massive jump for CPU usage for each OneOS VM
  - OneOS VMs use 8X more CPU resources / core
  - Our VxSim processes have the same CPU resource usage

```
top - 16:50:27 up 9 days,  1:29, 10 users,  load average: 3.23, 3.27, 3.22
Tasks: 315 total,   2 running, 313 sleeping,   0 stopped,   0 zombie
%Cpu(s): 11.6 us,  8.2 sy,  0.0 ni, 67.5 id,  0.0 wa,  0.0 hi,  0.0 si, 12.6 st
KiB Mem : 15880680 total,  3874300 free,  6478128 used,  5528252 buff/cache
KiB Swap:  4194300 total,  4181872 free,    12428 used.  8844948 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
23283 qemu      20   0 1541220 1.048g  11084 S  83.4  6.9 188:16.01 qemu-kvm
24799 qemu      20   0 1541224 1.048g  11084 S  82.7  6.9 183:47.79 qemu-kvm
24596 qemu      20   0 1543376 1.052g  11116 S  76.7  6.9 166:08.88 qemu-kvm
23084 qemu      20   0 1543376 1.050g  11116 S  76.1  6.9 171:53.35 qemu-kvm
 5620 ostelesc  20   0 2056008 256284  32824 S   2.3  1.6   7:30.54 gnome-shell
24992 root      20   0   68040    752    588 S   1.0  0.0   0:56.80 vxbridge
 4991 ostelesc  20   0  349528  62816  16156 S   0.7  0.4 349:56.31 Xvnc
22345 root      20   0  994880  75072  15856 S   0.7  0.5   0:36.72 virt-manager
22572 root      20   0  616544 556028 337620 S   0.7  3.5   0:29.80 gdb
22990 root      20   0  950252 314332  32736 S   0.7  2.0   1:56.80 ome_sp2_vx_appl
24293 root      20   0  950252 314396  32800 S   0.7  2.0   1:54.79 ome_sp2_vx_appl
   19 root      20   0       0      0      0 S   0.3  0.0  10:15.44 rcu_sched
 1236 root      20   0  553044  16348   5696 S   0.3  0.1   0:51.41 tuned
 5566 ostelesc  20   0 1165536  32984  12536 S   0.3  0.2   2:58.84 gnome-settings-
```

~ 8 X CPU usage
for each OneOS VM

VxSim is the same

Optimizing Virtual Machine Performance

From Libvirt / Virt-Manager

Avoiding unused devices

CPU performance options

Avoiding CPU overcomittment, copying CPU host configuration, single thread single core and mutiple sockets for a VM, CPU pinning to a NUMA node

DISK

SSD for the host, virtio drivers for VMs

Tuning Tools

tuned -> tuning profile delivery mechanism that adapts Red Hat Enterprise Linux for certain workload characteristics

Networking

Virtio, virthost-net

BLOCK I/O

Cache, threads, disk I/O throttling

Nested VM specific

Nested virtualization, **VMCS shadow,** VIRT-APIC

TOP running on VM workstation in the cloud with VMCS shadowing

- After enabling nested virtualization and VMCS shadowing the CPU usage drops
  - Enabling nested virtualization and VMCS shadowing drops the CPU usage to approximately half
  - We are still around 4X more resource intensive on the CPU when running on a designer workstation

```
op - 15:11:20 up 22 min,  3 users,  load average: 4.57, 3.98, 2.40
asks: 231 total,   2 running, 229 sleeping,   0 stopped,   0 zombie
Cpu(s): 31.0 us,  9.2 sy,  0.0 ni, 59.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.2 st
iB Mem : 16268364 total,  5591280 free,  5951980 used,  4725104 buff/cache
iB Swap:        0 total,        0 free,        0 used. 10059316 avail Mem

 PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
1853 qemu      20   0 1551788 1.045g  11312 S  57.1  6.7  12:03.91 qemu-kvm
3766 qemu      20   0 1551784 992.7m  11312 S  53.2  6.2   6:02.23 qemu-kvm
3820 qemu      20   0 1543588 0.978g  11320 S  51.5  6.3   4:08.87 qemu-kvm
1916 qemu      20   0 1552812 1.044g  11312 S  49.8  6.7   9:38.01 qemu-kvm
0761 ostelesc  20   0 1576816 222568  44960 S   2.0  1.4   0:56.88 gnome-shell
1898 root      20   0  950252 313584  31988 S   1.3  1.9   0:25.27 ome_sp2_vx_appl
0323 ostelesc  20   0  293060  67608  13296 S   1.0  0.4   0:17.50 Xvnc
3748 root      20   0  950252 313608  32012 S   0.7  1.9   0:18.03 ome_sp2_vx_appl
 709 chrony    20   0   22688   1292   1060 S   0.3  0.0   0:00.01 chronyd
1624 root      20   0  616552 556428 337616 S   0.3  3.4   0:08.94 gdb
3525 root      20   0  616552 556076 337620 S   0.3  3.4   0:06.69 gdb
3938 root      20   0   68040    748    588 S   0.3  0.0   0:03.24 vxbridge
```

~ 4 X CPU usage
for each OneOS VM

VxSim is the same

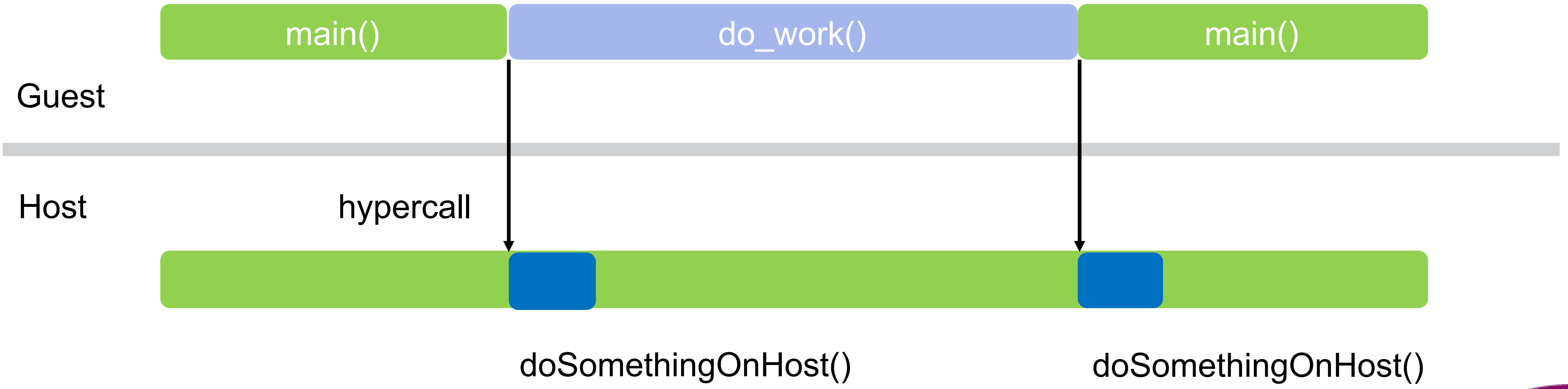
CPU as the main bottleneck towards SIM virtualization in the cloud

It becomes apparent very quickly that our SIM experiences a CPU bottle neck in the cloud as a consequence of nesting

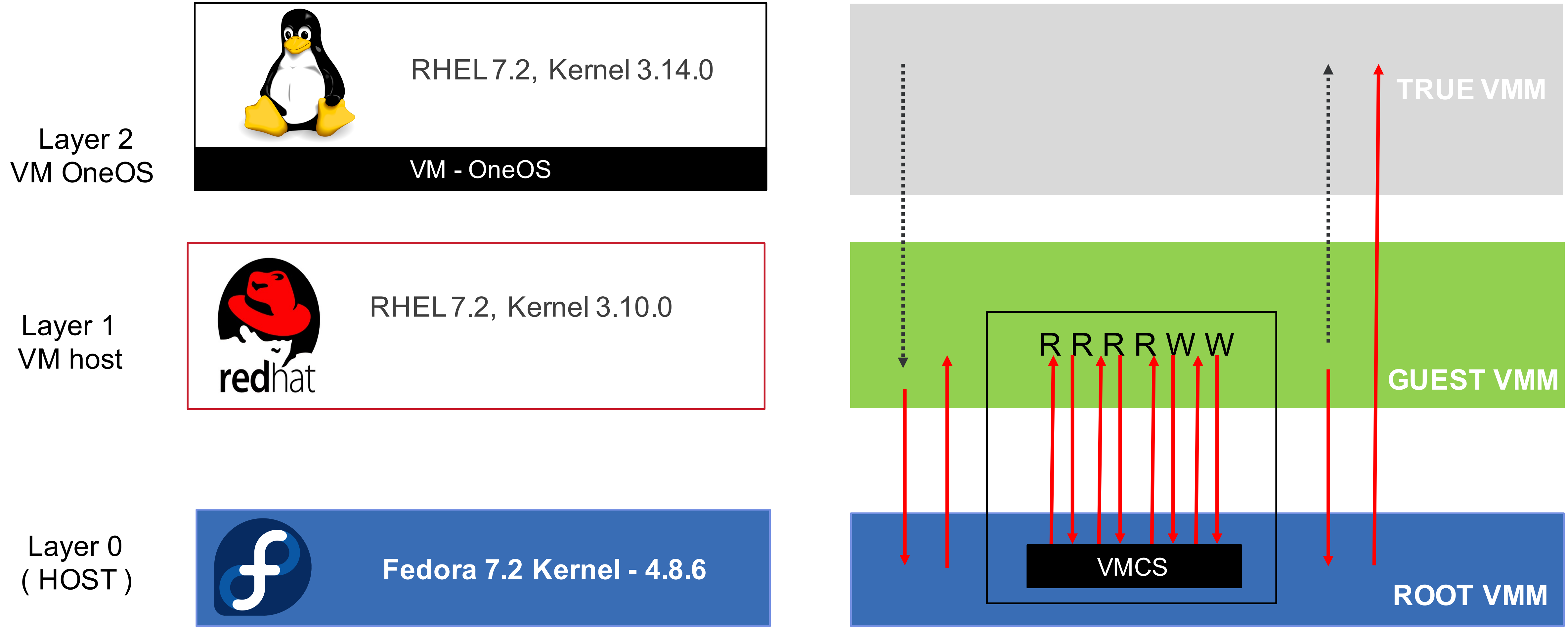
- To investigate this further we looked at a SIM network configuration in a steady state (the simulated network is up and hanging around)
  - On a standard designer workstation a single OneOS VM uses ~ 10% / CPU core
  - Running in the cloud we are at about ~ 50% / VCPU core
  - Without hardware features (VMCS shadow) that number jumps to ~ 80% VCPU core
- In our investigation we only had a redundant shelf with only two nodes, as we add more nodes we increase the number of OneOS VMs and the performance severely degrades
- We need to further understand this overhead and minimize it (if possible)

Some terminology…

- A hypervisor or virtual machine monitor (VMM), such as libvirt is a piece of computer software, firmware or hardware that creates and runs virtual machines.
- Hypercalls only exist with hardware assisted virtualization (specialized x86 instructions)
- Similar to an API between the VM and the hypervisor
- Privileged instructions are implemented by hypercalls to the hypervisor.
- VMCS is virtual machine control structure which is used to save the state of the VM/HOST as we transition from executing the code of the VM to that of the host

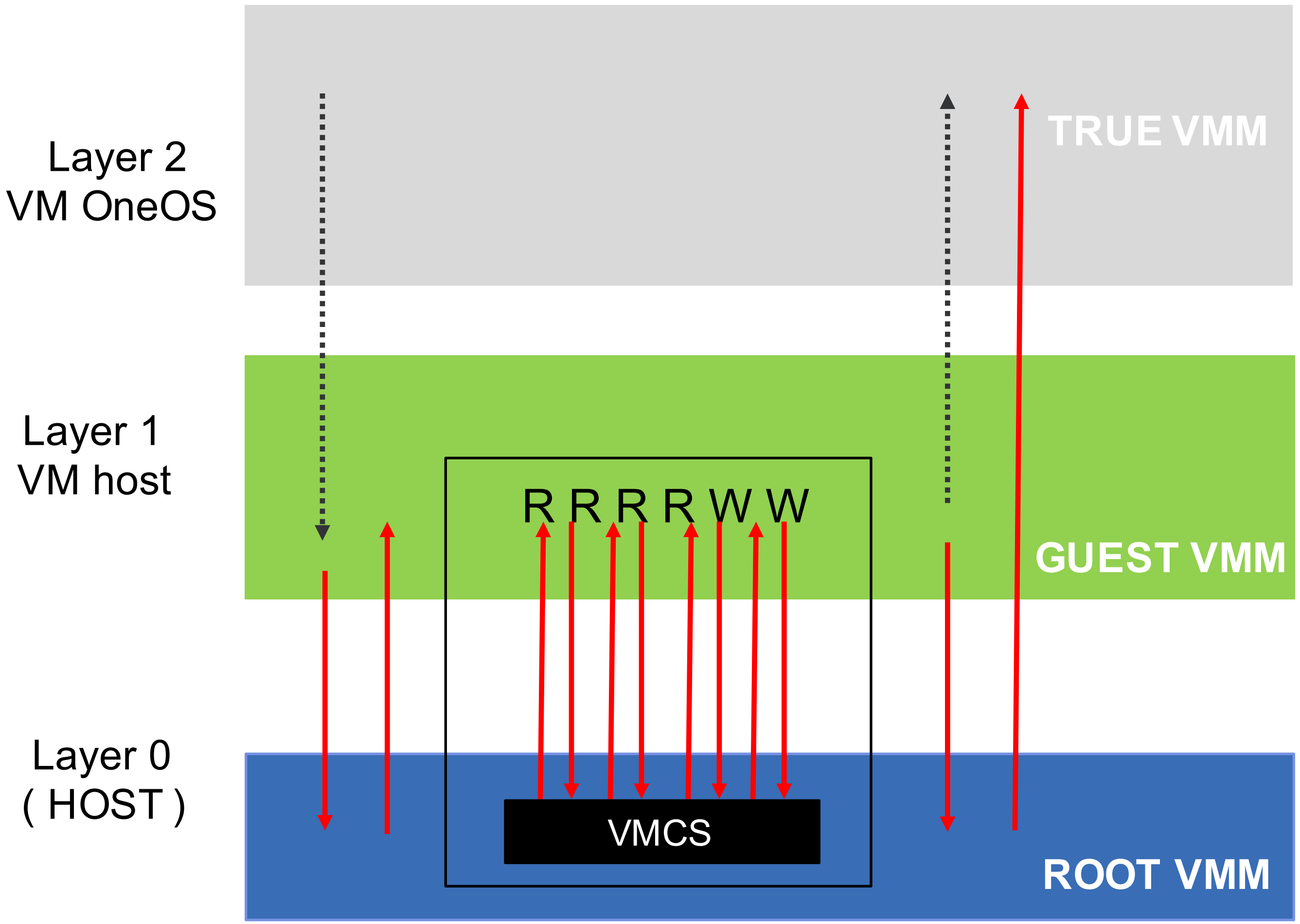


Nested VM Architecture for VMX



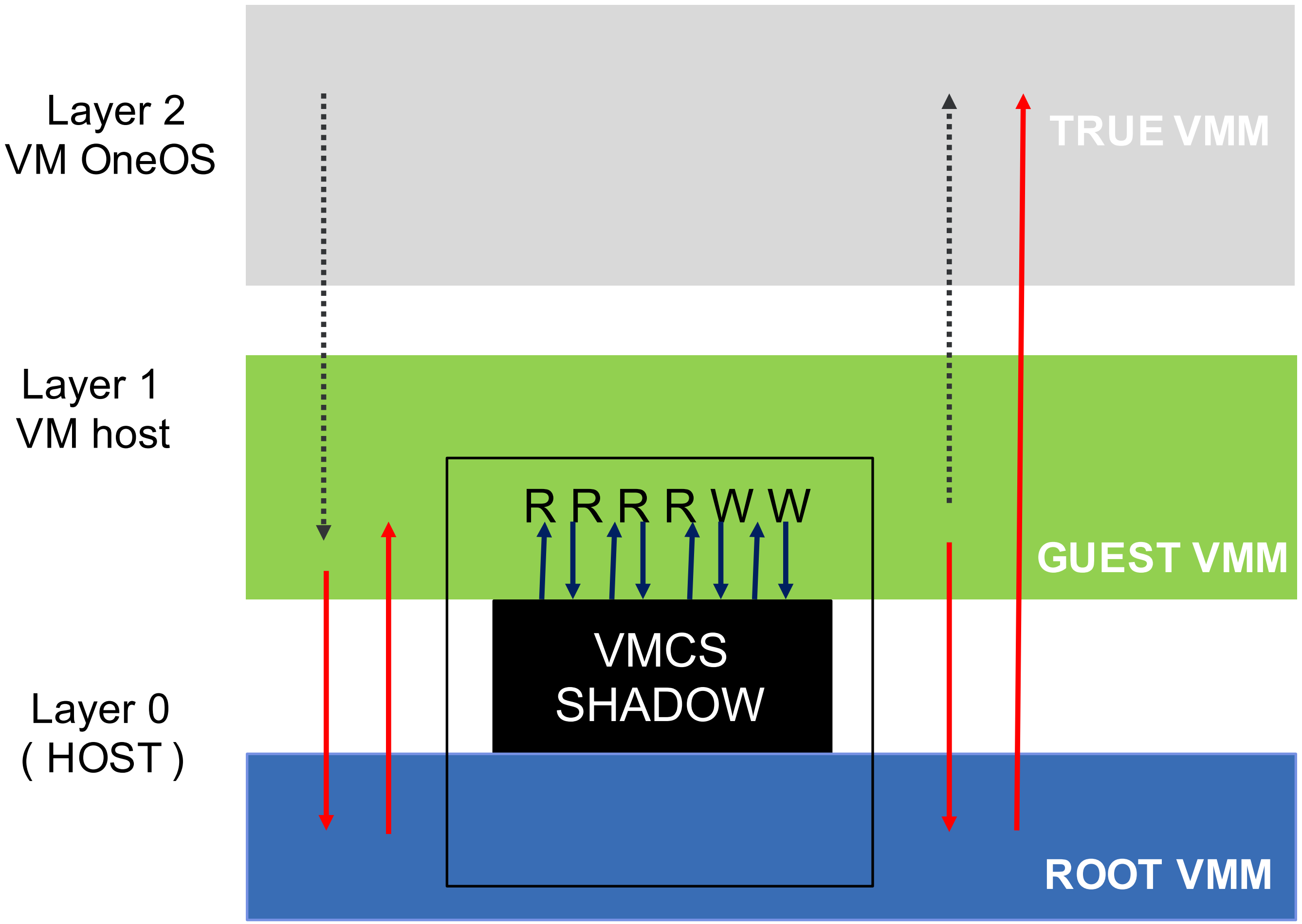
Nested VM Architecture for VMX (continued)

- **In a single-level architecture, executing any privileged instruction by any level of nested VMs returns to the host hypervisor (L0).**
  - The VM hypervisor (L1) has the illusion of running the code of the nested VM (L2) directly on the physical CPU
  - Privileged instructions of nested VMs are handled by the highest privileged level L0
  - The execution of any hypervisor level or VM privileged instructions causes the L0 trap handler to be executed
  - This VMX emulation can go to any level of nesting



Nested VM Architecture for VMX (continued)

- **To handle a single L2 exit, L1 does many things: read and write the virtual machine control structure (VMCS), disable interrupts, etc**
  - Those operations can trap, leading to exit multiplication
  - Exit multiplication: **a single L2 exit can cause 40-50 L1 exits!**
  - There is an optimization which allows us to execute a single exit faster and reduce frequency of exits
  - This is VMCS shadowing. VMCS shadowing directs the VMM VMREAD/VMWRITE to a VMCS shadow structure.
  - This reduces nesting induced VM exits.

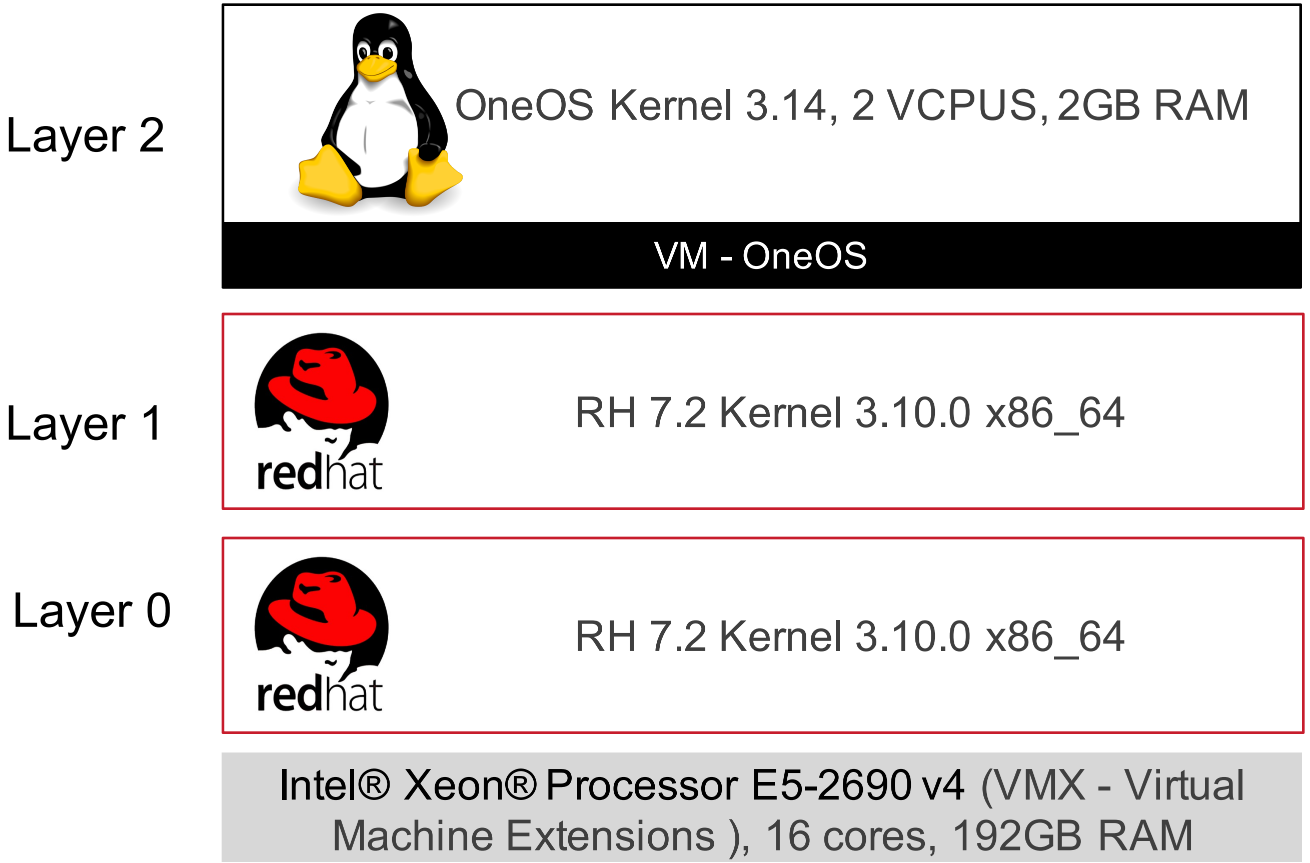


Tracing with LTTng

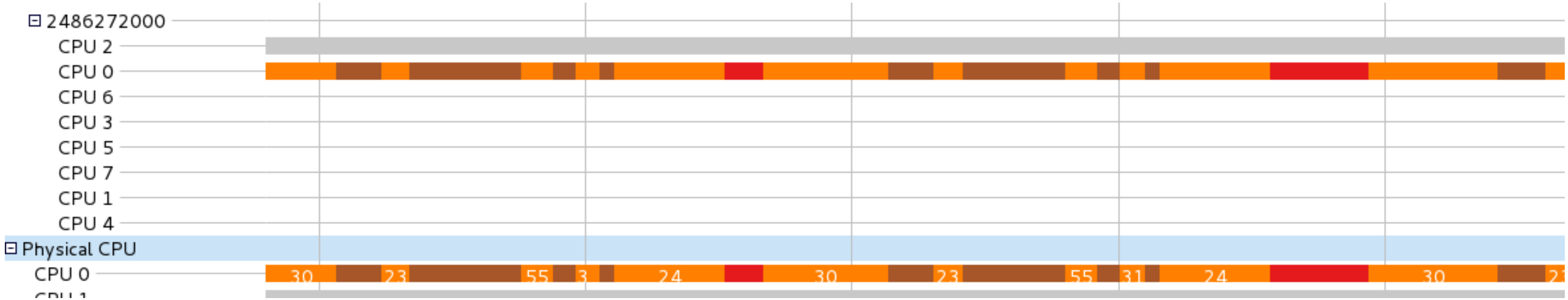
Ciena 6500 Product Simulator in the Cloud – Single Openstack node

- **Without access to the bottom layer L0 of the cloud, we created a nested configuration representing an isolated node in the cloud**
- **To investigate the execution flow of nested VMs we used the following setup:**
  - Layer 0 Host – RHEL 7.2
  - Layer 1 VM Host – RHEL 7.2
  - Layer 2 OneOS – Linux
- **For all intensive purpose this configuration resembled our setup in the cloud**

.



Execution Flow analysis of OneOS: L2 -> L1 -> L0



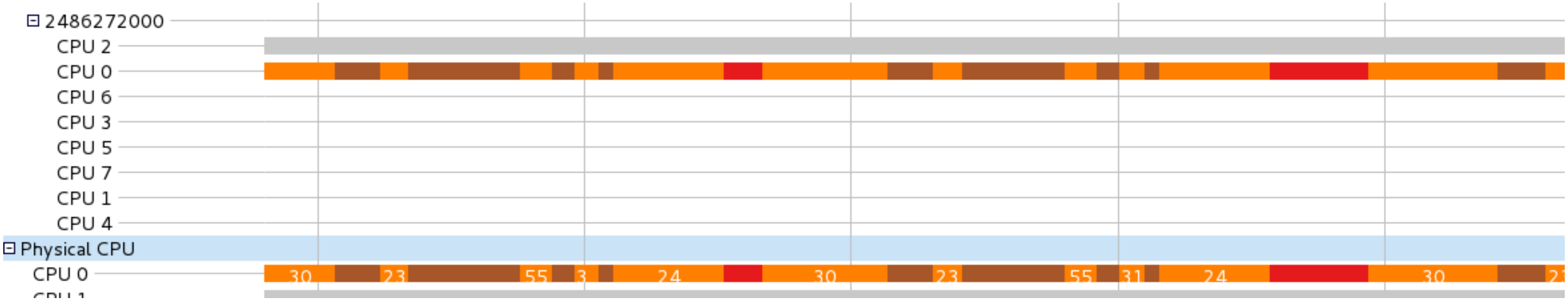
L2: OneOS

L1: VM Hypervisor

L0: Host Hypervisor

- We traced the execution flow of all levels finding out when the code of the host hypervisor, the VM hypervisor, and the nested VM is executing
- The code of L2 (OneOS) runs for a small period of time and then it exits to the L0 (host) to handle a privileged instruction.
- Most of the time, code of L0 and L1 execute and then for a small amount of time code of L2 executes.
- For further investigation, we look at the exit reason for each exit from L2 to L0.

Execution Flow analysis of OneOS: L2 -> L1 -> L0



L2: OneOS

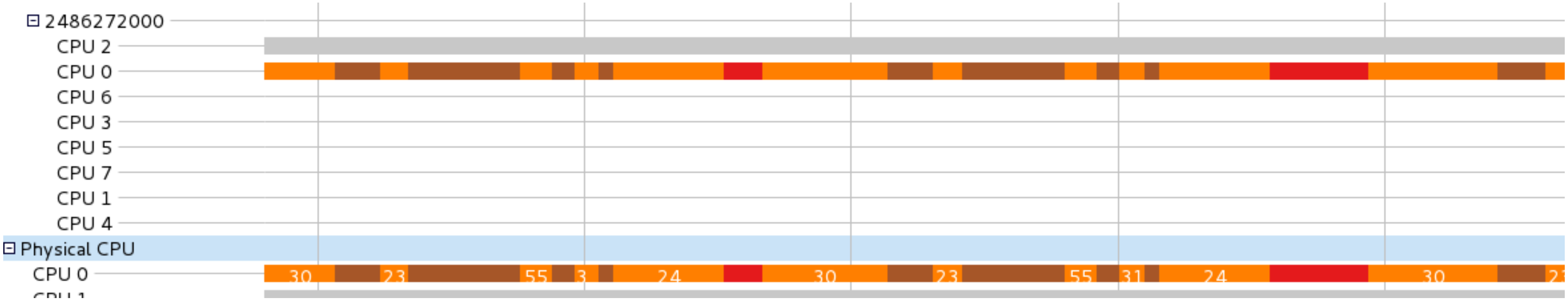
L1: VM Hypervisor

L0: Host Hypervisor

- Each vm_exit has a reason which is written in the exit_reason field.
- For example, if a syscall_read executes in the VM, it causes a vm_exit with exit reason of 30, which is I/O instruction.
- The frequency of each different exit reason could represent a lot of information about the instructions running in the VM.
- A high frequency of exit reason 30 shows intense I/O activity in a VM.

.

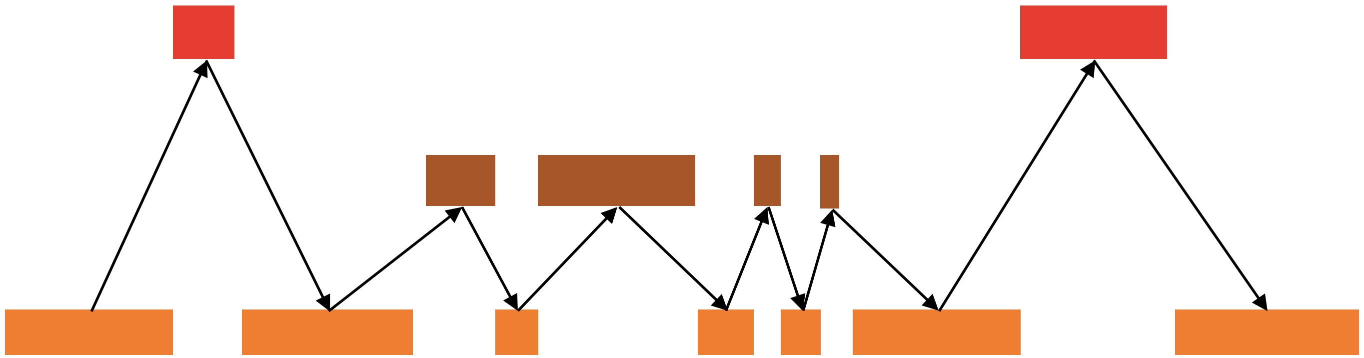
Execution Flow analysis of OneOS: L2 -> L1 -> L0



L2: OneOS

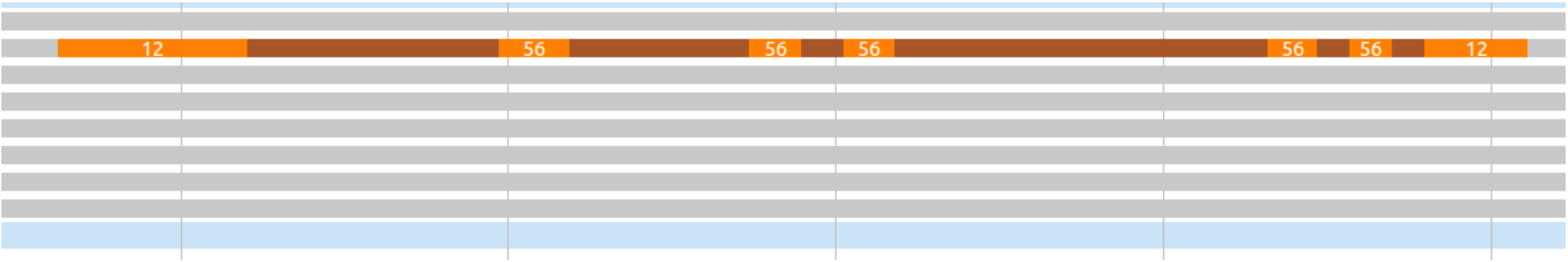
L1: VM Hypervisor

L0: Host Hypervisor



Use the bounce code

Execution Flow analysis: : L1 -> L0



L1: VM Hypervisor

L0: Host Hypervisor

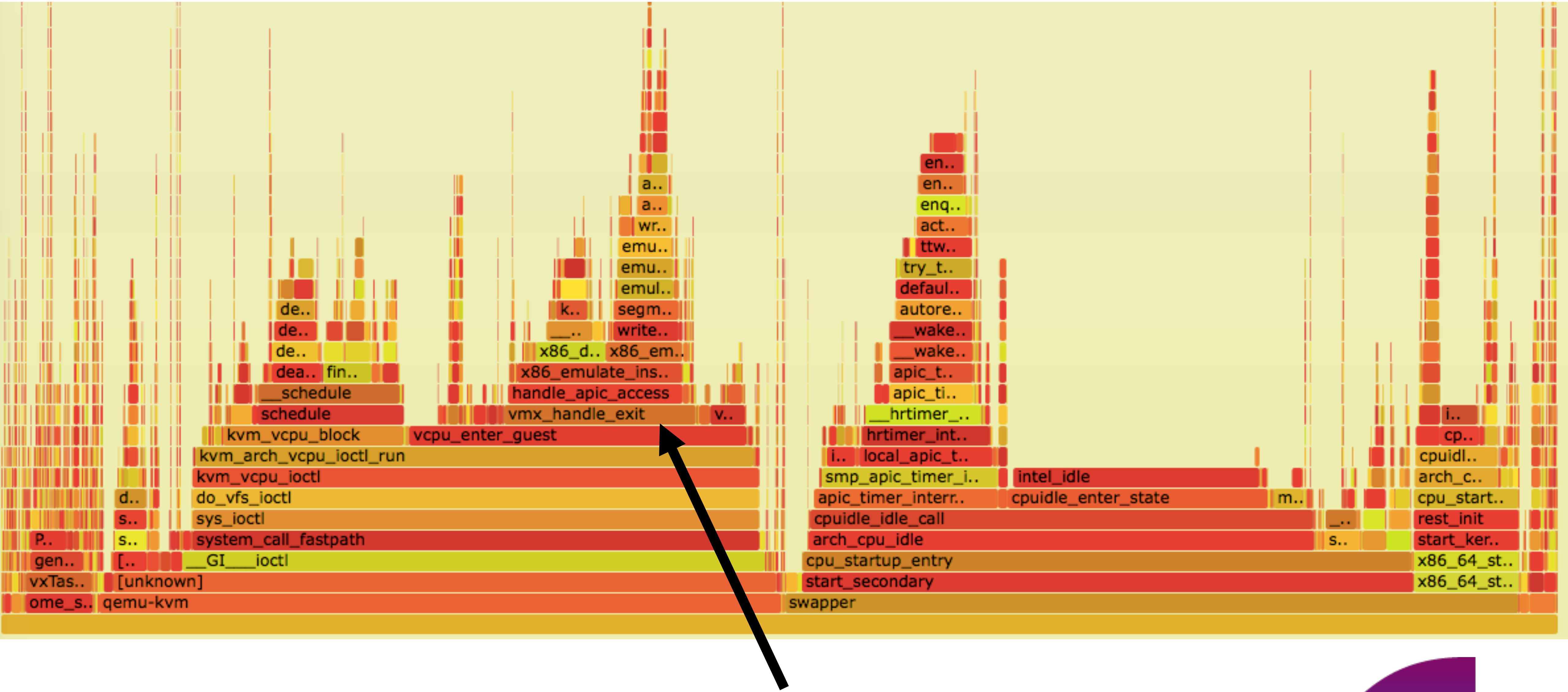
- The above shows the execution flow of the L1 layer traces on L0
- Majority of exit reasons are 56 which corresponds to APIC calls
- It looks more than likely that APIC is being emulated for nested VM L1
- Linux kernels above 4.0 introduce virtual APIC for nested VMs reducing the overhead associated with emulation

From the 4.8 kernel source:

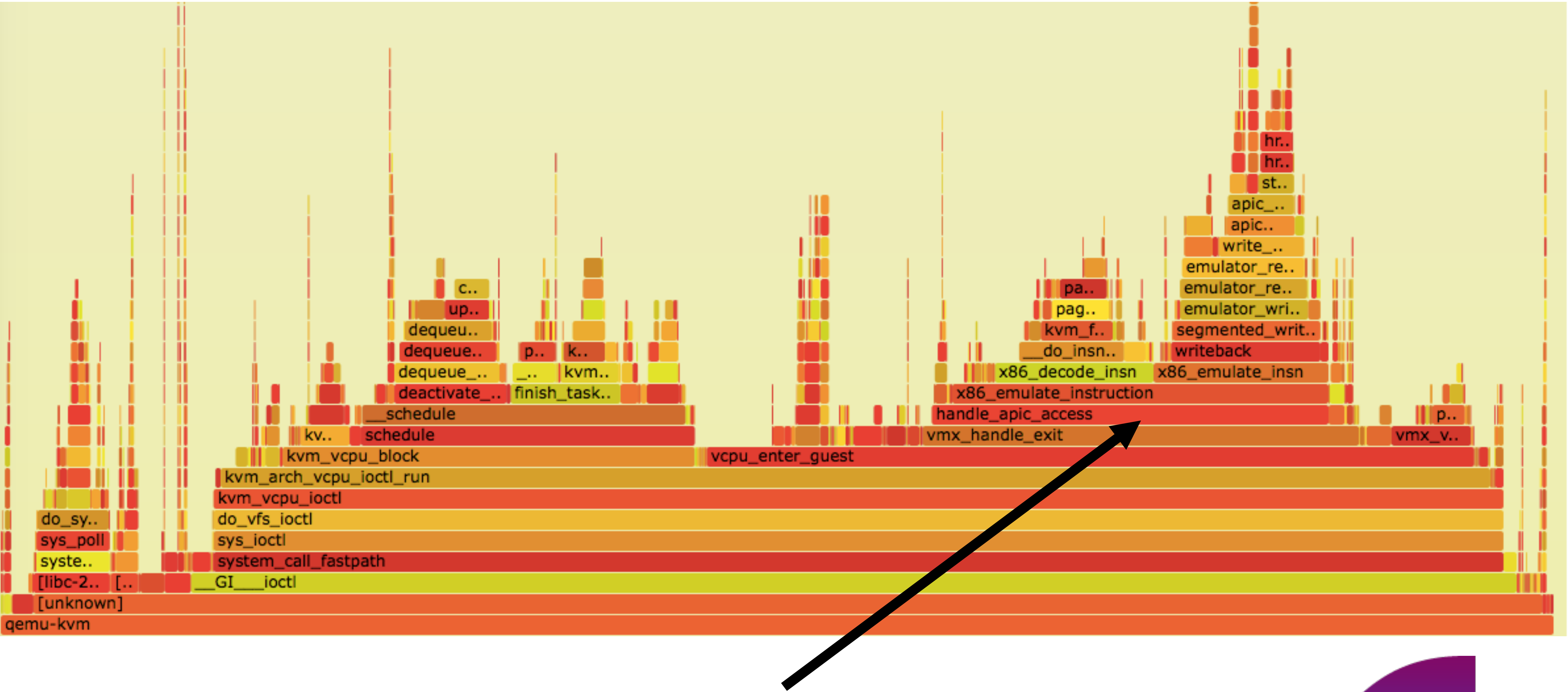
Linux/arch/x86/kvm/vmx.c

```
static inline bool nested_cpu_has_virt_x2apic_mode(struct vmcs12 *vmcs12)
{
        return nested_cpu_has2(vmcs12, SECONDARY_EXEC_VIRTUALIZE_X2APIC_MODE);
}

static inline bool nested_cpu_has_vpid(struct vmcs12 *vmcs12)
{
        return nested_cpu_has2(vmcs12, SECONDARY_EXEC_ENABLE_VPID);
}

static inline bool nested_cpu_has_apic_reg_virt(struct vmcs12 *vmcs12)
{
        return nested_cpu_has2(vmcs12, SECONDARY_EXEC_APIC_REGISTER_VIRT);
}

static inline bool nested_cpu_has_vid(struct vmcs12 *vmcs12)
{
        return nested_cpu_has2(vmcs12, SECONDARY_EXEC_VIRTUAL_INTR_DELIVERY);
}

static inline bool nested_cpu_has_posted_intr(struct vmcs12 *vmcs12)
{
        return vmcs12->pin_based_vm_exec_control & PIN_BASED_POSTED_INTR;
```

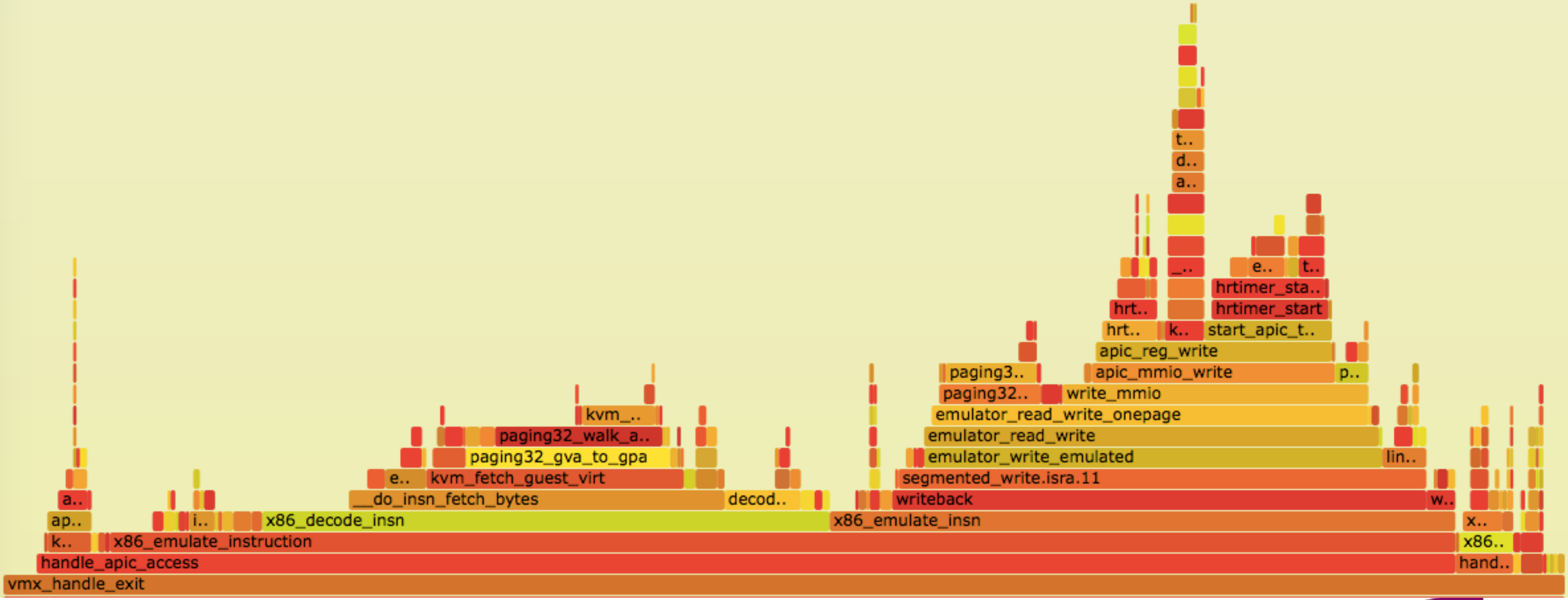
Profiling nested qemu-kvm with perf



Profiling nested qemu-kvm with perf



Profiling nested qemu-kvm with perf



What next?

- We're creating a lightweight SIM specific image for the L1 VM host which will have the latest nested virtualization kernel improvements with a smaller footprint
- Use LTTng to reduce any overhead that may be non specific to virtualization but impacting the performance of the product code
- We are looking to align views between VM exits and OneOS code so we can sync the execution of OneOS code along with the host
- Tracing the early boot of our OneOS with a bare metal tracer

POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

Thank You

ciena
Experience. Outcomes.