Notes on Nested Virtualization (KVM on KVM)

CloudOpen Eu 2013

Kashyap Chamarthy <kashyap@redhat.com>

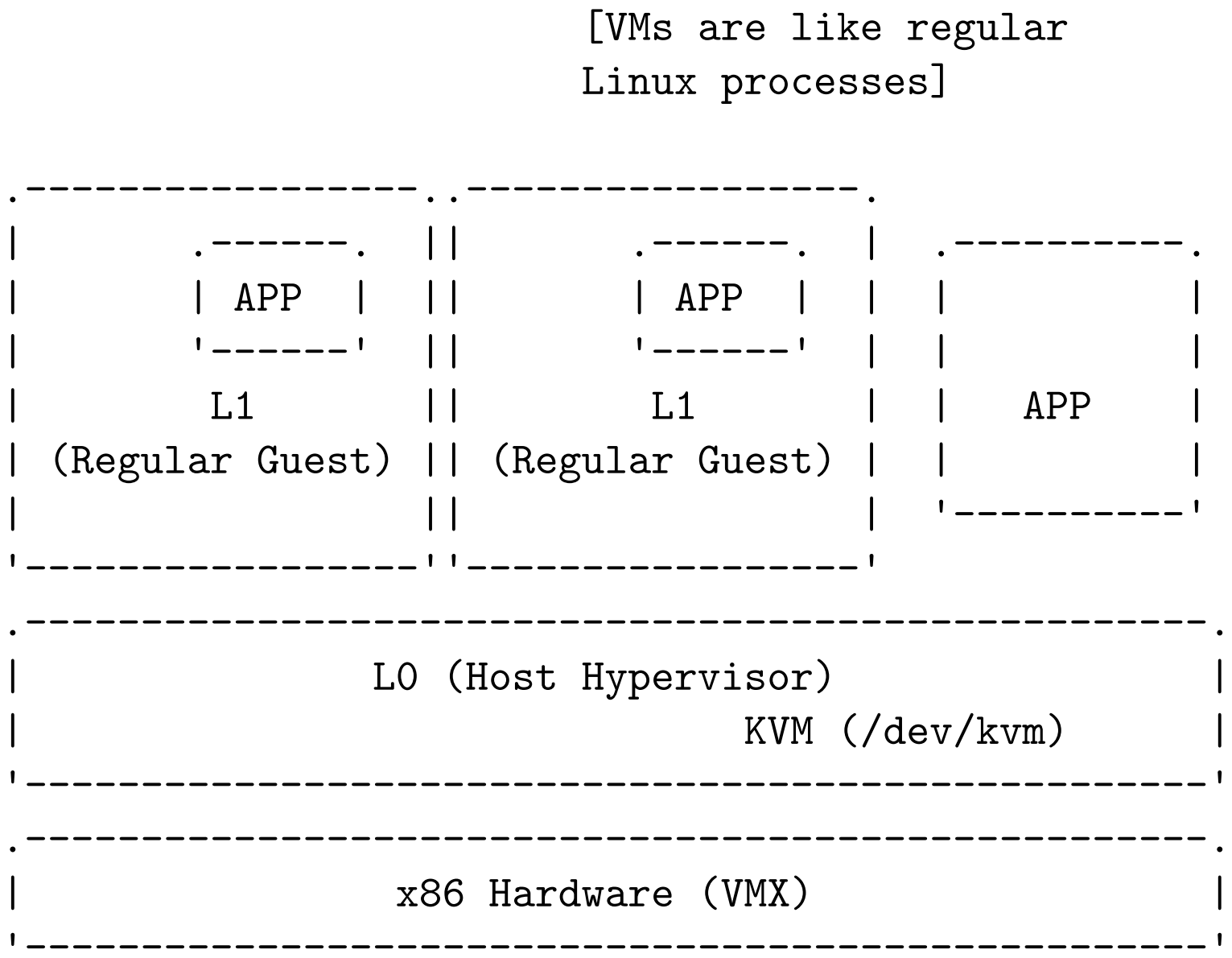
23 OCT 2013

Agenda

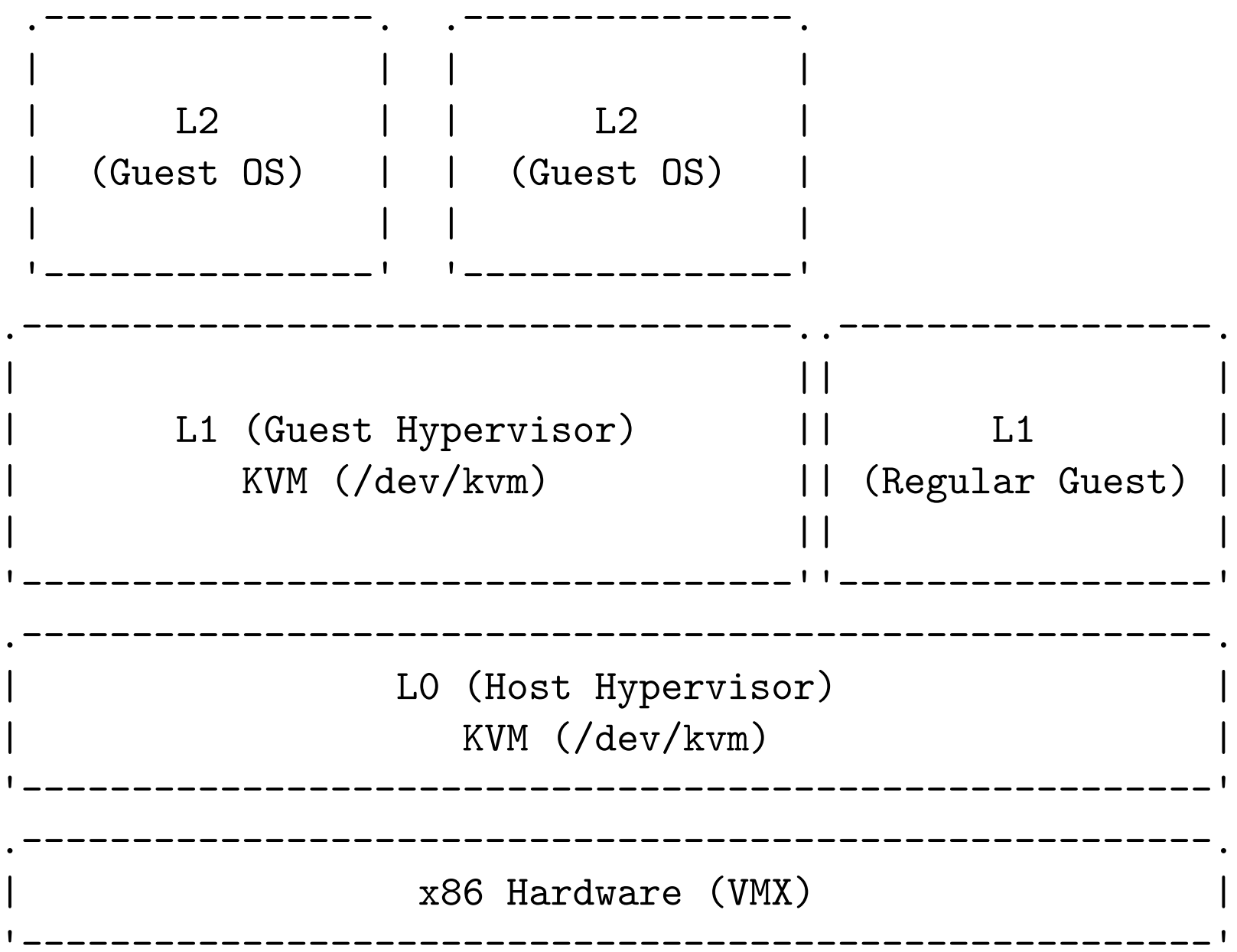
Section 1

Overview

Regular Virt



Nested VMX


L2
(Guest OS)
L2
(Guest OS)
L1 (Guest Hypervisor)
KVM (/dev/kvm)
L1
(Regular Guest)
L0 (Host Hypervisor)
KVM (/dev/kvm)
x86 Hardware (VMX)

Why Nested Virt?

- User-controlled hypervisors
- OpenStack in a VM
- Test/Demonstrate virtualization setups
- Live-migration of hypervisors

Bottleneck?

Performance overhead – Too many VM Exits & VM Entries

redhat.

Section 2

Optimizations for performance

Nested VMX Optimizations

- **VMCS Shadowing** (Intel)
  - Processor optimization that lets L0 define a shadow VMCS
  - Guest hypervisor can access shadow VMCS directly (in hardware)
    - Consequently, reducing the number of VM entries & exits

Nested MMU Optimizations

- EPT (Extended Page Tables) – Hardware optimization that provides a second level address translation.
- **Nested** EPT
  - Emulate EPT for L1
  - Allows L1 to use EPT when running a nested guest (L2)

Section 3

Configuration

On Host Hypervisor

- On L1 (Host Hypervisor):

```
$ cat /sys/module/kvm_intel/parameters/nested
Y
$ cat /sys/module/kvm_intel/parameters/enable_shadow_vmcs
Y
$ cat /sys/module/kvm_intel/parameters/ept
Y
```

Libvirt XML snippet

- Expose VMX instructions to L1

```
<cpu match='exact'>
<model>Haswell</model>
<feature policy='require' name='vmx'/>
</cpu>
```

[Or]

- Export host CPU model to guest CPU

```
<cpu mode='host-passthrough'>
```

L1 libvirt invocation

- L1 QEMU command-line prepared by libvirt
  - ```
    $ /usr/bin/qemu-system-x86_64 -machine \
    \ accel=kvm -name guest-hypervisor \
    -S -machine pc-i440fx-1.4,accel=kvm,usb=off \
    -cpu host \
    -m 10240 -smp 4,sockets=4,cores=1,threads=1 \
    -uuid 4ed9ac0b-7f72-dfcf-68b3-e6fe2ac588b2 \
    -nographic -no-user-config  -nodefaults \
    [...]
    ```
    ```

redhat.

Section 4

Initial workload with kvm.git

A simple test

- Compile kernel with *make defconfig*

```
#!/bin/bash

cd /home/test/src/linux

make defconfig

for i in {1..10}; do
     make clean;
     sync;
     sudo sh -c "echo 3 > /proc/sys/vm/drop_caches"
     time make;
done 2>&1 | tee make-timings.txt
```

Kernel compile with and without VMCS Shadowing

- *make* process (10 iterations)

VMCS Shadowing	L2	L1	L0
Without VMCS Shadowing	8m50s	7m21s	6m59s
With VMCS Shadowing	8m26s	7m18s	6m53s

- In L2's case, a **5%** improvement can be noticed *with* VMCS Shadowing enabled on L0.

Kernel compile with different MMU combinations

- A single *make* process (10 iterations)

MMU Combination	L2	L1	L0
Shadow-on-Shadow	55m54s	12m27s	2m14s
Shadow-on-EPT	11m39s	2m22s	2m14s
EPT-on-EPT (nEPT)	4m53s	3m56s	2m14s

- In L2's case, w/ nEPT: **10x** improvement vs Shadow-on-Shadow; **3x** vs Shadow-on-EPT

redhat.

Section 5

Conclusions & further to do

Summary

- L2 performance improvements are real w/ current **kvm.git** – which has nEPT & VMCS Shadowing support

- A *ton* of test combinations!

To do

- Run Autotest *virt-test* – exercises different guest OSes
- More fine-grained tests, worst-case workloads (cpuid?)
- KVM unit-tests in L1
- Disk and Network I/O – Multi-level device assignment

Testing from kvm.git

- To test from kvm.git *queue* (patches are tested here)

```
$ git clone \
  git://git.kernel.org/pub/scm/virt/kvm/kvm.git

$ git branch -d test_ept1 origin/queue

$ make -j6 && make bzImage && make modules

$ make modules_install && make install
```

Notes and References

Configuration notes with Haswell

`https://github.com/kashyapc/nvmx-haswell/blob/master/SETUP-nVMX.rst`

Nested Virtualization notes

`https://kashyapc.wordpress.com/tag/nested-virtualization/`

OpenStack RDO

`http://openstack.redhat.com`

KVM website

`http://www.linux-kvm.org/`

Autotest virt-test

`https://github.com/autotest/virt-test/`

The end.

Thanks for listening.