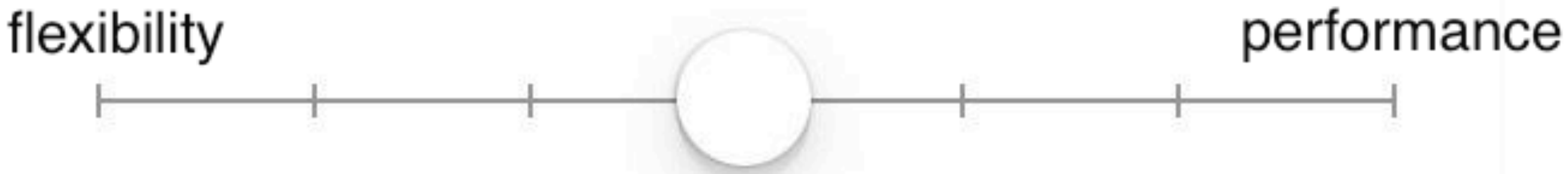
Helping Users Maximize VM Performance

Martin Polednik (@mpolednik)
Software Engineer @ Red Hat



flexibility

performance

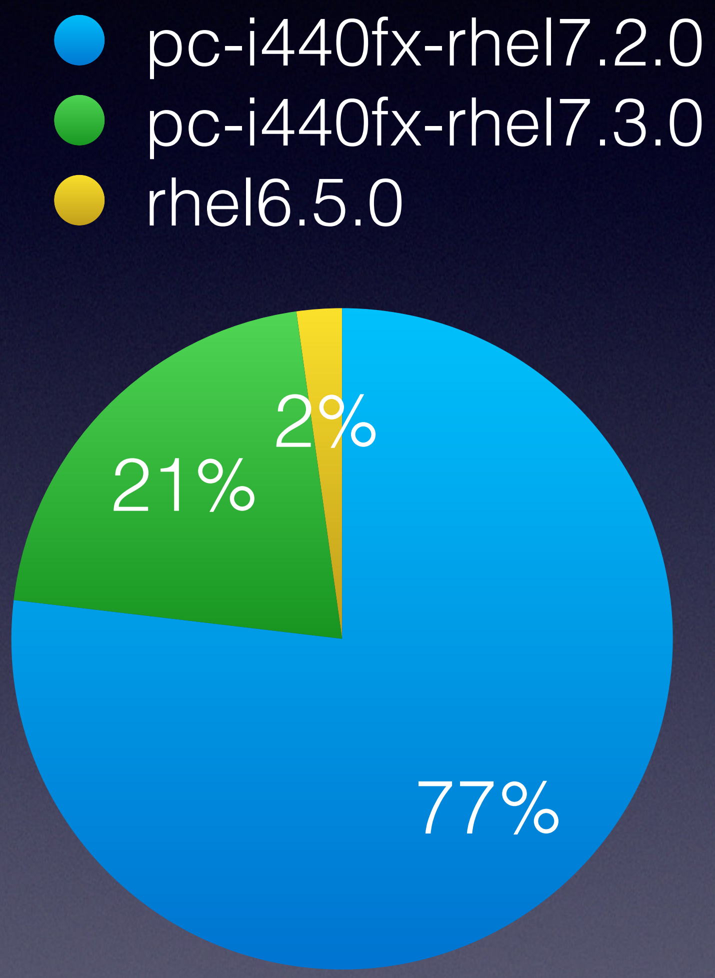
The Data

- oVirt databases from sosreports
- ~ 40,000 virtual machine (VM) definitions
- ~ 700 clusters*
- ~ 2,200 hosts
- ~ 60,000 disks

* oVirt specific entity that consists of hosts, VMs, disks, networks etc. Consider it a scheduling domain.

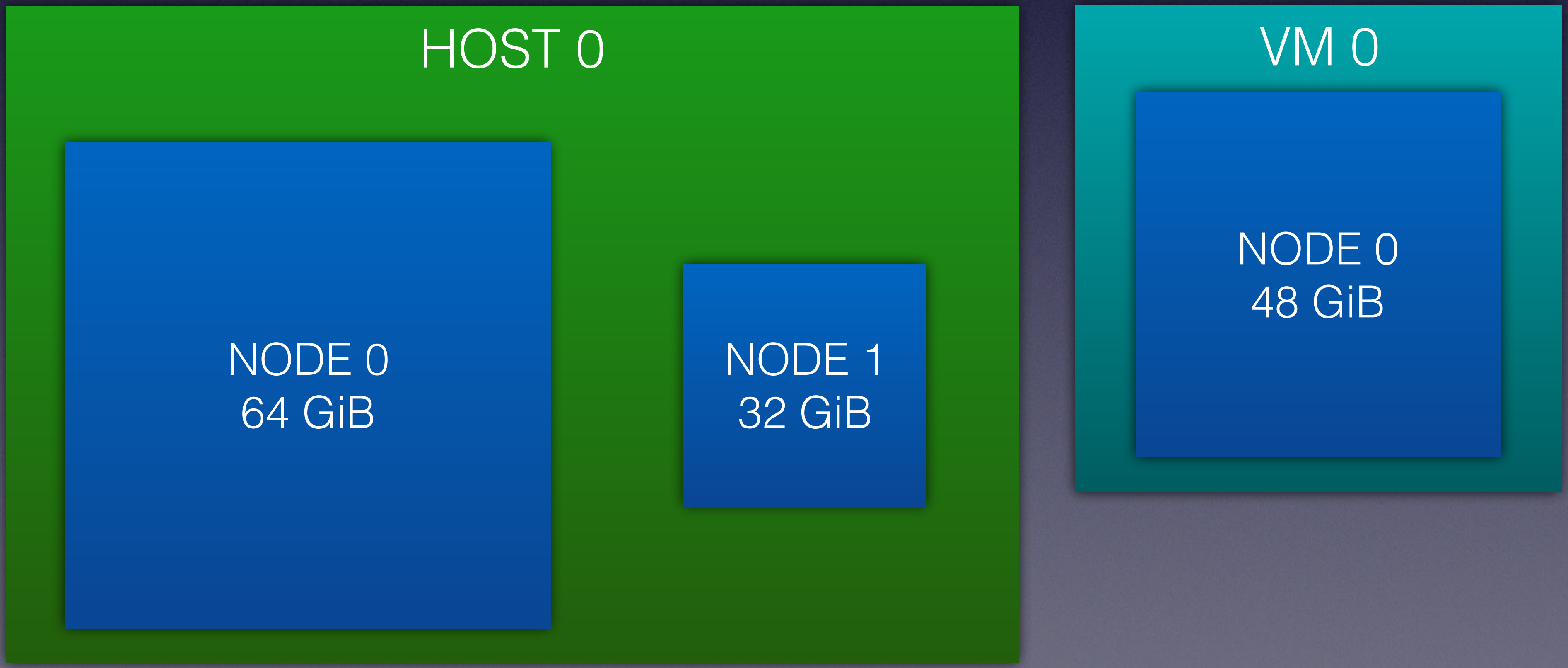
Machine Types

- clusters "group" VMs by machine type
- updating to a newer cluster is a nontrivial process



NUMA

- soft violation: VM does not fit within some of the host's NUMA nodes
- example: VM 0:NODE 0 doesn't fit within HOST 0:NODE 1
- could be solved by pinning

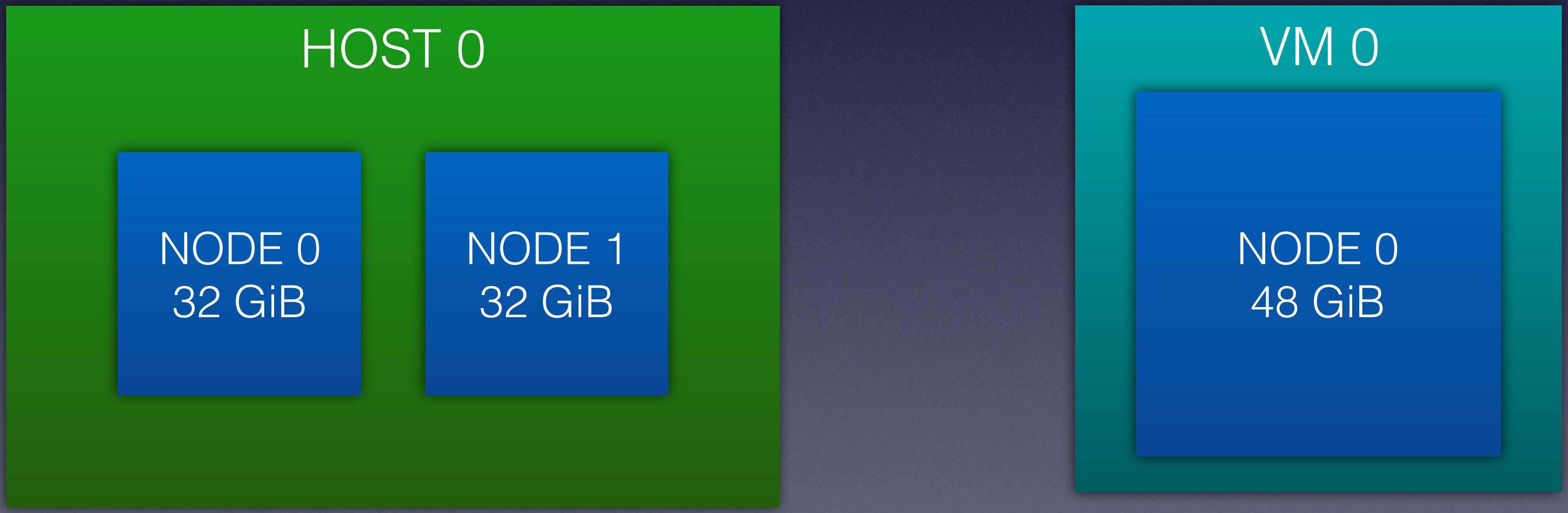


Soft NUMA Violations

- 17.01 % of VM definitions
- the query considered scheduling domains (clusters)
- "there exists a host in the cluster whose NUMA node is smaller than the NUMA node of the VM"
- worst case in cluster AND host scheduling

NUMA

- hard violation: VM does not fit within any of the host's NUMA nodes
- example: VM 0:NODE 0 doesn't fit within HOST 0:NODE 0 or HOST 0:NODE 1



Hard NUMA Violations

- 9.74 % of VM definitions
- scheduling domains were considered
- "there exists a host in the cluster whose NUMA nodes are smaller than the NUMA node of the VM"
- worst case in cluster scheduling

Solution

- warn the user about suboptimal NUMA topology
  - easy to determine on the cluster level
  - important for specific applications (huge DBs)
- future: create the nodes automatically?

NUMA & CPU pinning

- low adoption, why?
  - no migration (disabled at management level)
  - HA is hard, breaks cluster logic (only HA between subset of hosts)
  - limited scheduling (pin to host)
- can we change that?

NUMA & CPU pinning

- host-passthrough CPU (aka copy features)
- automatically pin CPUs
- e.g. 4 NUMA nodes, 12 CPUs per node
    - node CPU0, CPU1 ~> "service" CPUs (emulation thread, IO thread, virt daemons)
    - CPU2 through CPU11 ~> compute CPUs
    - if #vCPU > 10, ask the user to add a virtual node
    - easy to think about RT too!

Hugepages

- platform default + extended sizes
- either preallocated or dynamically allocated
- at least for x86_64 1 GiB (pdpe1gb) preferred, other sizes configurable
- THP is hit or miss performance-wise

Hugepages

- no cluster-level overcommit
- no memory hot(un)plug, limited migration (management layer constraints)
- "hard" resource limit
- NUMA-aware allocation

Hugepages Allocation

- could cause VM start delays
- opt-out at the host level, disabled in scheduler
- reserved hugepages concept (DPDK etc.)
  - `max(vm_hugepages - free_hugepages, 0)`

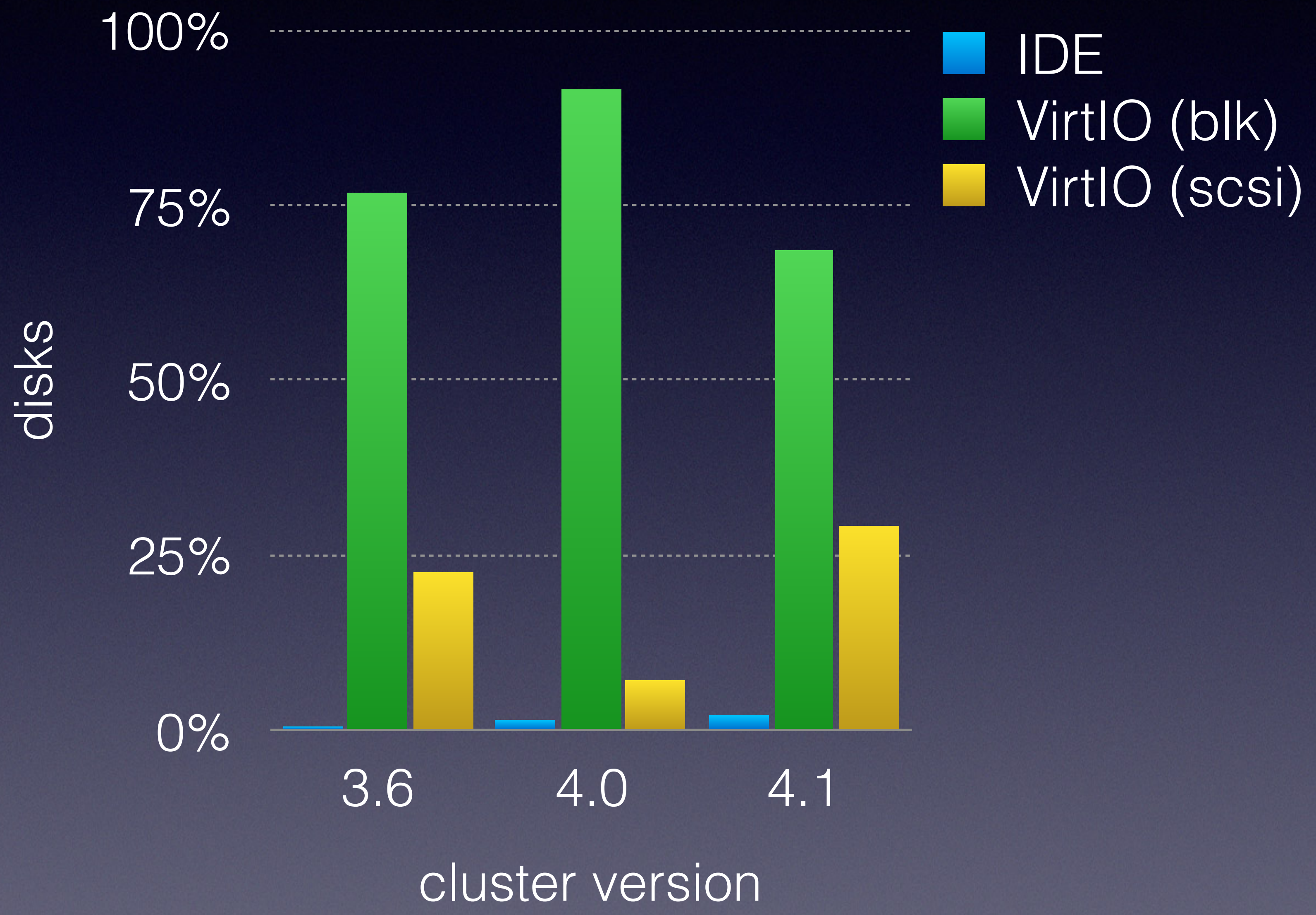
L3 cache

- https://git.qemu.org/?p=qemu.git;a=commit;h=14c985cffa6cb177fc01a163d8bcf227c104718c
- QEMU: -cpu foo,l3-cache=on
- libvirt: <cpu><cache level='3' mode='emulate'/></cpu>
- less inter-processor interrupts (IPIs) -> less VMEXITs
- essential for SAP workloads

Disk Interface

- choice between IDE, VirtIO-blk, VirtIO-SCSI (+ passthrough)
- 3.6, 4.0 defaults to VirtIO-blk, 4.1+ to VirtIO-SCSI
- VirtIO-SCSI controller by default in VMs (hotplug capability) :(
- TRIM is important to people!

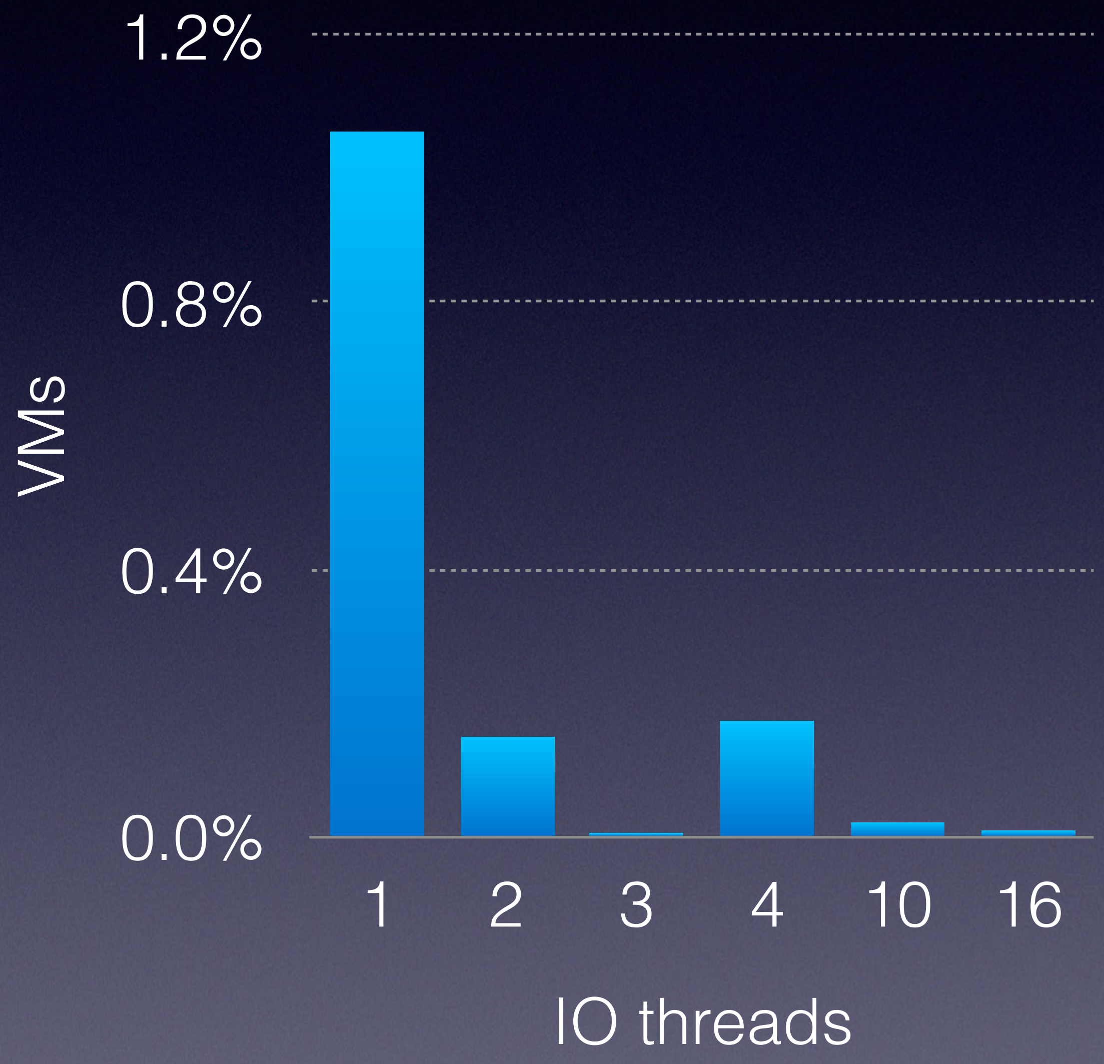
Disk Interface



IO Threads

- 3.6, 4.0, 4.1 allow specifying # of IO threads
- no hints about which number to use

IO Threads



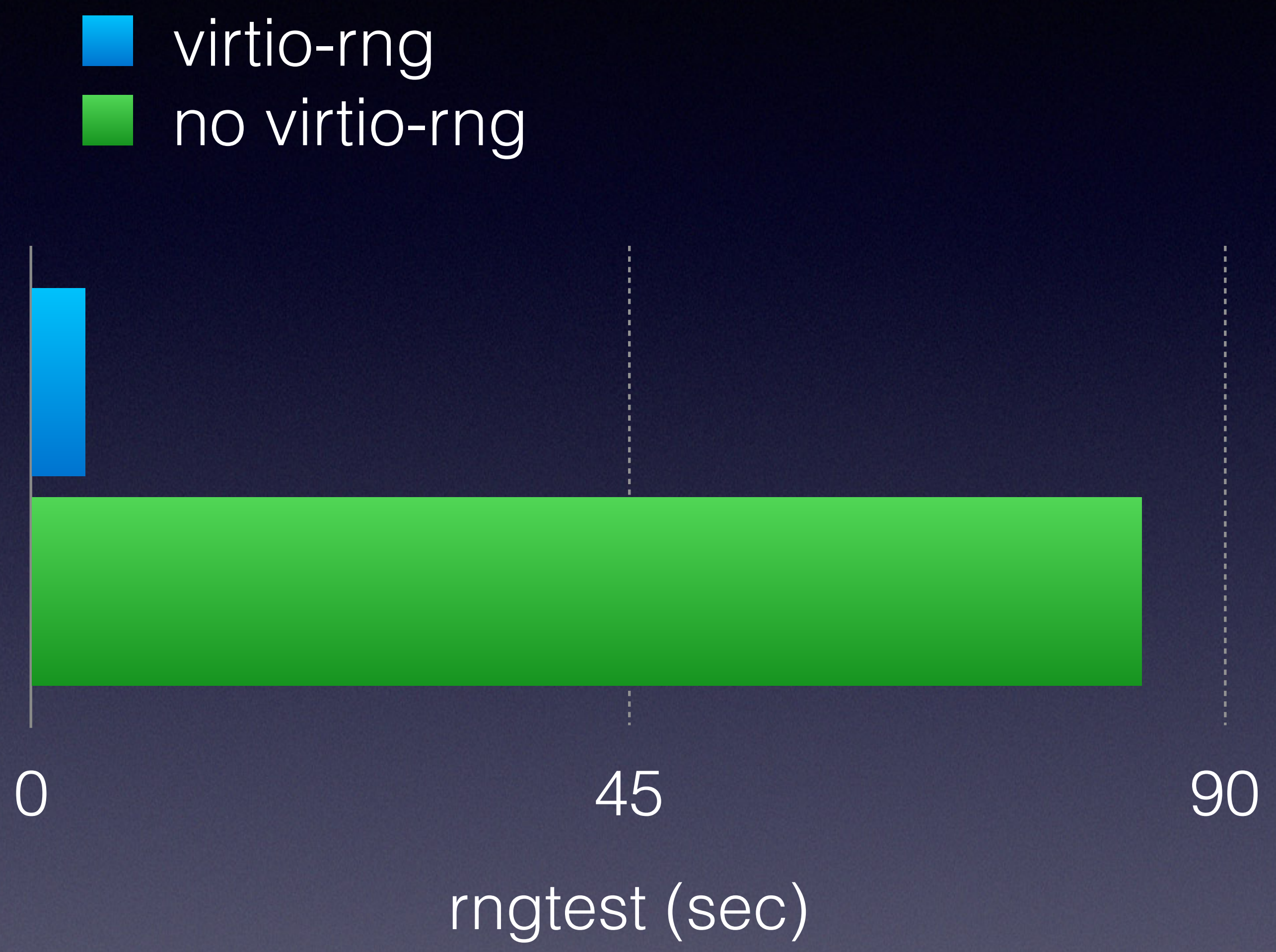
IO Threads

- testing has shown the "sweet spot" to be 1 IO thread
  - therefore, oVirt no longer (easily) allows arbitrary numbers
  - override via hooks
- https://mpolednik.github.io/2017/01/23/virtio-blk-vs-virtio-scsi/

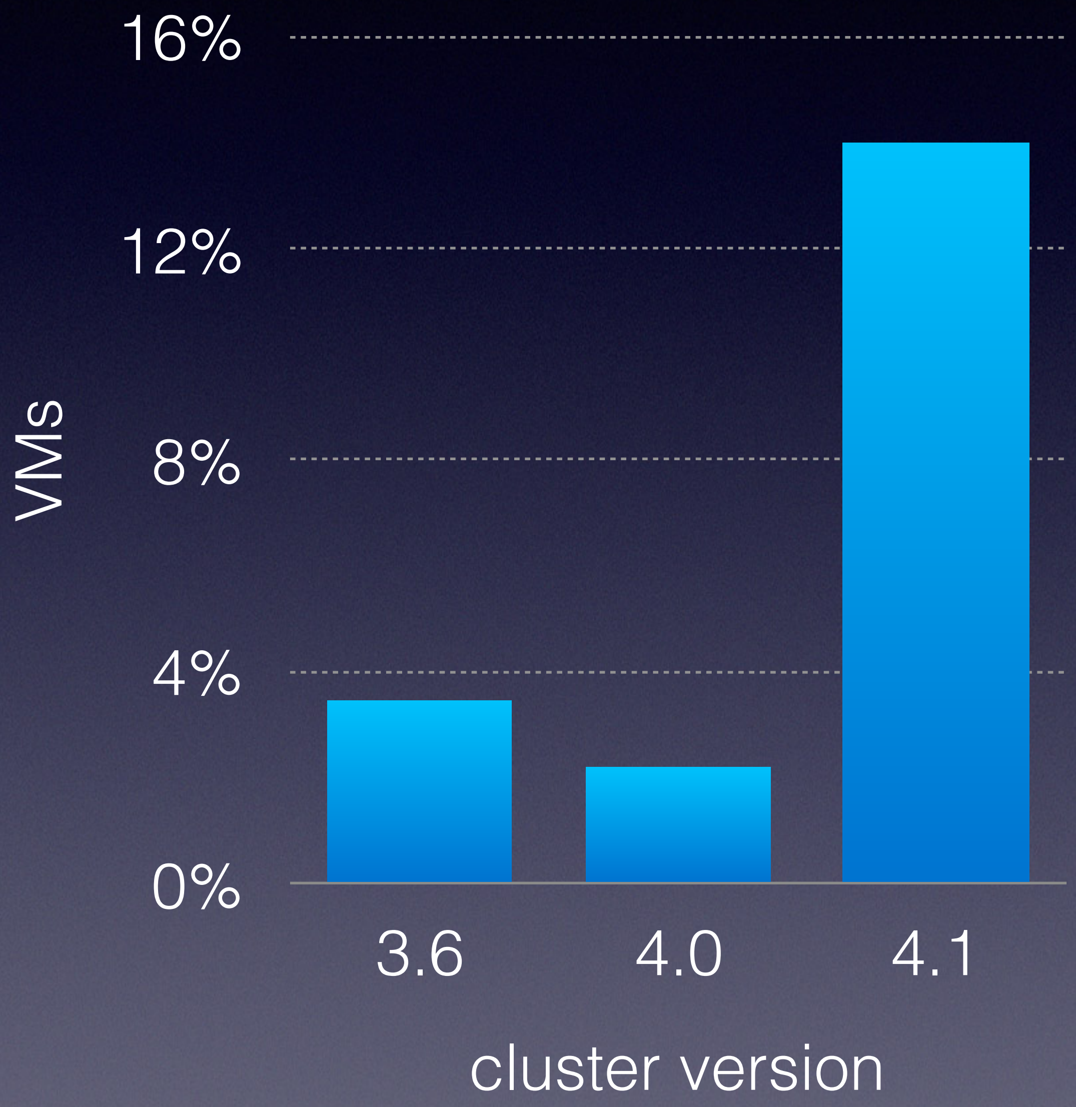
VirtIO RNG

- "low hanging fruit"
- improves virtually any operation that uses PRNG (e.g. OS installation, GPG key generation)
- optional in 3.6, 4.0, default in 4.1 - no downsides?

VirtIO RNG perf



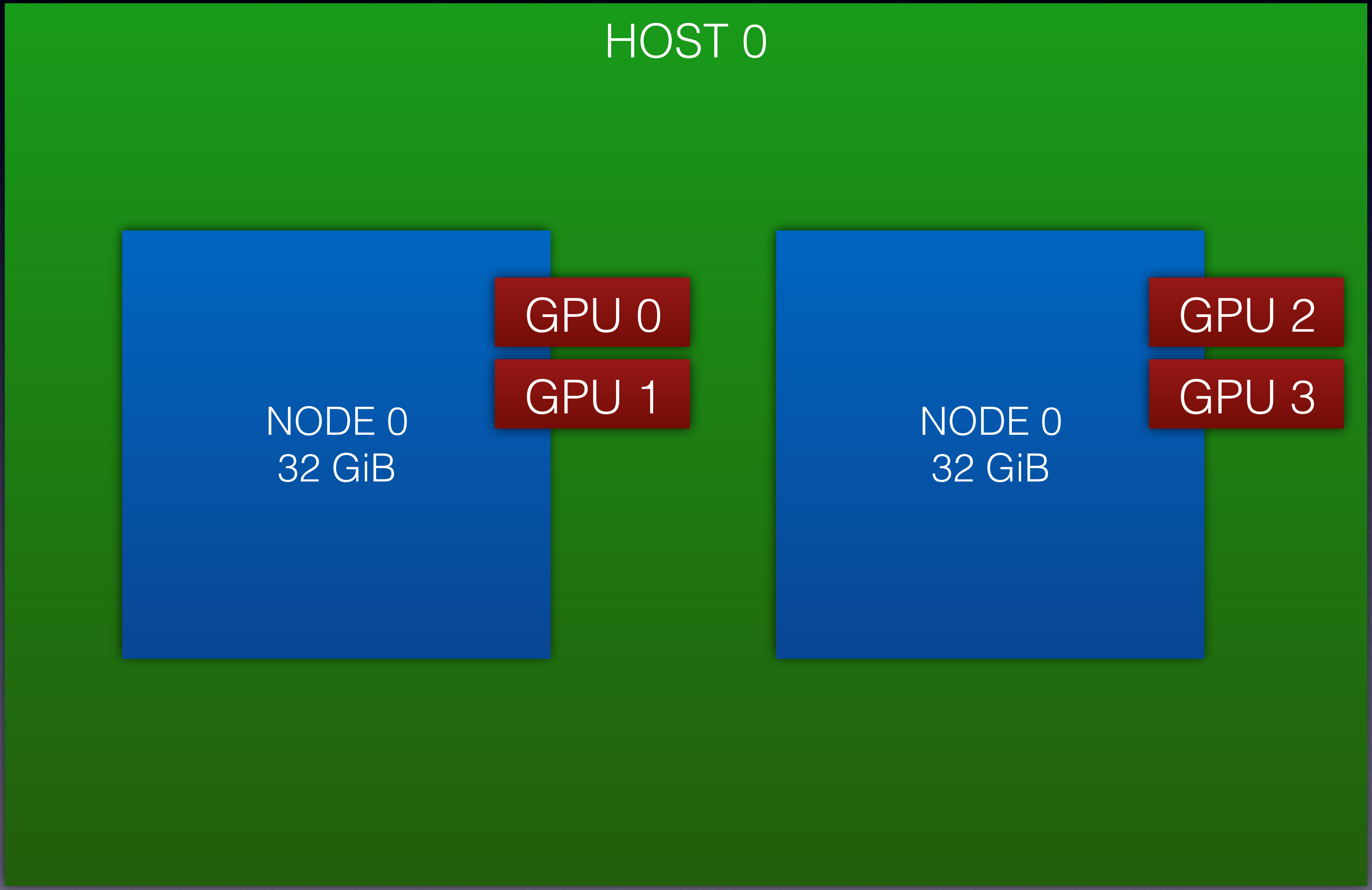
VirtIO RNG



Host Devices

- using real hardware to accelerate the VMs
- GPUs, NICs, NVMe disks
- reduced CPU load
- should still honor NUMA locality
- hard resource limit

Host Devices



Host Devices

- easy to tune numa automatically for simple case (all host devices within single numa node)
- more complicated if host devices origin from multiple NUMA nodes

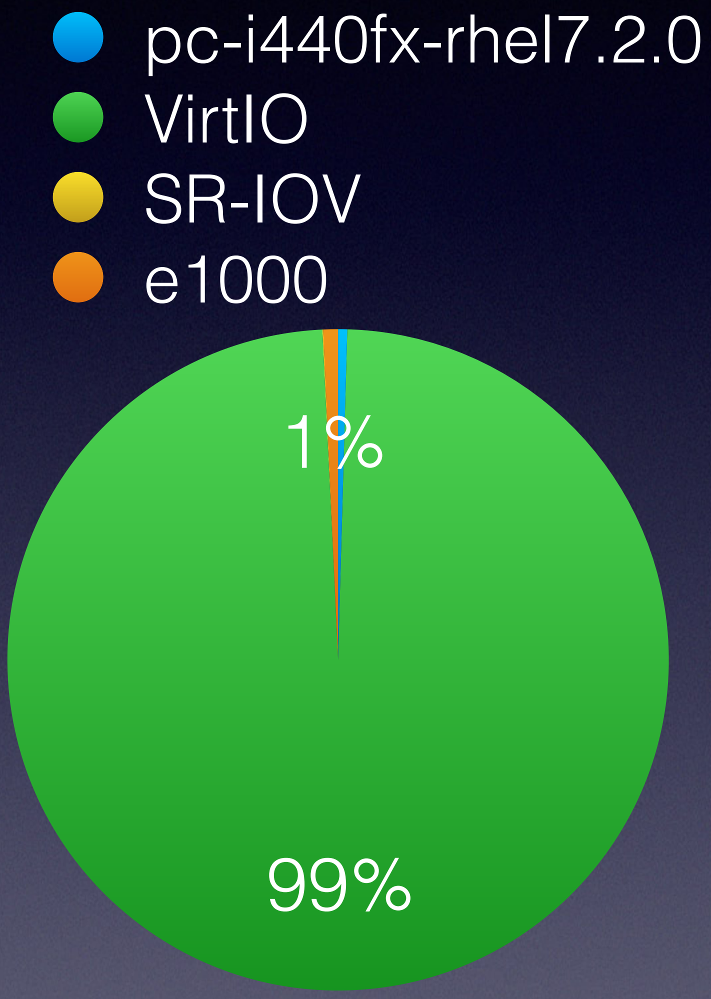
Network

- VirtIO is the preferred "flexibility" choice

- SR-IOV for performance/NFV, migration enabled

- emulated NICs for compatibility

- looks good as it is



Migration Performance

- relevant for clusters
- maximum downtime incremented in steps
- limit number of inbound/outbound migrations to avoid oversaturated network
- post copy - needs to be enabled explicitly, success chance dependent on user's network
    - don't expect high bandwidth, redundant network in every case

Migration Performance

	Legacy	Minimal downtime	Suspend workload if needed	Post copy
20 GiB RAM	Failed After 12 min	41 min 31 sec	31 min 42 sec	25 min
20 GiB RAM, 50 msec latency	Failed After 17min	47 min 24 sec	1 h 12 min 31 sec	48 min 10 sec
40 VM,1 GiB RAM	AVG: 1 min 40 sec	AVG: 1 min 50 sec	AVG: 4 min	AVG: 1 min 30 sec

KSM

- hugetlbfs not scanned by ksmd
- no overcommit for VMs that are considered high performance
- waste of CPU cycles?

Devices

- graphics, video, USBs, smartcard, watchdog, balloon
- do we need them?
- no known (to us) performance effects
  - removing them shouldn't hurt
  - no data though

Devices

- some functionality tradeoffs (ballon and memory hot(unplug) in the future)
- running headless
  - no graphics
  - no video
  - no spice/vnc, just console connectivity
  - console proxy to connect to the guests

Implementation

- do as many "safe" tweaks as possible
    - with a single NUMA node, go for device locality
- warn about suboptimal configuration
    - NUMA violation => suggest a vNODE
- inform about tradeoffs
    - VirtIO-blk vs VirtIO-SCSI
- allow user to override as many tunes as possible!

Benchmarks

- synthetic benchmarks show 0-15 % performance improvement
  - pgbench ~ 10 % improvement
  - pts/enclode-flac ~ 0.1 % improvement
- more data in the future as reports come in

Summary

- align everything with NUMA topology
- suggest pinning where possible (incl. IO thread, emulator thread)
- suggest hugepages
- expose l3 cache
- VirtIO-RNG
- host devices (hardware) > VirtIO > emulation
- remove unneeded devices

Summary

- benchmark your workload and tune accordingly!

Questions?

Thank you!
Slides & Blog @ https://mpolednik.github.io/