Performance Analysis and Tuning Red Hat Enterprise Linux 6 and 7

D. John Shakshober
Director /Sr Consulting Engineer
Red Hat Performance Engineering

Performance Analysis of RHEL6/7

- Performance Tools and Tuned
- Performance Analysis Utilities
  - Perf, Pcp,Tuna
  - Out of the Box Experience
- Disk / NUMA Tuning
- Network / Low Lat / NFV
- RHEV KVM / Open Stack
- Atomic / Open Shift

Red Hat Performance Engineering

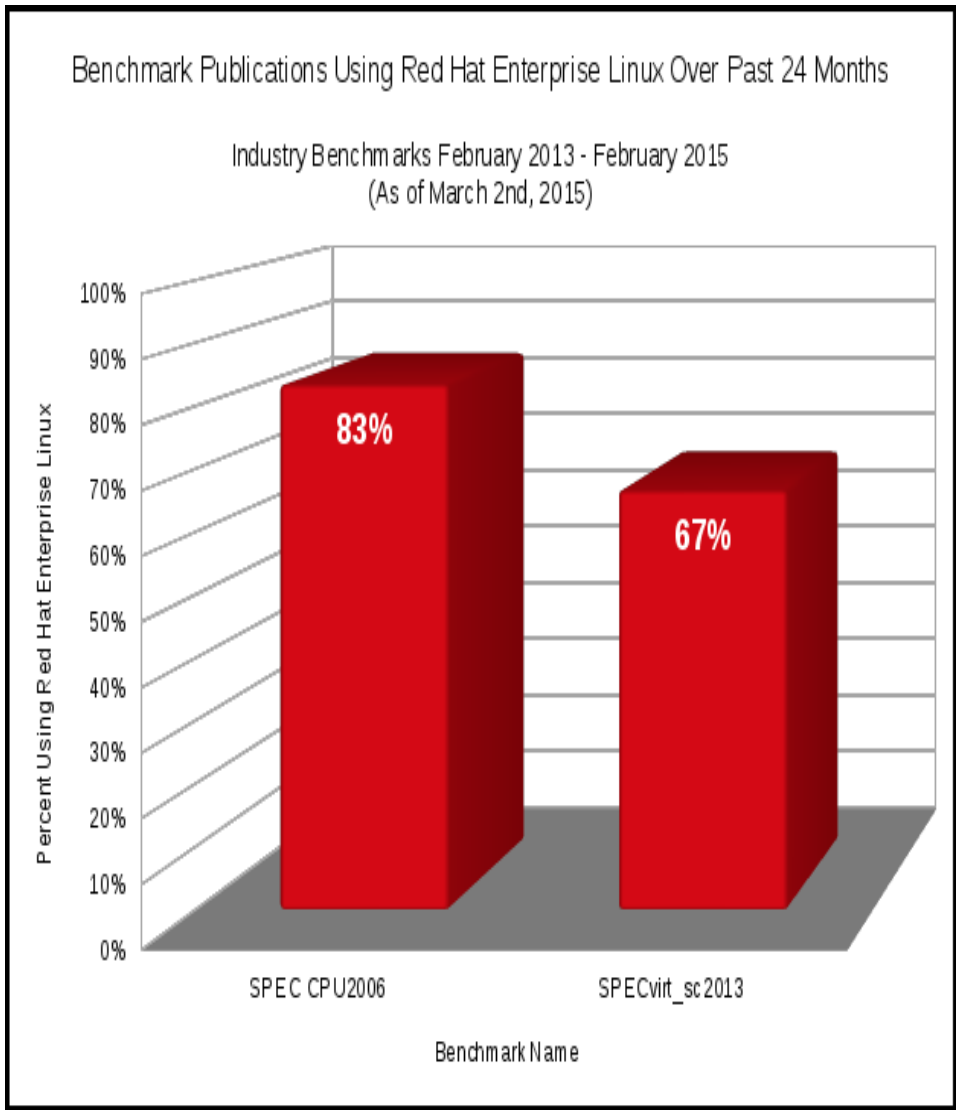
- Benchmarks – code path coverage
  - CPU – linpack, lmbench
  - Memory – lmbench, McCalpin Streams
  - Disk IO – Iozone, aiostress – scsi, FC, iSCSI
  - Filesystem – IOzone, postmark– ext3/4, xfs. gfs2,gluster
  - Network – Netperf – 10 Gbit, 40 Gbit IB, PCI3
- Bare Metal, RHEL6/7 KVM, Atomic Container
- White box Intel/Arm/Power/AMD w/OEMs

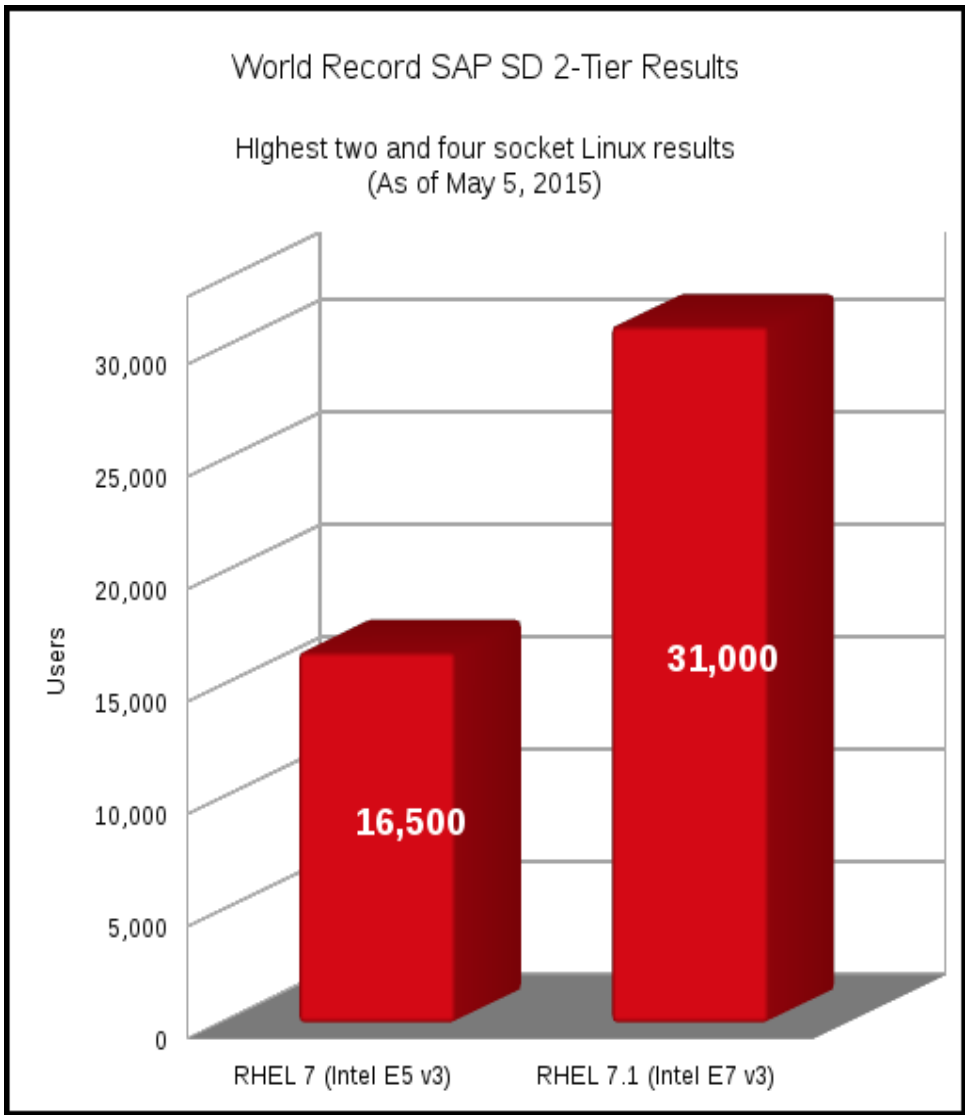
RHEL Performance Evolution

RHEL5	RHEL6	RHEL7	RH Cloud
Static Hugepages	Transparent Hugepages	Tuned - throughput-performance (default)	RHEV tuned profile
CPU Sets	Tuned - Choose Profile	Automatic NUMA-balancing	RHOP Tuned, NUMA, SR-IOV
Ktune on/off	NUMAD - userspace	RHEL Realtime	RHEL Atomic Host
CPU Affinity (taskset)	cgroups	Containers, Docker	OpenShift v3
NUMA Pinning (numactl)	irqbalance - NUMA enhanced	irqbalance - NUMA enhanced	CloudForms
irqbalance			

RHEL / Intel Benchmark Haswell EX

(http://rhelblog.redhat.com/2015/05/06/red-hat-delivers-leading-application-performance-with-the-latest-intel-xeon-processors/





Subsystem Analysis

Subsystem Analysis: CPU

```
# mpstat -P 1 1
04:27:56 PM  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
04:27:57 PM    1    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00  100.00
04:27:58 PM    1   73.27    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00   26.73
04:27:59 PM    1  100.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
04:28:00 PM    1  100.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
04:28:01 PM    1  100.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
04:28:02 PM    1  100.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
04:28:03 PM    1   26.00    0.00    1.00    0.00    0.00    0.00    0.00    0.00    0.00   73.00
04:28:04 PM    1    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00  100.00
```

Subsystem Analysis: Memory

```
# vmstat 1

procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 2  0      0 25554152 986896 13833932    0    0     0     0  307  266  0  1 99  0  0
 1  0      0 24454408 986896 13833932    0    0     0     0 1286  548  0  4 96  0  0
 1  0      0 23428300 986896 13833932    0    0     0     0 1288  508  0  4 96  0  0
 1  0      0 22371768 986896 13833932    0    0     0     8 1120  150  0  4 96  0  0
 1  0      0 21571872 986896 13833932    0    0     0     0 1162  305  1  4 96  0  0
 1  0      0 21571872 986896 13833932    0    0     0     0 1040   67  4  0 96  0  0
 1  0      0 21571872 986896 13833932    0    0     0     0 1067   90  4  0 96  0  0
 1  0      0 21571872 986896 13833932    0    0     0     0 1045   70  4  0 96  0  0
 0  0      0 25773696 986896 13833932    0    0     0    24  487  130  2  0 98  0  0
```

Subsystem Analysis: Memory

```
# numastat -c qemu  Per-node process memory usage (in Mbs)

PID              Node 0 Node 1 Node 2 Node 3 Total
---------------  ------ ------ ------ ------ -----
10587 (qemu-kvm)   1216   4022   4028   1456 10722
10629 (qemu-kvm)   2108     56    473   8077 10714
10671 (qemu-kvm)   4096   3470   3036    110 10712
10713 (qemu-kvm)   4043   3498   2135   1055 10730
---------------  ------ ------ ------ ------ -----
Total             11462  11045   9672  10698 42877

# numastat -c qemu

Per-node process memory usage (in Mbs)

PID              Node 0 Node 1 Node 2 Node 3 Total
---------------  ------ ------ ------ ------ -----
10587 (qemu-kvm)      0  10723      5      0 10728
10629 (qemu-kvm)      0      0      5  10717 10722
10671 (qemu-kvm)      0      0  10726      0 10726
10713 (qemu-kvm)  10733      0      5      0 10738
---------------  ------ ------ ------ ------ -----
Total             10733  10723  10740  10717 42913
```

Subsystem Analysis: Memory

```
# numastat -mczs

Per-node system memory usage (in MBs):
                Node 0 Node 1 Node 2 Node 3 Node 4 Node 5 Node 6 Node 7  Total
               ------ ------ ------ ------ ------ ------ ------ ------ ------
MemTotal        32766  32768  32768  32768  32768  32768  32768  32752 262126
MemFree         31863  31965  32120  32086  32098  32080  32114  32062 256388
MemUsed           903    803    648    682    670    688    654    690   5738
FilePages          11     26      8     37     21     18      9     45    176
Slab               25     16      7     10     12     36     10     10    126
Active              5     13      4     25     10      9      6     41    113
Active(file)        4     11      3     23      8      6      3     40     99
SUnreclaim         19     10      6      6      9     33      7      7     97
Inactive            7     15      4     14     12     12      6      6     76
Inactive(file)      7     15      4     14     12     12      6      6     76
SReclaimable        7      6      2      4      3      3      3      2     29
Active(anon)        2      1      1      2      2      2      3      2     14
AnonPages           2      1      1      2      2      2      3      2     14
Mapped              0      0      0      1      4      3      1      1     11
KernelStack         9      0      0      0      0      0      0      0     10
PageTables          0      0      0      0      1      1      0      1      3
Shmem               0      0      0      0      0      0      0      0      0
Inactive(anon)      0      0      0      0      0      0      0      0      0
```

Subsystem Analysis: Disk

```
# iostat -N 1
Linux 3.10.0-229.el7.x86_64 (jerms-lab7.perf.lab.eng.rdu.redhat.com)
03/12/2015   _x86_64_ (24 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.03    0.00    0.01    0.00    0.00   99.95

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
vda               5.42        17.19        59.87   19633224   68398872
vdb               0.46         0.98         5.48    1119751    6263272
vg0-swap          0.00         0.00         0.00       1008          0
vg0-root          5.46        17.18        59.87   19625293   68396764


avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.04    0.00    0.71    3.31    0.00   95.94

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
vda            7043.00         0.00     12326.00          0      12326
vdb               0.00         0.00         0.00          0          0
vg0-swap          0.00         0.00         0.00          0          0
vg0-root       7042.00         0.00     12322.00          0      12322
```

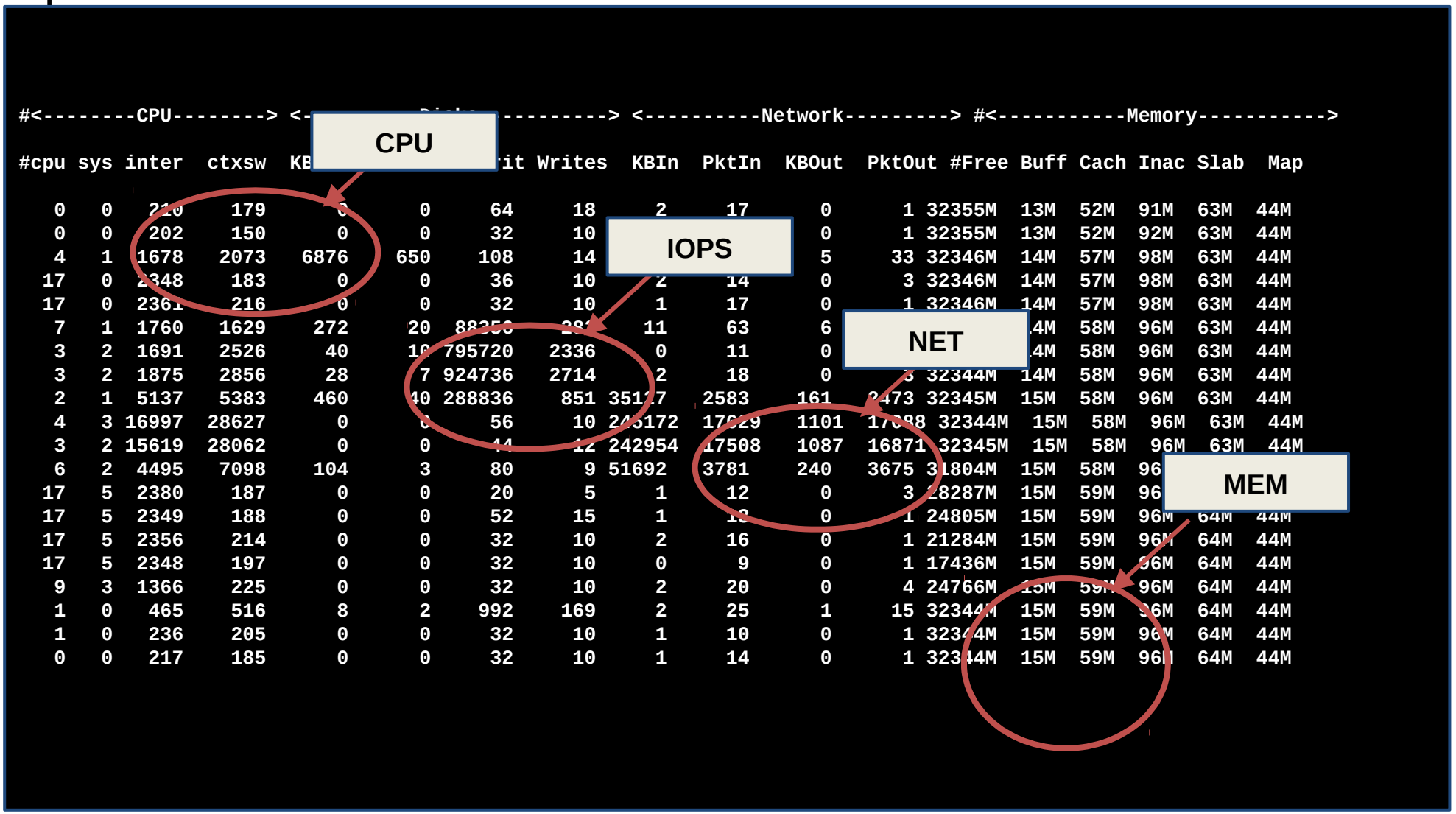
Subsystem Analysis: Network

ifpps -d <device>

```
3.10.0-231.el7.bz1200859.x86_64, p1p1 (sfc 10000Mbit/s link:yes), t=1000ms, cpus=5+1/16
(consider to increase your sampling interval, e.g. -t 10000)
rx:          339.255 MiB/t     443006 pkts/t        668 drops/t          0 errors/t
tx:           10.589 MiB/t      91660 pkts/t          0 drops/t          0 errors/t

rx:       815972.271 MiB   1261075090 pkts      1033625 drops          0 errors
tx:       859607.086 MiB   1308314479 pkts            0 drops          0 errors

sys:           94436 cs/t         1569 procs            2 running          0 iowait

mem:         128728M total       4109M used         2062M active        521M inactive
swap:          1023M total          0M used            0M cached

cpu13 +:         1.0% usr/t       15.5% sys/t        83.5% idl/t         0.0% iow/t
cpu15 |:         1.0% usr/t       13.5% sys/t        85.4% idl/t         0.0% iow/t
cpu 1 |:         1.0% usr/t       12.4% sys/t        86.6% idl/t         0.0% iow/t
cpu 3 |:         1.0% usr/t       12.1% sys/t        86.9% idl/t         0.0% iow/t
cpu 5 |:         2.0% usr/t       10.2% sys/t        87.8% idl/t         0.0% iow/t
cpu14 -:         0.0% usr/t        0.0% sys/t       100.0% idl/t         0.0% iow/t
avg:             0.0%              4.4%              95.6%               0.0%

cpu13 +:       19658 irqs/t              19239 sirq rx/t                74 sirq tx/t
cpu15 |:       14126 irqs/t              13910 sirq rx/t                57 sirq tx/t
cpu 9 |:       12391 irqs/t              12364 sirq rx/t                27 sirq tx/t
cpu 1 |:       11082 irqs/t              10996 sirq rx/t                33 sirq tx/t
cpu 5 |:        9962 irqs/t               9829 sirq rx/t                68 sirq tx/t
```

Subsystem Analysis: ALL

pmcollectl -s cdnm



But...what if we have a problem ?

- Automatic not enough...
- Need to eek out the last X percent
- Need Determinism

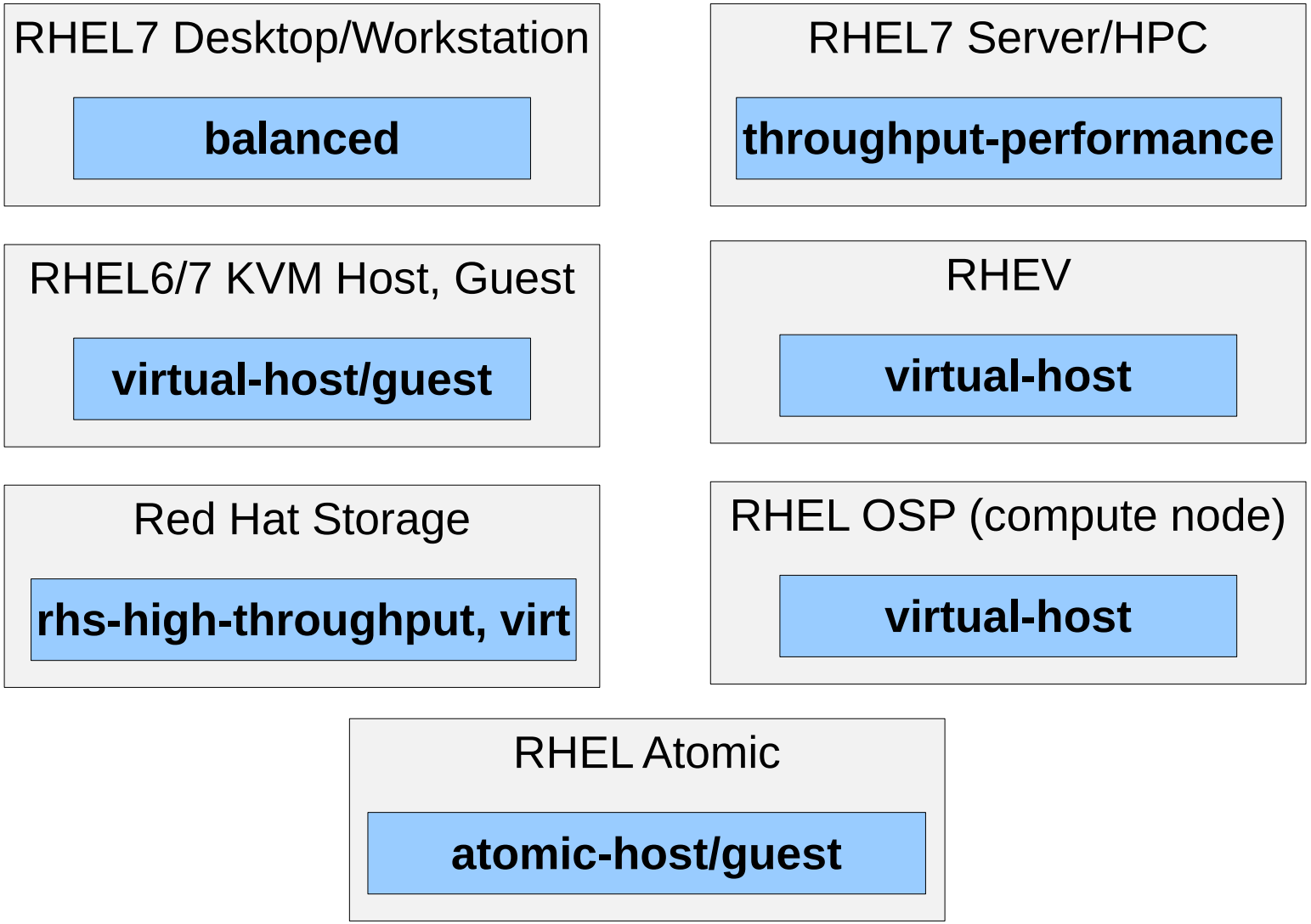
Tuned Updates for RHEL7

- Installed by default!
- Profiles updated for RHEL7 features and characteristics
- Profiles automatically set based on installation
  - Desktop/Workstation: balanced profile
  - Server/HPC: throughput-performance profile
- Optional hook/callout capability
- Concept of Inheritance (just like httpd.conf)

RHEL “tuned” package

```
Available profiles:
- balanced
- desktop
- latency-performance
- myprofile          << Easy to add your own
- network-latency
- network-throughput
- throughput-performance
- virtual-guest
- virtual-host
Current active profile: myprofile
```

Tuned Profiles throughout Red Hat's Product Line



Perf

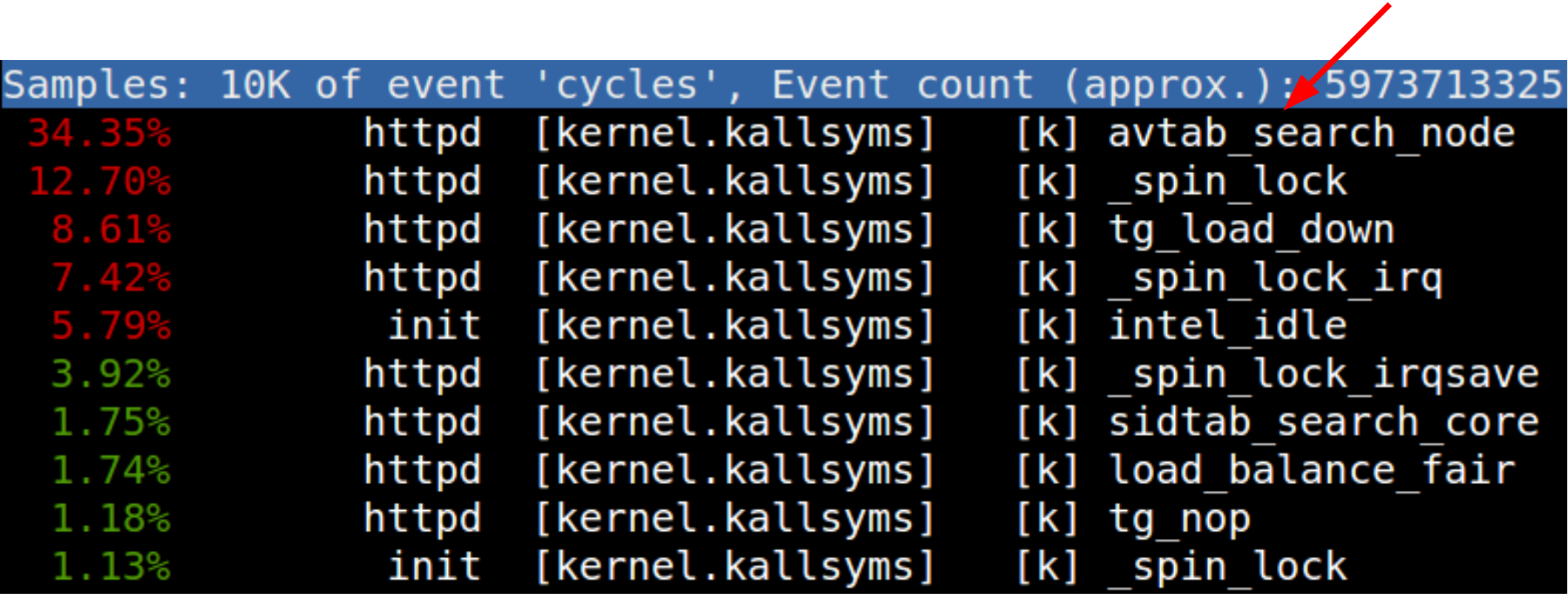
perf list

List counters/tracepoints available on your system

```
# perf list

List of pre-defined events (to be used in -e):
  cpu-cycles OR cycles                               [Hardware event
  instructions                                       [Hardware event
  cache-references                                   [Hardware event
  cache-misses                                       [Hardware event
  branch-instructions OR branches                    [Hardware event
  branch-misses                                      [Hardware event
  cpu-clock                                          [Software event
  task-clock                                         [Software event
  page-faults OR faults                              [Software event
  context-switches OR cs                             [Software event
  cpu-migrations OR migrations                       [Software event
  minor-faults                                       [Software event
  major-faults                                       [Software event
```

perf top

System-wide 'top' view of busy functions



perf record

- Record system-wide (-a)
  - perf record -a sleep 10
  - perf record -a // Hit ctrl-c when done.
- Or record a single command
  - perf record myapp.exe
- Or record an existing process (-p)
  - perf record -p <pid>
- Or add call-chain recording (-g)
  - perf record -g ls -rl /root
- *Or only record specific events (-e)*
  - *perf record -e branch-misses -p <pid>*

perf record

- Record system-wide (-a)

 # perf record -a dd if=/dev/zero of=test bs=1M count=1000 conv=fdatasync oflag=direct

 # perf report --stdio |head -20

```
# To display the perf.data header info, please use --header/--header-only options.
#
# Samples: 2K of event 'cycles'
# Event count (approx.): 438884664
#
# Overhead       Command      Shared Object                                     Symbol
# ........  ............  ..................  ................................................
#
    38.18%            dd  [kernel.kallsyms]  [k] __clear_user
     5.08%            dd  [kernel.kallsyms]  [k] do_blockdev_direct_IO
     3.66%            dd  [kernel.kallsyms]  [k] gup_pte_range
     3.52%            dd  [kernel.kallsyms]  [k] put_page
     1.85%  swapper  [kernel.kallsyms]  [k] native_safe_halt
     1.79%            dd  [kernel.kallsyms]  [k] __domain_mapping
     1.67%            dd  [kernel.kallsyms]  [k] __bio_add_page
     1.64%            dd  [kernel.kallsyms]  [k] __blk_segment_map_sg
     1.29%            dd  [kernel.kallsyms]  [k] rb_prev
     1.11%  swapper  [kernel.kallsyms]  [k] irq_entries_start
     0.99%            dd  [kernel.kallsyms]  [k] sg_next
     0.97%            dd  [kernel.kallsyms]  [k] dm_table_find_target
```

perf report

```
# Overhead  Command      Shared Object
# ........  .......  .....................  ..............................
#
    43.53%       dd  [kernel.kallsyms]  [k] __clear_user
                 |
                 --- __clear_user
                     |
                     |--99.75%-- read_zero.part.5
                     |           read_zero
                     |           vfs_read
                     |           sys_read
                     |           system_call_fastpath
                     |           __GI___libc_read
                     --0.25%-- [...]

     5.37%       dd  [kernel.kallsyms]  [k] do_blockdev_direct_IO
                 |
                 --- do_blockdev_direct_IO
                     __blockdev_direct_IO
                     xfs_vm_direct_IO
                     generic_file_direct_write
                     xfs_file_dio_aio_write
                     xfs_file_aio_write
                     do_sync_write
```

/dev/zero

oflag=direct

Compare 2 perf recordings

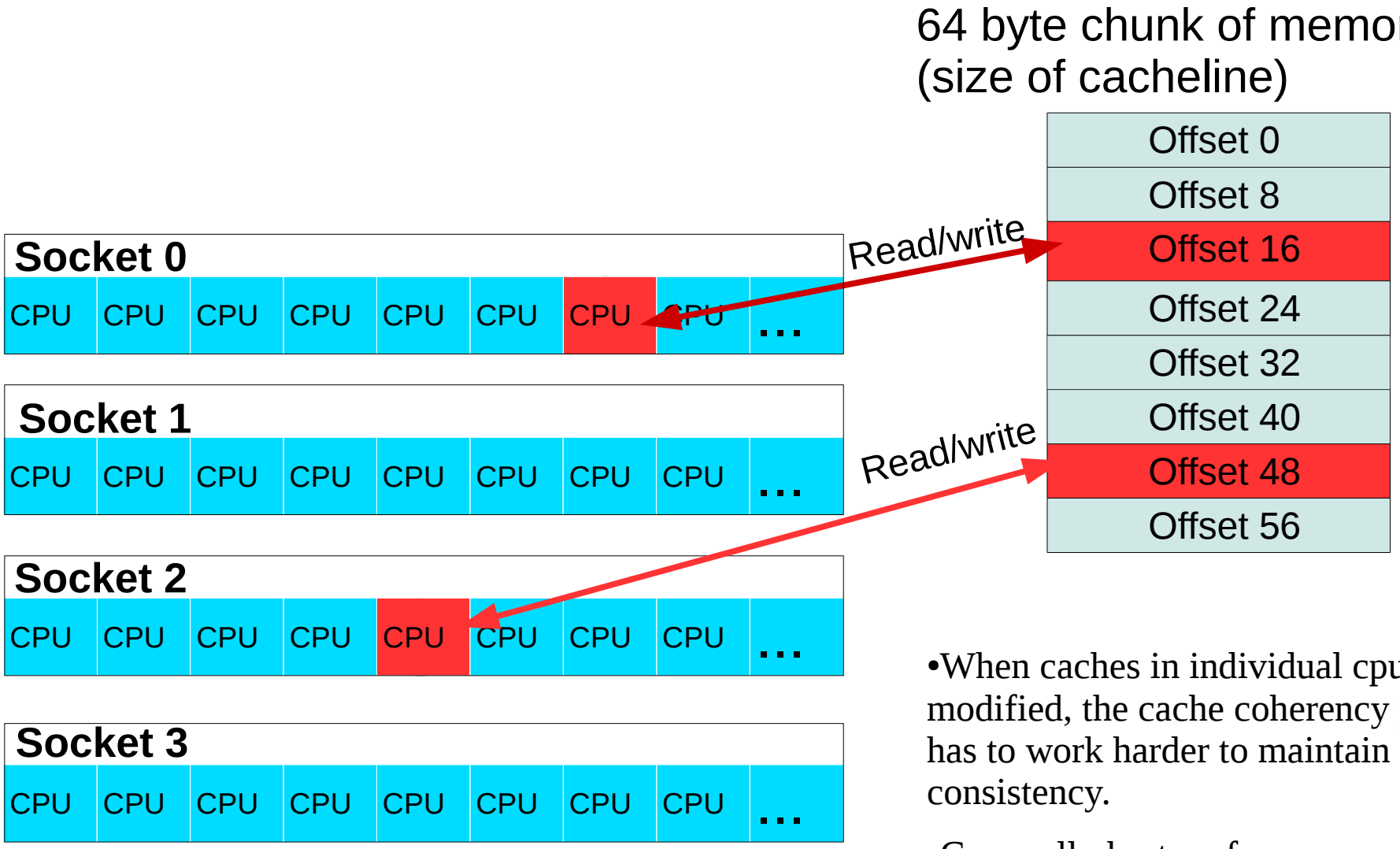
```
# perf diff

# Event 'cycles'
#
# Baseline    Delta          Shared Object                       Symbol
# ........  .......  ..................................  ........................
#
    12.88%  -12.27%  [kernel.kallsyms]                   [k] __lookup_mnt
    11.97%  -11.17%  systemd                             [.] 0x0000000000064968
     4.32%   +6.43%  libdbus-1.so.3.7.4                  [.] 0x0000000000029258
     4.06%   +4.72%  dbus-daemon                         [.] 0x000000000014a6e
     3.79%   -3.79%  libglib-2.0.so.0.3600.3            [.] 0x000000000088d6a
     3.72%   +0.25%  [kernel.kallsyms]                   [k] seq_list_start
```

perf list

grep for something interesting, maybe to see what numabalance is doing ?

```
# perf list | grep sched: | grep numa
  sched:sched_move_numa                              [Tracepoint event]
  sched:sched_stick_numa                             [Tracepoint event]
  sched:sched_swap_numa                              [Tracepoint event]
```

New C-2-C RHEL7.3
Cacheline Contention – high level



- When caches in individual cpus are modified, the cache coherency protocol has to work harder to maintain consistency.
- Can really hurt performance.

Output from “c2c data sharing” tool

Cache #	Refs	Stores	Data Address	Pid	Tid	Inst Address	Symbol	Object	CPU Participants
0	118789	273709	0x602380	37878					
	17734	136078	0x602380	37878	37878	0x401520	read_wrt_thread	a.out	0{0};
	13452	137631	0x602388	37878	37883	0x4015a0	read_wrt_thread	a.out	0{1};
	15134	0	0x6023a8	37878	37882	0x4011d7	reader_thread	a.out	1{5};
	14684	0	0x6023b0	37878	37880	0x4011d7	reader_thread	a.out	1{6};
	13864	0	0x6023b8	37878	37881	0x4011d7	reader_thread	a.out	1{7};
1	31	69	0xffff88023960df40	37878					
	13	69	0xffff88023960df70	37878	***	0xffffffff8109f8e5	update_cfs_rq_blocked_load	vmlinux	**0{0,1,2}; 1{14,16};**
	17	0	0xffff88023960df60	37878	***	0xffffffff8109fc2e	__update_entity_load_avg_contrib	vmlinux	**0{0,1,2}; 1{14,16};**
	1	0	0xffff88023960df78	37878	37882	0xffffffff8109fc4e	__update_entity_load_avg_contrib	vmlinux	0{2};

This shows us:

- The hottest contended cachelines
- The process names, data addr, ip, pids, tids
- The node and CPU numbers they ran on,
- And how the cacheline is being accessed (read or write)

HSW EX Brickland – recent “perf c2c” activity

Offset 0x00:
Heavily read. Tiny number of writes.

Cnt	inst_addr	Symbol
814	0x7f70fd202c99	libhdbbasis.so:ltt::allocated_refcounted::destroyImp
347	0x7f70fd202cad	libhdbbasis.so:ltt::allocated_refcounted::destroyImp

Offset 0x00
Heavily Read

Offset 0x10:
Heavily read. Tiny number of writes.

Cnt	inst_addr	Symbol
1427	0x7f70fd12afb4	libhdbbasis.so:MemoryManager::PoolAllocator::deallocateImpl
148	0x7f70fd12f891	libhdbbasis.so:MemoryManager::PoolAllocator::allocateNoThrowImpl
662	0x7f7109cb8ab0	libhdbcs.so:ltt::allocated_refcounted::release
2810	0x7f7109cb8aba	libhdbcs.so:ltt::allocated_refcounted::release
4934	0x7f7109cb8ae0	libhdbcs.so:ltt::allocated_refcounted::addReference
15	0x7f7109cb8ae6	libhdbcs.so:ltt::allocated_refcounted::addReference

Offset 0x10
Heavily Read

Offset 0x20
Heavily Written

Offset 0x30:
Heavily written. Tiny number of reads.

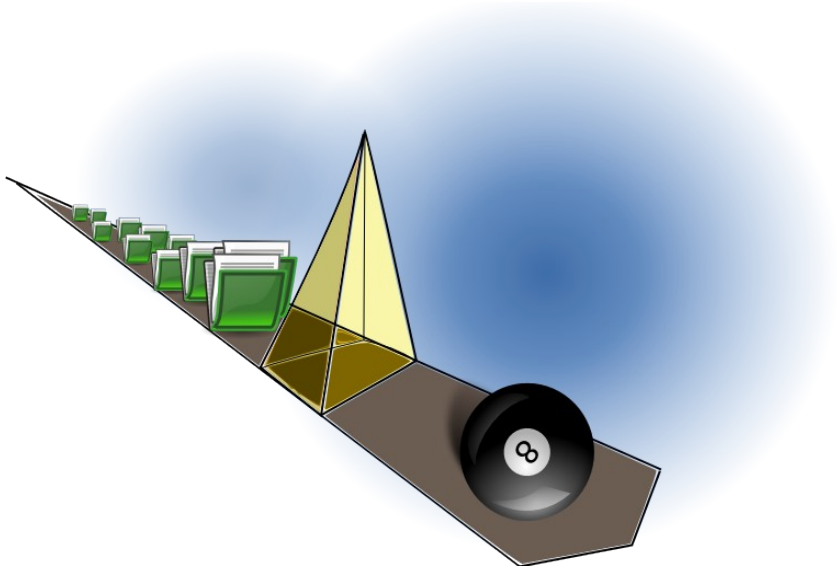
Cnt	inst_addr	Symbol
8074	0x7f70fd12b230	libhdbbasis.so:MemoryManager::PoolAllocator::addReference
19	0x7f70fd12b236	libhdbbasis.so:MemoryManager::PoolAllocator::addReference
474	0x7f70fd130378	libhdbbasis.so:MemoryManager::PoolAllocator::release
3919	0x7f70fd130383	libhdbbasis.so:MemoryManager::PoolAllocator::release

Reports include:

- All simultaneous readers and writers to a cacheline.
- The sockets and cpus the accesses are coming from.
- The data addr, pid, tid, load latencies, where data was sourced from.

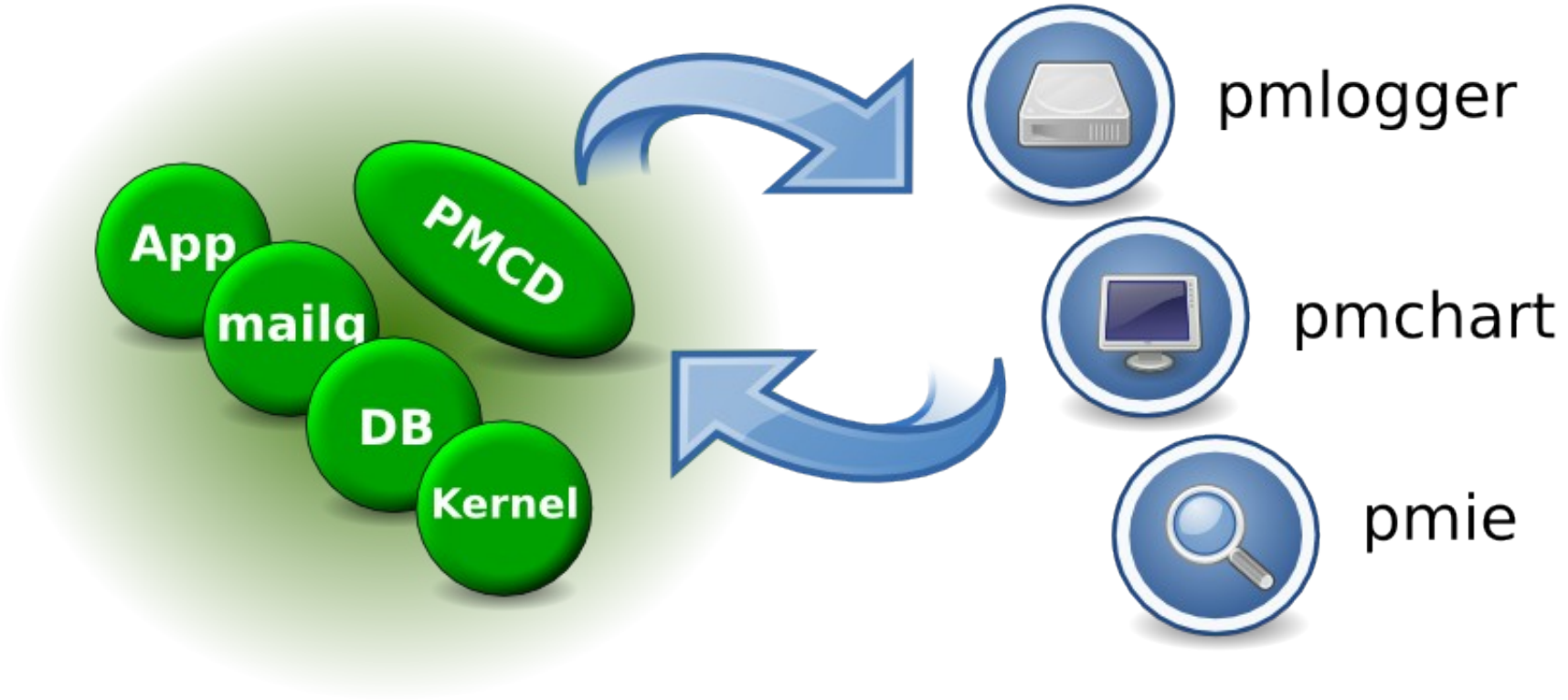
Performance Co-Pilot - Overview

- What is PCP?
  - Open source **toolkit**
  - System-level analysis
  - Live and historical
  - Extensible (monitors, collectors)
  - Distributed



Supported in RHEL7.0 and RHEL6.6!

Performance Co-Pilot - Architecture



Performance Co-Pilot - Installation

```
# yum install pcp*
# systemctl enable {pmcd,pmlogger,pmwebd}
# systemctl start {pmcd,pmlogger,pmwebd}
```

Verify it's working:

```
# pmclient -t1
```

Performance Co-Pilot - Firewall Config

RHEL6:

-A INPUT -p tcp -m state --state NEW -m tcp --dport 44321 -j ACCEPT

-A INPUT -p udp -m state --state NEW -m udp --dport 44321 -j ACCEPT

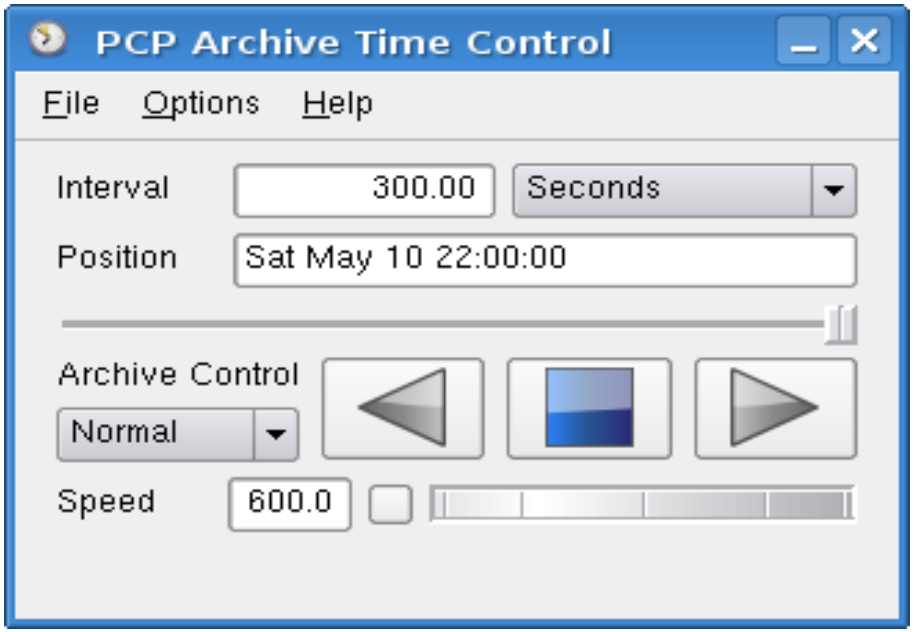
RHEL7:

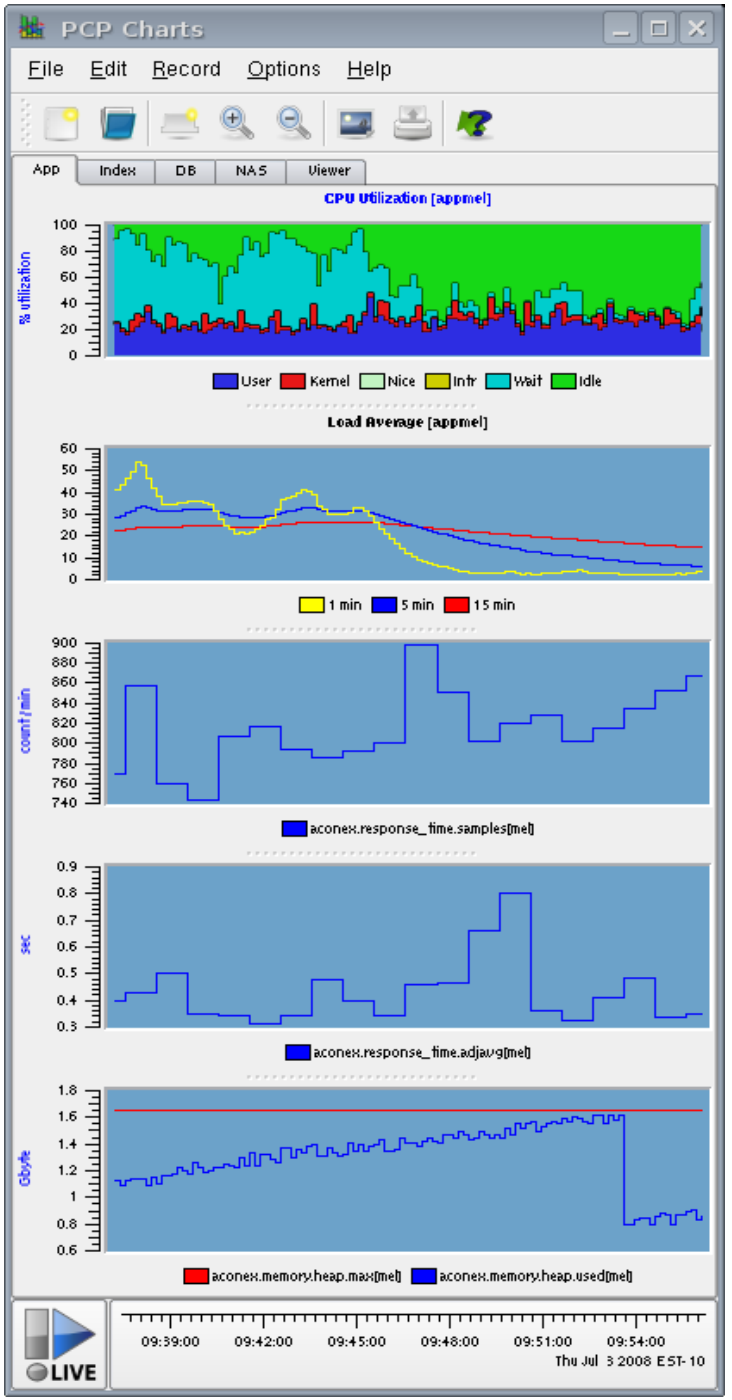
firewall-cmd --permanent --zone=public --add-service=pmcd

firewall-cmd --reload

Client toolkit - *pmchart*

- Arbitrary charts
- Load / Save views
- VCR-style playback

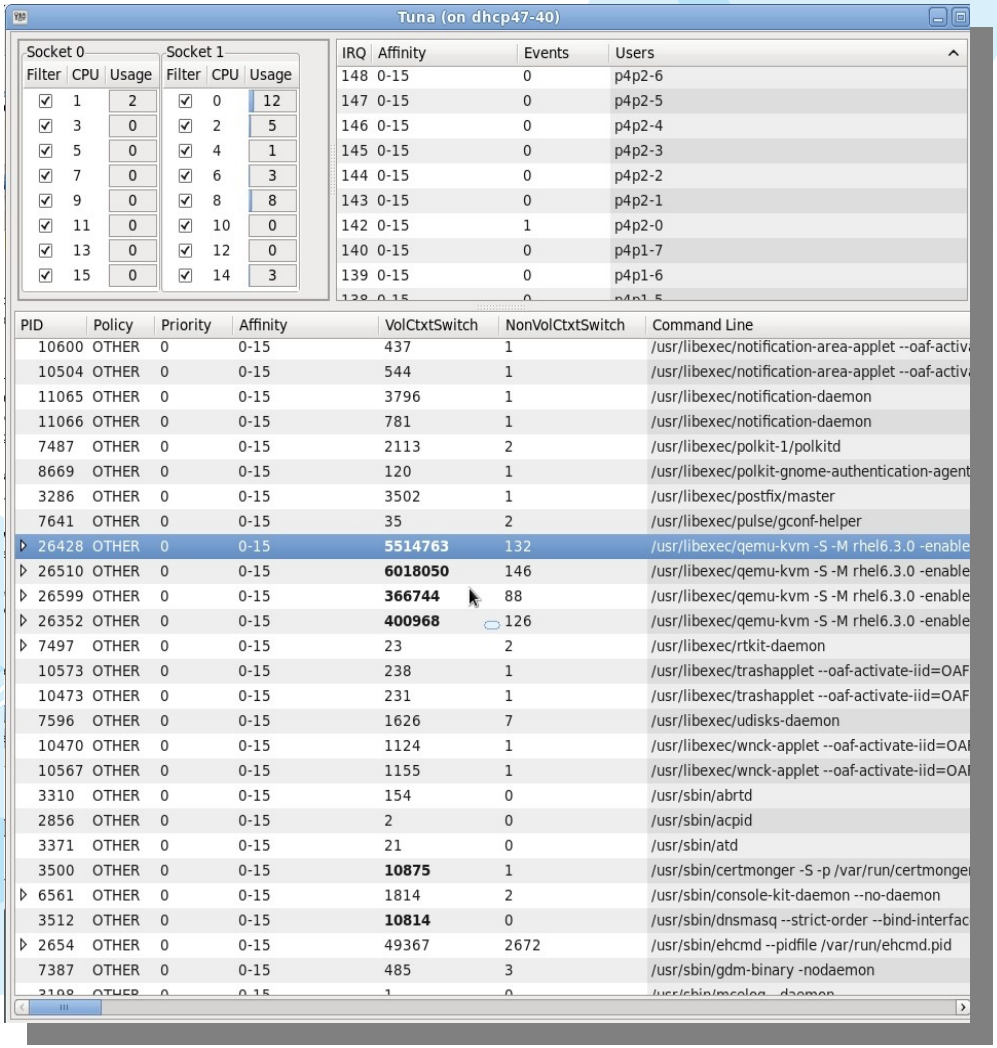




Tuna

System Tuning Tool - tuna

- Tool for fine grained control
- Display applications / processes
- Displays CPU enumeration
- Socket (useful for NUMA tuning)
- Dynamic control of tuning
  - Process affinity
  - Parent & threads
  - Scheduling policy
  - Device IRQ priorities, etc

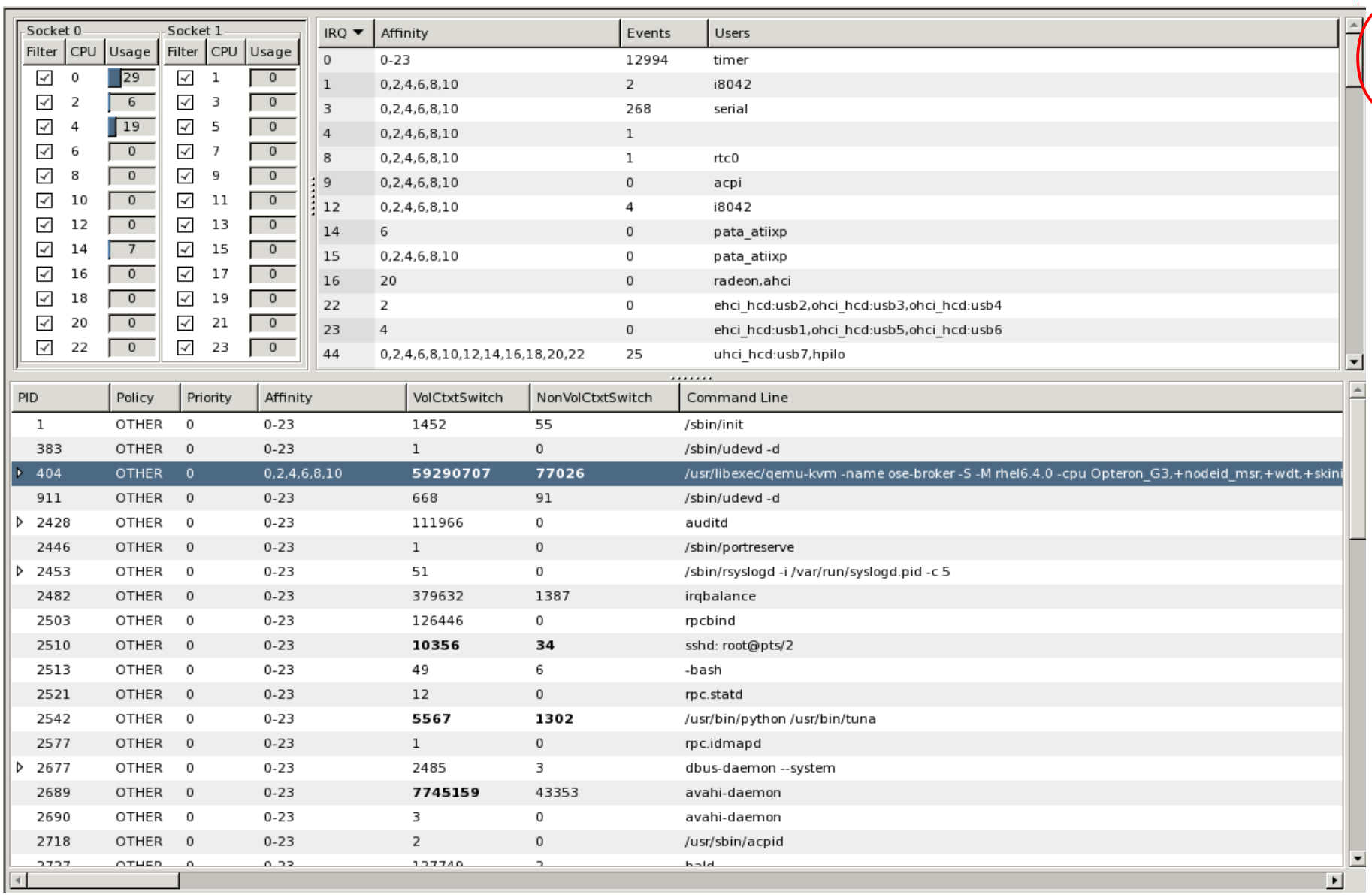


Tuna (RHEL6/7)

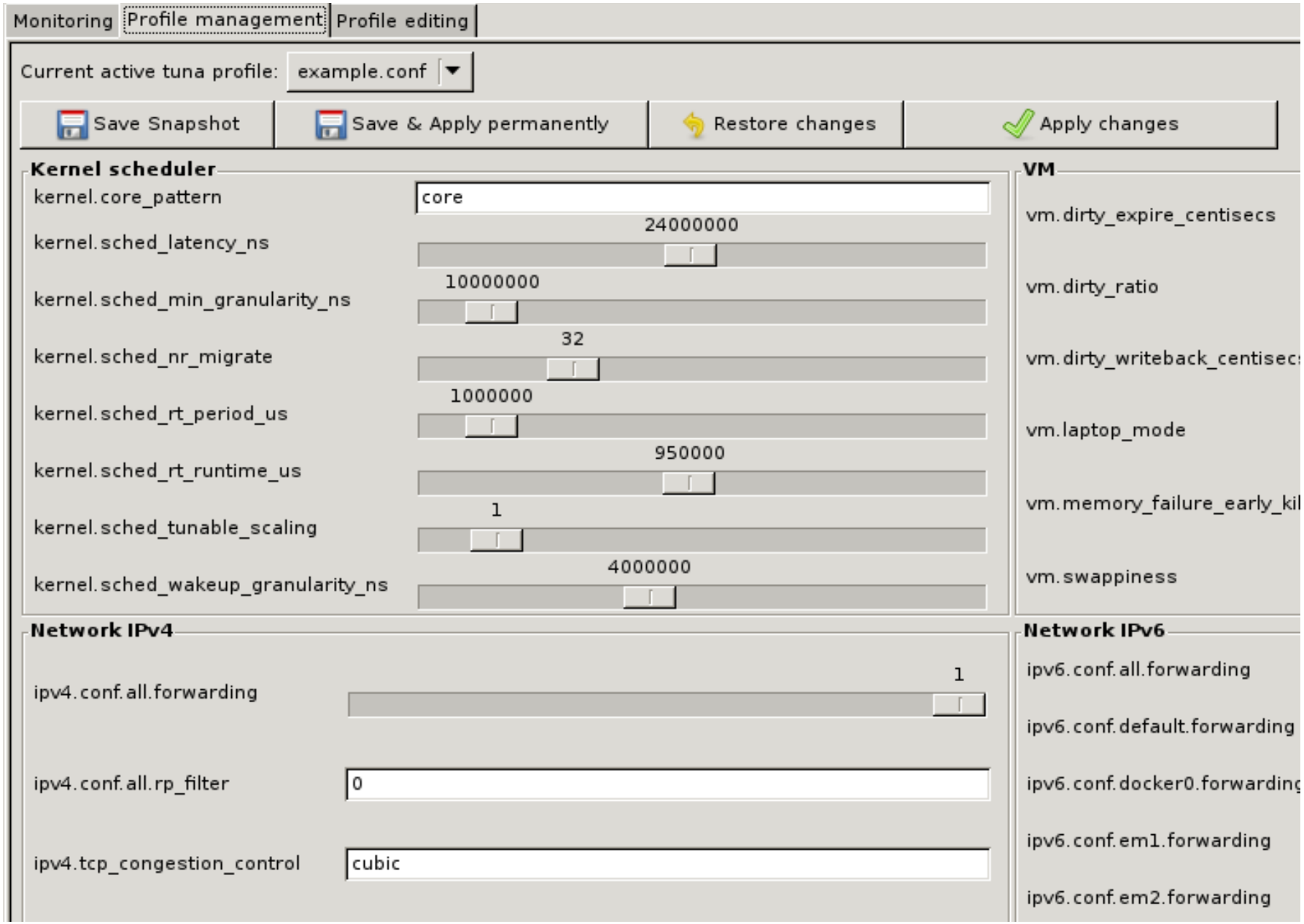
1

2

3



Tuna GUI Capabilities Updated for RHEL7



Helpful Utilities

Supportability

- redhat-support-tool
- sos
- kdump
- perf
- psmisc
- strace
- sysstat
- systemtap
- trace-cmd
- util-linux-ng

NUMA

- hwloc
- Intel PCM
- numactl
- numad
- numatop (01.org)

Networking

- dropwatch
- ethtool
- netsniff-ng (EPEL6)
- tcpdump
- wireshark/tshark

Power/Tuning

- cpupowerutils (R6)
- kernel-tools (R7)
- powertop
- tuna
- tuned

Storage

- blktrace
- iotop
- iostat

Performance Optimizations RHEL7

CPU

- Support for all new CPUs
- AVX2 instruction support
- RHEL-RT released March 5

Memory

- Automatic NUMA Balancing
- Tunable workqueues (writeback)

Networking

- Full support for PTP1588v2
- Route cache → F.I.B.
- irqbalance handles NUMA
- busy_poll, tcp_fastopen
- nohz_full (tickless while active)
- Byte Queue Limits
- TCP Small Queues

Power Management

- intel_pstate
- tuned does most heavy lifting

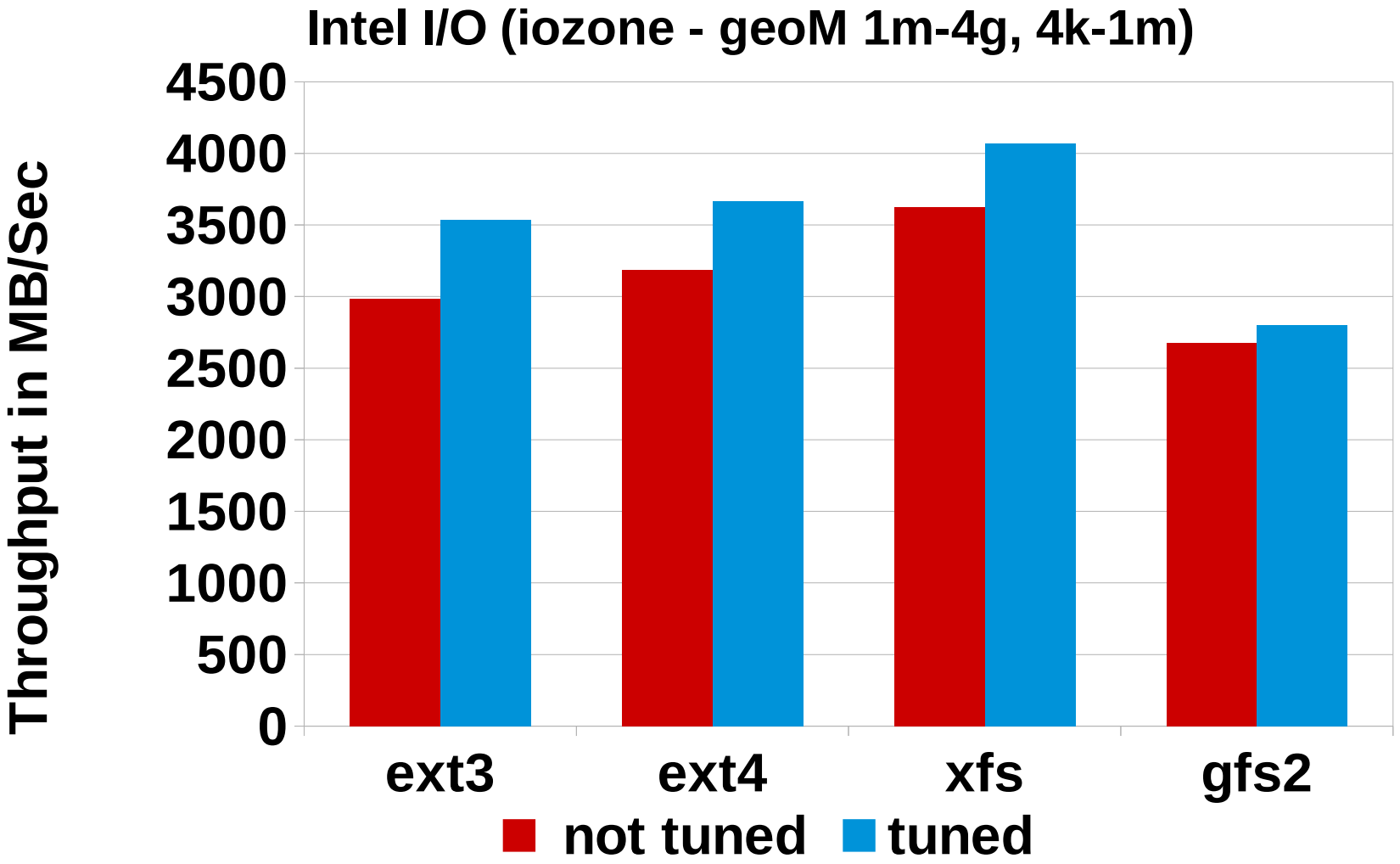
Disk IO

Tuned-adm profile throughput-performance (R7 default)

- governor=performance
- energy_perf_bias=performance
- min_perf_pct=100
- readahead=4096
- kernel.sched_min_granularity_ns = 10000000
- kernel.sched_wakeup_granularity_ns = 15000000
- vm.dirty_background_ratio = 10
- vm.swappiness=10

Tuned: Storage Performance Boost

RHEL7 File System In Cache Perf



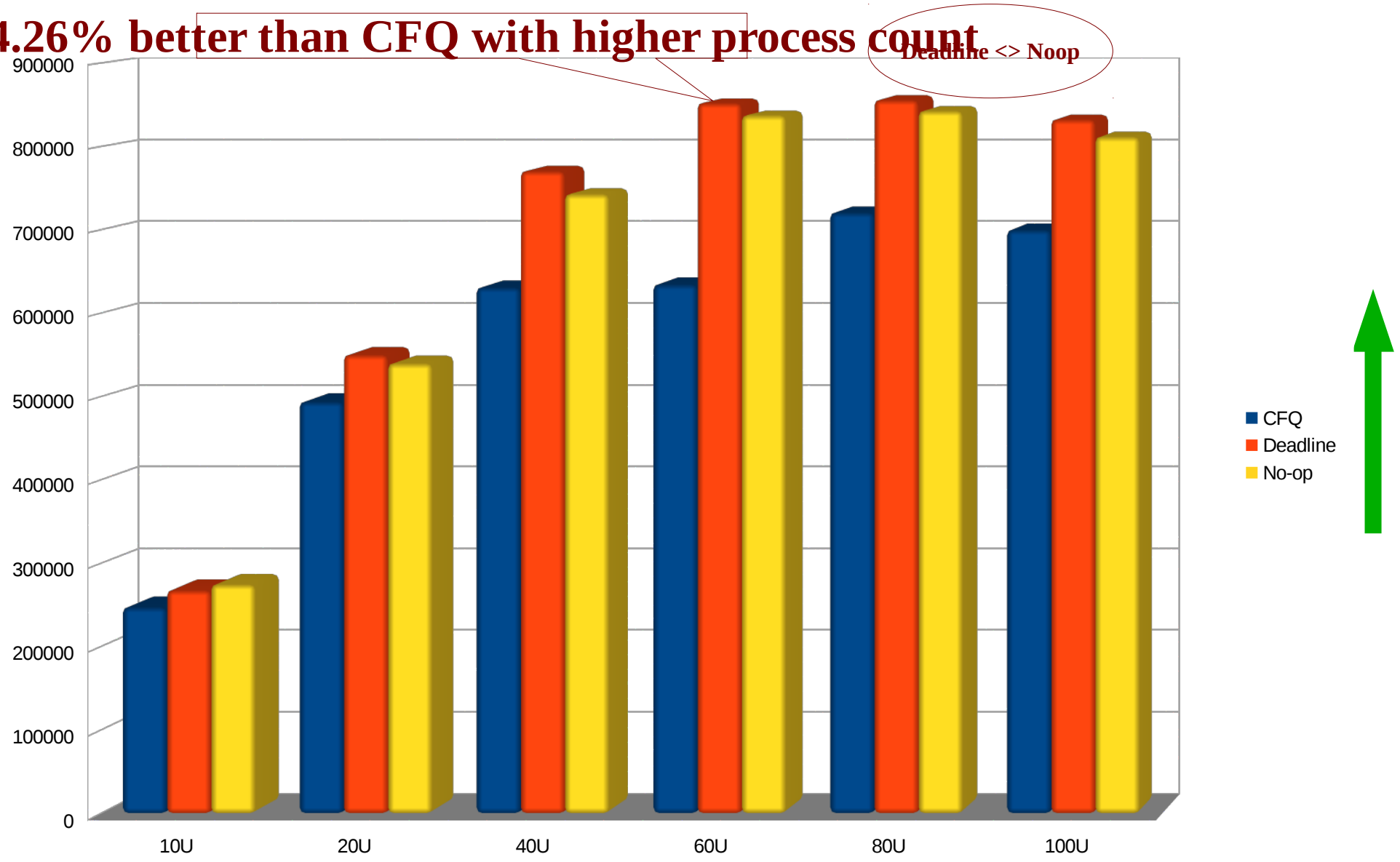
Larger is better

I/O Tuning – Configuring I/O Elevators

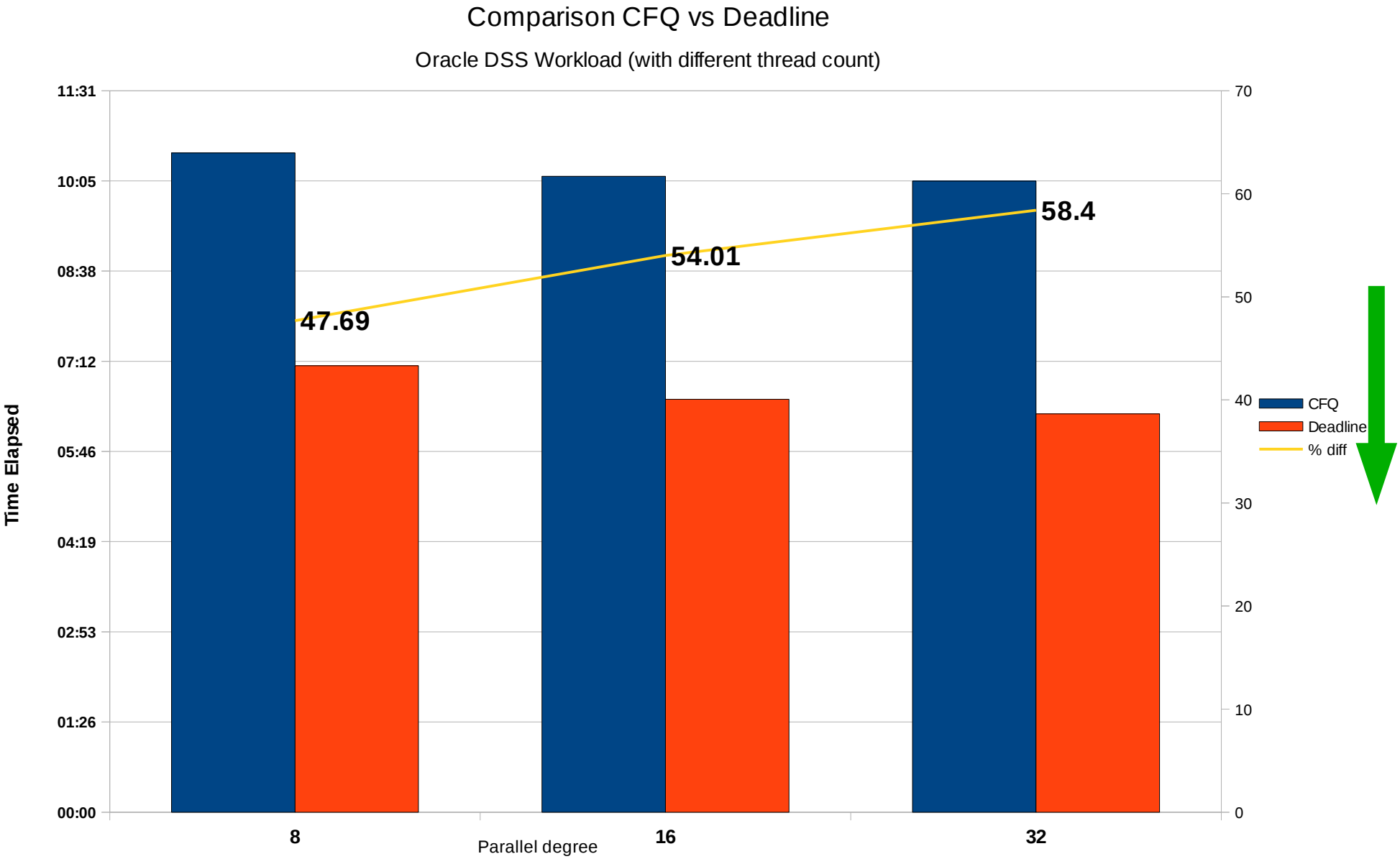
- **Boot-time**
  - Grub command line – elevator=deadline/cfq/noop
- **Dynamically, per device**
  - echo “deadline” > /sys/class/block/sda/queue/scheduler
- **tuned (RHEL6 utility)**
  - tuned-adm profile throughput-performance
  - tuned-adm profile enterprise-storage

Impact of I/O Elevators – OLTP Workload

34.26% better than CFQ with higher process count



Impact of I/O Elevators – DSS Workload

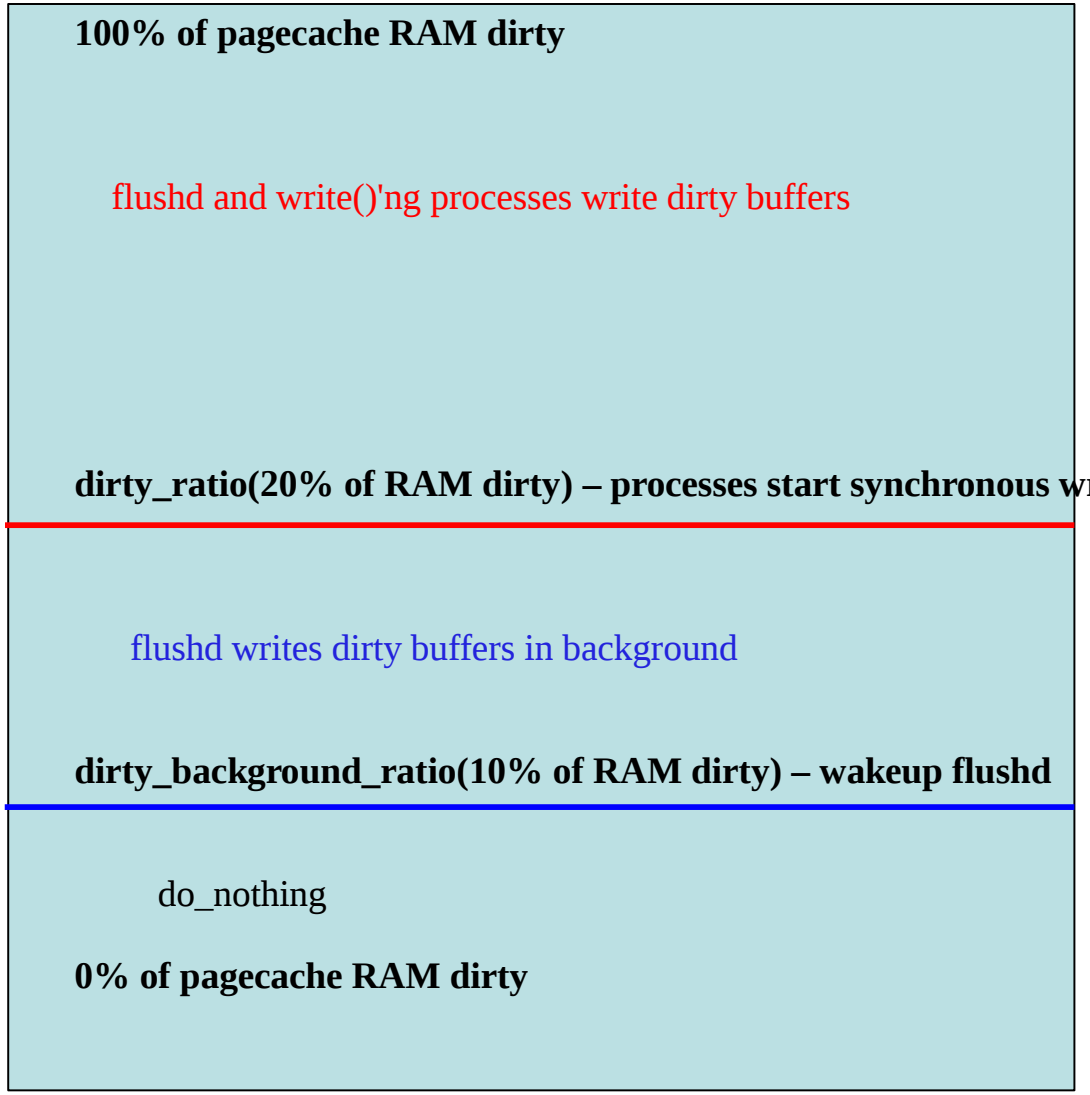


Tuning Memory – Flushing Caches

- Drop unused Cache
  - ✔ Frees unused memory
  - ✔ File cache
  - ✗ If the DB uses cache, may notice slowdown

- **Free pagecache**
  - echo 1 > /proc/sys/vm/drop_caches
- **Free slabcache**
  - echo 2 > /proc/sys/vm/drop_caches
- **Free pagecache and slabcache**
  - echo 3 > /proc/sys/vm/drop_caches

dirty_ratio and dirty_background_ratio



If there is a lot of pagecache pressure one would want to start background flushing sooner and delay the synchronous writes. This can be done by

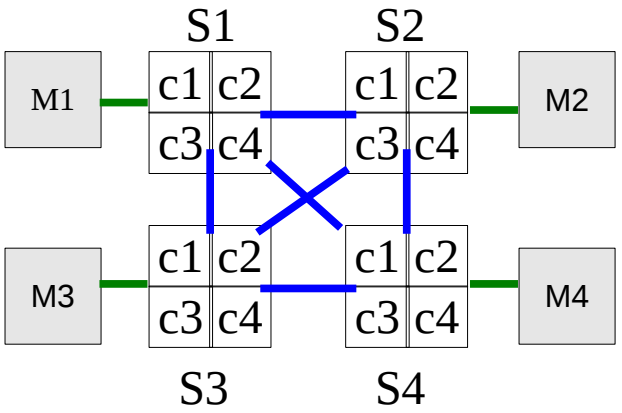
- **Lowering the dirty_background_ratio**
- **Increasing the dirty_ratio**

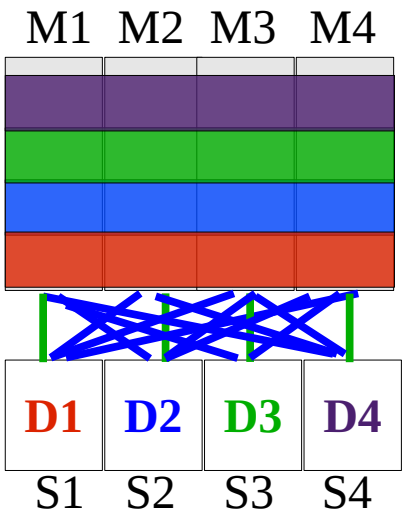
Tuning Memory – swappiness

- Not needed as much in RHEL7
- Controls how aggressively the system reclaims “mapped” memory:
- Default - 60%
- Decreasing: more aggressive reclaiming of unmapped pagecache memory, thereby delaying swapping
- Increasing: more aggressive swapping of mapped memory

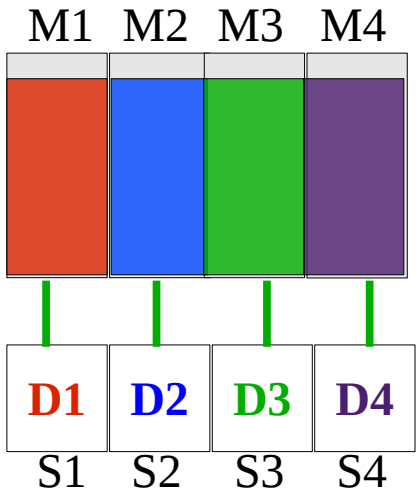
NonUniform Memory Access
NUMA

Understanding NUMA (Non Uniform Memory Access)





No NUMA optimization



NUMA optimization

- Multi Socket – Multi core architecture
  - NUMA required for scaling
  - RHEL 5 / 6 completely NUMA aware
  - Additional performance gains by enforcing NUMA placement

Understand the Configuration

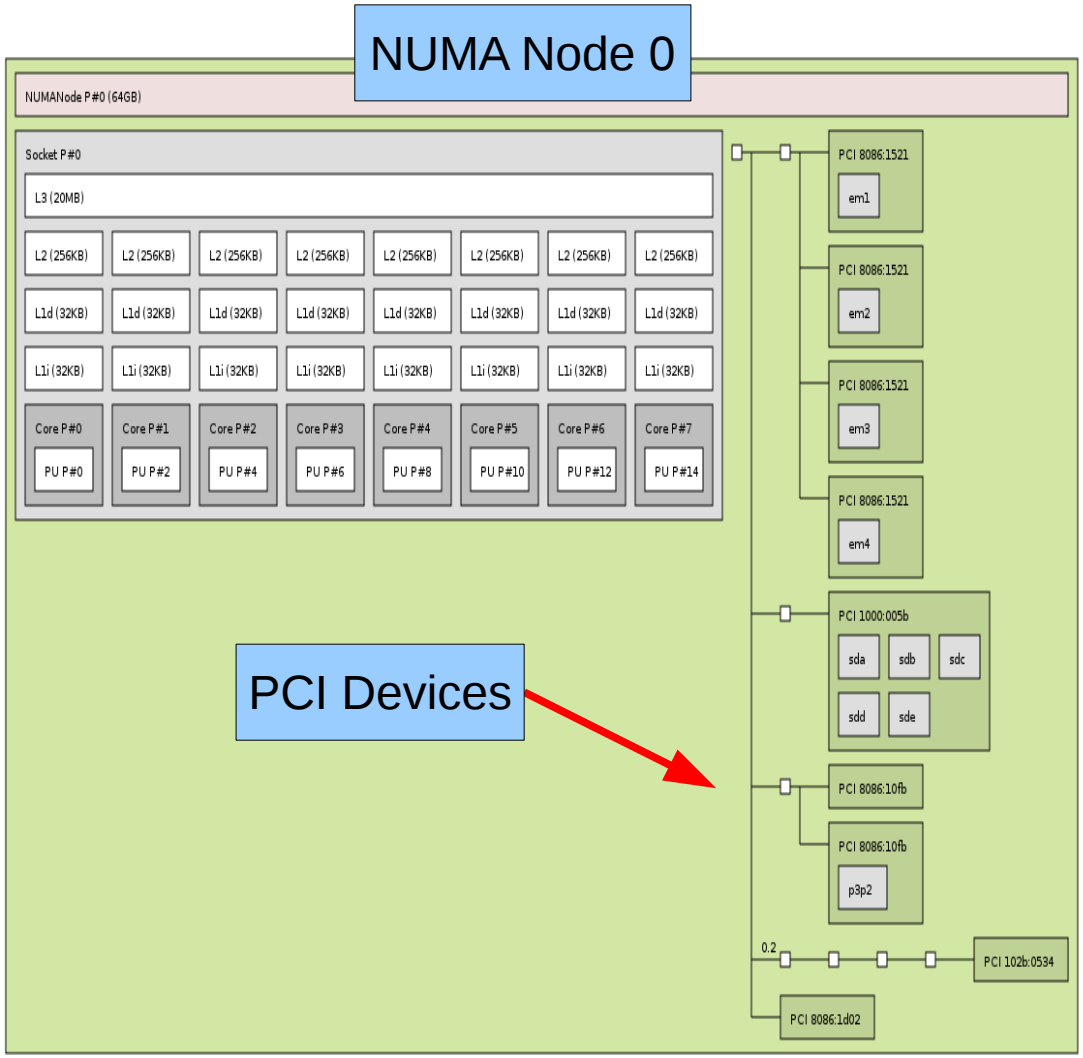
Sampling of tools we often use.
I'm sure you have your own favorites.

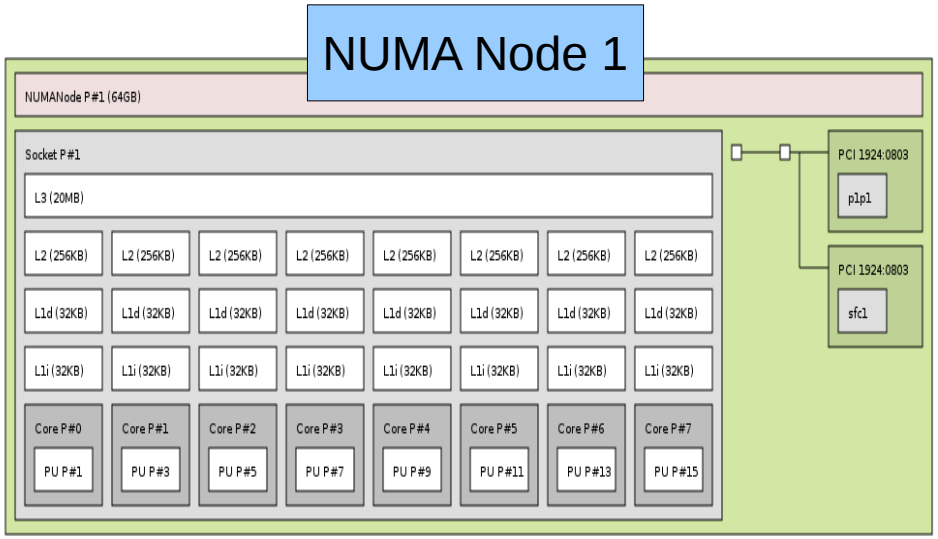
- lscpu: Display information about the CPU architecture
- lstopo: Show the topology of the system
- numactl --hardware: Show inventory of available nodes

- dmidecode: DMI table decoder
- lspci -t -vv: List all PCI devices
- lsblk: List block devices
- blkid: Locate & print block device attributes
- cat /proc/cmdline: See flags kernel booted with.
- ifconfig -a: Display available network interfaces

- … <and all your favorites>

Visualize NUMA Topology: lstopo





How can I visualize my system's NUMA topology in Red Hat Enterprise Linux?

https://access.redhat.com/site/solutions/62879

RHEL NUMA Scheduler

- RHEL6
  - numactl, numastat enhancements
  - numad – usermode tool, dynamically monitor, auto-tune
- RHEL7 – auto numa balancing
  - Moves tasks (threads or processes) closer to the memory they are accessing.
  - Moves application data to memory closer to the tasks that reference it.
  - A win for most apps.
  - Enable / Disable
    - sysctl kernel.numabalancing={0,1}

numastat: per-node meminfo (new)

```
# numastat -mczs
```

	Node 0	Node 1	Total
MemTotal	65491	65536	131027
MemFree	60366	59733	120099
MemUsed	5124	5803	10927
Active	2650	2827	5477
FilePages	2021	3216	5238
Active(file)	1686	2277	3963
Active(anon)	964	551	1515
AnonPages	964	550	1514
Inactive	341	946	1287
Inactive(file)	340	946	1286
Slab	380	438	818
SReclaimable	208	207	415
SUnreclaim	173	230	403
AnonHugePages	134	236	370

numastat – per-PID mode

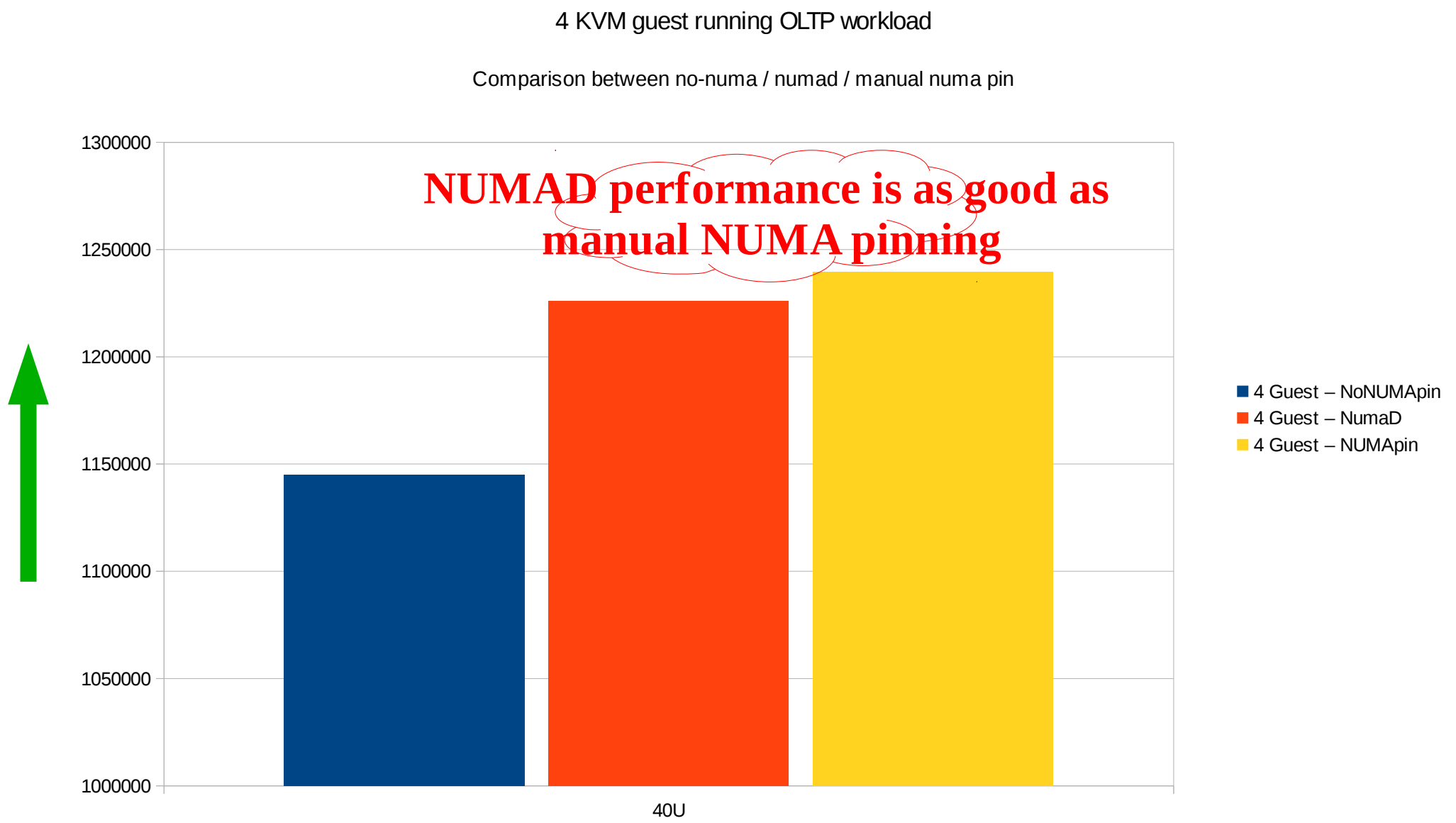
 # numastat -c java (*default scheduler – non-optimal*)

```
Per-node process memory usage (in MBs)
PID              Node 0 Node 1 Node 2 Node 3 Total
---------------  ------ ------ ------ ------ -----
57501 (java)        755   1121    480    698  3054
57502 (java)       1068    702    573    723  3067
57503 (java)        649   1129    687    606  3071
57504 (java)       1202    678   1043    150  3073
---------------  ------ ------ ------ ------ -----
Total              3674   3630   2783   2177 12265
```

 # numastat -c java (*numabalance close to opt*)

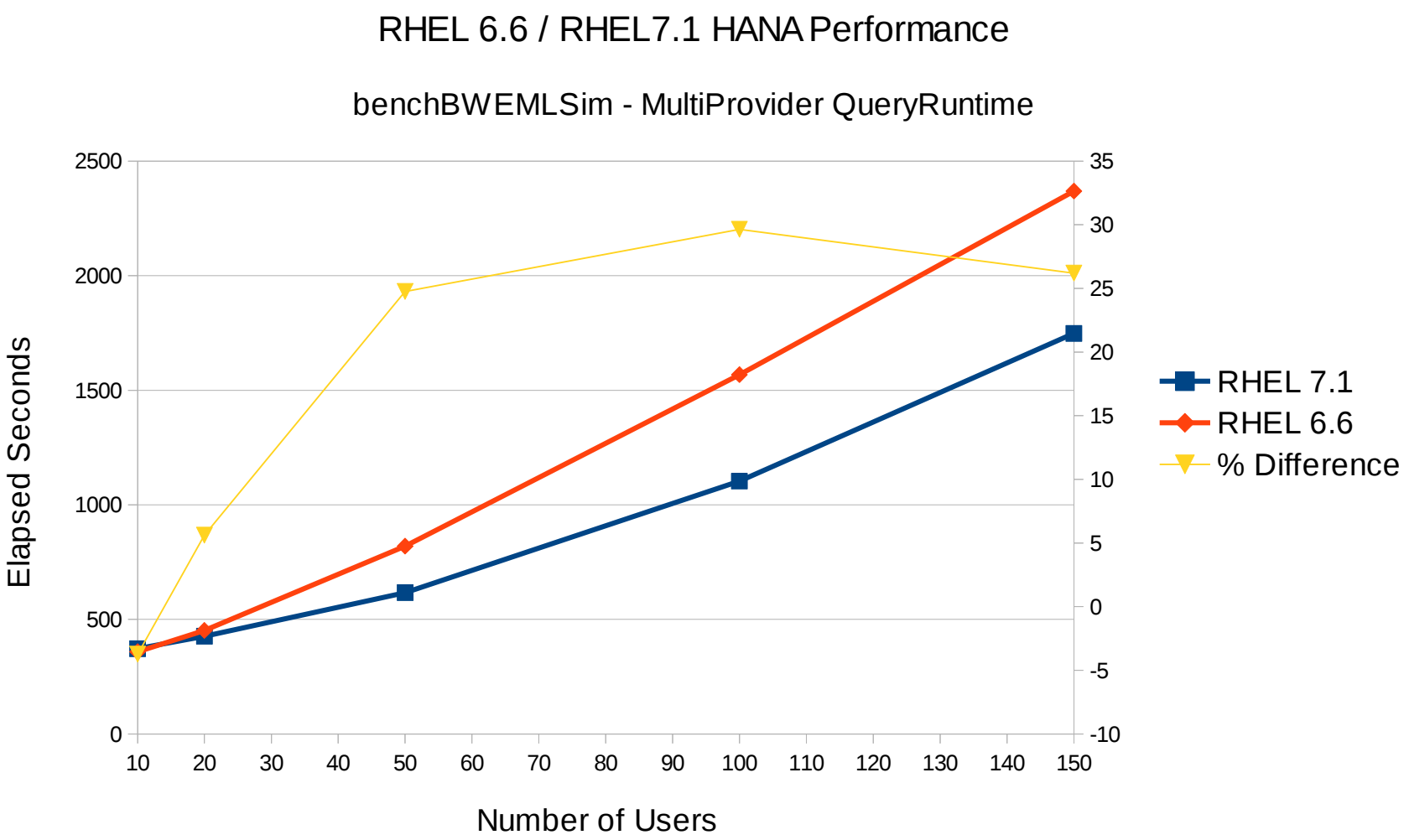
```
Per-node process memory usage (in MBs)
PID              Node 0 Node 1 Node 2 Node 3 Total
---------------  ------ ------ ------ ------ -----
56918 (java)         49   2791     56     37  2933
56919 (java)       2769     76     55     32  2932
56920 (java)         19     55     77   2780  2932
56921 (java)         97     65   2727     47  2936
---------------  ------ ------ ------ ------ -----
Total              2935   2987   2916   2896 11734
```

Memory Tuning – Effect of “numa”



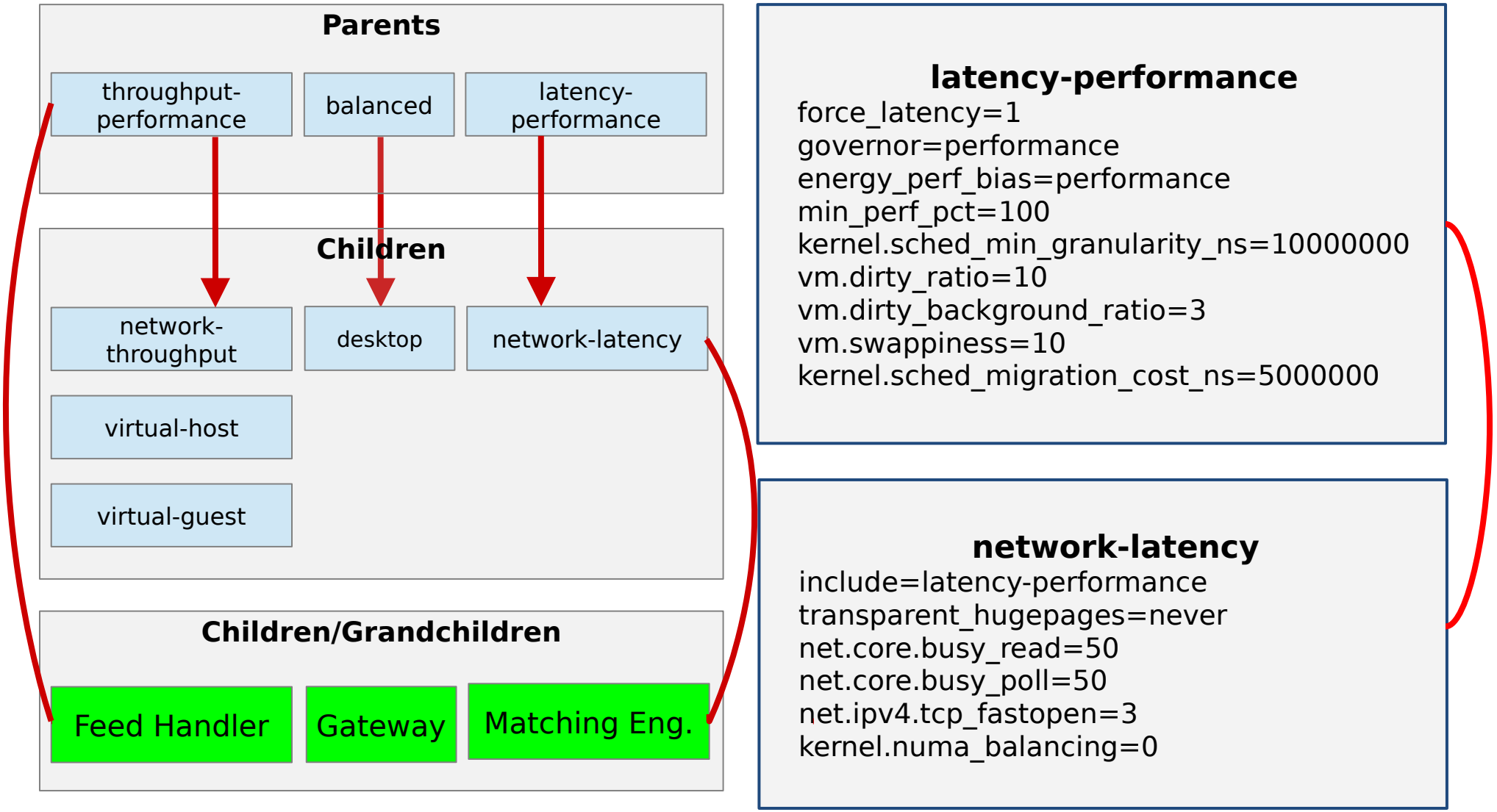
RHEL 6.6 vs RHEL 7.1 SAP HANA Performance

All due to Auto NUMA balancing (kernel.numa_balancing = 1)



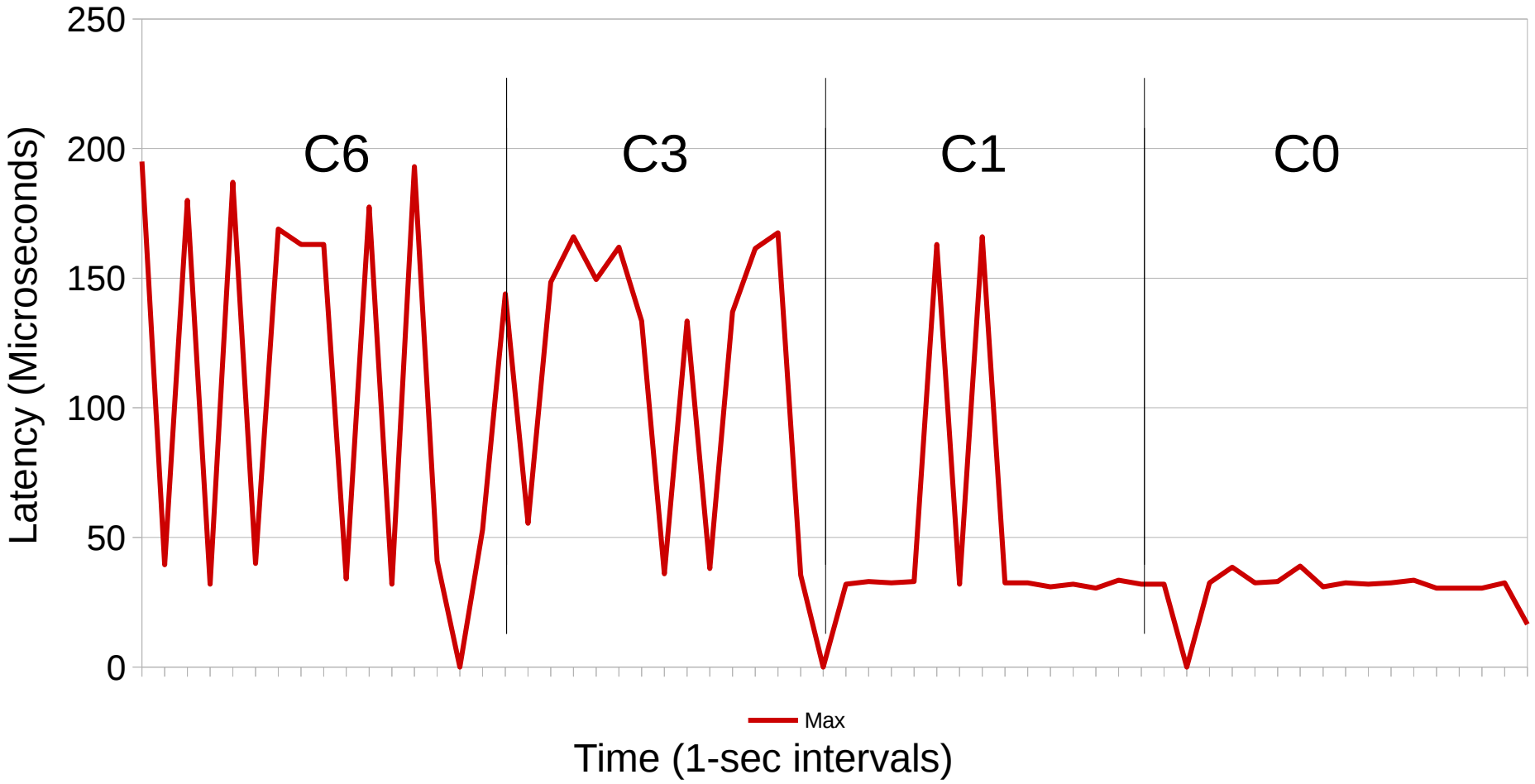
Power Management
Networking
Low Latency

Tuned *network-latency* Profile



Tuned: Network-Latency Performance Boost

C-state lock improves determinism, reduces jitter



turbostat shows P/C-states on Intel CPUs

turbostat begins shipping in RHEL6.4, cpupowerutils package

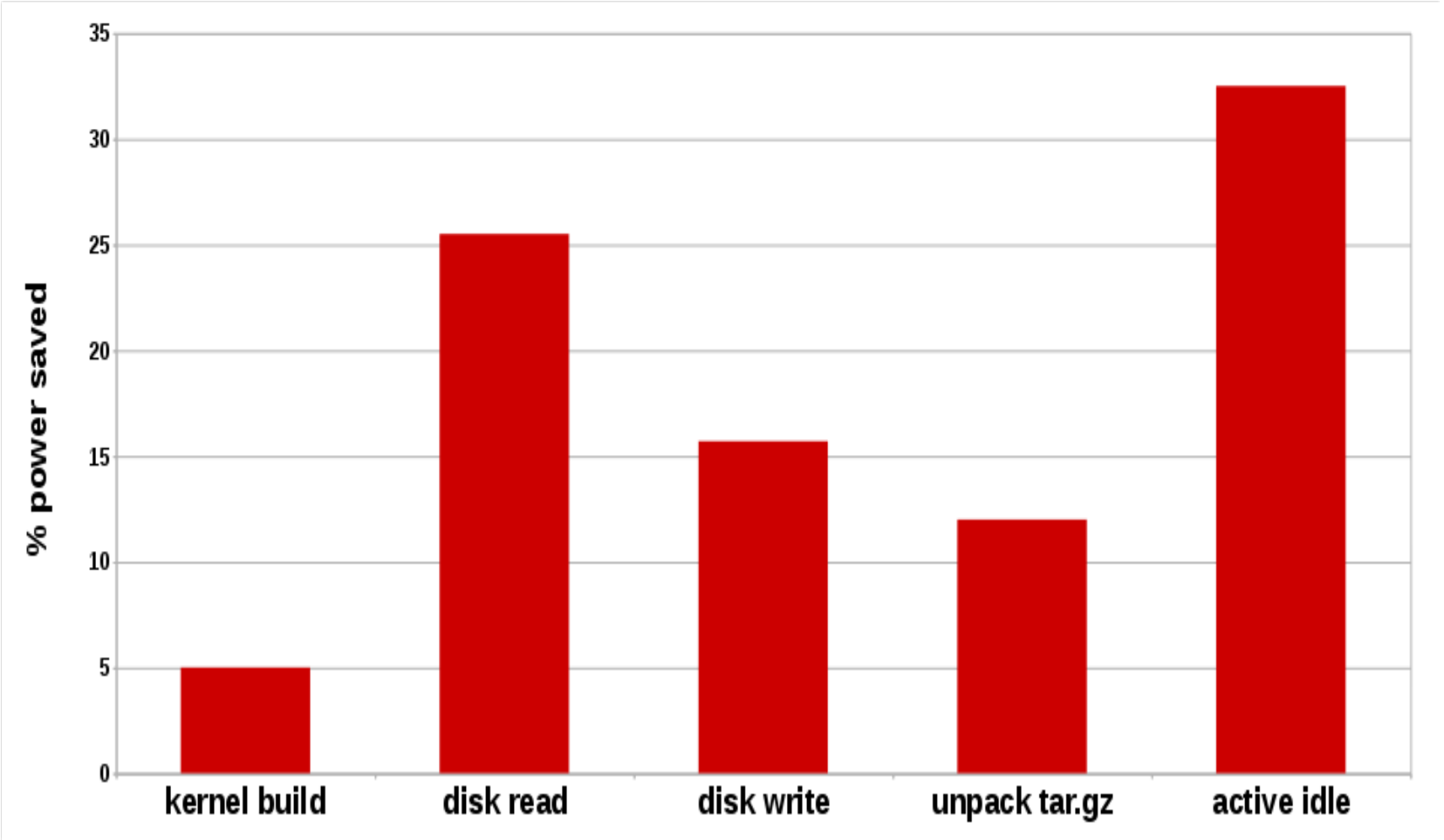
Default

pk	cor	CPU	%c0	GHz	TSC	%c1	%c3	%c6	%c7
0	0	0	0.24	2.93	2.88	5.72	1.32	0.00	92.72
0	1	1	2.54	3.03	2.88	3.13	0.15	0.00	94.18
0	2	2	2.29	3.08	2.88	1.47	0.00	0.00	96.25
0	3	3	1.75	1.75	2.88	1.21	0.47	0.12	96.44

latency-performance

pk	cor	CPU	%c0	GHz	TSC	%c1	%c3	%c6	%c7
0	0	0	0.00	3.30	2.90	100.00	0.00	0.00	0.00
0	1	1	0.00	3.30	2.90	100.00	0.00	0.00	0.00
0	2	2	0.00	3.30	2.90	100.00	0.00	0.00	0.00
0	3	3	0.00	3.30	2.90	100.00	0.00	0.00	0.00

Impact of CPU Idle Drivers (watts per workload)



Low Latency Performance Tuning Guide for Red Hat Enterprise Linux 7

- Tactical tuning overview for latency-sensitive workloads.
- Emphasizes impactful new features included in RHEL7:
  - CPU/power management
  - NUMA
  - tuned profiles
  - scheduling
  - network tunables
  - kernel timers.
  - "de-jittering" CPU cores
  - tracing techniques

https://access.redhat.com/articles/1323793

Full DynTicks Patchset - nohz_full=1

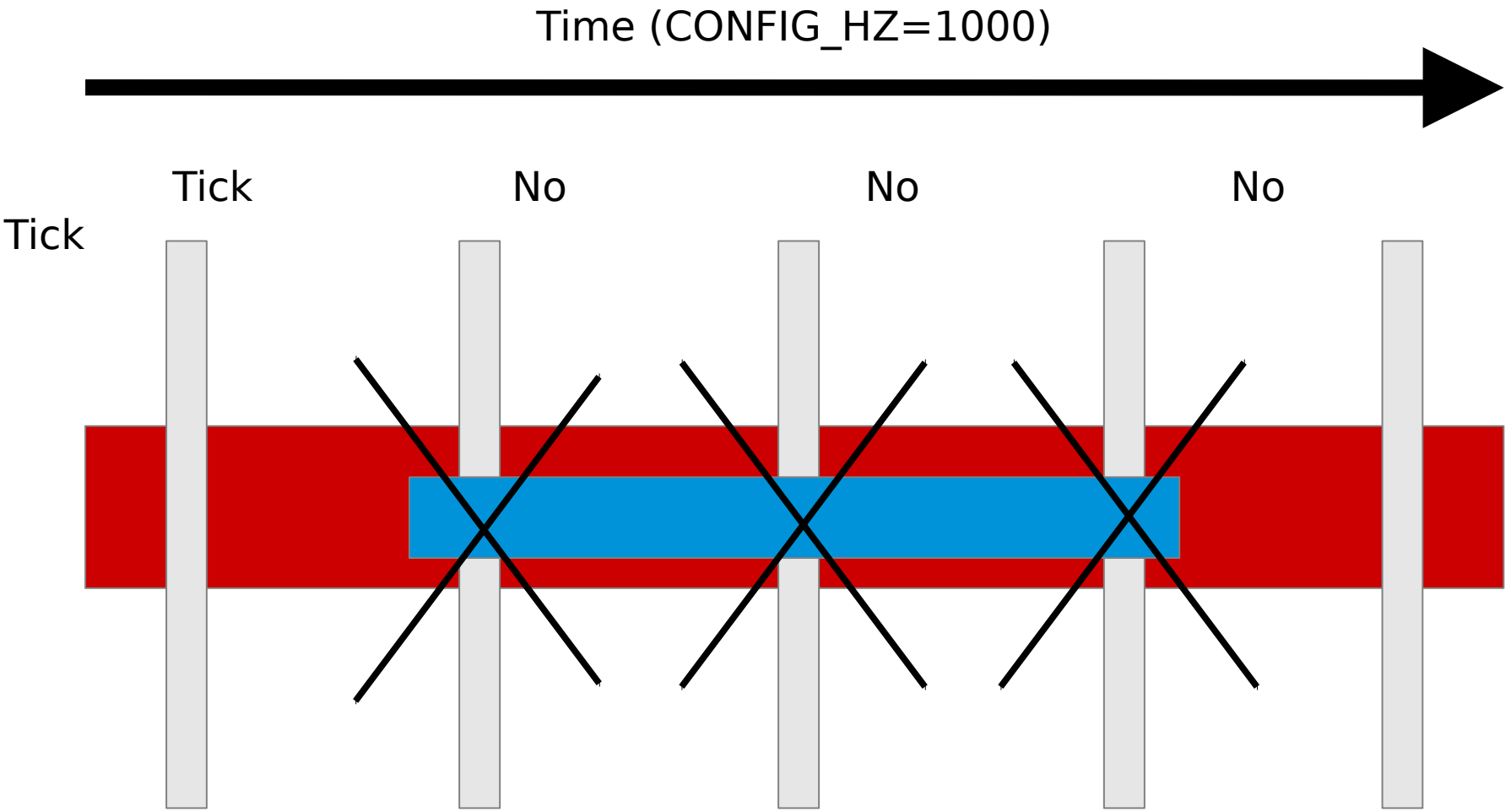
Patchset Goal:

- Stop interrupting userspace tasks
- Move timekeeping to non-latency-sensitive cores
- If nr_running=1, then scheduler/tick can avoid that core
- Default disabled...Opt-in via nohz_full cmdline option

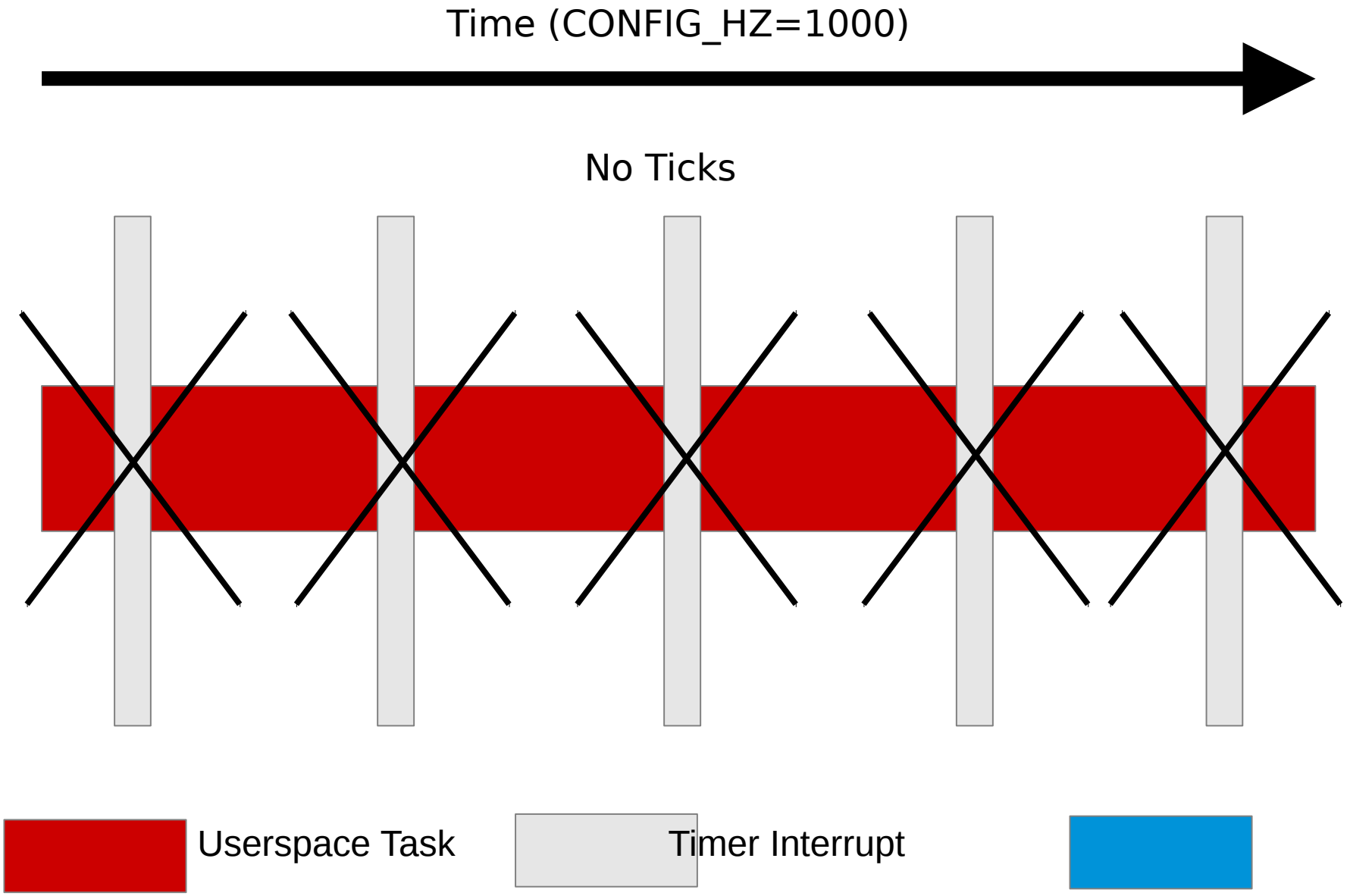
Kernel Tick:

- timekeeping (gettimeofday)
- Scheduler load balancing
- Memory statistics (vmstat)

RHEL6 and 7 Tickless

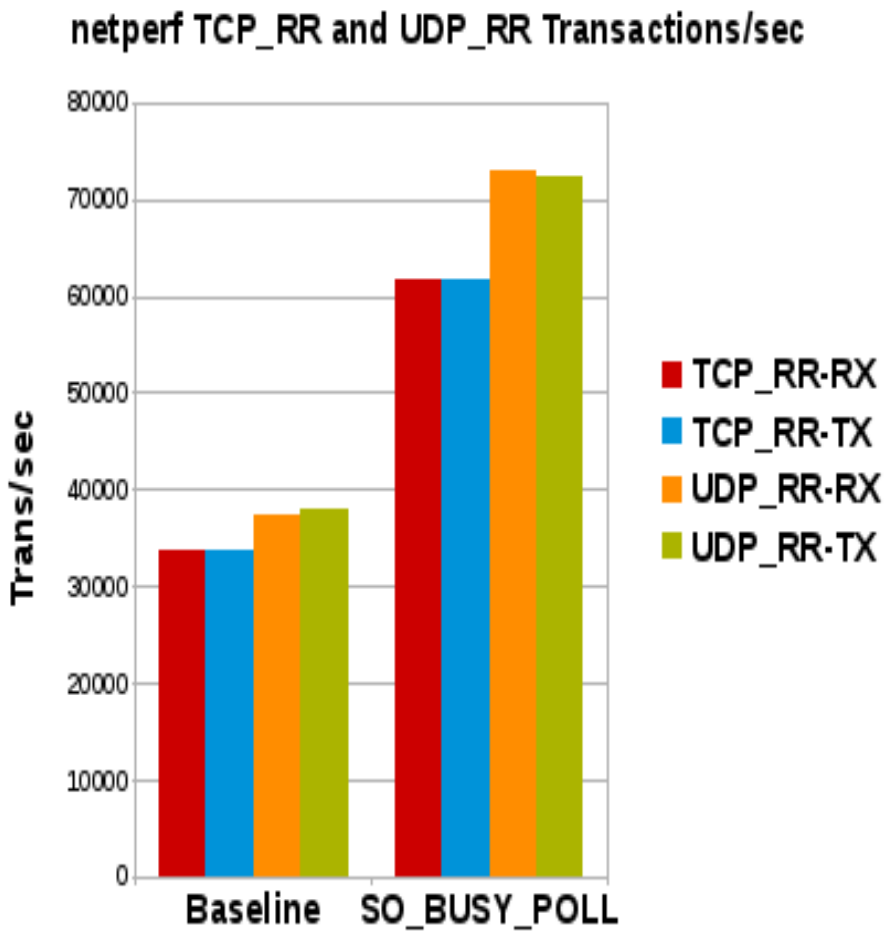


RHEL7 nohz_full



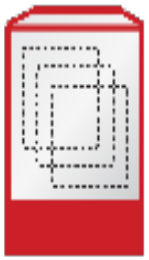
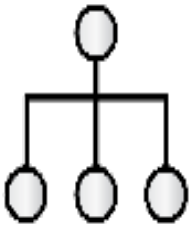
SO_BUSY_POLL Socket Option

- Socket-layer code polls receive queue of NIC
- Replaces interrupts and NAPI
- Retains full capabilities of kernel network stack

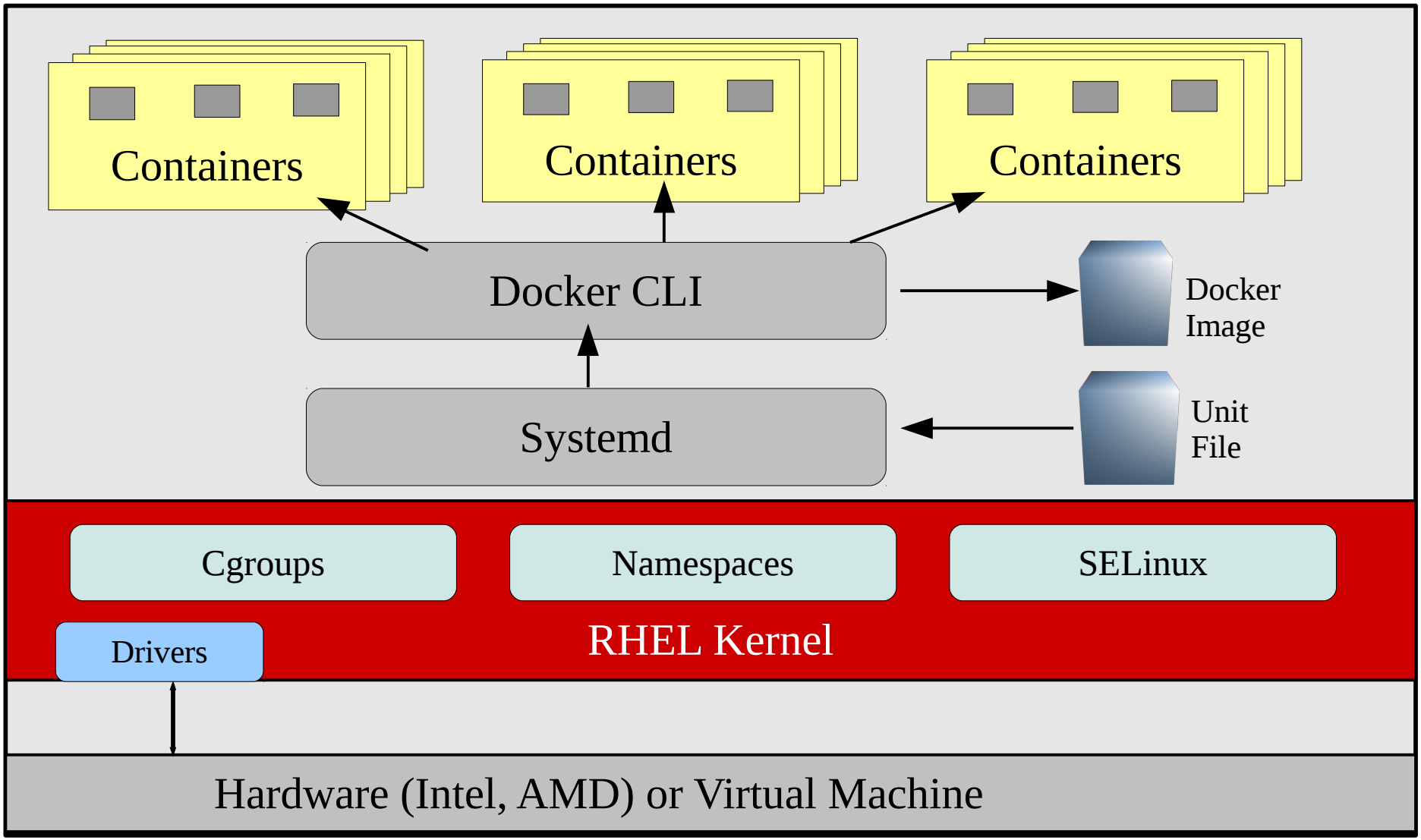


RHEL Atomic / Open Shift

Red Hat Enterprise Linux Atomic Host

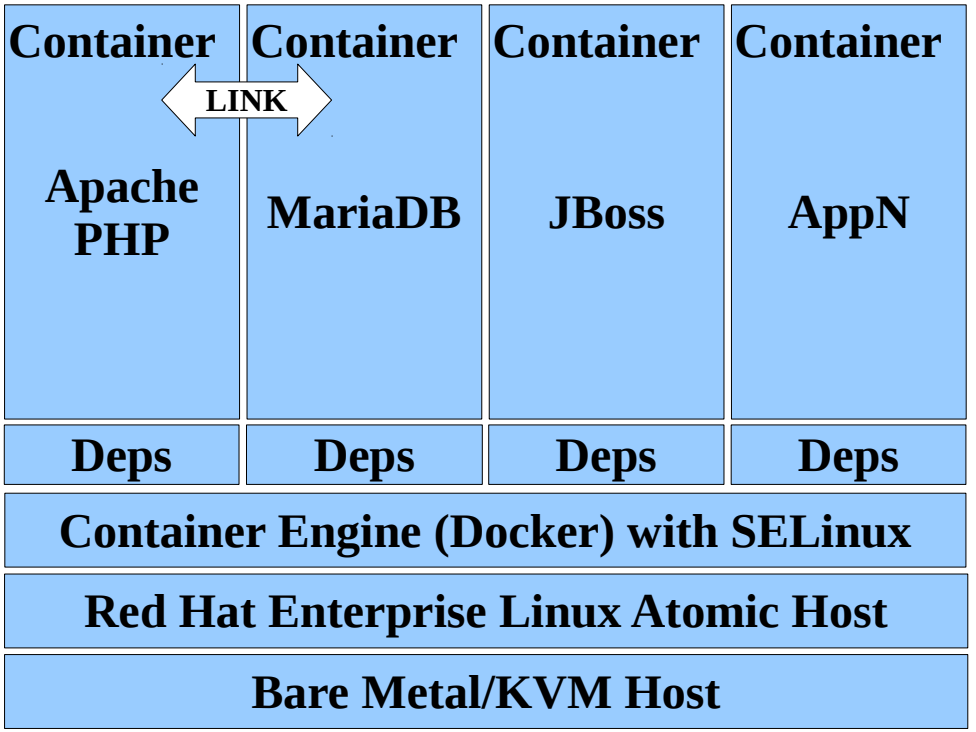
IT IS RED HAT ENTERPRISE LINUX	OPTIMIZED FOR CONTAINERS		
	 MINIMIZED FOOTPRINT	 SIMPLIFIED MAINTENANCE	 ORCHESTRATION AT SCALE
Inherits the complete hardware ecosystem, military-grade security, stability and reliability for which Red Hat Enterprise Linux is known.	Minimized host environment tuned for running Linux containers while maintaining compatibility with Red Hat Enterprise Linux.	Atomic updating and rollback means it's easy to deploy, update, and rollback using imaged-based technology.	Build composite applications by orchestrating multiple containers as microservices on a single host instance.

RHEL 7 Containers Architecture

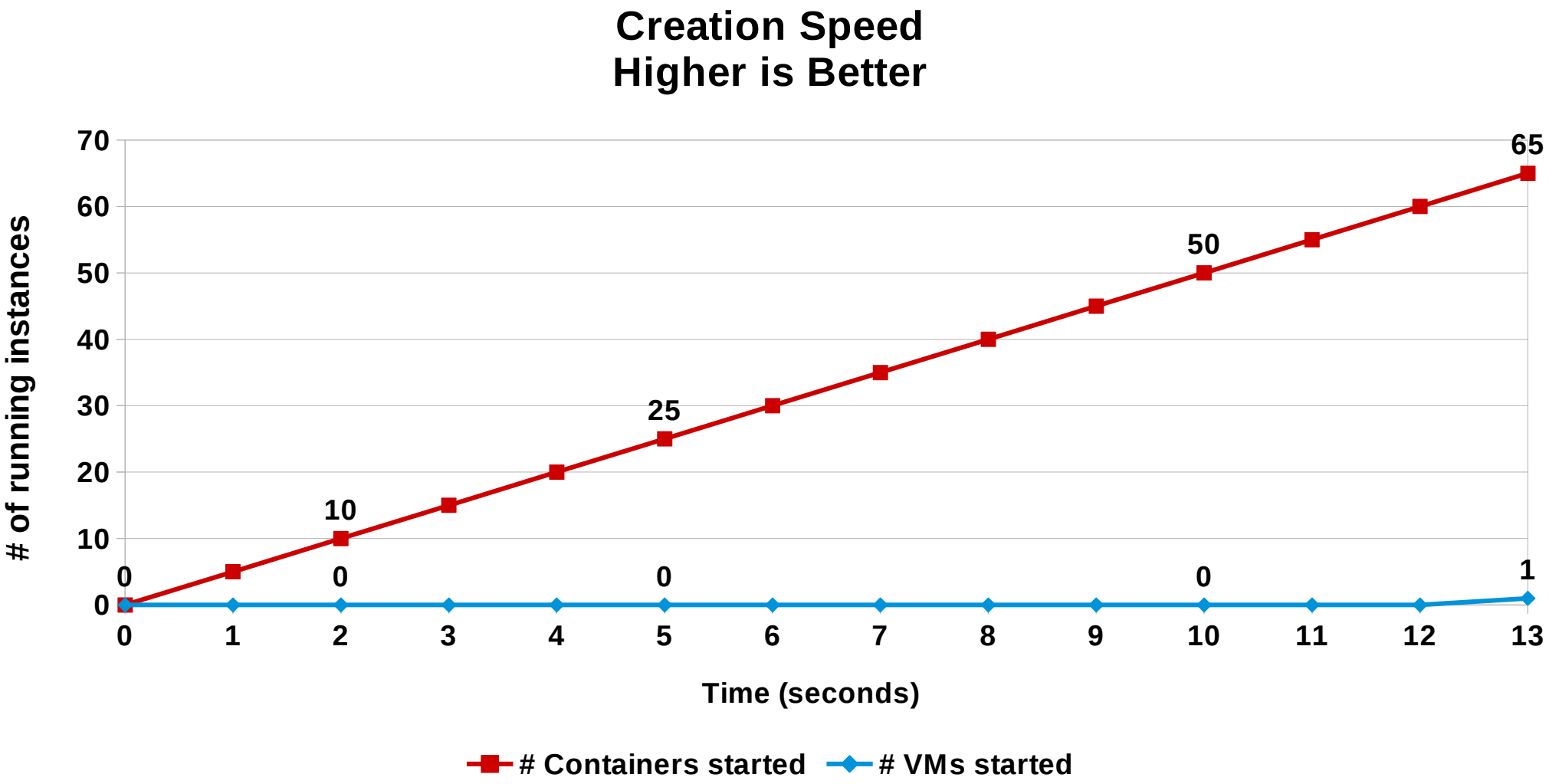


Density

- Depends on
  - Hardware/Cloud capabilities
  - Workload requirements
  - Overcommit configuration
  - Network capabilities/config
    - ~1000 using Linux Bridge
    - Higher with SDN
  - Storage capabilities/config
    - Loop-LVM: no setup, Good performance...for development
    - Direct-LVM: requires setup, best performance/scale for production

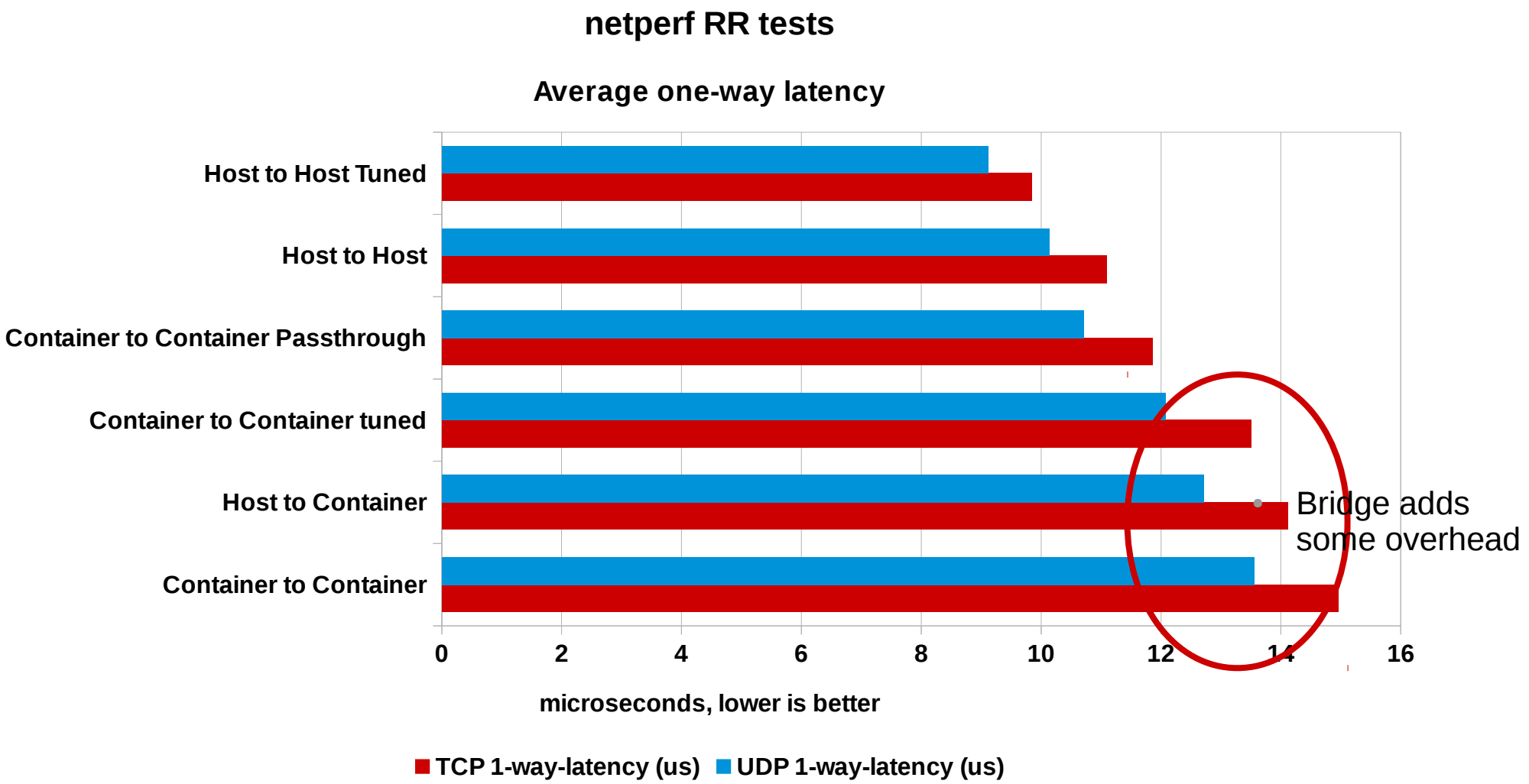


Container creation speed vs VMs

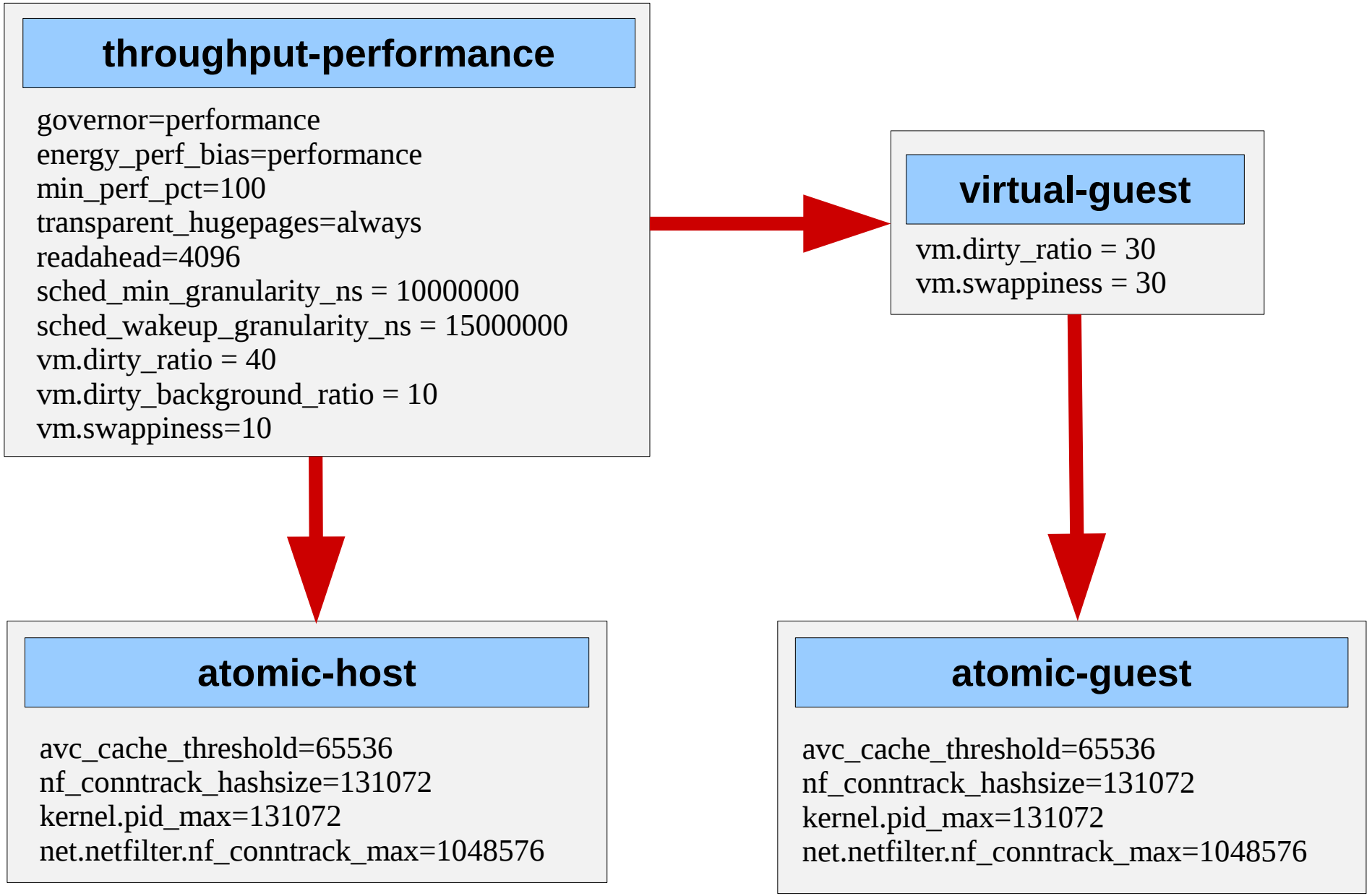


Network Performance: netperf, RR tests (latency)

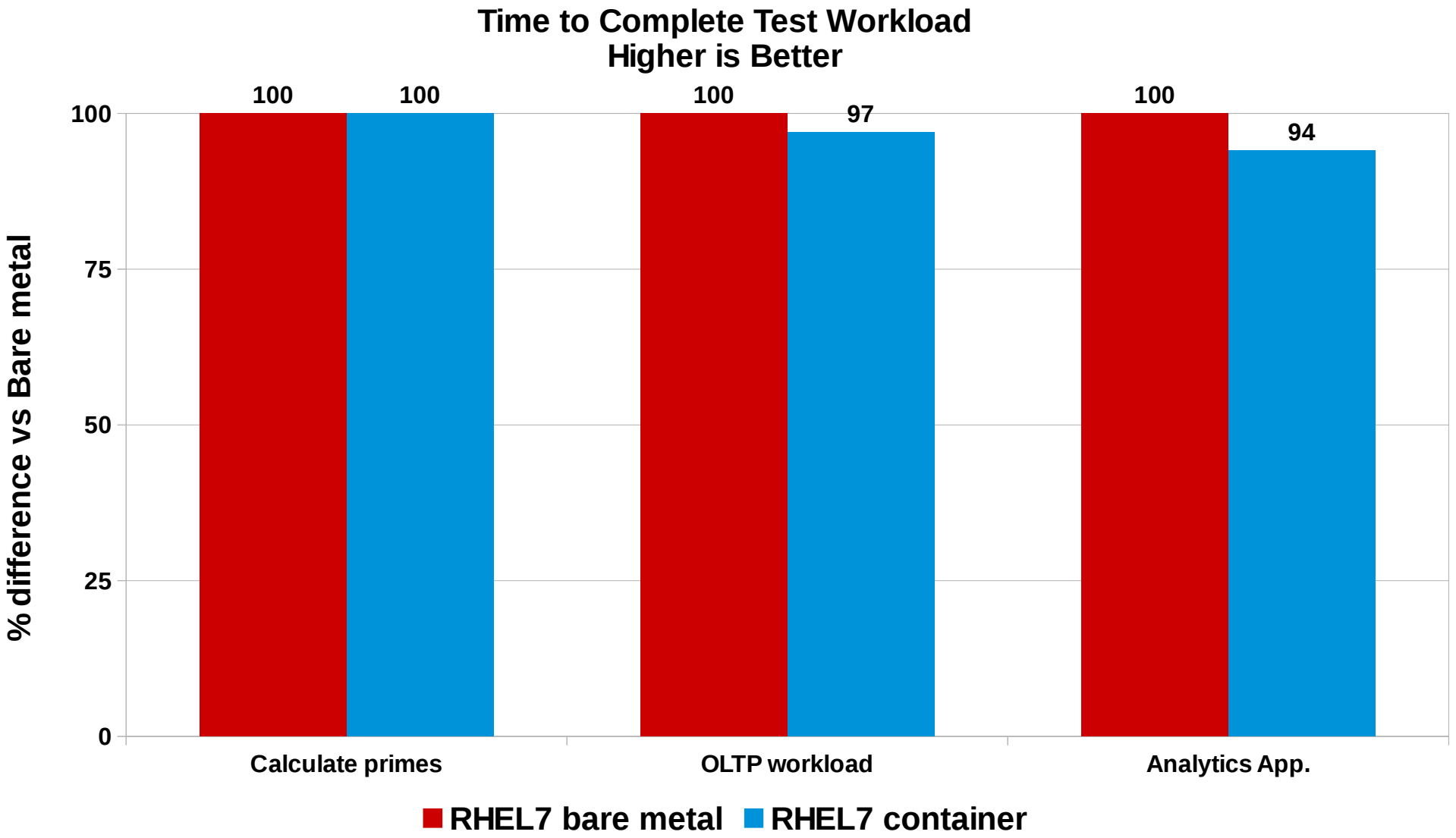
- With cpu pinning and IRQ Affinity in place



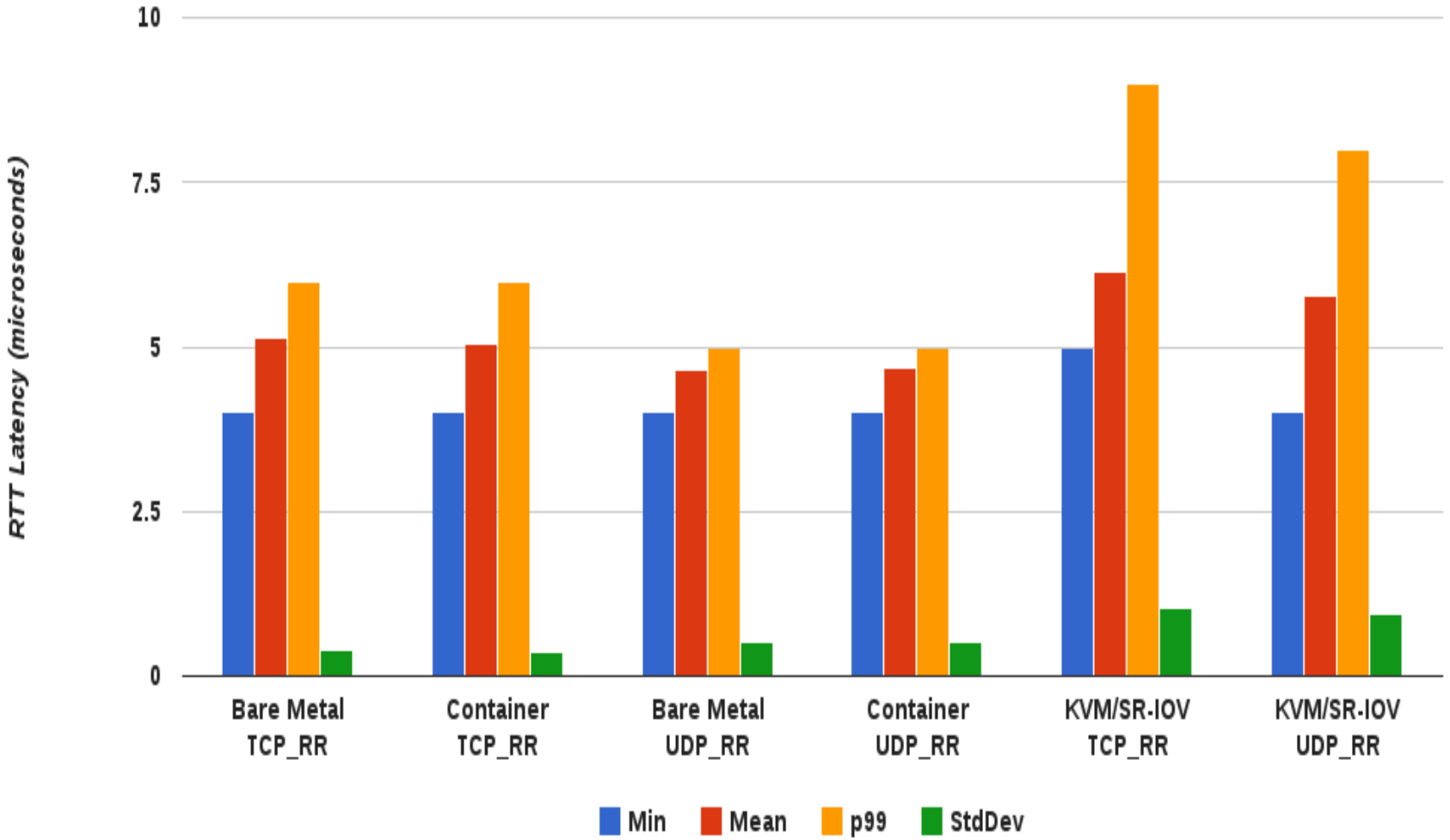
Tuned: RHEL Atomic Profile Inheritance



Container performance across multiple workloads



RHEL 7.1 + OpenOnload
Bare Metal/Containers/SR-IOV



References

Low Latency Tuning Guide for Red Hat Enterprise Linux 7

https://access.redhat.com/articles/1323793

Accelerating Red Hat Enterprise Linux 7-based Linux Containers with Solarflare OpenOnload

https://access.redhat.com/articles/1407003

How do I create my own tuned profile on RHEL7 ?

https://access.redhat.com/solutions/731473

KVM / NFV / Realtime

Quick Overview – KVM Architecture

- Guests run as a process in userspace on the host
- A virtual CPU is implemented using a Linux thread
  - The Linux scheduler is responsible for scheduling a virtual CPU, as it is a normal thread
- Guests inherit features from the kernel
  - NUMA
  - Huge Pages
  - Support for new hardware

Virtualization Tuning – Caching

Figure 1

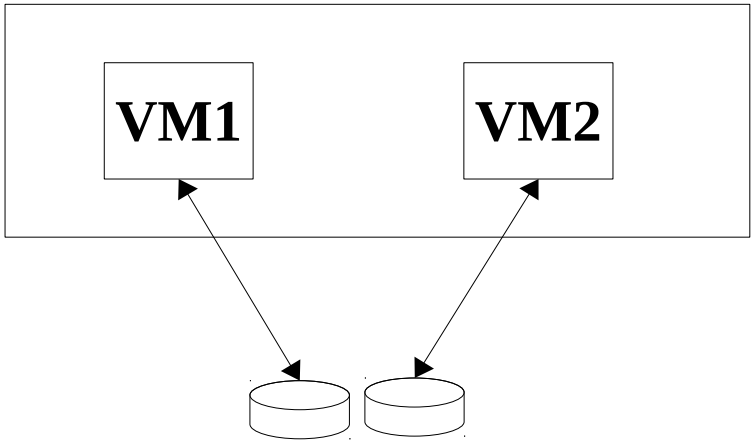
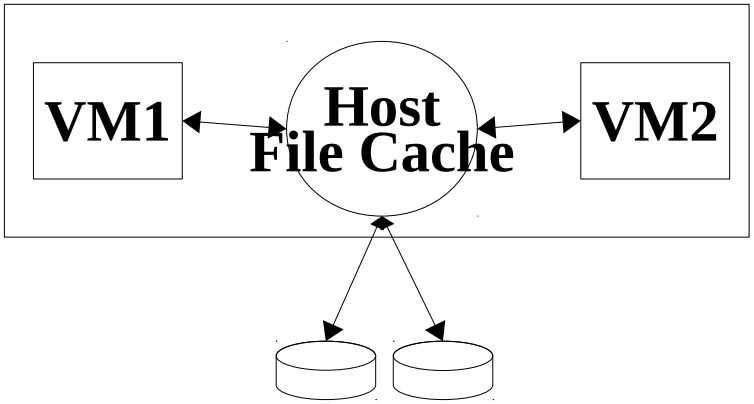
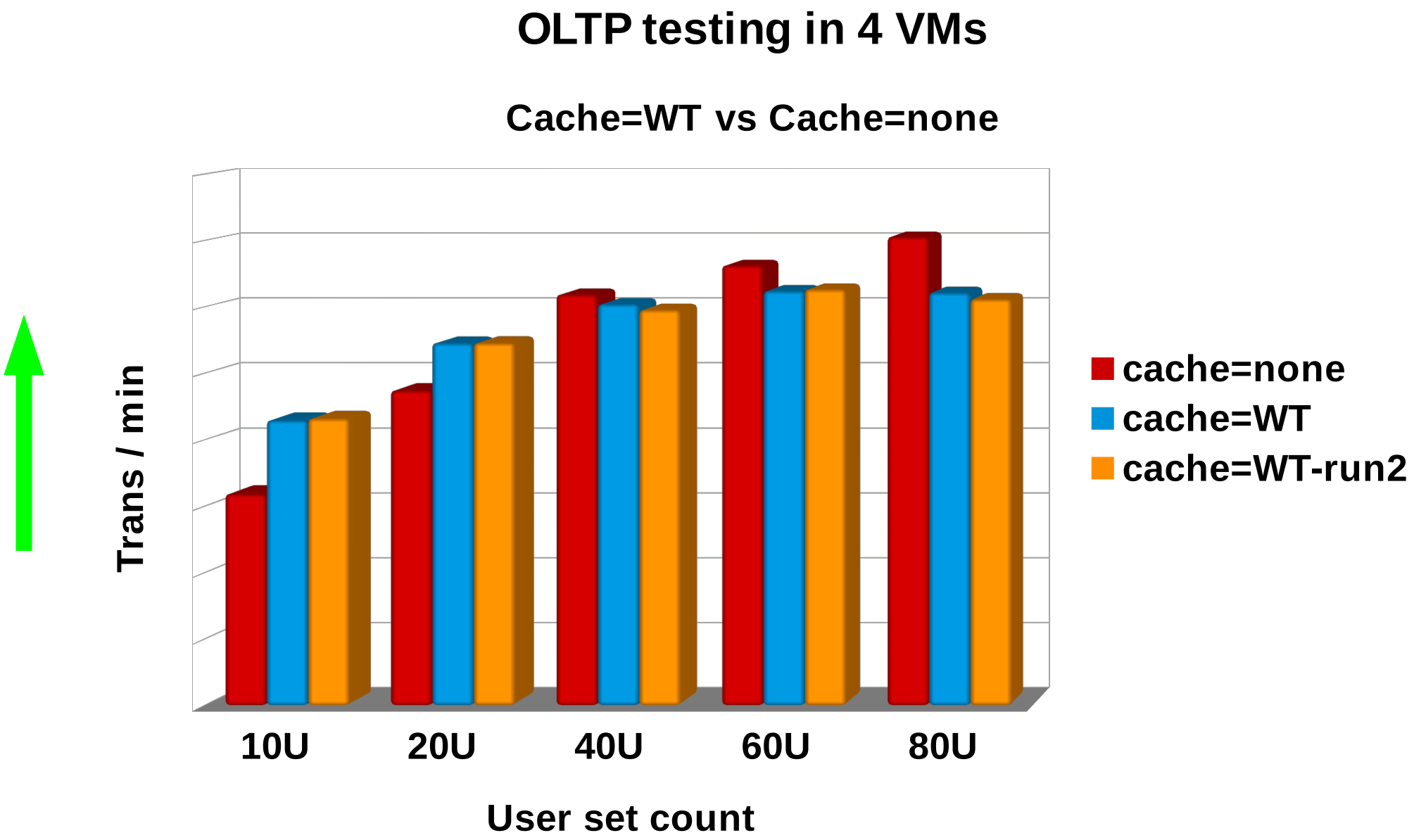


Figure 2

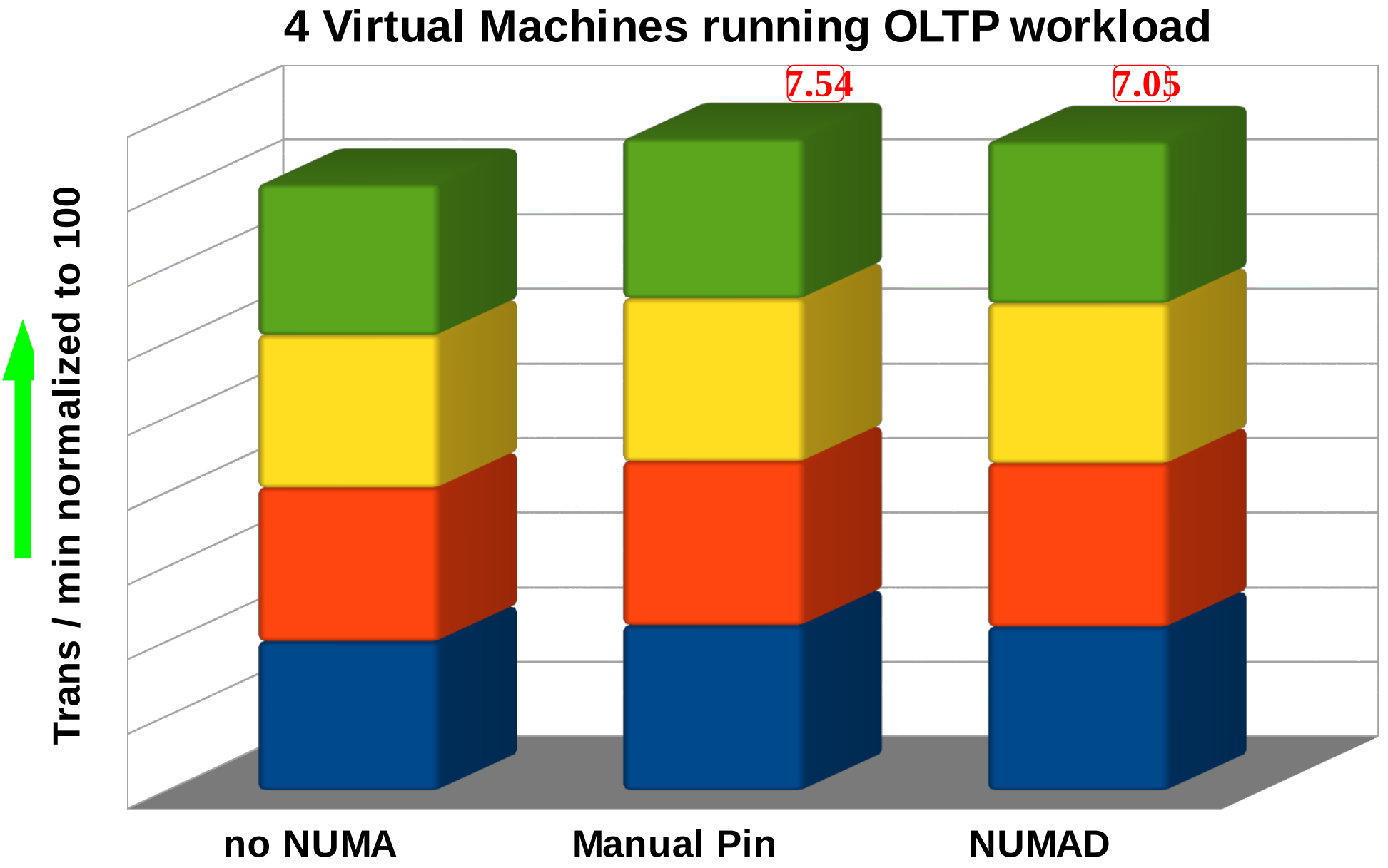


- **Cache = none (Figure 1)**
  - I/O from the guest is not cached on the host
- **Cache = writethrough (Figure 2)**
  - I/O from the guest is cached and written through on the host
    - Works well on large systems (lots of memory and CPU)
    - Potential scaling problems with this option with multiple guests (host CPU used to maintain cache)
    - Can lead to swapping on the host

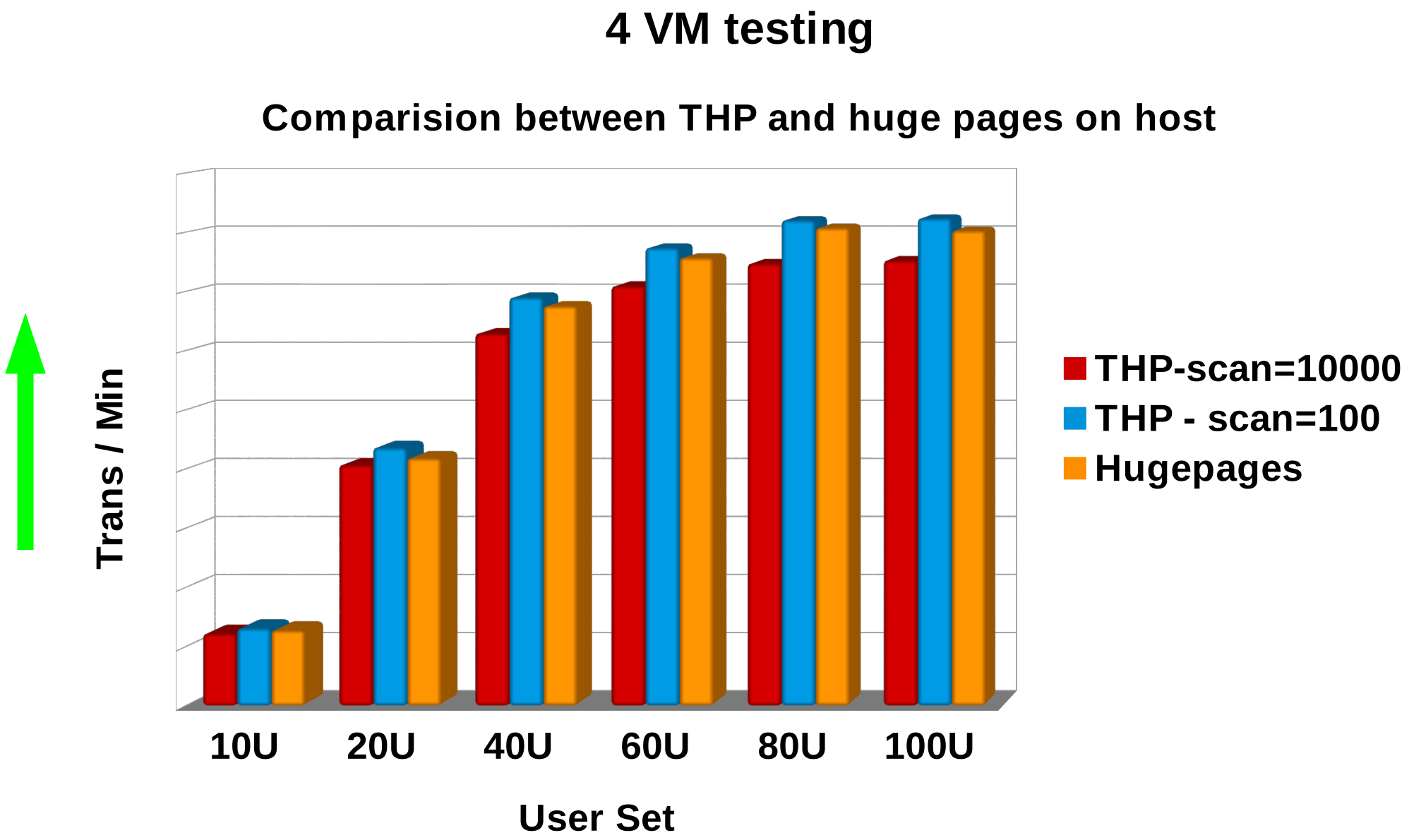
Virt Tuning – Effect of I/O Cache Settings



Virt Tuning – Using NUMA



Virt Tuning – Tuning Transparent Huge Pages

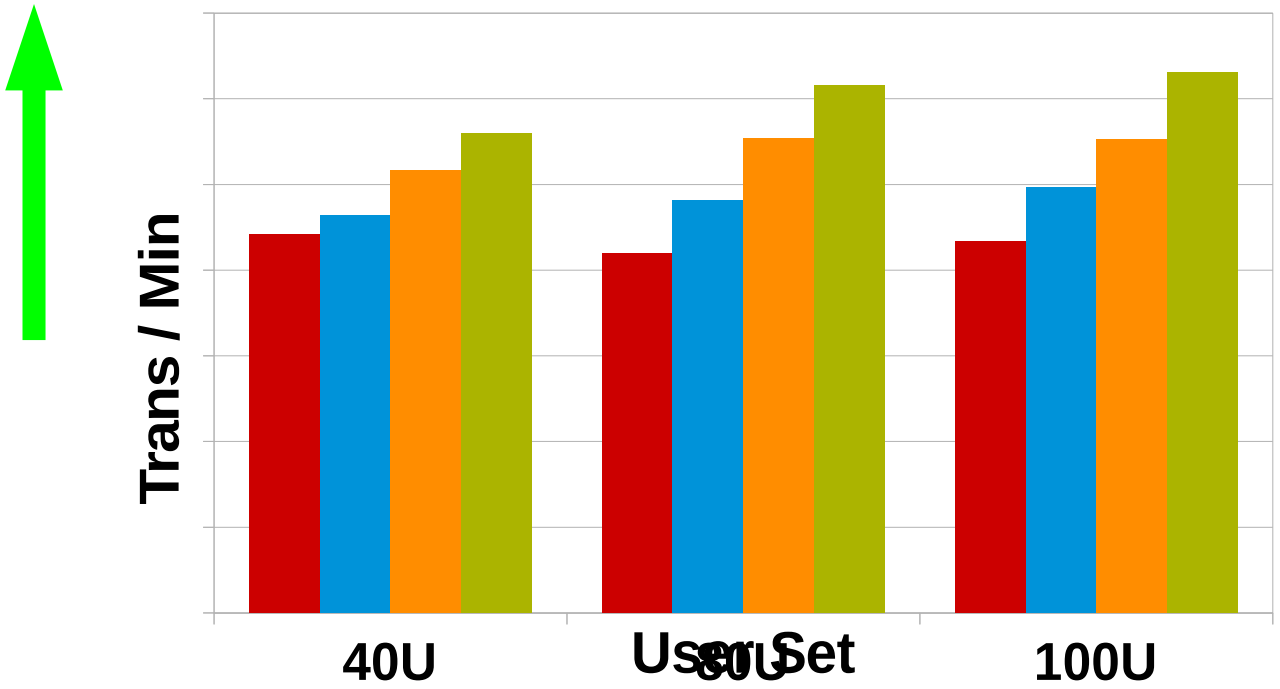


Virt Tuning – Kernel Samepage Merging (KSM)

KSM breaks down THP, so performance advantage of THP is lot.

Some of it is recovered by lowering the scan interval

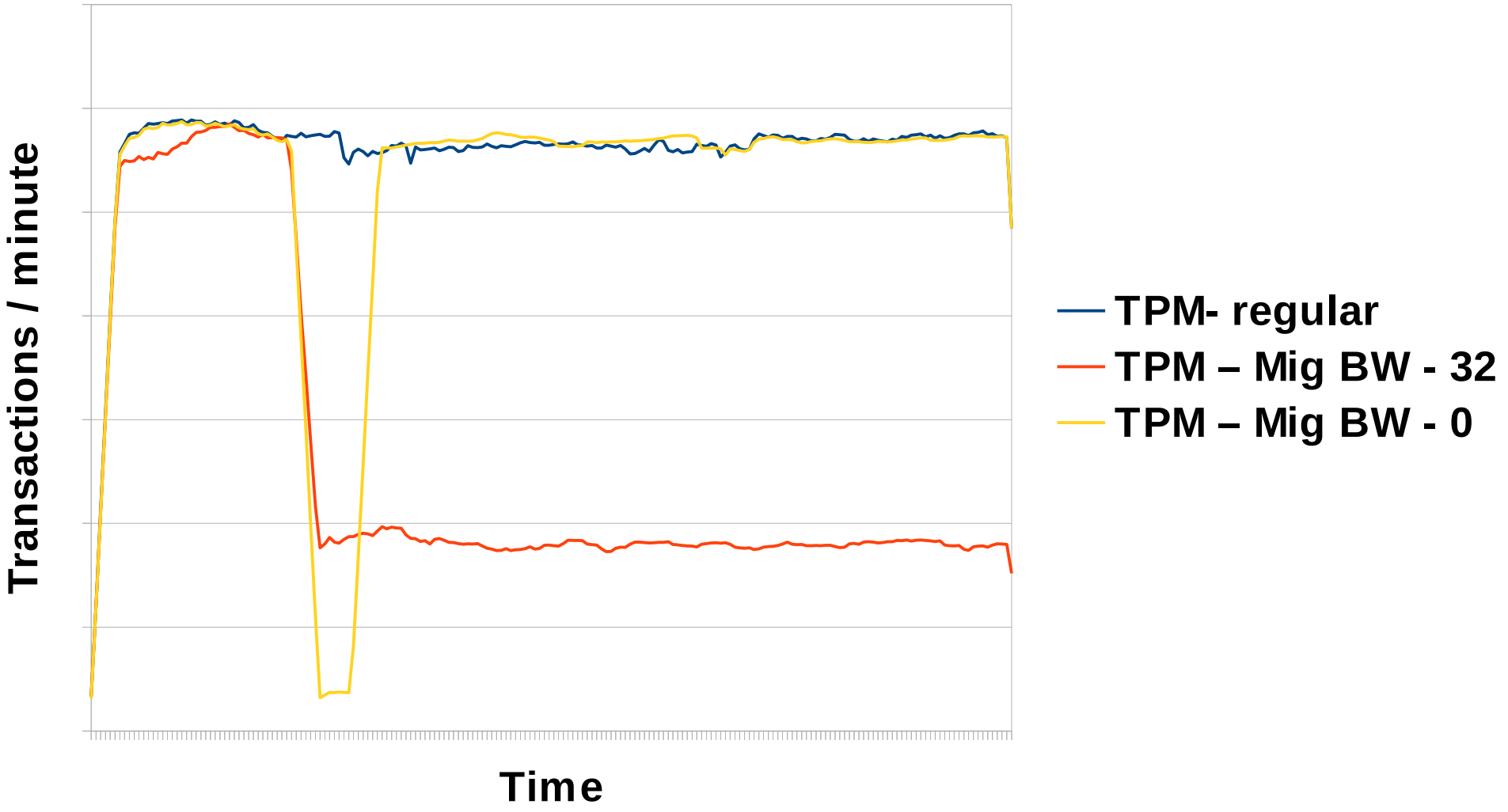
CPU overhead of KSM results in less CPU time for application



- THP 10000scan, ksm on, mem opt server (150)
- THP 100scan, ksm on, mem opt server (150)
- THP 100scan, ksm off, mem opt server(150)
- THP 100scan, ksm off, no memory opt

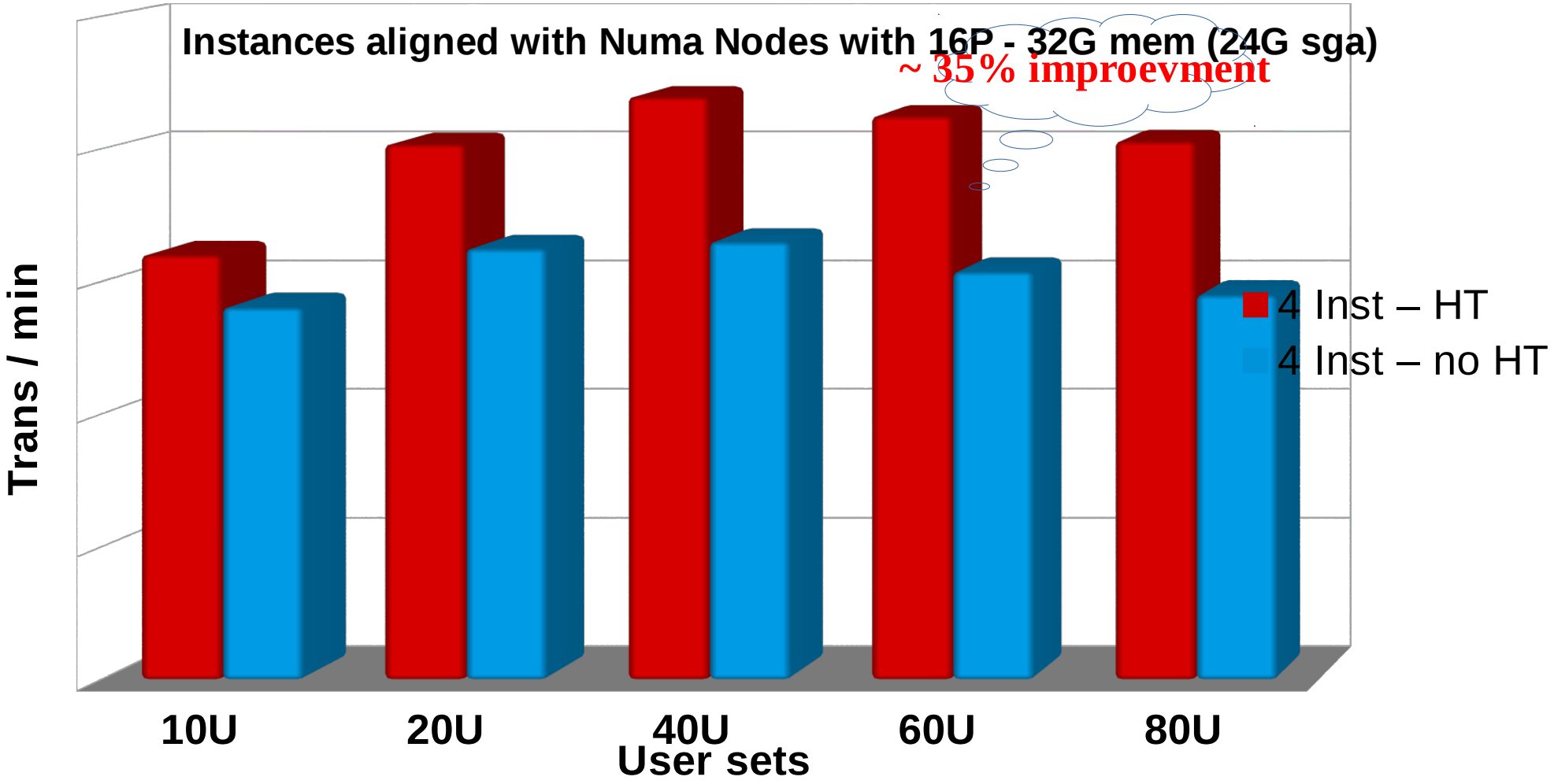
RHEV 3.3 – Migration

Migrating a 108G VM running OLTP ~ 500K Trans/min



Configure – migration_max_bandwidth = <Value> in

Multi Instances of database with and without Hyperthreads

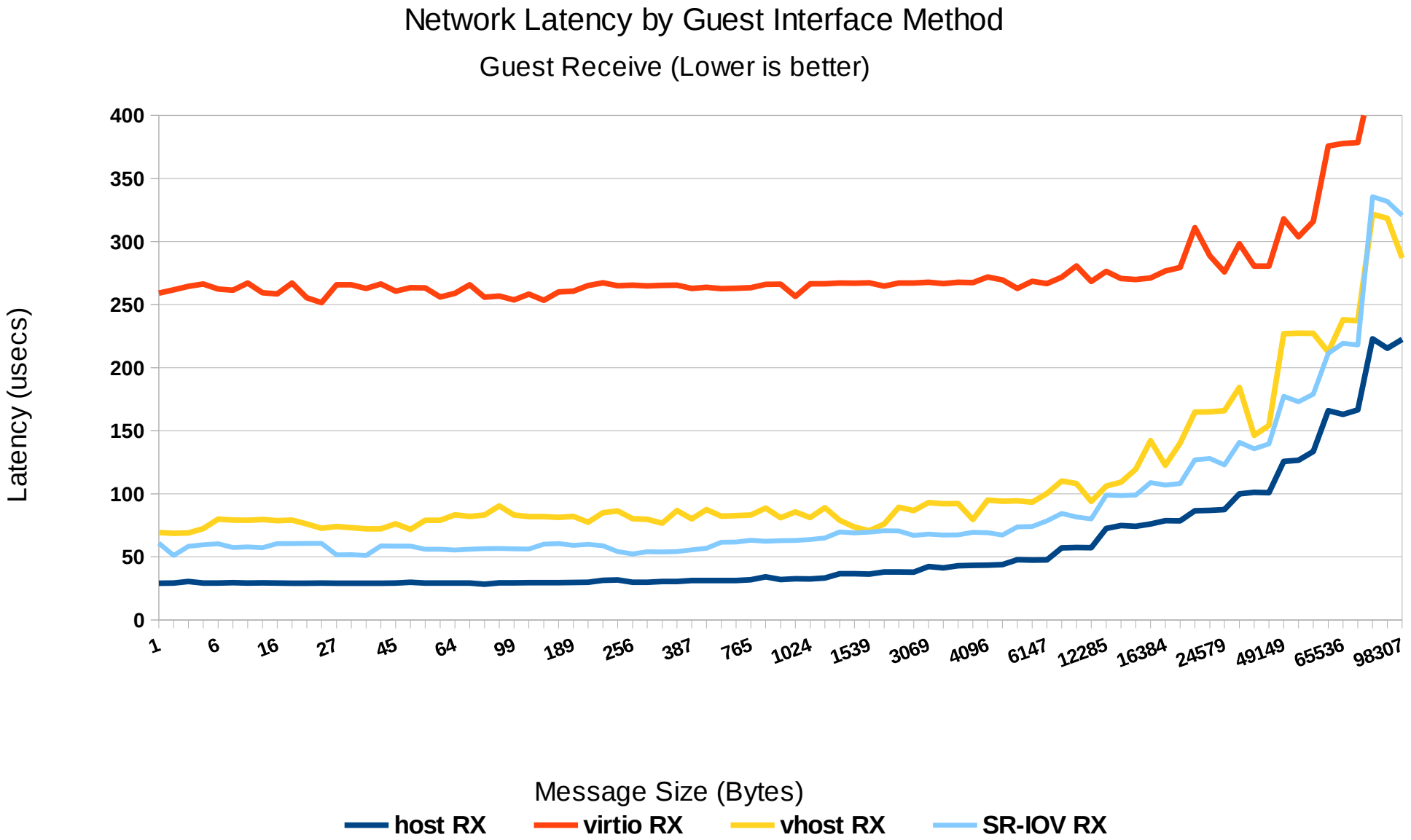


Each of the 4 instances were aligned to an individual NUMA node. This test shows the best gain in performances as other factors influencing performance like NUMA, I/O are not a factor

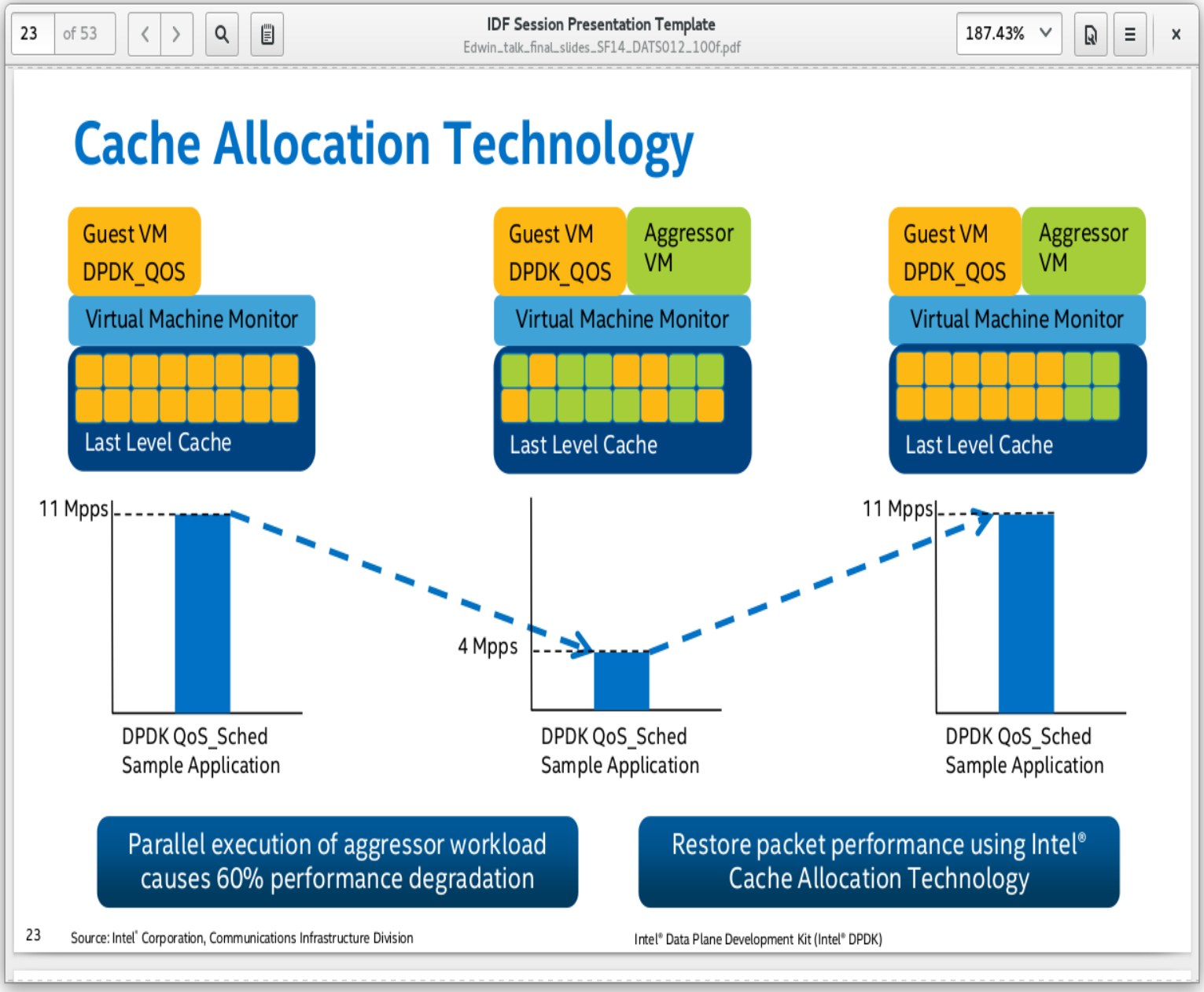
Virtualization Tuning – Network

- VirtIO
  - ✔ VirtIO drivers for network
- vhost_net (low latency – close to line speed)
  - ✔ Bypass the qemu layer
- PCI pass through
  - ✔ Bypass the host and pass the PCI device to the guest
  - ✔ Can be passed only to one guest
- SR-IOV (Single root I/O Virtualization)
  - ✔ Pass through to the guest
  - ✔ Can be shared among multiple guests
  - ✔ Limited hardware support

Virtualization Tuning – Network – Latency Comparison



- **Intel Cache Monitoring / Cache Allocation Tech**



- **Intel Cache Monitoring / Cache Allocation Tech**

Intel® Xeon® Processor E5 v4 Product Family Resource Director Technology (RDT)

Feature	Benefit	How Does it Work?
Cache Monitoring Technology (CMT)	Ability to monitor Last Level Cache occupancy for a set of threads	Each thread assigned a RMID (Resource Monitoring ID)
Cache Allocation Technology (CAT)	Ability to partition Last Level Cache, enforcement on a per thread basis Enables workload prioritization, consolidation, and resource partitioning Enables control over noisy neighbors	Each thread assigned a Class of Service Each Class of Service restricted to portion of LLC
Code and Data Prioritization (CDP)	A specialized extension of CAT which enables separate masks for code and data. This allows code to be protected at the L3 cache level for instance	Half of the masks are associated with code, the other half of the masks are associated with data
Memory Bandwidth Monitoring (MBM)	Monitors Memory Bandwidth utilization on an RMID basis. Identify memory bandwidth conflict issues and enable thread migration	RMIDs can be associated with one or a group of threads / applications

Intel Cache Monitoring / Cache Allocation Tech

- **Examples -** Lock target cache lines into the cache via CAT
- **Set the default way mask to not include the last two cache ways**

  - wrmsr 0xc90 0xFFFFC

- **Set a second way mask to only include the last two cache ways**

  - wrmsr 0xc91 0x3

− Set core 1 to use the second way mask

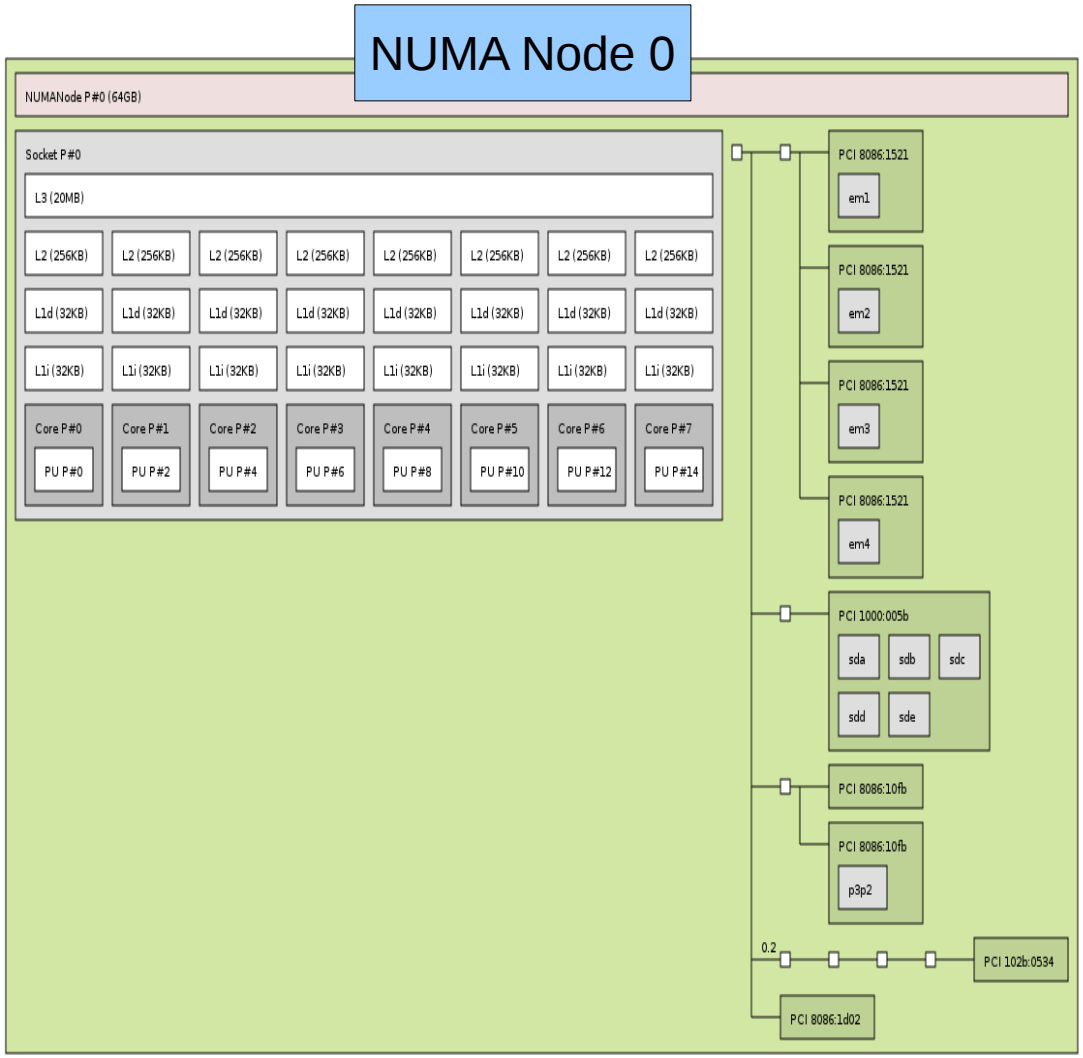
  - wrmsr –p 1 0xc8f 0x100000000

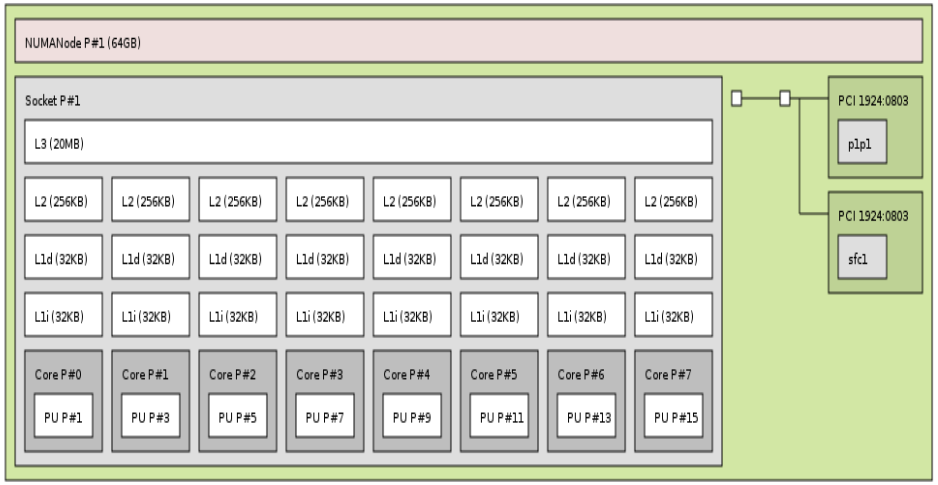
− Use core 1 to touch all of the memory locations one wishes to lock permanently into the cache

− Set core 1 to use the default (first) way mask

  - wrmsr –p 1 0xc8f 0x00000000

Know your hardware: lstopo





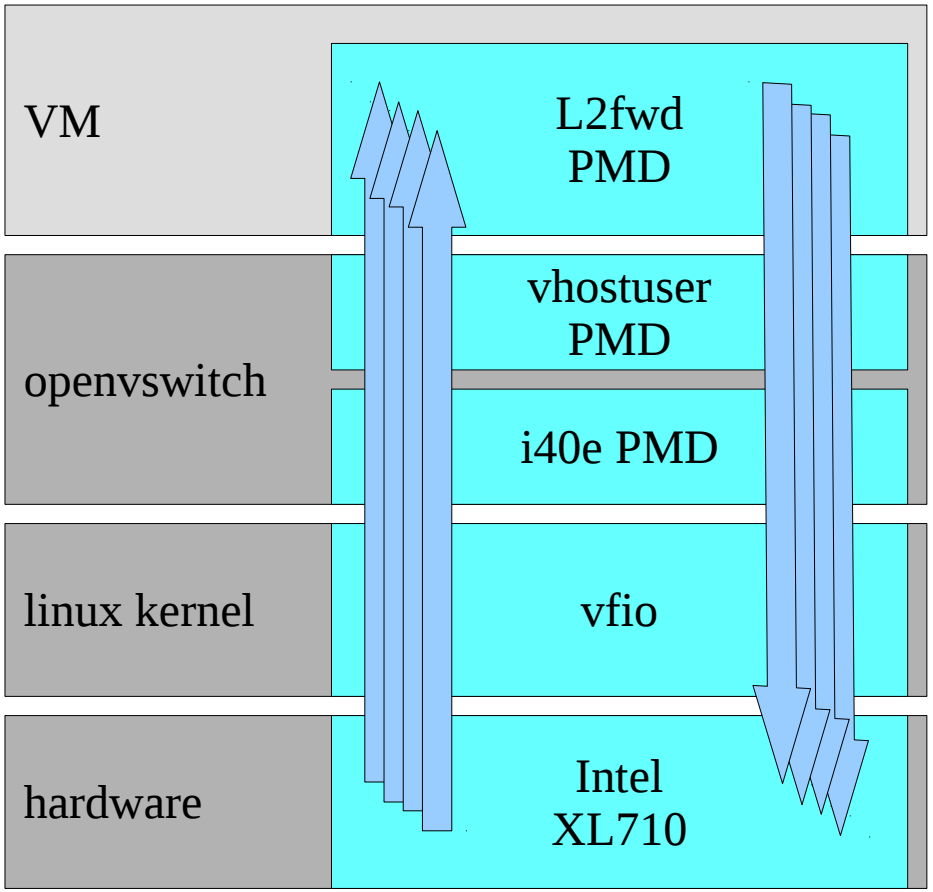
How can I visualize my system's NUMA topology in Red Hat Enterprise Linux?

https://access.redhat.com/site/solutions/62879

DPDK Accelerated Virtual Switch

- Enables packet switching from the physical NIC on the host (hypervisor) to the VNF application in the Guest VM and between Guest VMs to be handled almost entirely in user-space
- Support for both DPDK physical and KVM/QEMU vhost-user ports
  - vhost-user is a QEMU feature that provides efficient virtio-net I/O between a guest VM and a user-space vswitch
- DPDK support for OVS is still under development in the upstream
  - OVS 2.4.0 Single Queue DPDK , Shipped in RHEL7.2, OSP8
  - OVS 2.5.0 Multi Queue DPDK support
    - *2.5.0 has been held up because of upstream conntrack work; for RHEL 7.3, 7.2z*
- OpNFV – RH Performance Team contributing with Office of Tech, and Kernel / QE
  - VSperf group w/ consortia
  - Develop “MoonGen” (University Munich), control flows, measure loss,
  - Work on other HW packet generators – Xena, XIA, Sprient

RHEL 7.2 vhostuser – multi-queue



Upstream ovs-dpdk (2.5), Intel 40Gb XL710
64-byte frames
33.4 Million packets per second!
OVS: 8 cores (16 threads), 2 bridges, each using 4 i40e PMD threads + 4 vhostuser PMD threads
VM 4: cores (8 threads), 2 vhostuser interfaces, each using 4 virtio PMD thread

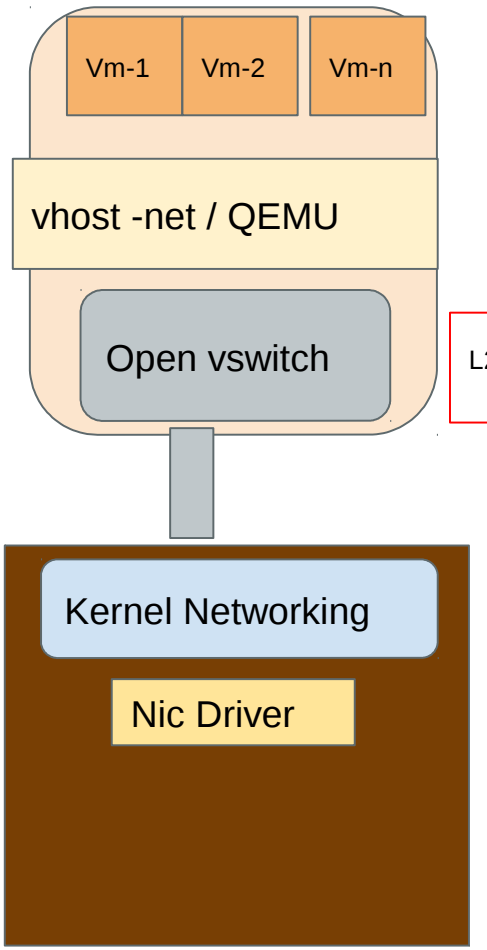
Performance with tiny frames 64 Bytes

Kernel Networking

L2 Fwd **0.52** Mpps in one VM out another VM Single Queue

L2 Fwd **0.90** Mpps Till VMs Multi-Queue

L2 Fwd **9.05** Mpps Till OVS

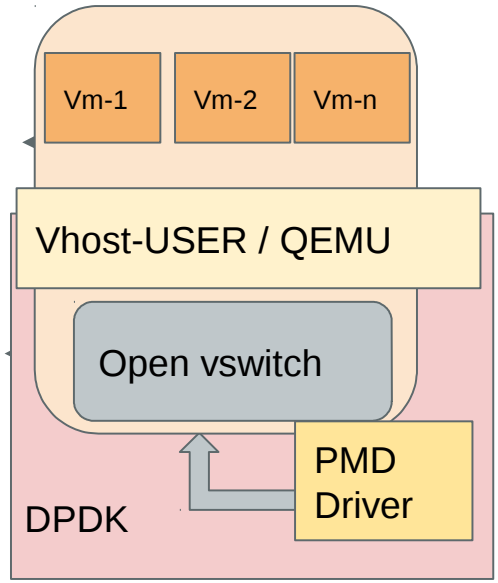


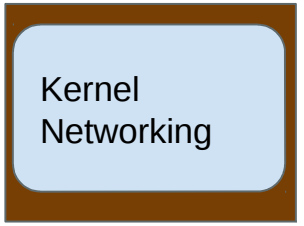
DPDK kernel bypass

L2 Fwd **4.04** Mpps in one Vm out another VM Single Queue

L2 Fwd **14.21** Mpps Till VMs Multi-Queue

L2 Fwd **14.88** Mpps Till OVS

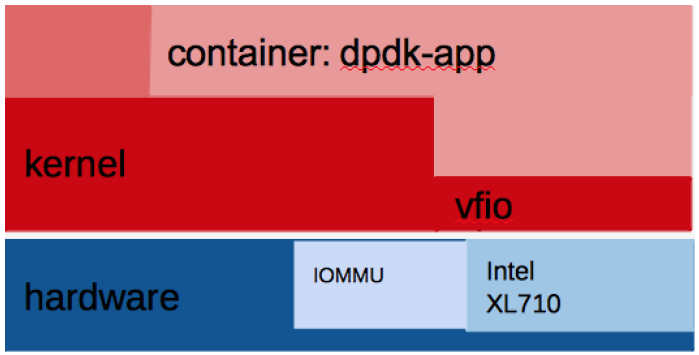




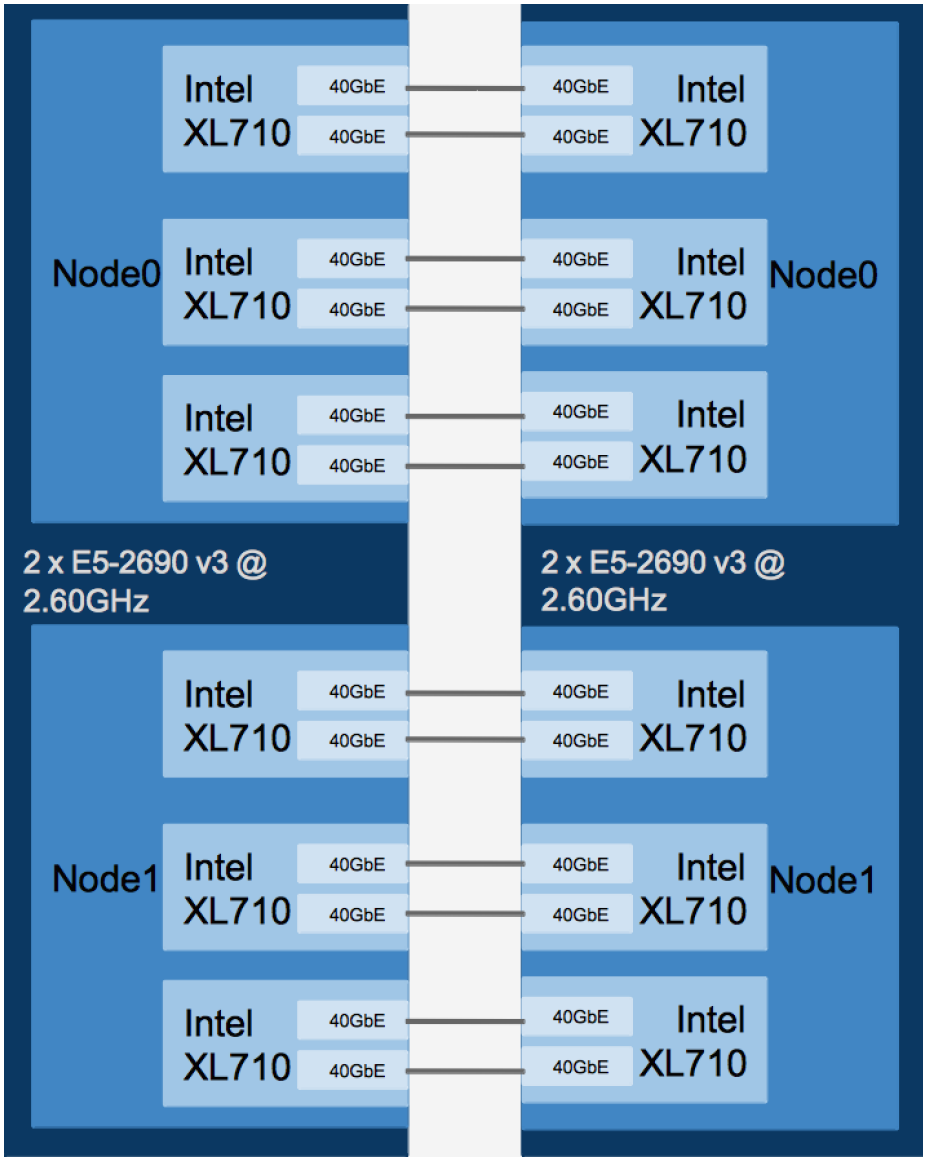
64 Byte frames, 10G link, Theoretical limit 14.88 Million Packets Per Second (Mpps)

Scalability w/ 6 – 40Gb adapters i40e Packets/sec Passthrough (RHEL7.1 + DPDK)

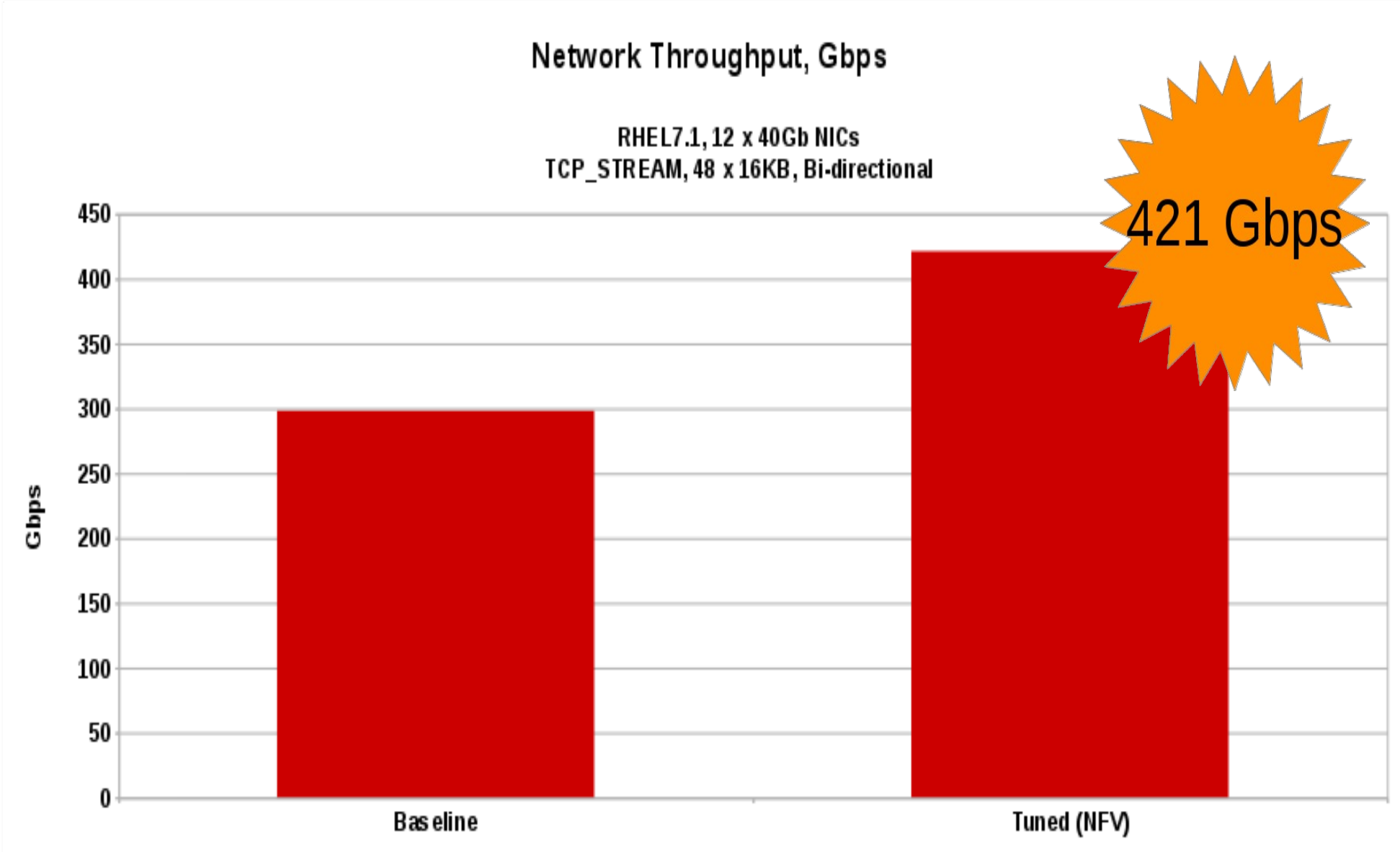
Configurations: BM/Atomic/KVM



- **Boot options**
  - CPU cstate=1
  - Reserve 1GB hugepages
  - isolate CPUs
- **DPDK setup**
  - Allocate PCI physical functions [using vfio]
- **Run a DPDK application in container (ovs-dpdk, pktgen, l2fwd, or testpmd)**
  - container includes dpdk packages, access to PCI dev
  - specify which poll-mode-driver library to use (one for each network interface type)
  - specify which PCI function(s) to use
  - specify which CPUs and memory to use
- **Current software versions tested:**
  - dpdk-2.0.0-6
  - pktgen-dpdk-2.8.4
  - openvswitch-2.3.90-10031.gitf097013a.3

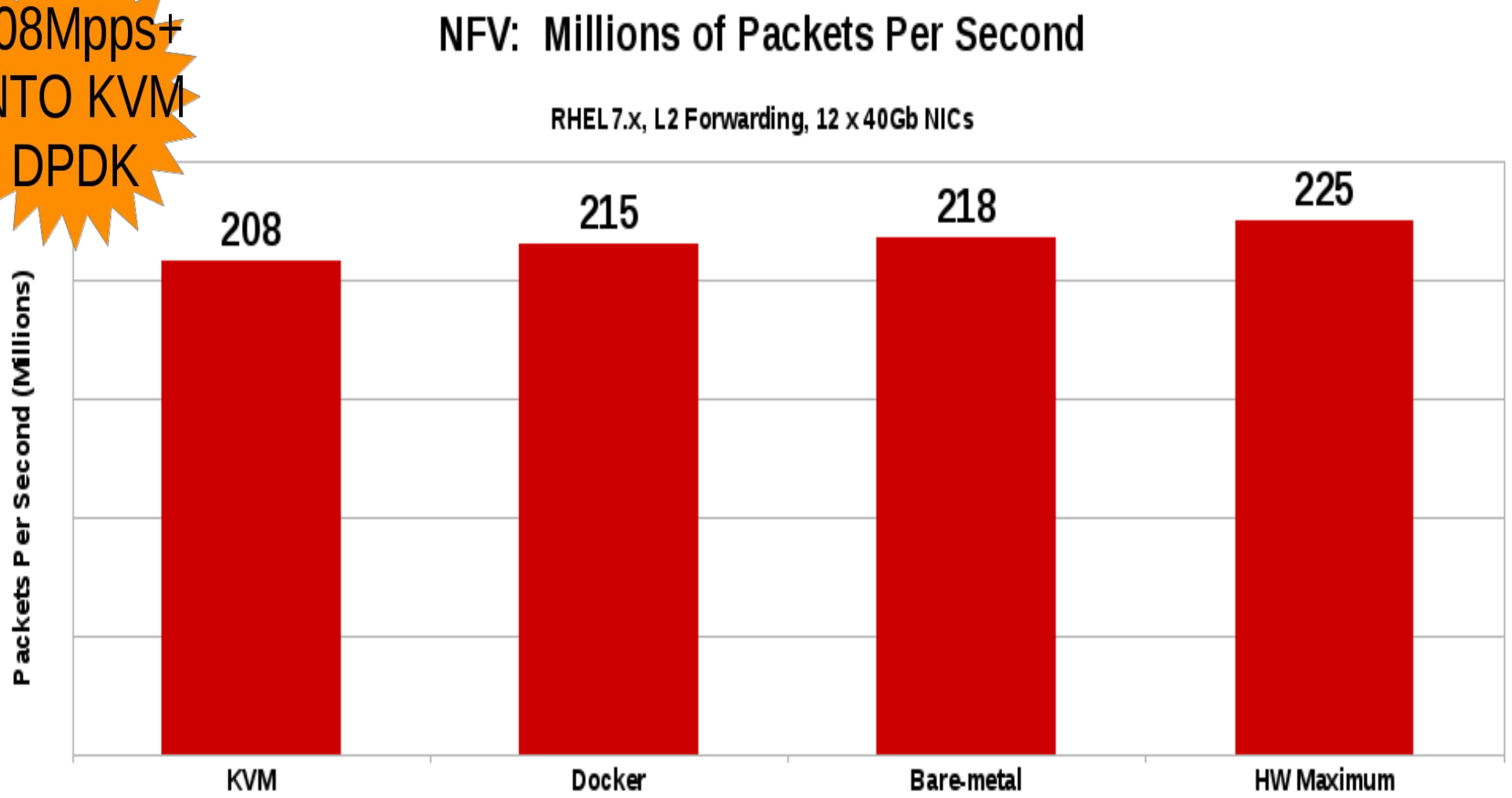


40G Network Data/Tuned Networks

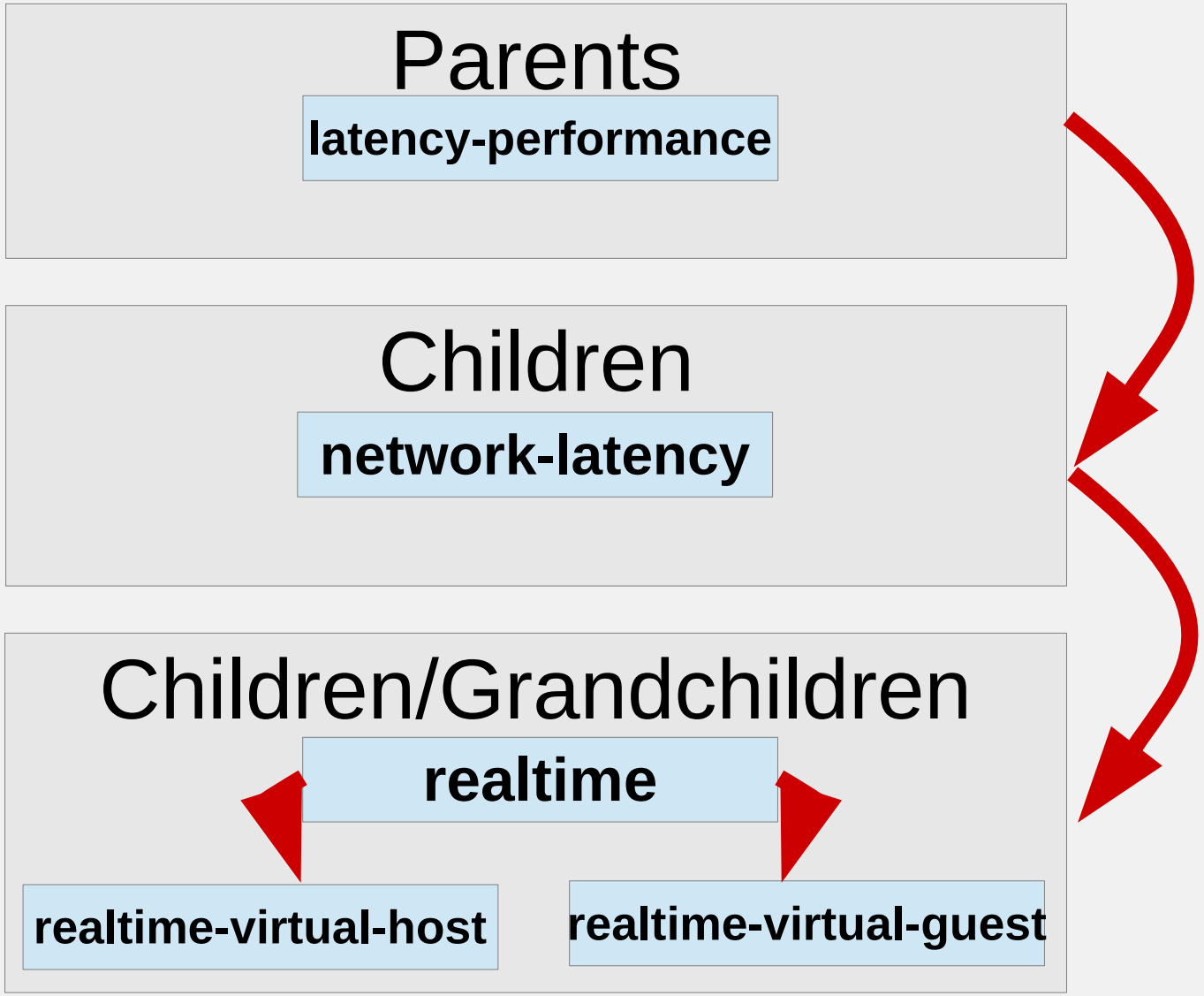


NFV 40G Packets/Sec DPDK (64 byte UDP)

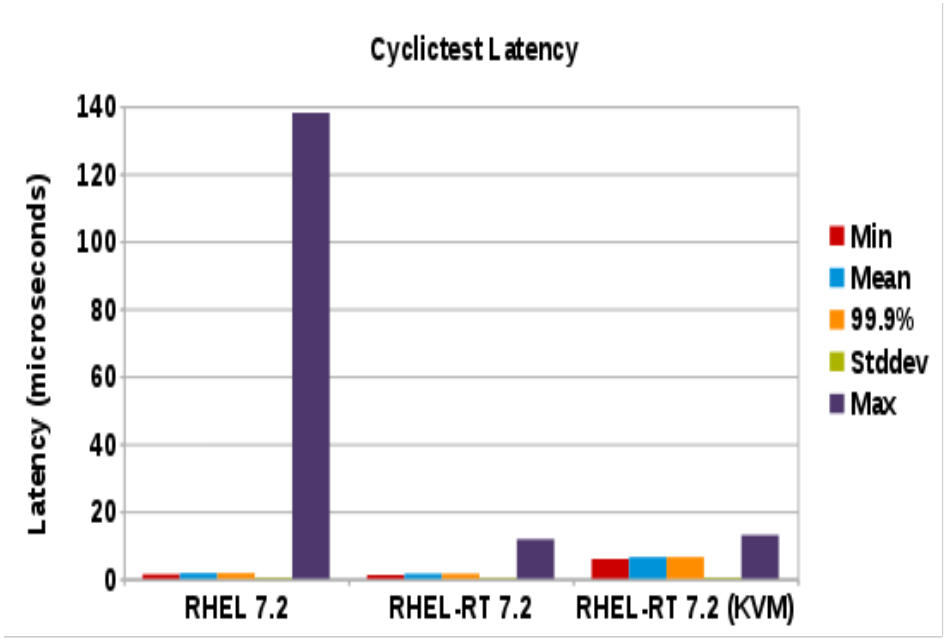
208Mpps+ INTO KVM DPDK



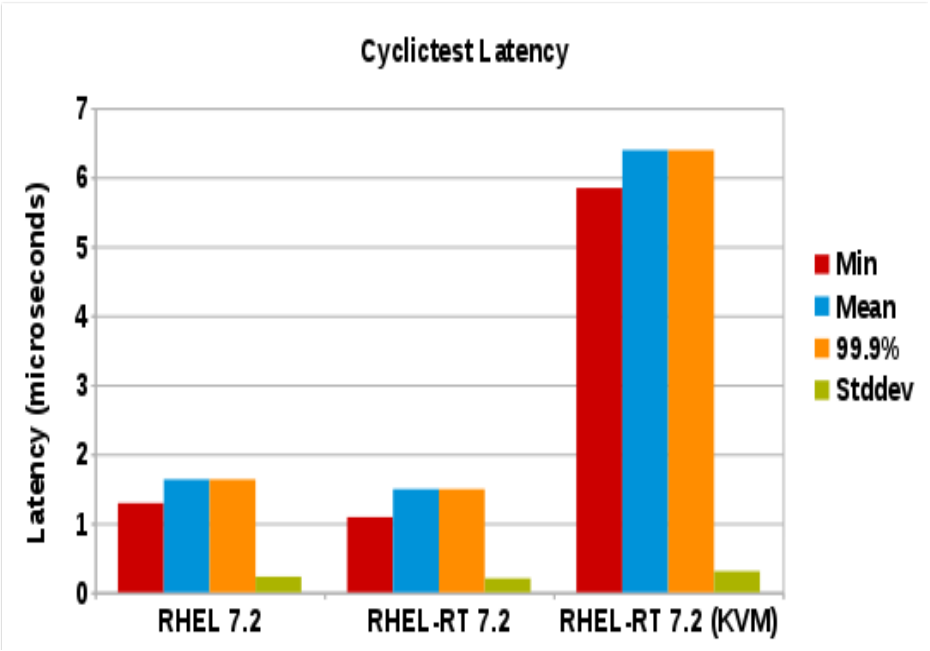
Realtime, Realtime KVM/NFV Tuned Profiles



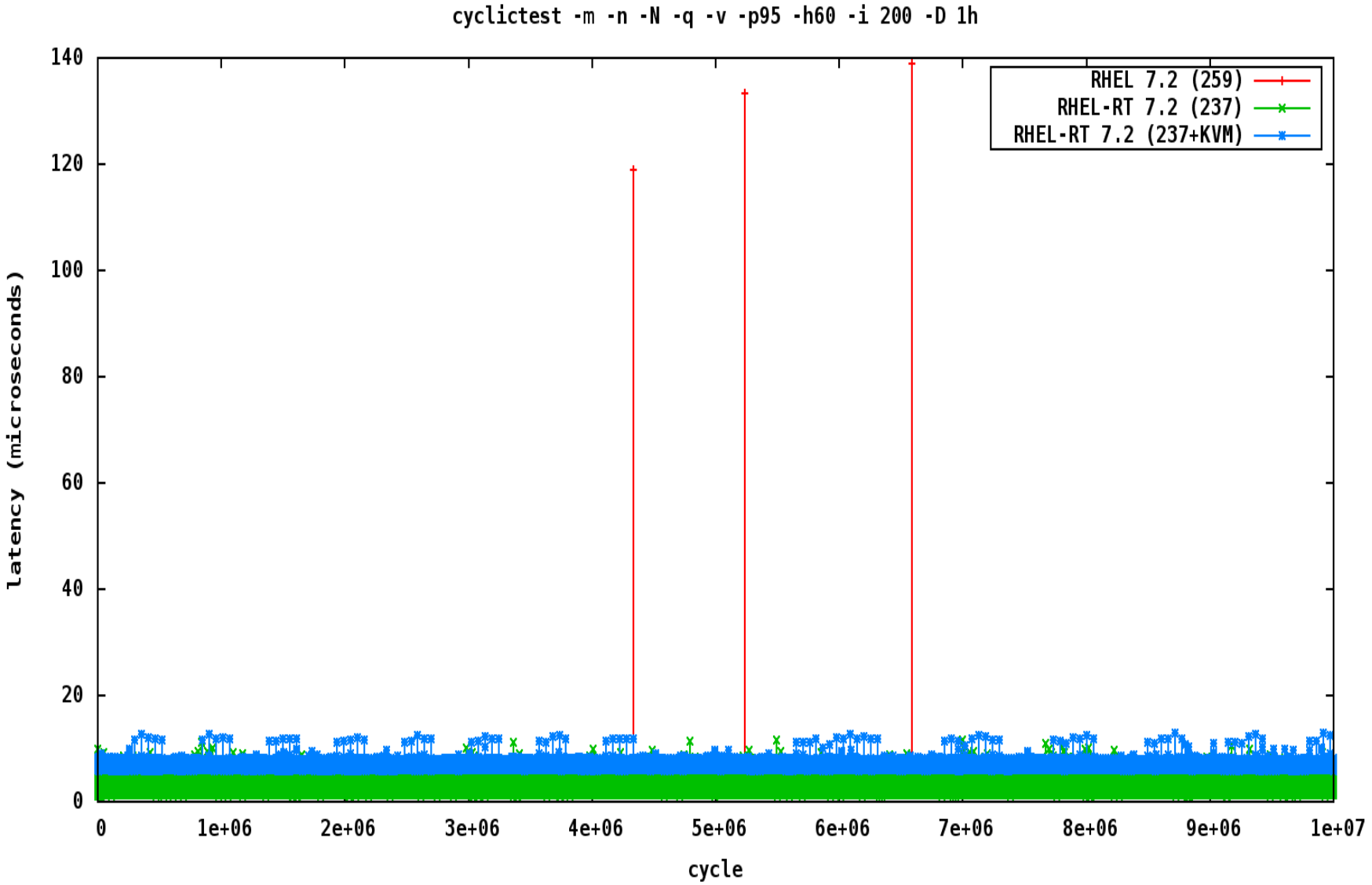
Scheduler Latency (cyclictest)



Remove maxes to zoom in



Realtime Scheduler Latency Jitter Plot



Valuable Links

- Red Hat Performance Tuning Guide
- Red Hat Low Latency Tuning Guide
- Red Hat Virtualization Tuning Guide
- Resource Management and LXC Guide
- Comprehensive Overview of Storage Scalability in Docker
- RHEL Blog / Developer Blog
- Blog: http://www.breakage.org/ or @jeremyeder
- Reference Architectures on RH Portal
  - Ex:Deploying Oracle RAC Database 12c on RHEL 7 - Best Practices
- Key RH Summit Presentation:
  - Performance analysis & tuning of Red Hat Enterprise Linux: Part I
  - Performance analysis & tuning of Red Hat Enterprise Linux: Part II

Questions

Performance Utility Summary

Supportability

- redhat-support-tool
- sos
- kdump
- perf
- psmisc
- strace
- sysstat
- systemtap
- trace-cmd
- Util-linux-ng
- pcp

NUMA

- hwloc
- Intel PCM
- numactl
- numad
- numatop (01.org)

Networking

- dropwatch
- ethtool
- netsniff-ng (EPEL6)
- tcpdump
- wireshark/tshark

Power/Tuning

- cpupowerutils (R6)
- kernel-tools (R7)
- powertop
- tuna
- tuned

Storage

- blktrace
- iotop
- iostat