Red Hat Customer Convergence

#rhconvergence

Performance Tuning and Analysis of Red Hat Enterprise Linux 7

Joe Mario
Senior Principal Software Engineer
Feb 2015

Agenda

- Red Hat Performance Team
- RHEL7.1, RHEL6.6 items
- Tuned
- NUMA Tuning
- perf
- Linux Containers

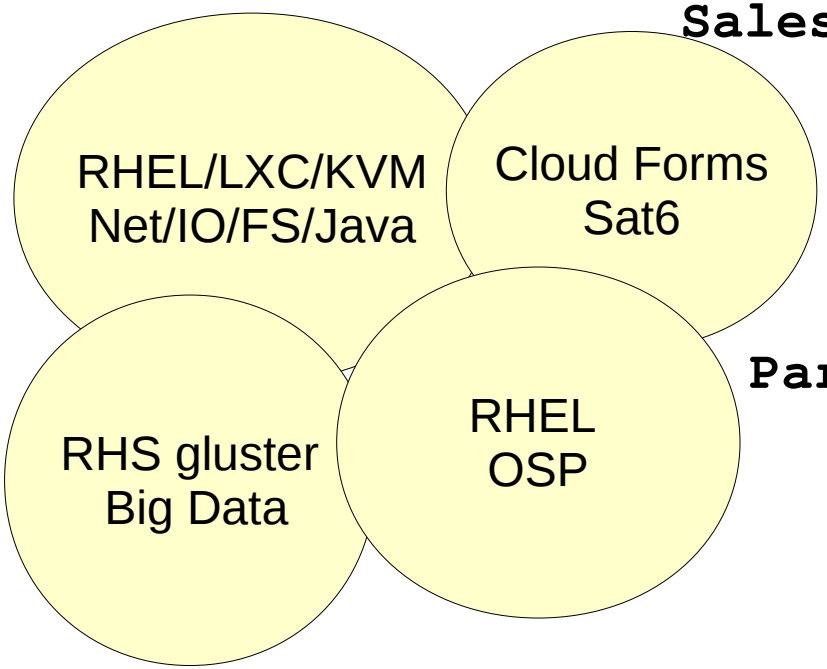
Red Hat Performance Engineering

Benchmarking SPEC/TPC/STAC

Field Support

ISV / Applications

Sales / Field Consulting



New Technology
cpu/nic/ssd

Partner / HW OEM / Chips

Competitive Analysis

Reference Architectures
Perf Briefs / Guides

RHEL Platform(s) Performance Coverage

Benchmarks – code path coverage

- CPU – linpack, lmbench
- Memory – lmbench, McCalpin STREAM
- Disk IO – iozone, fio – SCSI, FC, iSCSI
- Filesystems – iozone, ext3/4, xfs, gfs2, gluster
- Networks – netperf – 10/40Gbit, Infiniband/RoCE, Bypass
- Bare Metal, RHEL6/7 KVM
- White box AMD/Intel, with our OEM partners

Application Performance

- Linpack MPI, SPEC CPU, SPECjbb 05/13
- AIM 7 – shared, filesystem, db, compute
- Database: DB2, Oracle 11/12, Sybase 15.x , MySQL, MariaDB, PostgreSQL, Mongo
- OLTP – Bare Metal, KVM, RHEV-M clusters – TPC-C/Virt
- DSS – Bare Metal, KVM, RHEV-M, IQ, TPC-H/Virt
- SPECsfs NFS
- SAP – SLCS, SD
- STAC = FSI (STAC-N)

RHEL Performance Evolution

Red Hat Enterprise Linux Performance Evolution

RHEL5

Static Hugepages

Ktune – on/off

CPU Affinity (taskset)

NUMA Pinning (numactl)

irqbalance

Red Hat Enterprise Linux Performance Evolution

RHEL5	RHEL6
Static Hugepages	Transparent Hugepages
Ktune – on/off	Tuned – choose profile
CPU Affinity (taskset)	CPU Affinity (ts/numactl)
NUMA Pinning (numactl)	NUMAD – userspace tool
irqbalance	irqbalance – NUMA enhanced

Red Hat Enterprise Linux Performance Evolution

RHEL5	RHEL6	RHEL7
Static Hugepages	Transparent Hugepages	Transparent Hugepages
Ktune – on/off	Tuned – choose profile	Tuned – throughput-performance (default)
CPU Affinity (taskset)	CPU Affinity (ts/numactl)	CPU Affinity (ts/numactl)
NUMA Pinning (numactl)	NUMAD – userspace tool	Autonuma-Balance
irqbalance	irqbalance – NUMA enhanced	irqbalance – NUMA enhanced

tuned

What is “tuned” ?

Tuning profile delivery mechanism

Red Hat ships *tuned profiles* that improve performance for many workloads...hopefully yours!

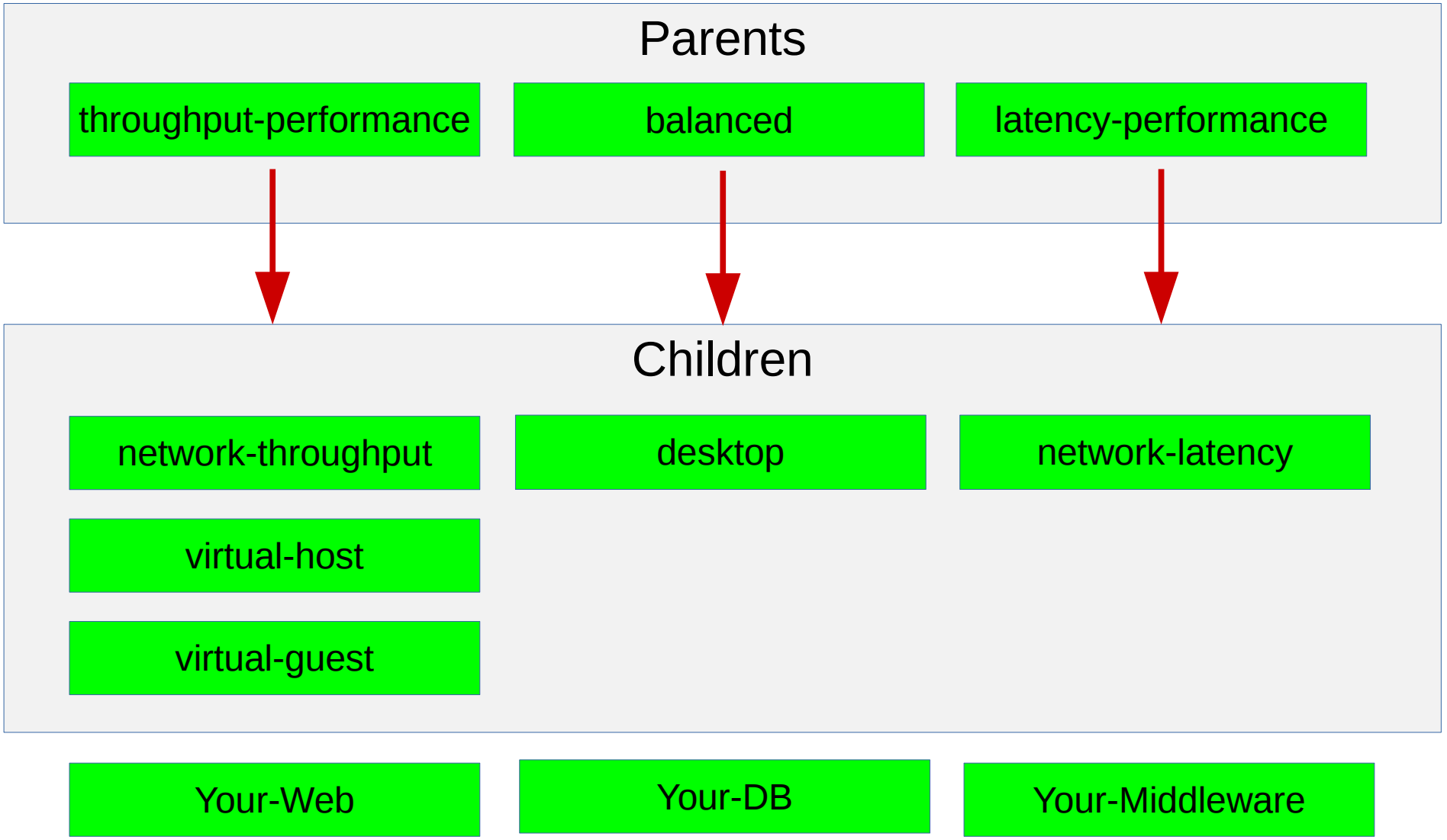
Tuned: Updates for RHEL7

- Re-written for maintainability and extensibility.
  - Configuration is now consolidated a single tuned.conf file
  - Optional hook/callout capability
  - Adds concept of Inheritance (just like httpd.conf)
  - Profiles updated for RHEL7 features and characteristics

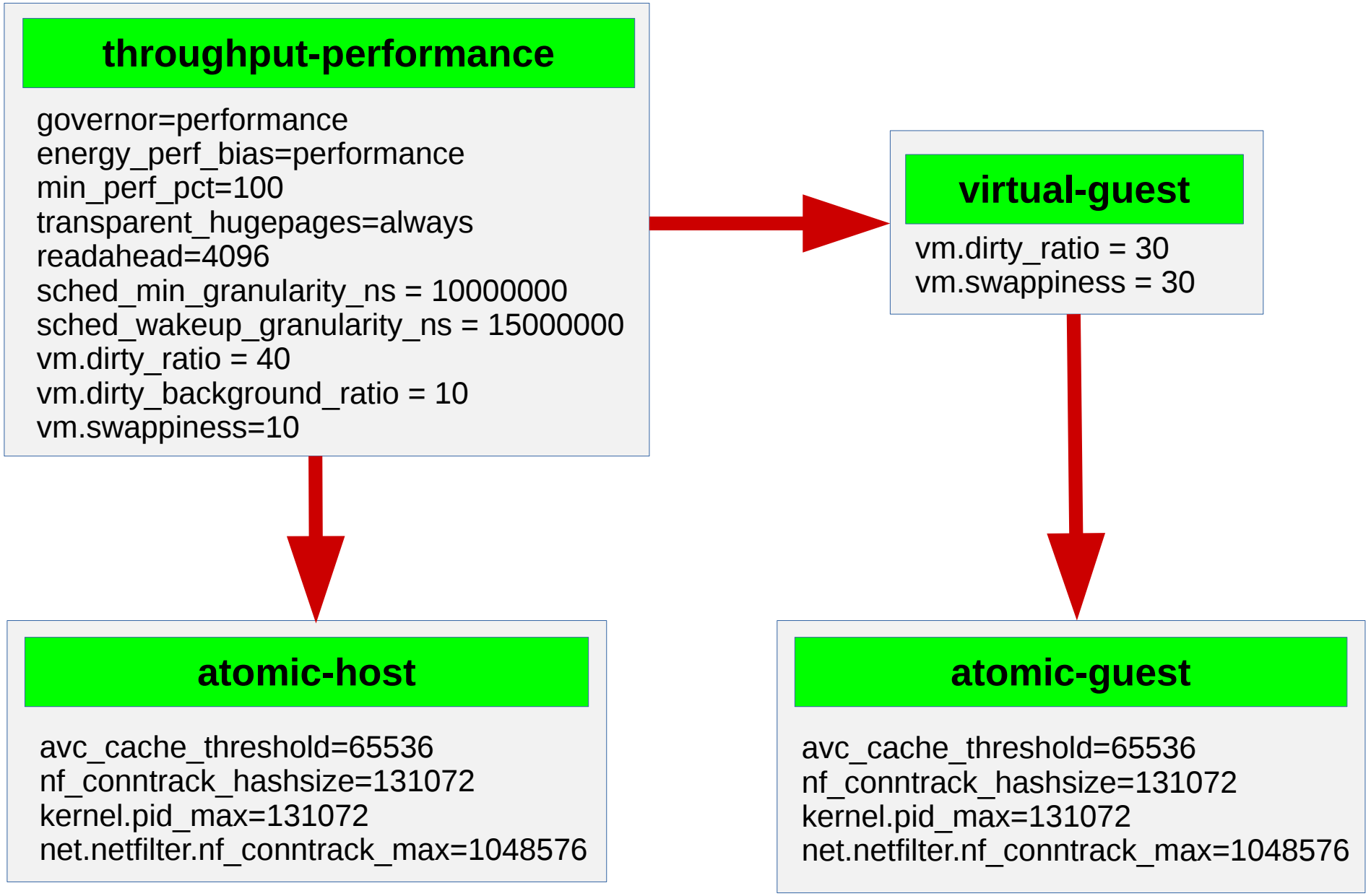
RHEL “tuned” package

```
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: Profile Inheritance



Tuned: RHEL Atomic Profile Inheritance



NoFile=1048576

NPROC=1048576

Tuned Profiles throughout Red Hat's Product Line

RHEL7 Desktop/Workstation

balanced

RHEL7 Server/HPC

throughput-performance

RHEL6/7 KVM Host, Guest

virtual-host/guest

RHEV

virtual-host

Red Hat Storage

rhs-high-throughput, virt

RHEL OSP (compute node)

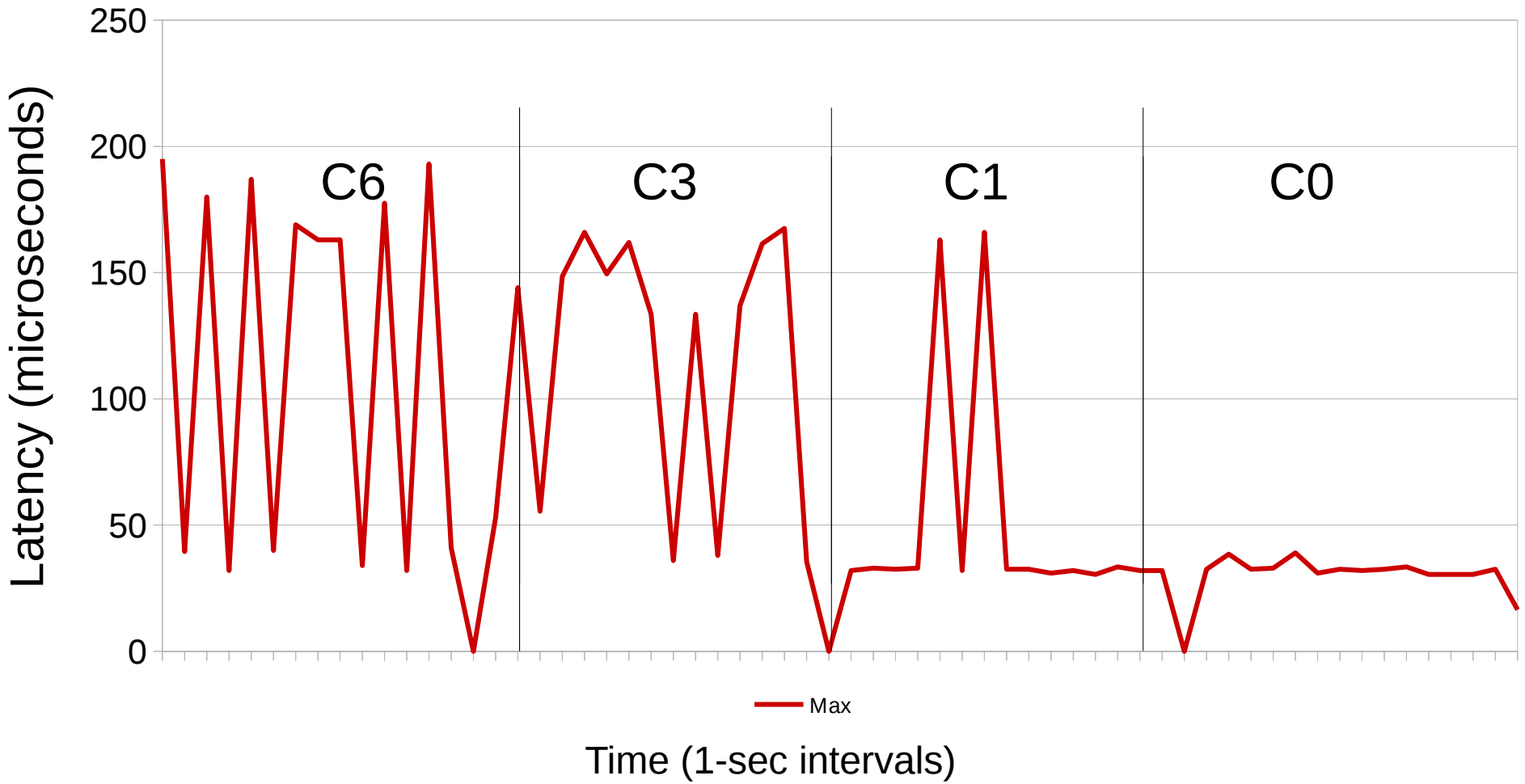
virtual-host

RHEL Atomic

atomic-host/guest

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
```

turbostat shows P/C-states on Intel CPUs

turbostat begins shipping in RHEL6.4, cpupowerutils package

Default

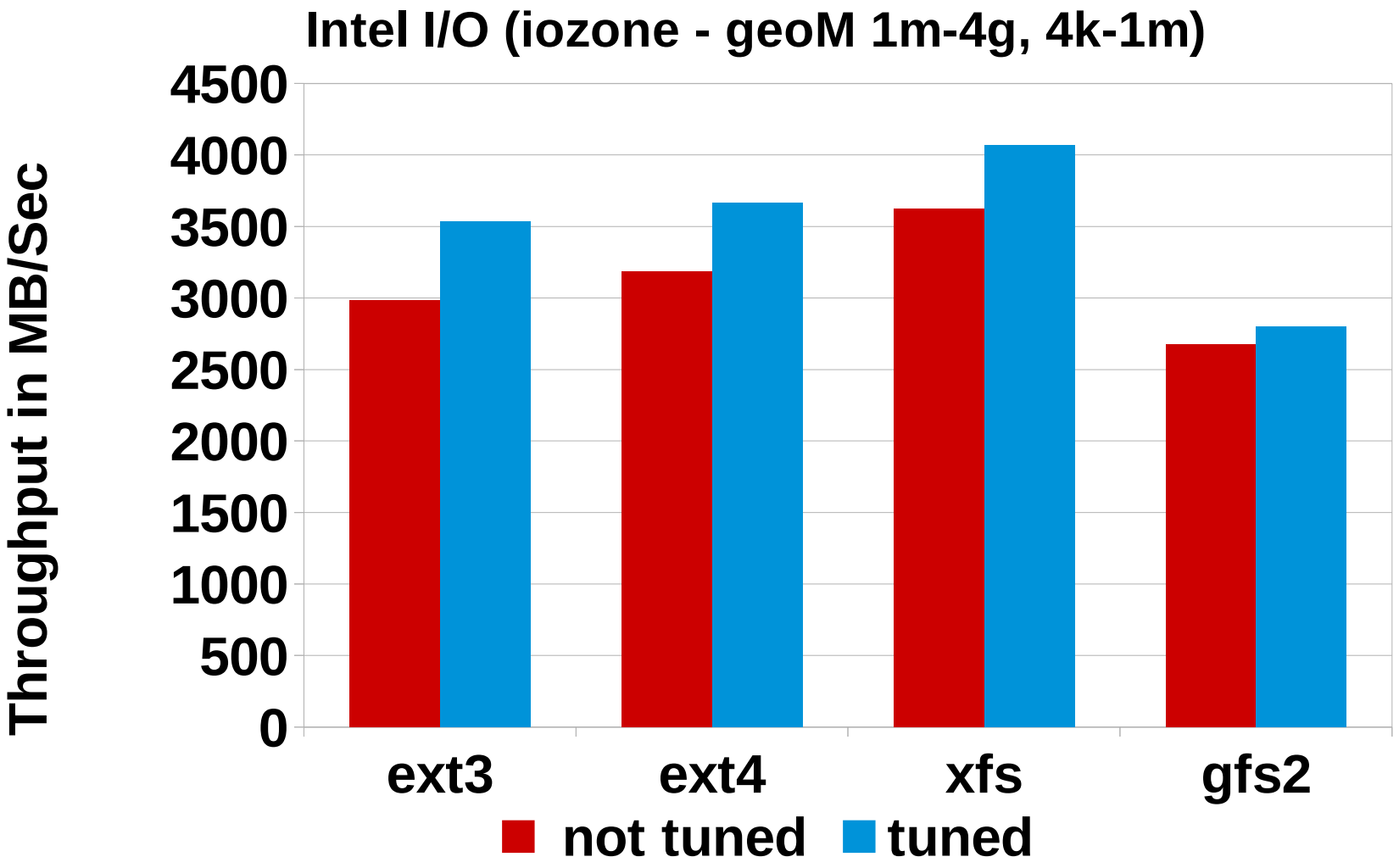
pk	cor	CPU	%c0	GHz	TSC	%c1	%c3	%c6	%c7
0	0	0	0.24	2.93	2.88	5.72	1.32	0.00	92.72
0	1	1	2.54	3.03	2.88	3.13	0.15	0.00	94.18
0	2	2	2.29	3.08	2.88	1.47	0.00	0.00	96.25
0	3	3	1.75	1.75	2.88	1.21	0.47	0.12	96.44

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

Tuned: Storage Performance Boost

RHEL7 RC1 3.10 File System In Cache Perf



Larger is better

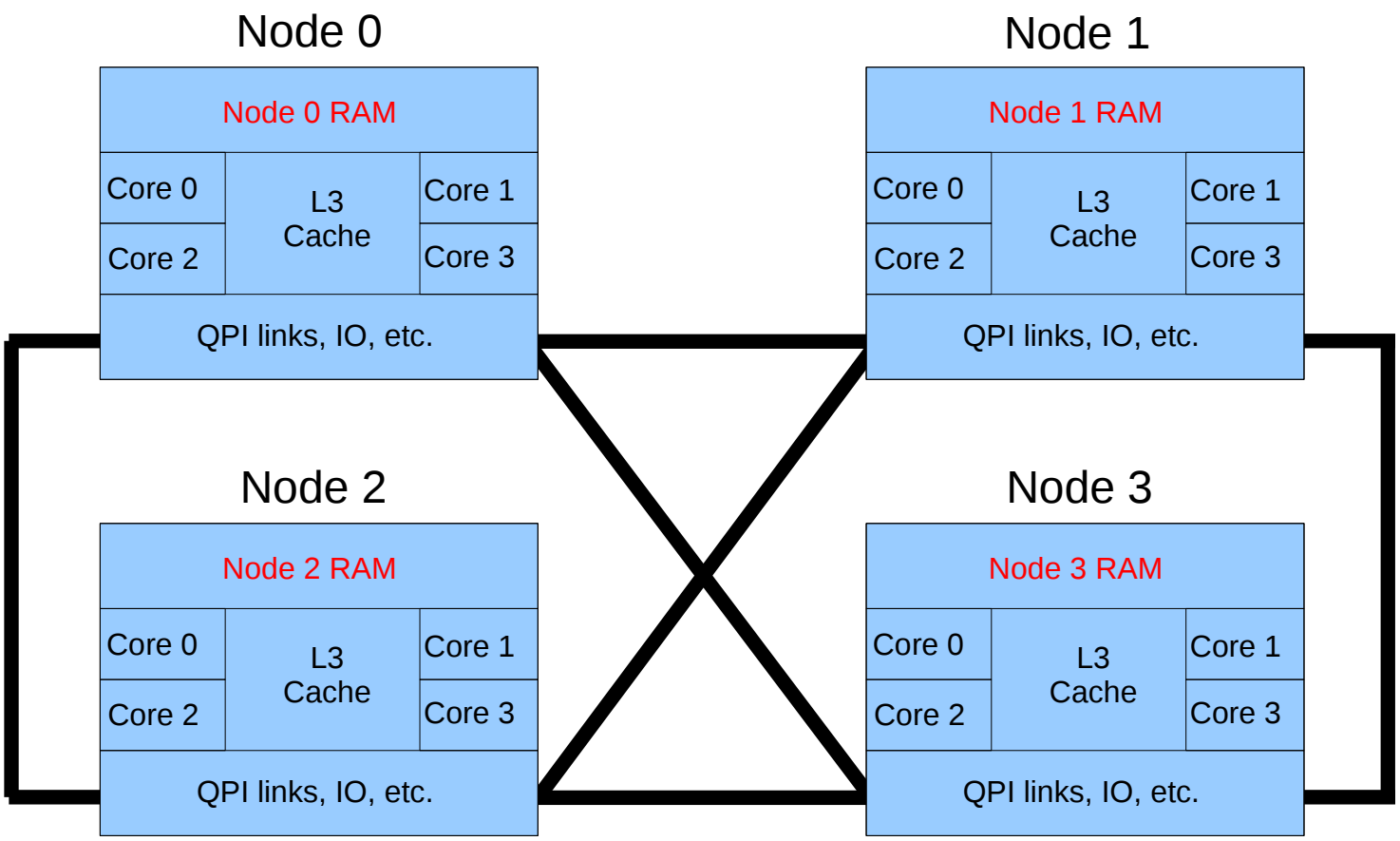
NUMA Tuning

Discovery

What is NUMA?

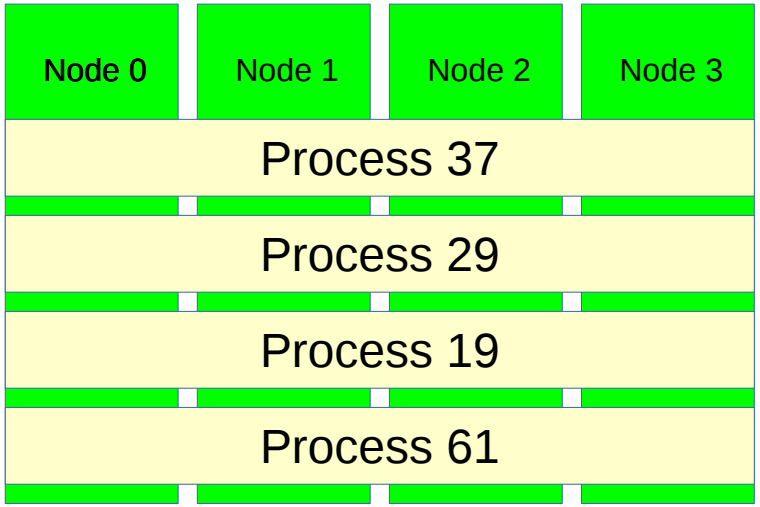
- Non Uniform Memory Access
- A result of making bigger systems more scalable by distributing system memory near individual CPUs....
- Practically all multi-socket systems have NUMA
  - Most servers have 1 NUMA node / socket
  - Recent AMD systems may have 2 NUMA nodes / socket

Typical NUMA System

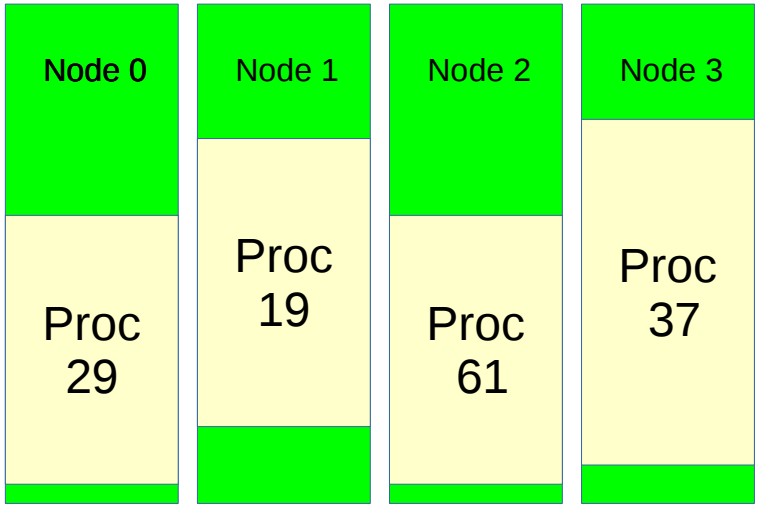


Goal: Align process memory and CPU threads within nodes

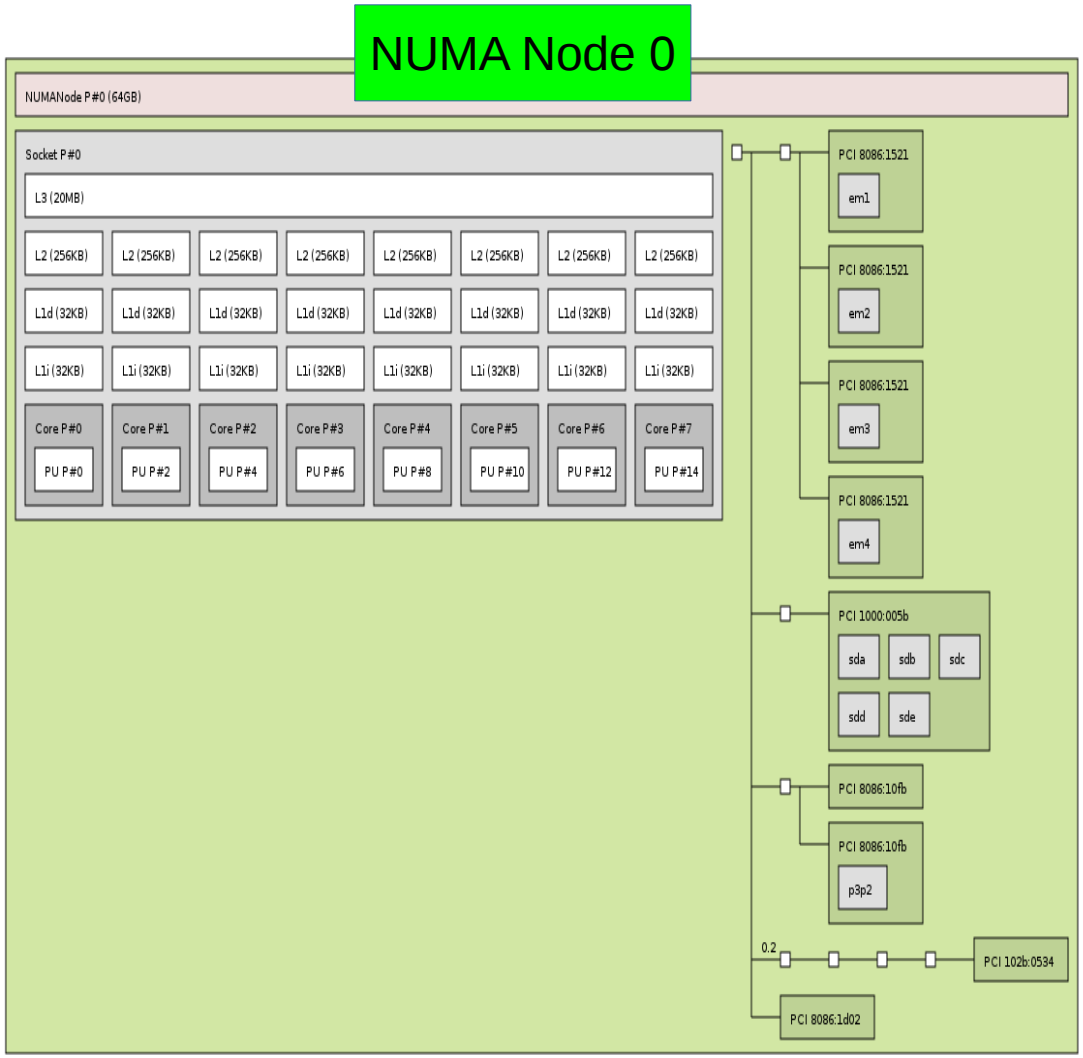
Before: processes use cpus & memory from multiple nodes.

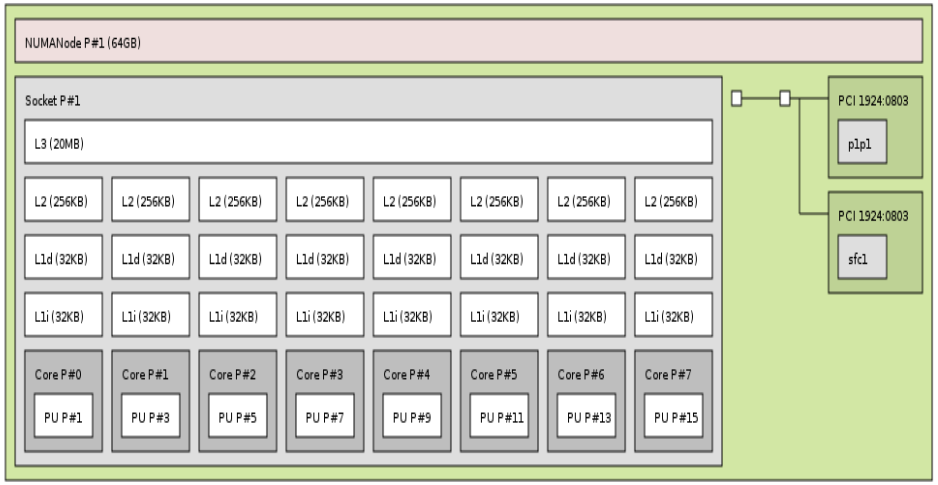


After: processes have more “localized” cpu & memory usage.



Know your hardware: lstopo

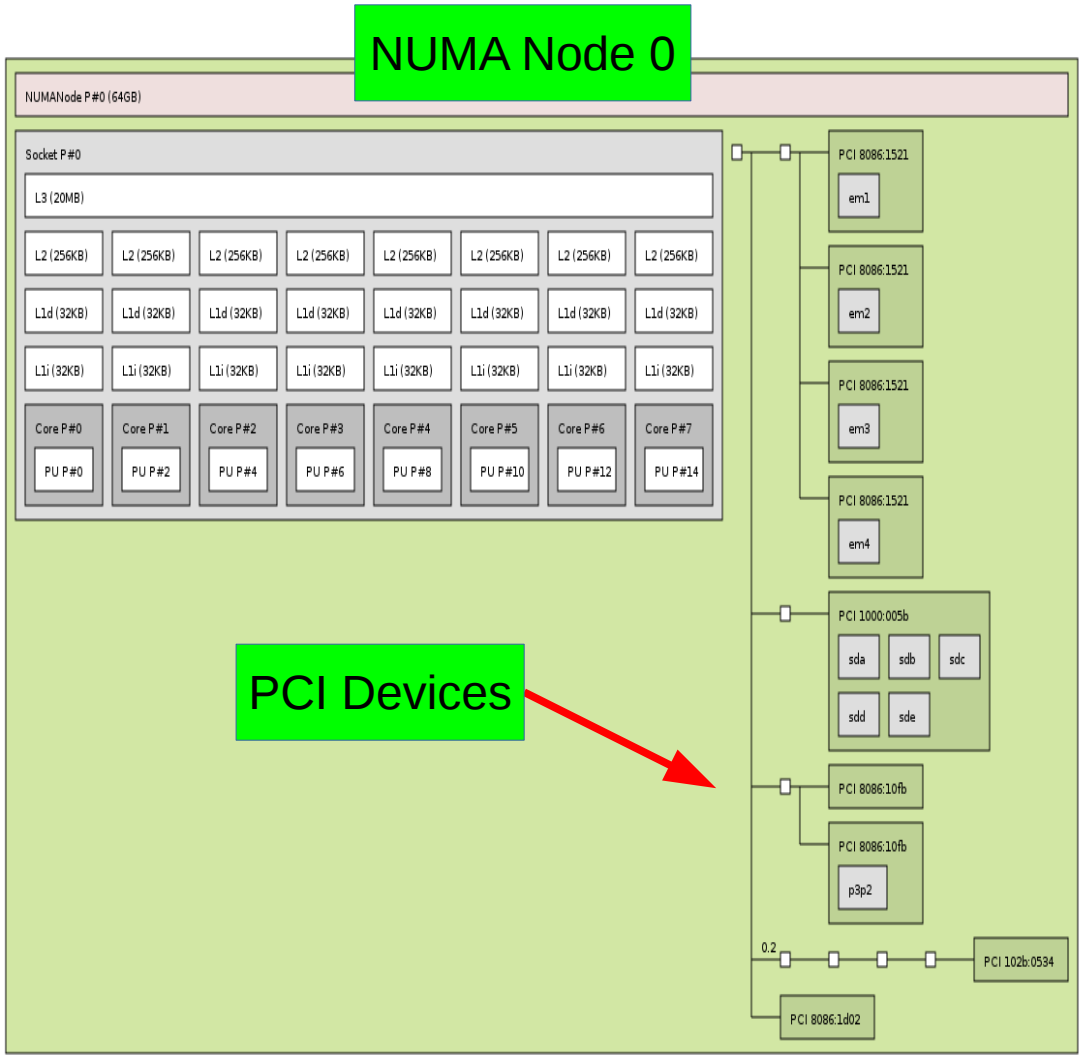


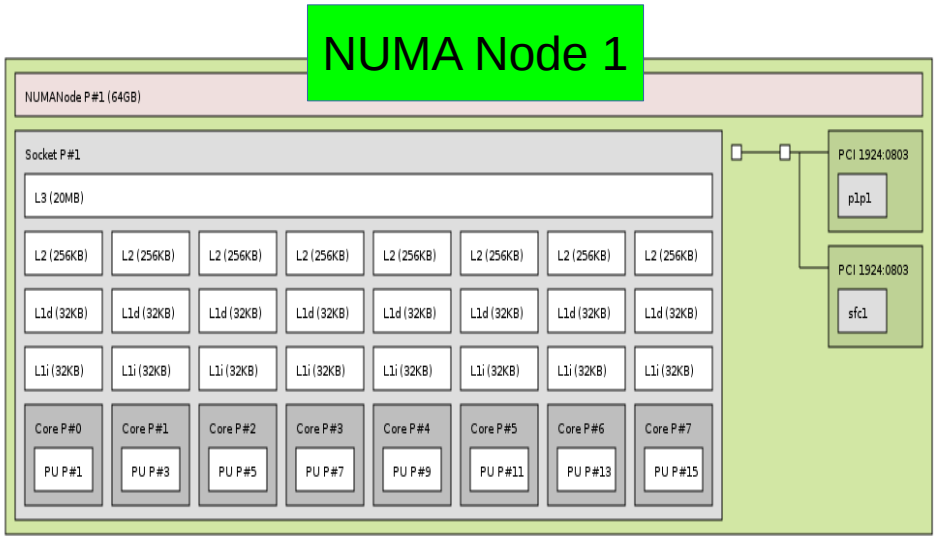


How can I visualize my system's NUMA topology in Red Hat Enterprise Linux?

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

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

NUMA layout via numactl

```
# numactl --hardware
available: 4 nodes (0-3)
node 0 cpus: 0 4 8 12 16 20 24 28 32 36
node 0 size: 65415 MB
node 0 free: 63482 MB
node 1 cpus: 2 6 10 14 18 22 26 30 34 38
node 1 size: 65536 MB
node 1 free: 63968 MB
node 2 cpus: 1 5 9 13 17 21 25 29 33 37
node 2 size: 65536 MB
node 2 free: 63897 MB
node 3 cpus: 3 7 11 15 19 23 27 31 35 39
node 3 size: 65536 MB
node 3 free: 63971 MB
node distances:
node   0   1   2   3
  0:  10  21  21  21
  1:  21  10  21  21
  2:  21  21  10  21
  3:  21  21  21  10
```

NUMA Topology and PCI Bus

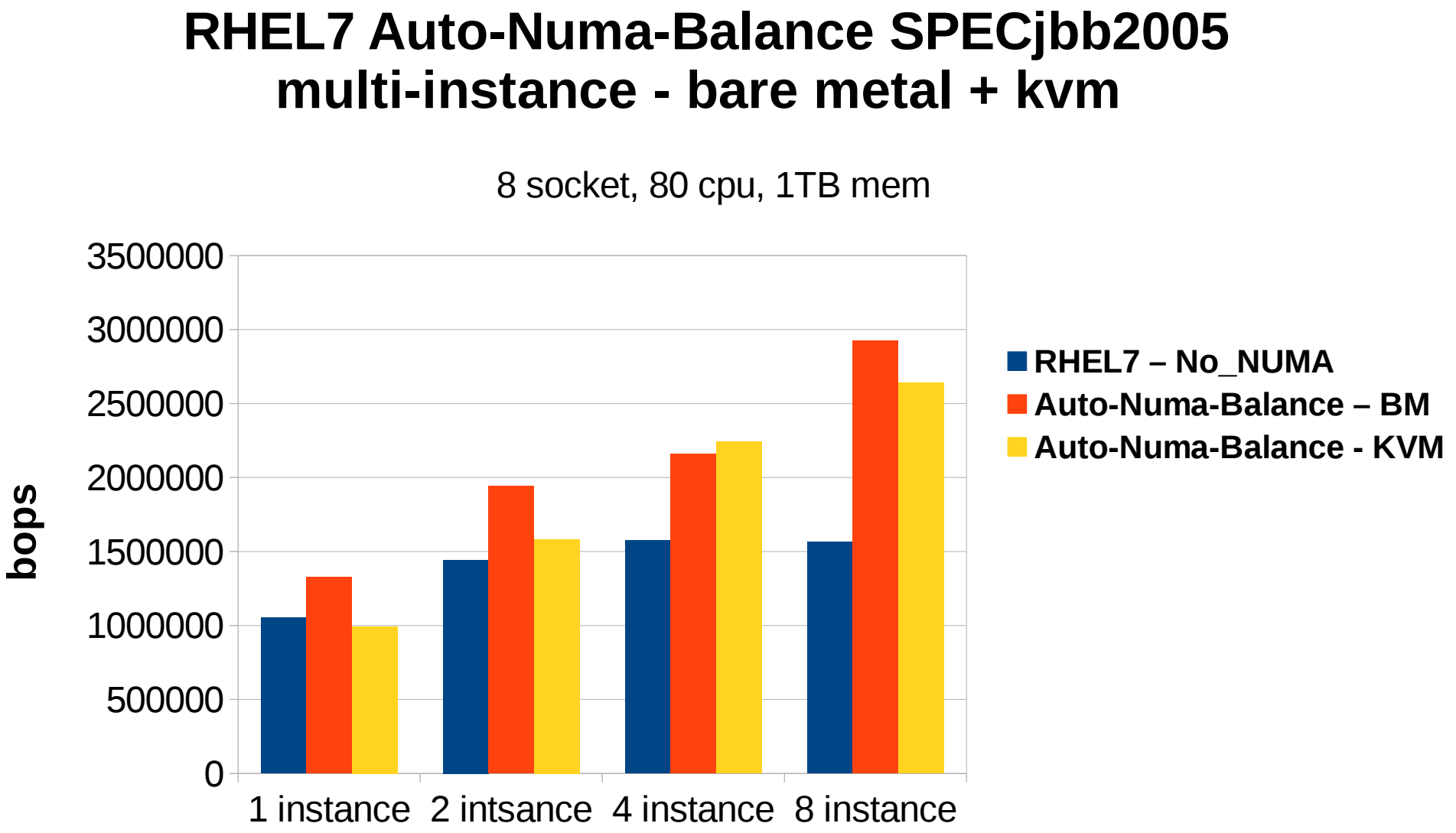
- Install adapters “close” to the CPU that will run the performance critical application.

- When BIOS reports locality, irqbalance handles NUMA/IRQ affinity automatically.

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}

NUMA Performance – SPECjbb2005 on DL980 Westmere EX



NUMA Tools: numastat

Completely rewritten for RHEL6.4

Per-node /proc/meminfo

Backwards compatible

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 – 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
```

Overview of Performance Analysis Utilities

perf

perf

Userspace tool to read CPU counters and kernel tracepoints

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
```

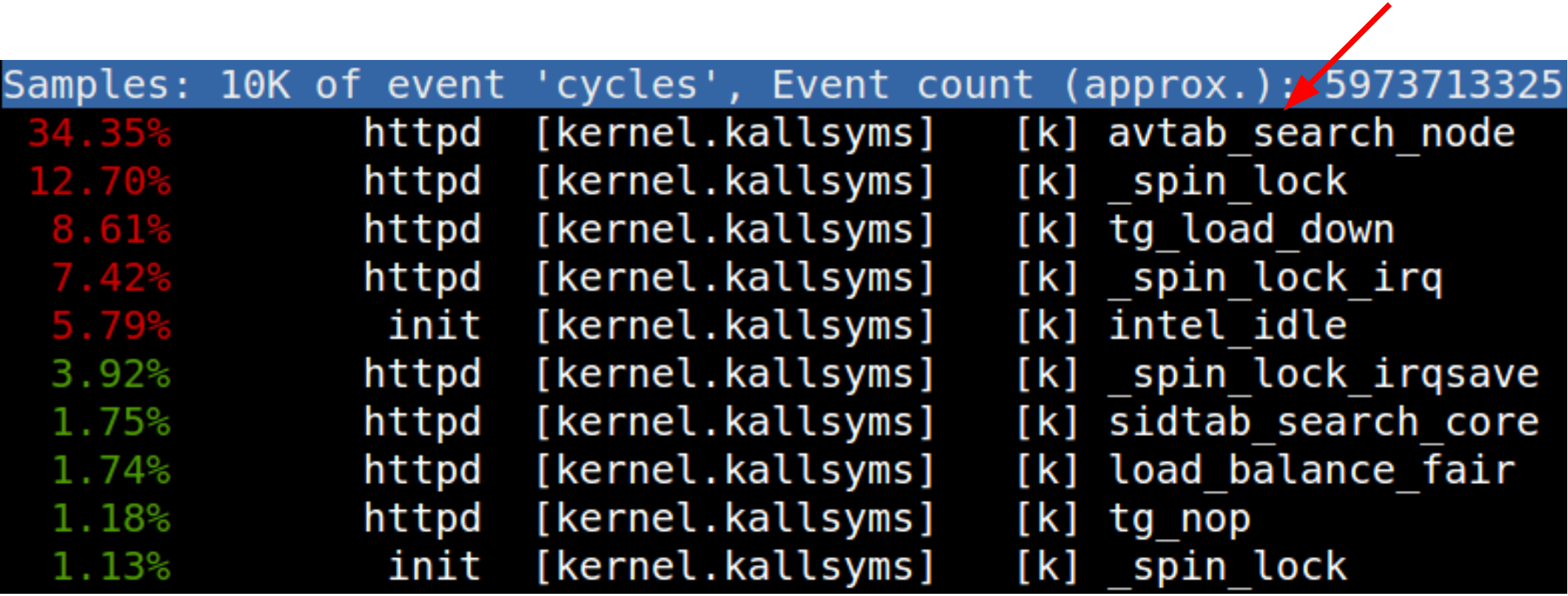
perf list

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]
```

perf top

System-wide 'top' view of busy functions



perf record

- *Record system-wide (-a)*

perf record

- Record system-wide (-a)
- *Or record a single command*

perf record

- Record system-wide (-a)
- Or record a single command
- *Or record an existing process (-p)*

perf record

- Record system-wide (-a)
- Or record a single command
- Or record an existing process (-p)
- *Or add call-chain recording (-g)*

perf record

- Record system-wide (-a)
- Or record a single command
- Or record an existing process (-p)
- Or add call-chain recording (-g)
- *Or only record specific events (-e)*

perf record

- Record system-wide (-a)

perf record -a dd if=/dev/zero of=test bs=1M count=1000 conv=fdatasync oflag=direct
1000+0 records in
1000+0 records out
1048576000 bytes (1.0 GB) copied, 6.26617 s, 167 MB/s
[perf record: Woken up 1 times to write data]
[perf record: Captured and wrote 0.305 MB perf.data (~13311 samples)]

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 - analyzing the raw data for mem accesses.

(c2c output, more later)

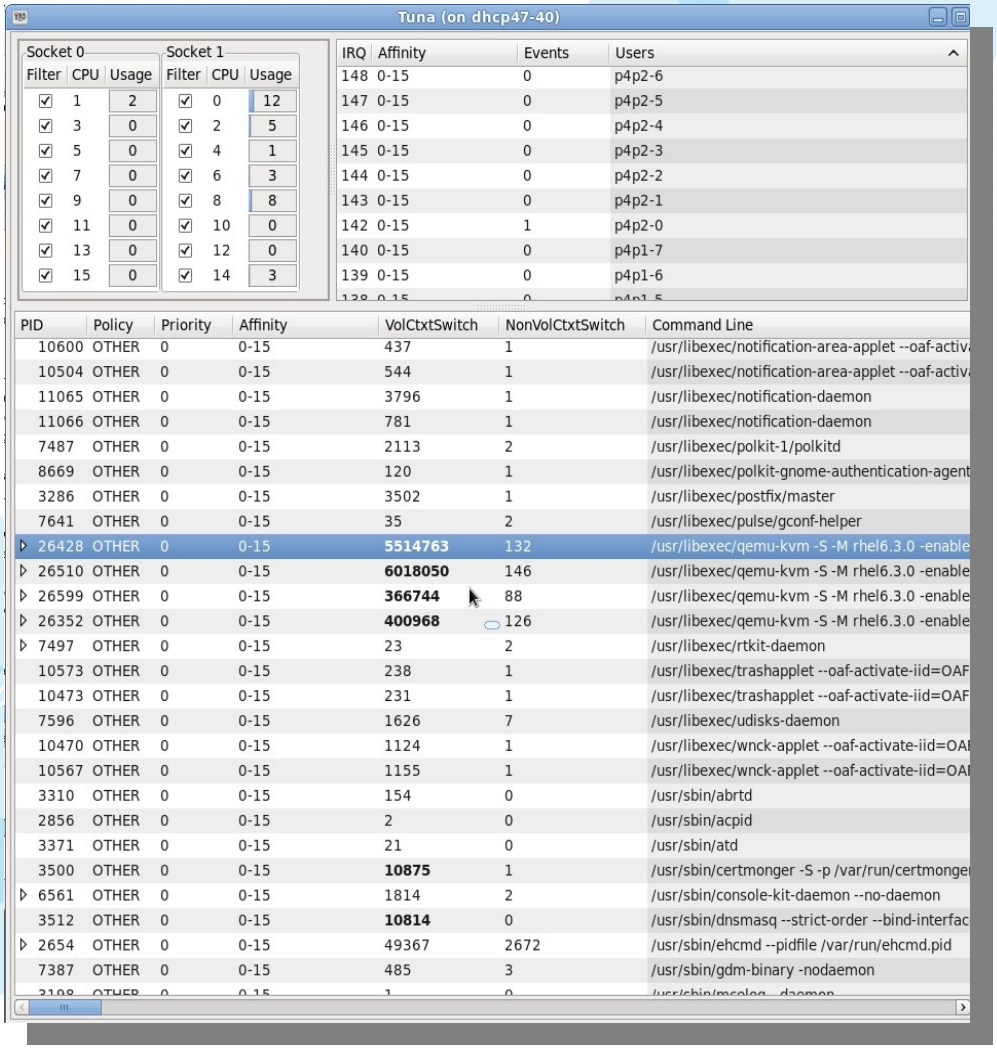
```
      Count  Response Type
       -------  ---------------
     152729 [LOAD,L2,HIT,SNP NONE]                          // Hit in the L2 cache
     143821 [LOAD,LFB,HIT,SNP NONE]                         // "In flight" loads to Line Fill Buffer
     116187 [LOAD,RMT_RAM,HIT,SNP NONE,SNP MISS]            // Hit in a remote node's ram
      89248 [LOAD,L1,HIT,SNP NONE]                          // Hit in the L1 cache
      40723 [LOAD,LCL_RAM,HIT,SNP NONE,SNP MISS]            // Hit in local node's ram
      40614 [LOAD,LCL_LLC,HIT,SNP NONE]                     // Hit in local last level cache
        826 [STORE,L1,HIT]
        769 [STORE,L1,MISS]
        402 [LOAD,LCL_LLC,HIT,SNP MISS]
        279 [LOAD,LCL_LLC,MISS,SNP NA]
        185 [LOAD,RMT_LLC,HIT,SNP HIT]
        159 [LOAD,LCL_LLC,HIT,SNP HIT]
        134 [LOAD,RMT_LLC,HIT,SNP HITM]    // False Sharing across numa nodes!
        100 [LOAD,UNCACHED,HIT,SNP NONE]
         40 [LOAD,LCL_LLC,HIT,SNP HITM]
         27 [LOAD,L1,HIT,SNP NONE,LOCKED]
          4 [LOAD,RMT_RAM,HIT,SNP NONE,SNP MISS,LOCKED]
```

Load latencies – key info to know, not shown here, but in the raw data

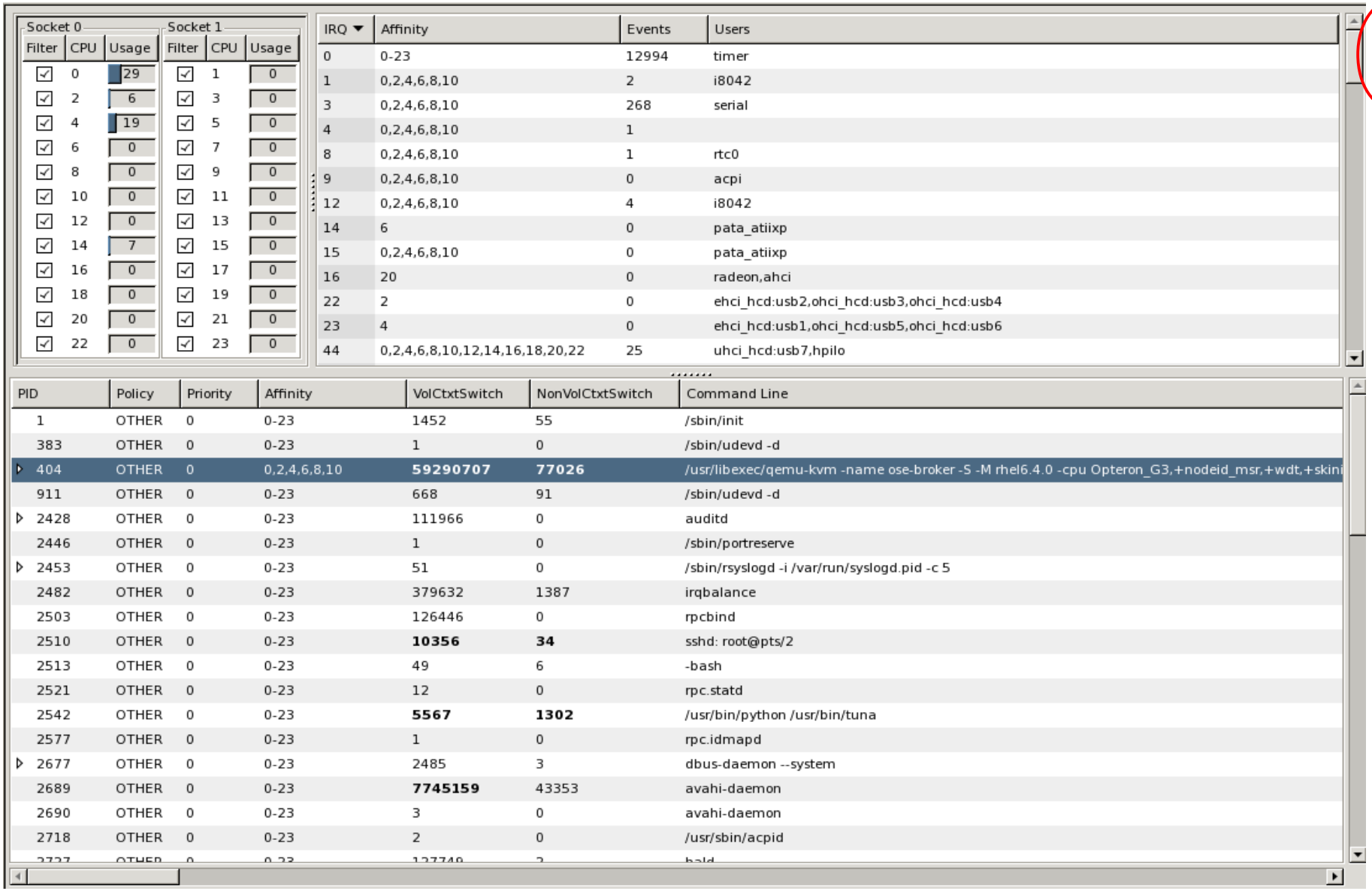
Tuna

System Tuning Tool - tuna

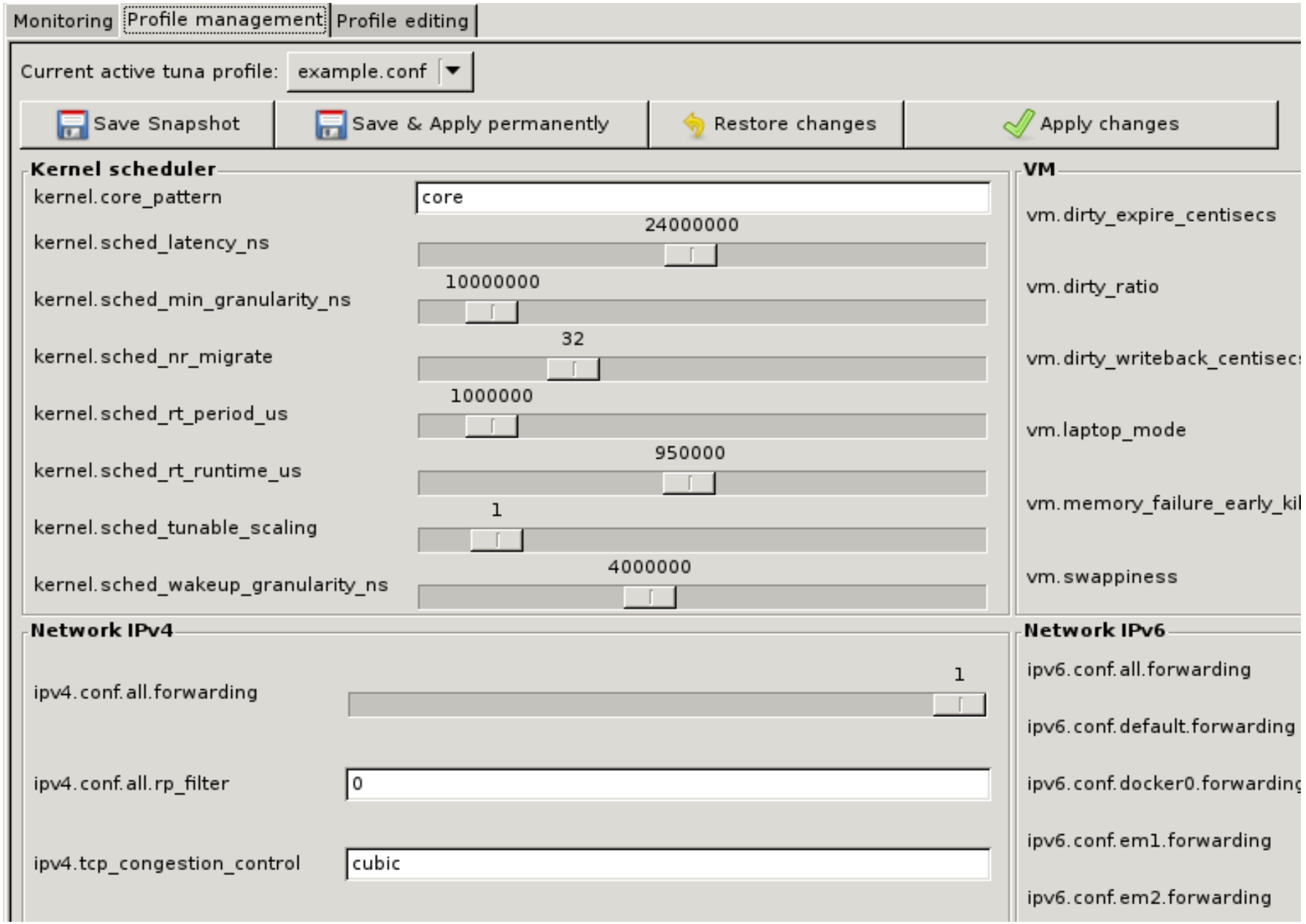
- Tool for fine grained control
- Display applications / processes
- Displays CPU enumeration
- Socket (useful for NUMA tuning)
- Dynamic control of tuning
  - Process affinity
  - Parent & threads
  - Scheduling policy
  - Device IRQ priorities, etc



Tuna (RHEL6/7)

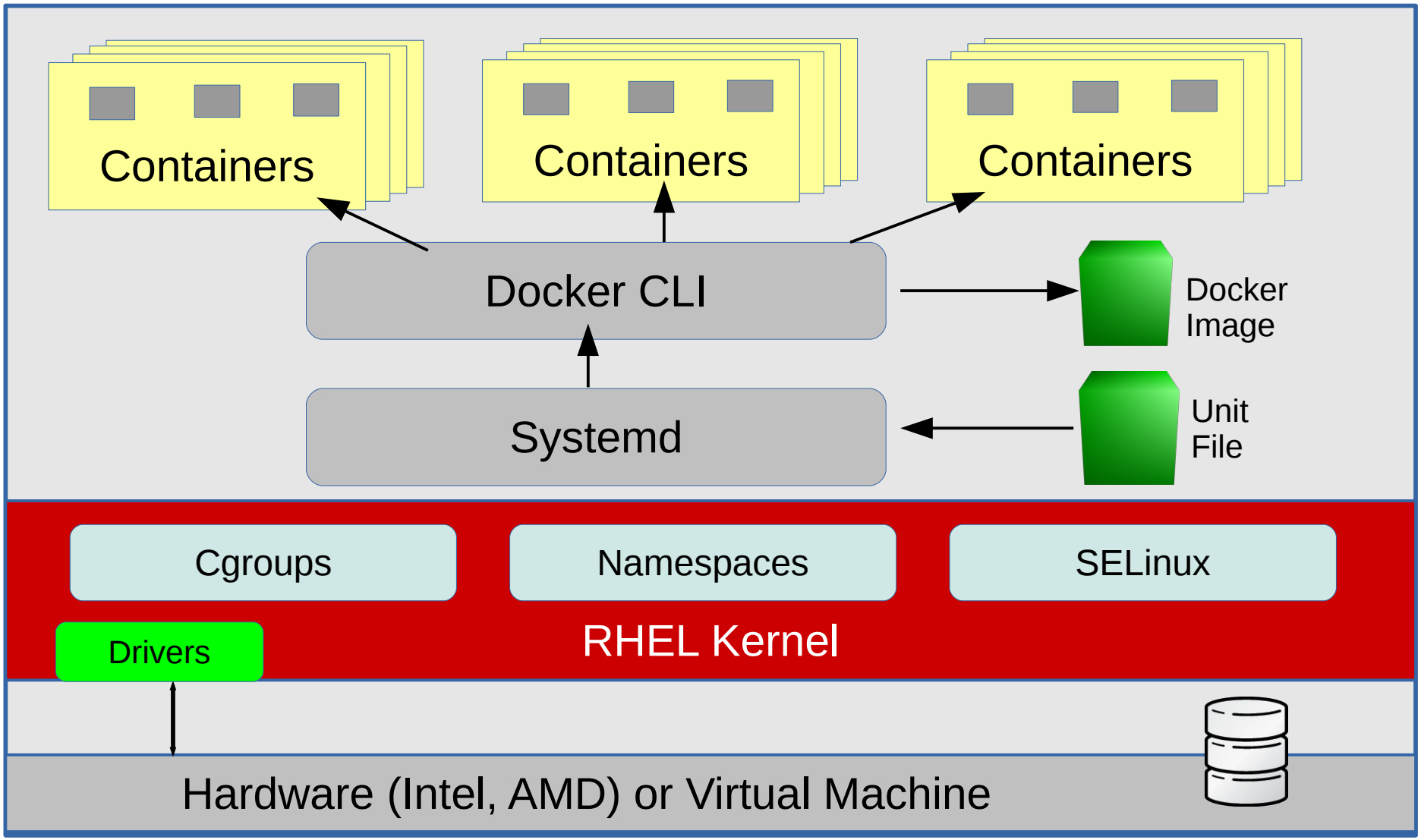


Tuna GUI Capabilities Updated for RHEL7

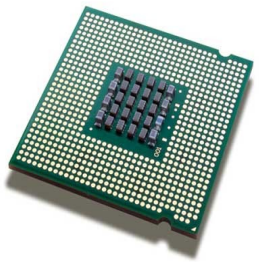


Containers

RHEL 7 Containers Architecture



Performance food groups – still relevant in containers



CPU

Memory

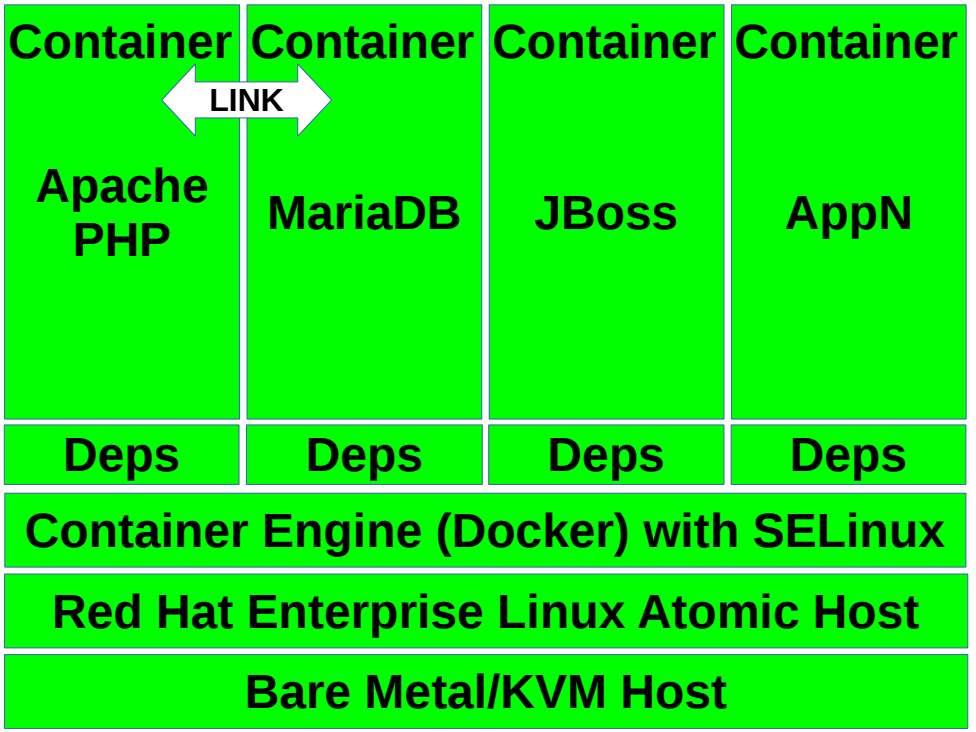
Storage

Networking

- numa_balancing
- irqbalance
- Storage/Networking passthrough
- Short circuits via shared namespaces
- Network and Storage tuning still applies
- Power management

Density

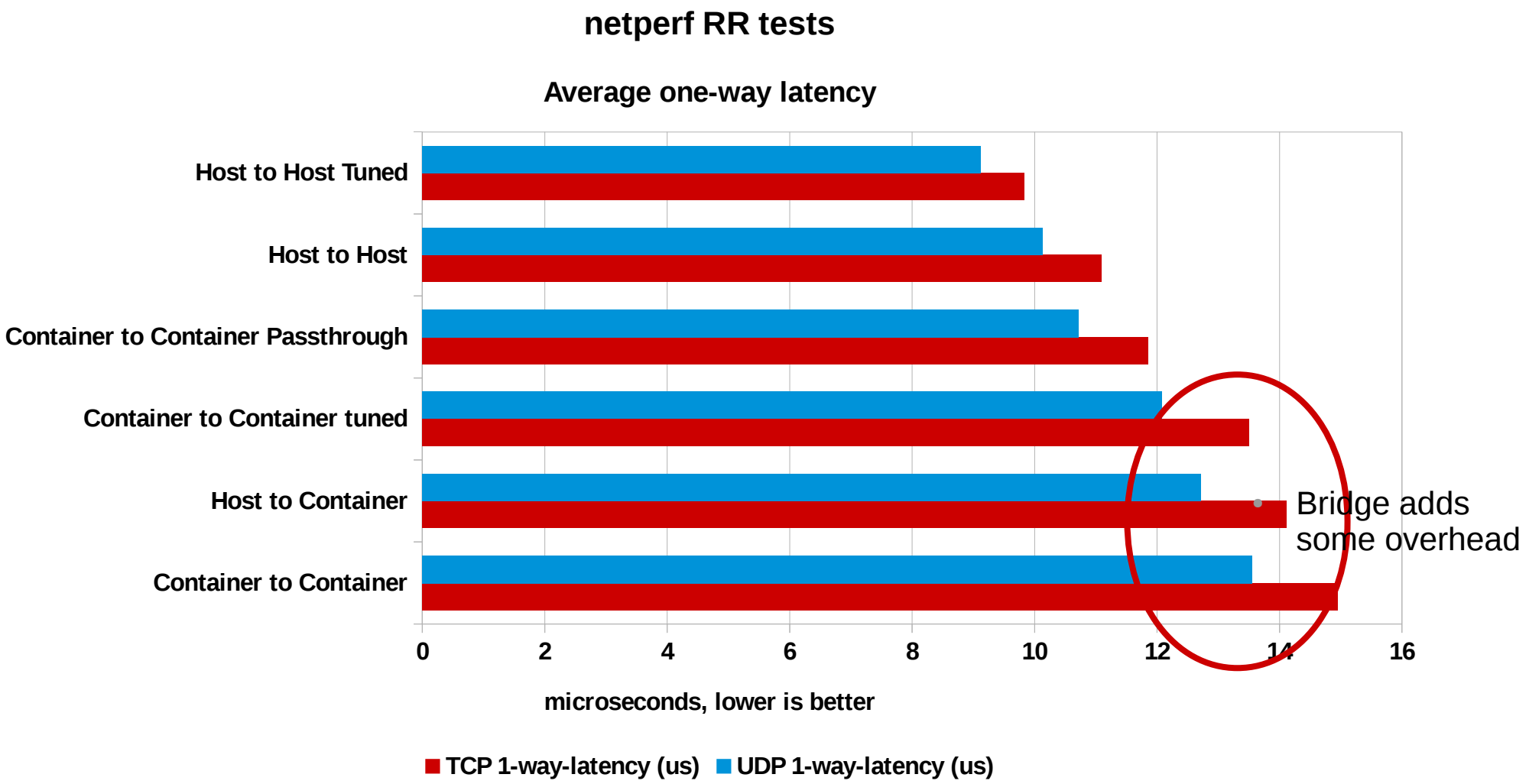
- Depends on
  - Hardware/Cloud capabilities
  - Workload requirements
  - Overcommit configuration
  - Network capabilities/config
    - ~1000 using Linux Bridge
    - Higher with SDN
  - Storage capabilities/config
    - Loop-LVM: no setup, Good performance...for development
    - Direct-LVM: requires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



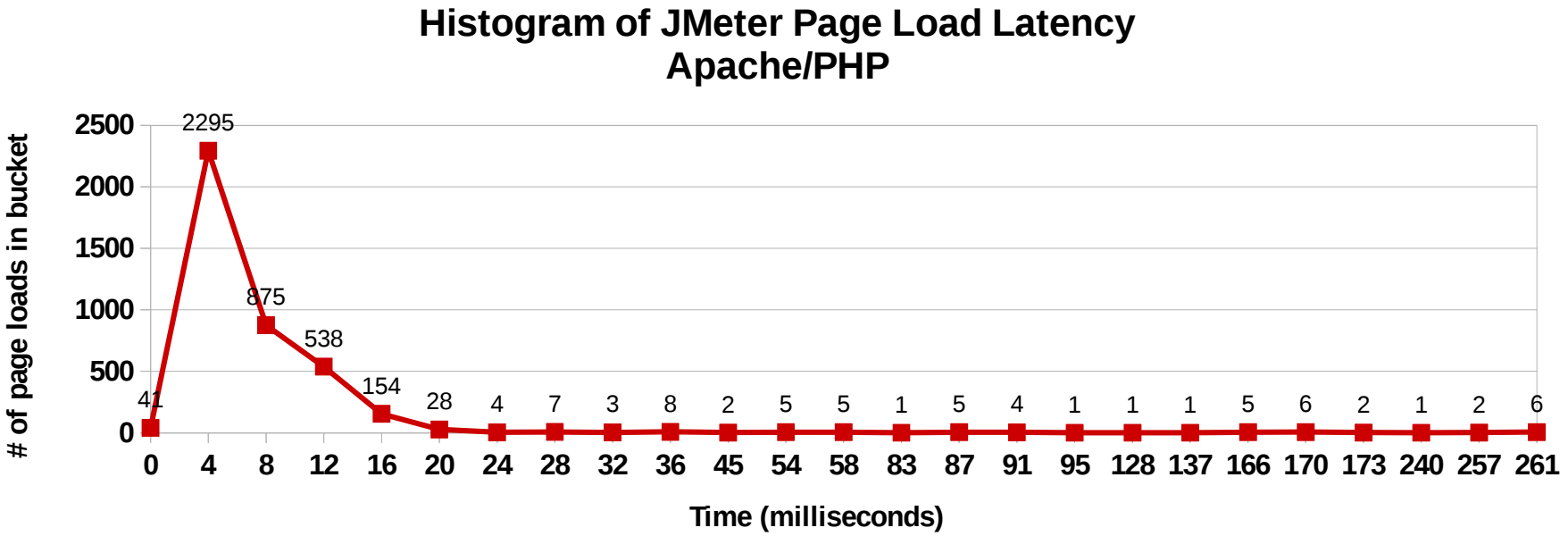
Container Performance
vs
Bare Metal

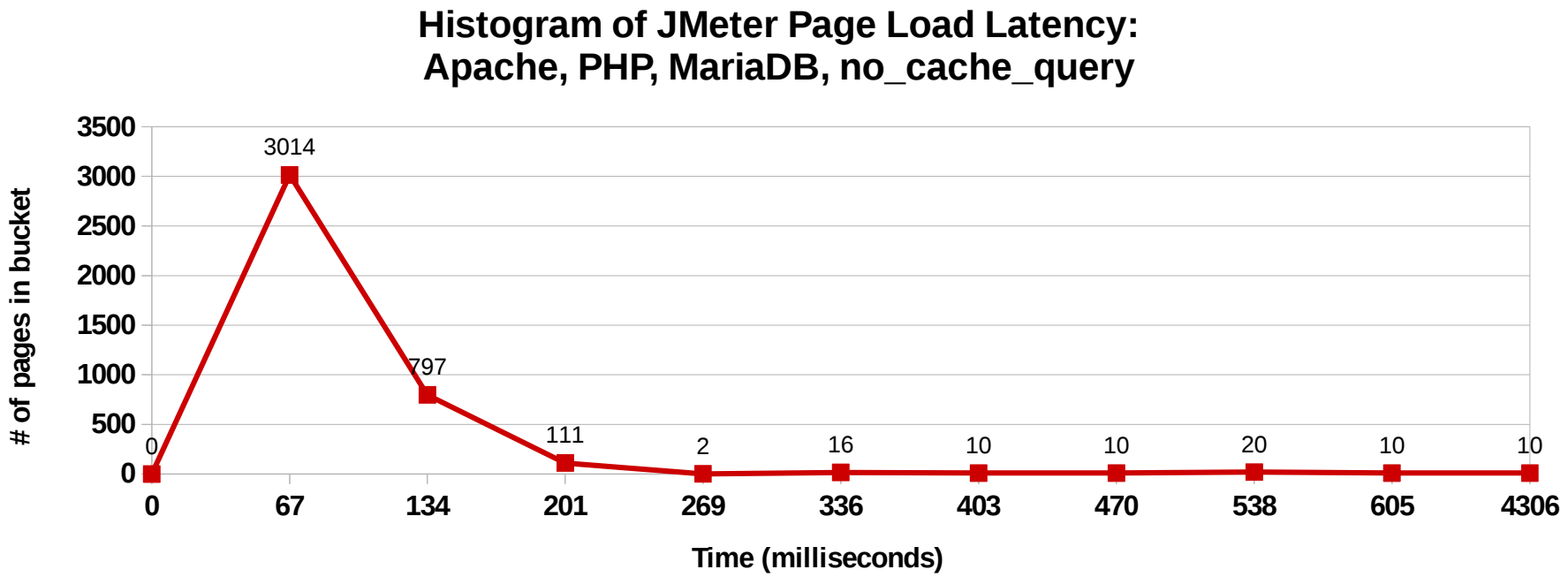
Network Performance: netperf, RR tests (latency)

- With cpu pinning and IRQ Affinity in place



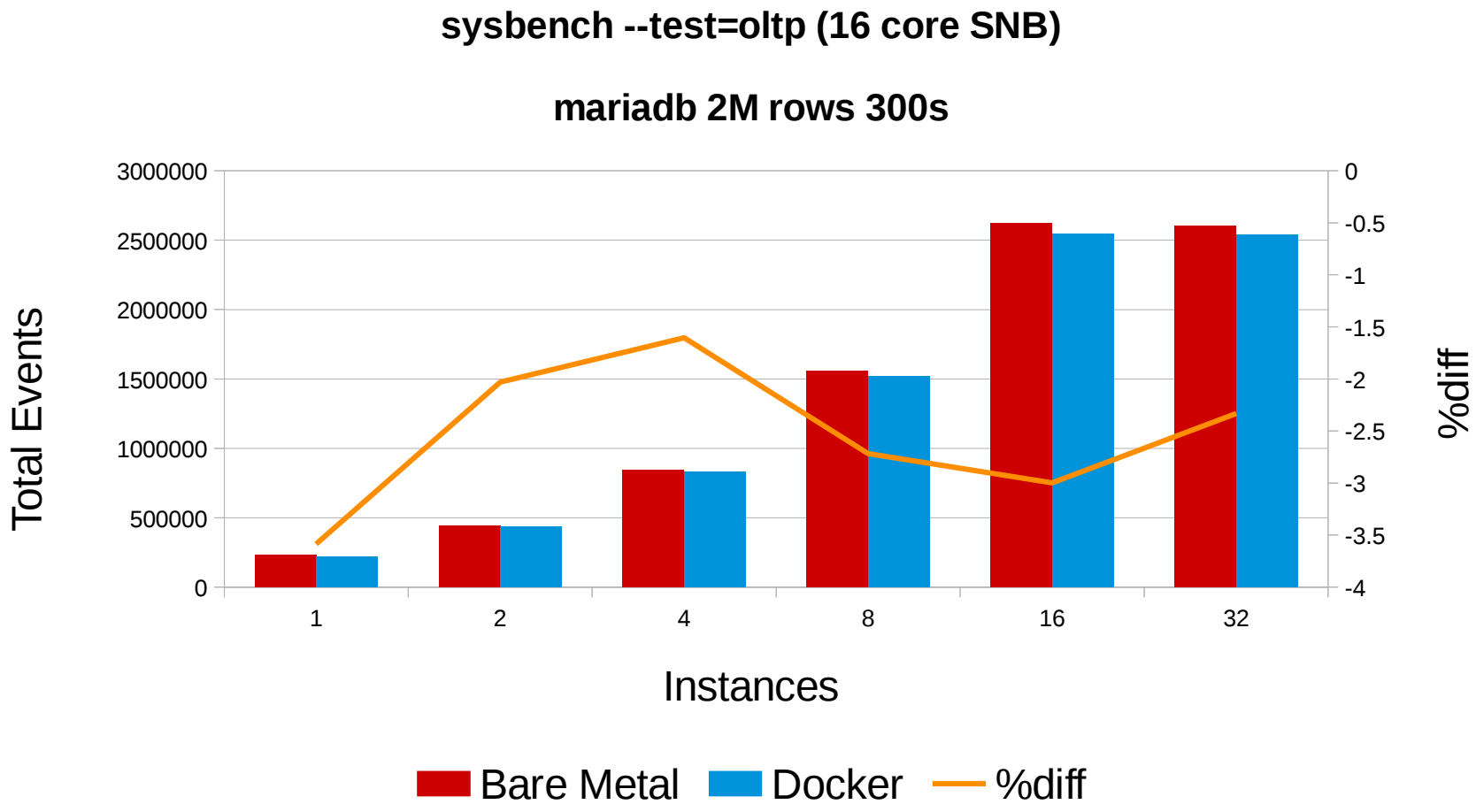
Apache, PHP, MariaDB in a container [remote JMeter]



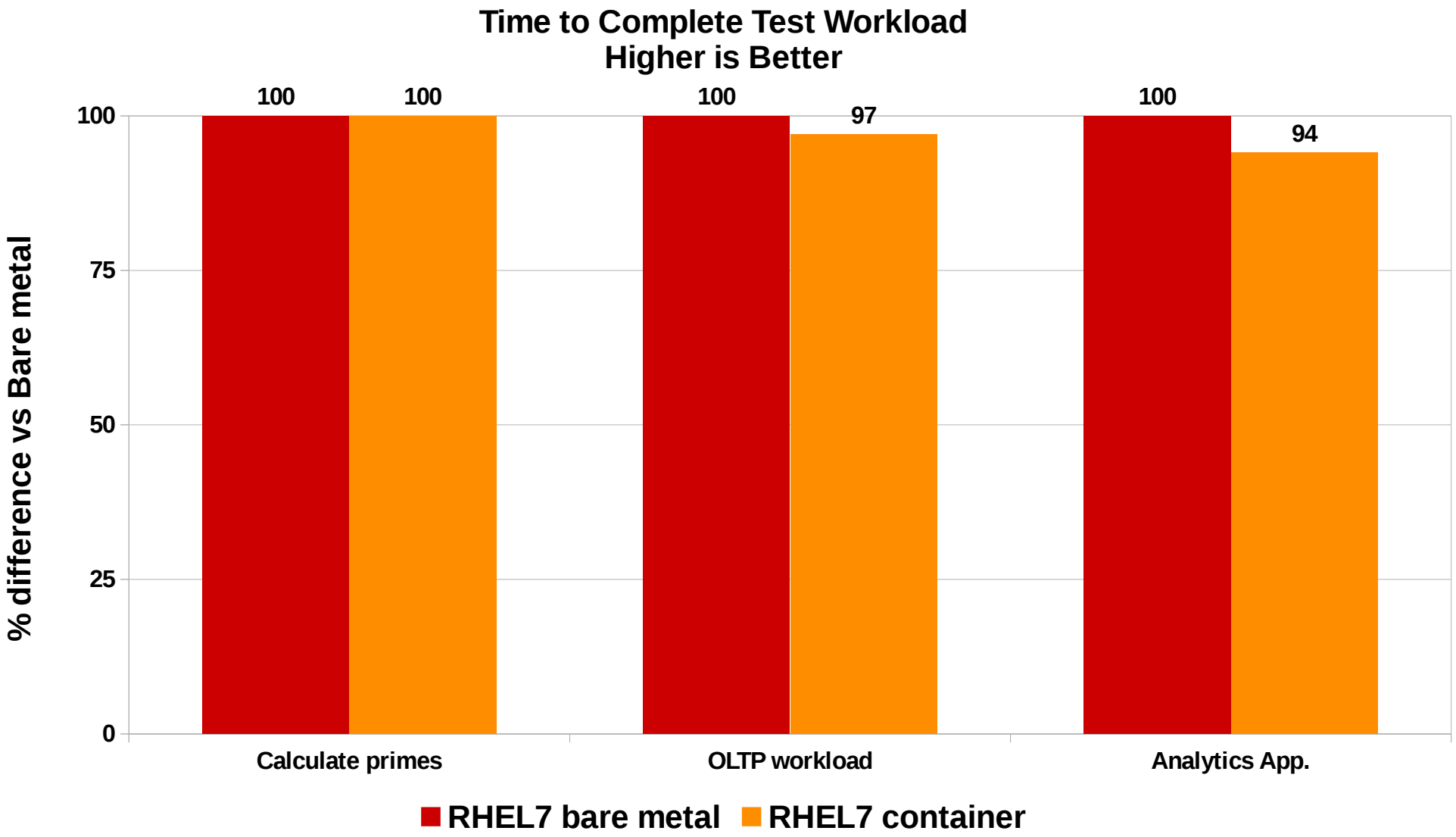


CPU/Memory Performance Test Plan: sysbench oltp

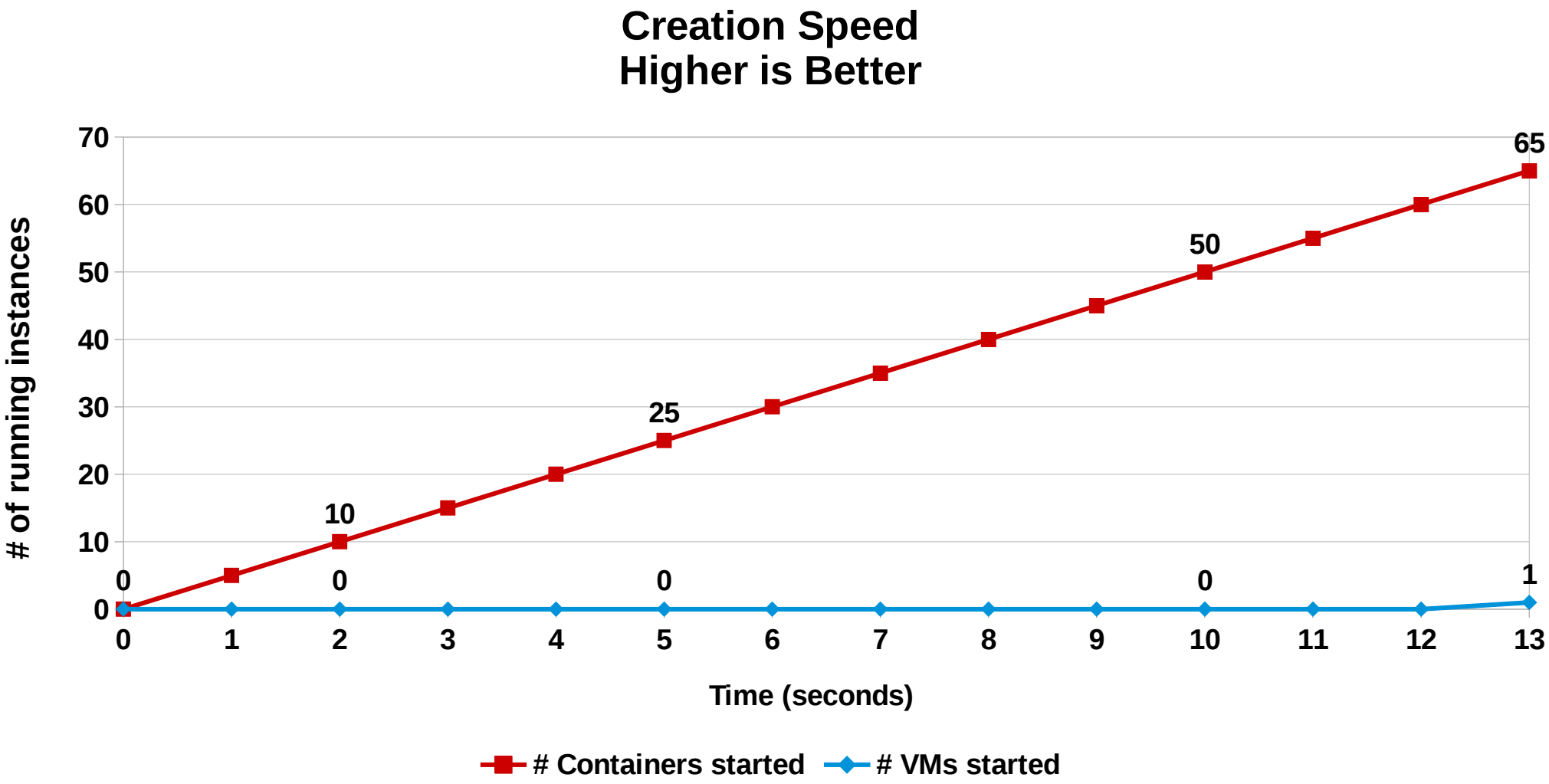
- sysbench –test=oltp mariadb 2M rows, 300s
- Within 1-4% of bare metal.
- 32 threads = overcommit



Container performance across multiple workloads



Container creation speed vs VMs



Dockerfile for Performance Analysis

- Includes all the common debug/analysis tools
  - Some tests run with additional tracing
- Sets up the environment
  - Tests must know if they're run in a container or not
- Pulls in necessary git repos
  - Repos include all the tests/workload generators
- Mounts volumes:
  - For test state information, multi-host semaphore/locking
  - Tests write output to persistent storage

Red Hat delivering this in a 'rhel-tools' container image. (GA with Atomic)

Lastly...

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

Useful Utilities

Supportability

- redhat-support-tool
- sos
- kdump
- perf
- psmisc
- strace
- sysstat
- systemtap
- ftrace / 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

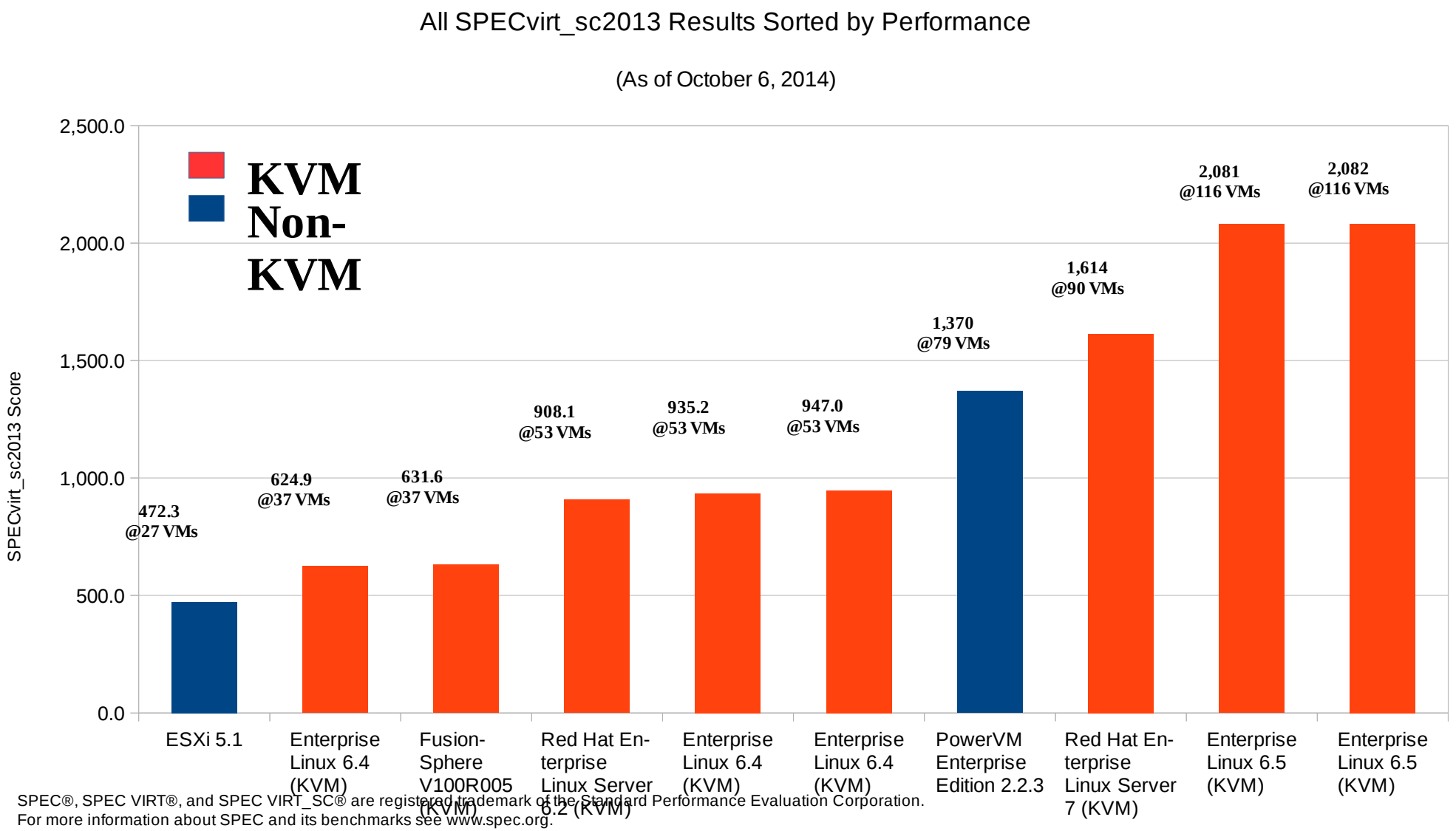
- blktrace
- iotop
- iostat

Questions

Backup slides

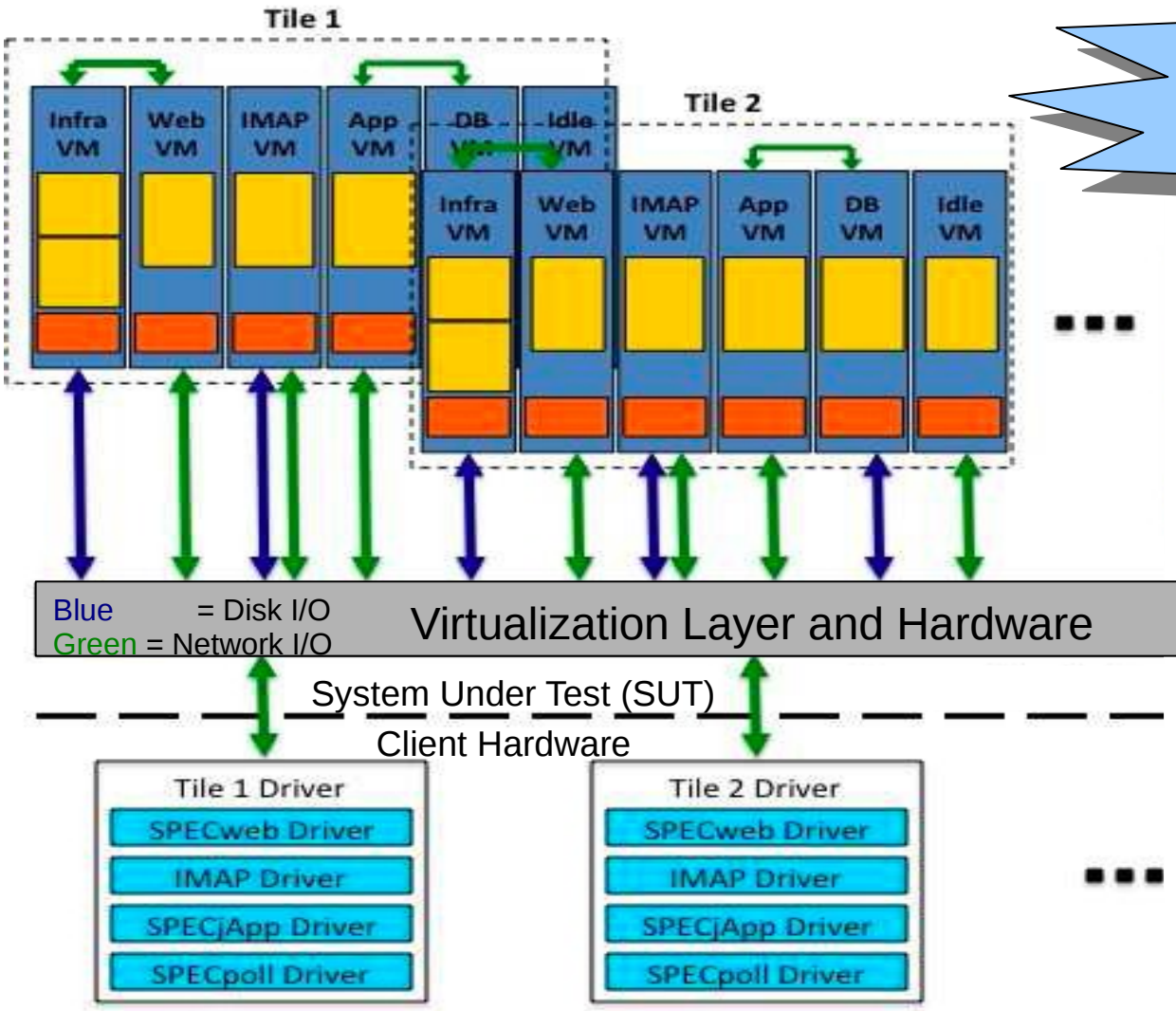
SPECvirt

Latest SPECvirt 2013 results



SPECvirt2013:

RHEL 7/6and KVM Post Industry Leading Results



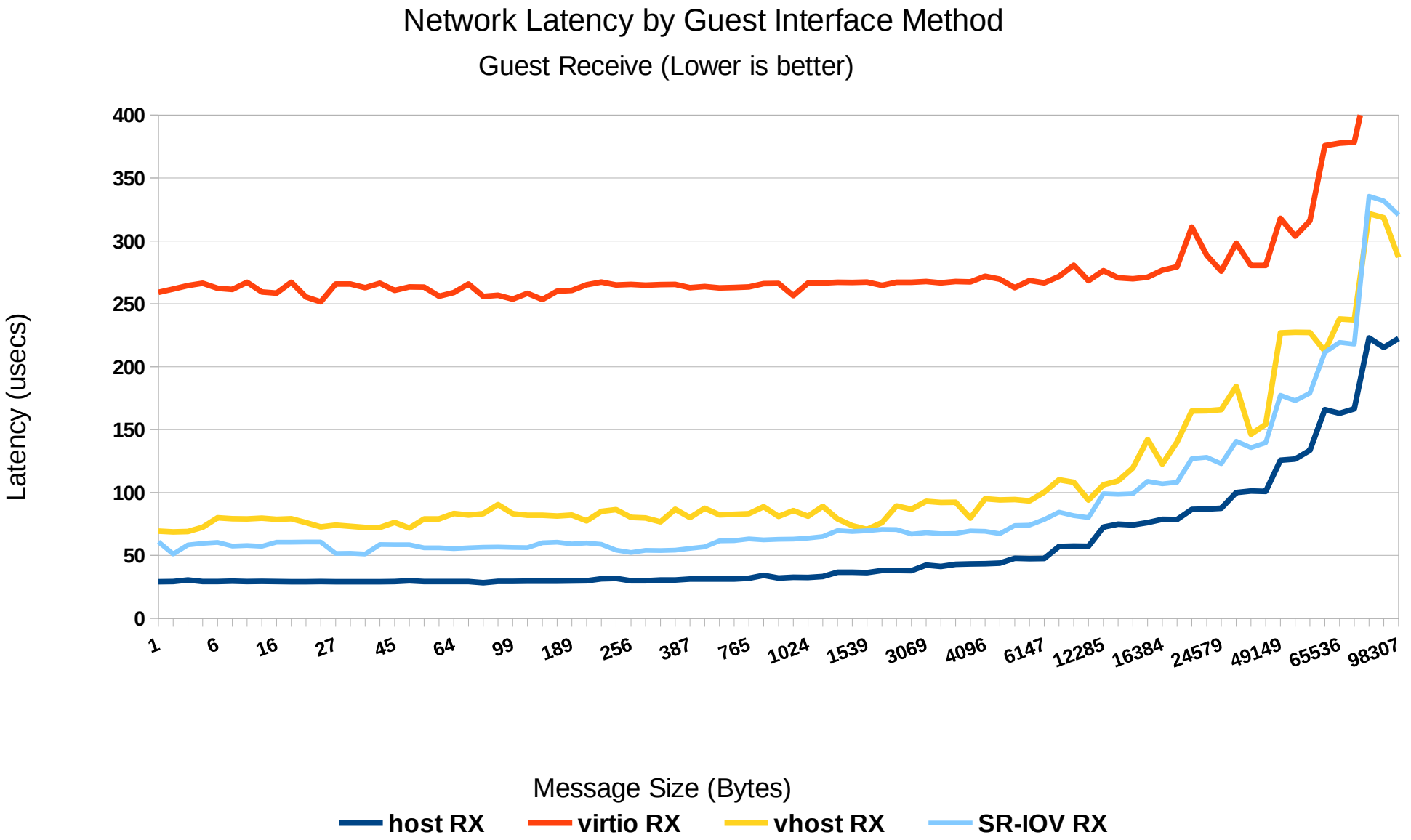
> 1 SPECvirt Tile/core

Key Enablers:

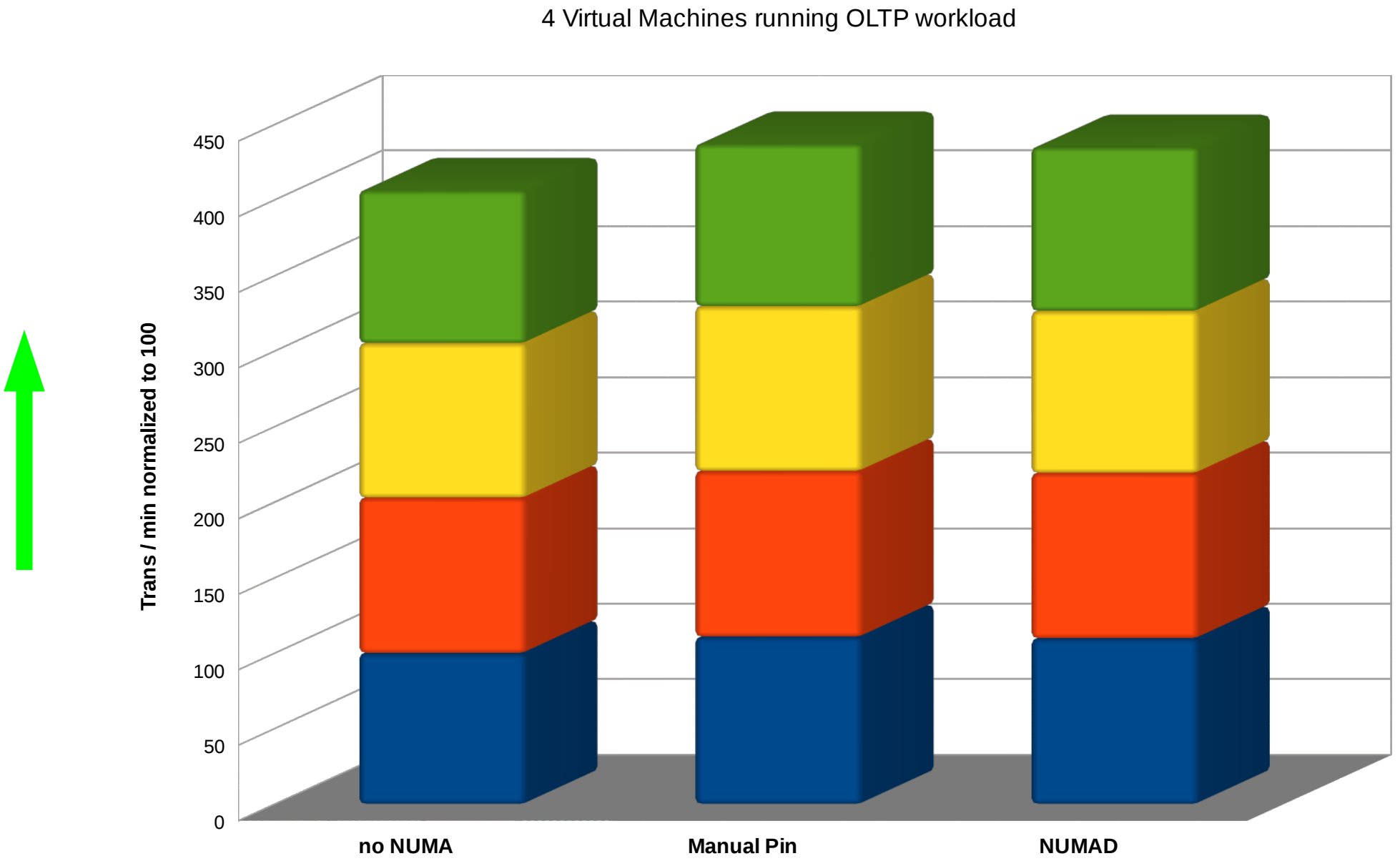
- ✓ SR-IOV
- ✓ Huge Pages
- ✓ NUMA
- ✓ Node Binding

http://www.spec.org/virt_sc2013/results/

Virtualization Tuning – Network – Latency Comparison



Virt Tuning – Using NUMA



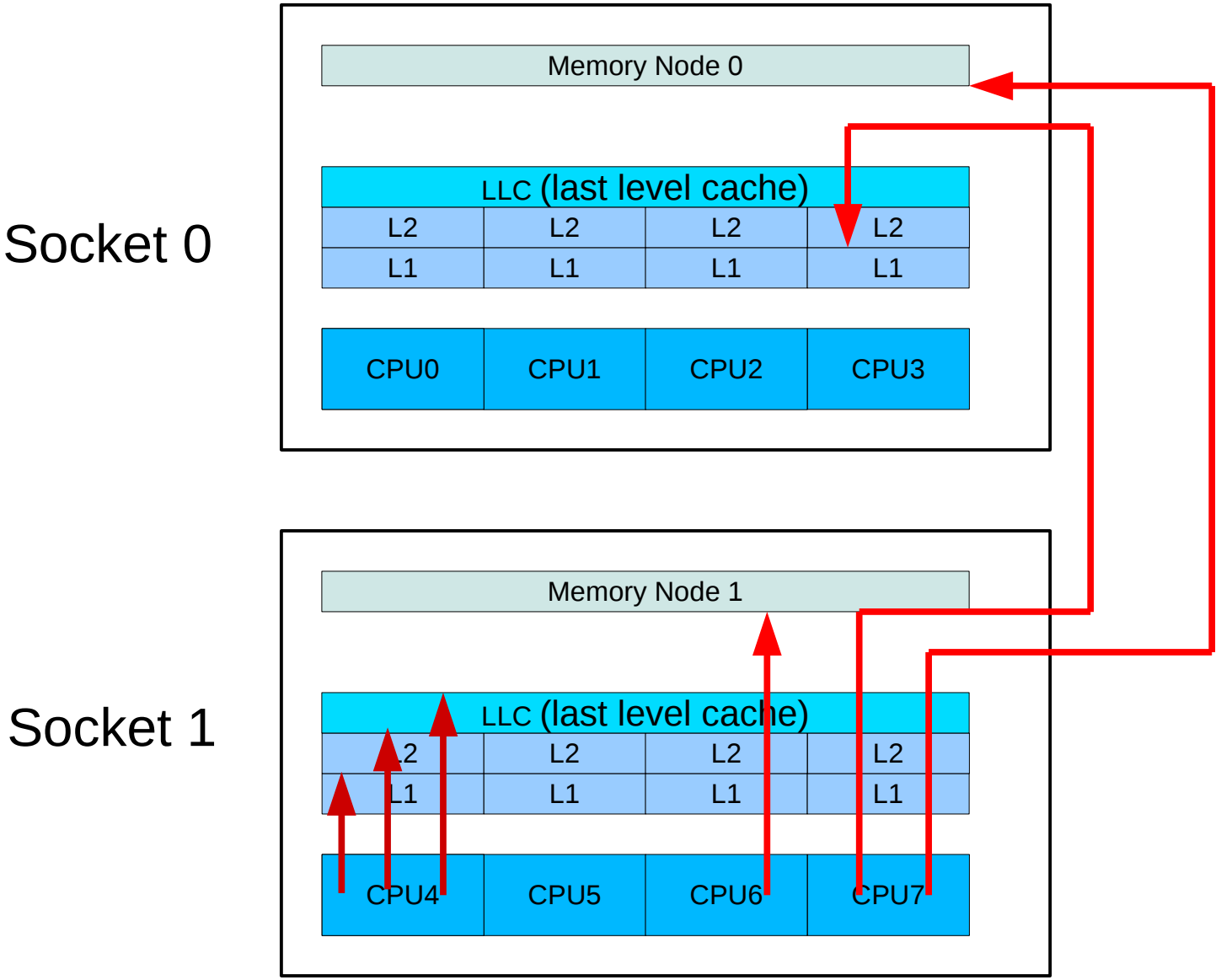
Backup

Detecting cache to cache contention.

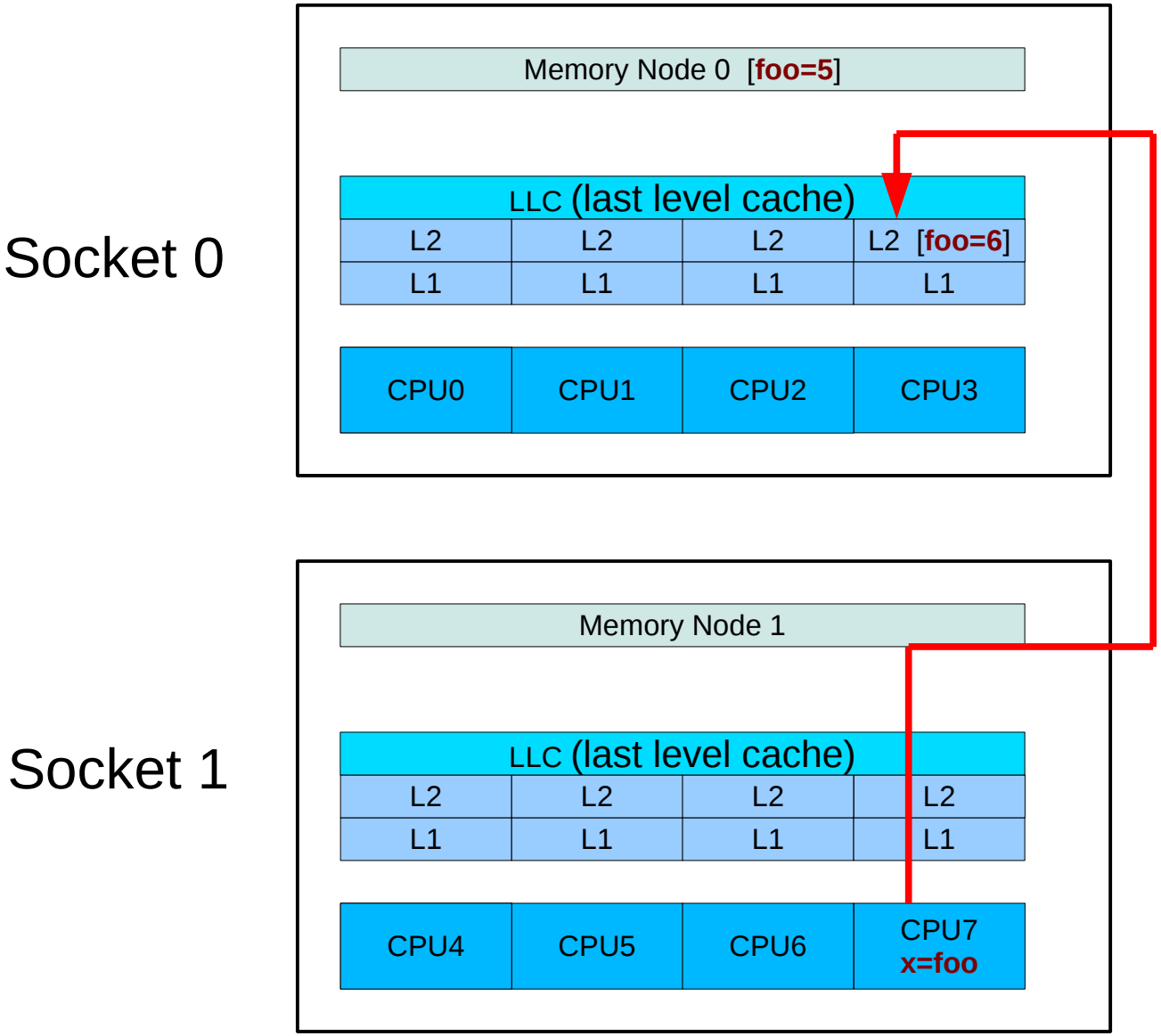
Cache-to-Cache false sharing tool

- Red Hat developed an enhancement to the 'perf' tool to detect simultaneous cross-numa cacheline contention.
- Critical to:
  - Shared memory applications
  - Multi-threaded apps spanning multiple numa nodes
- Code being reviewed upstream
- Integrating it into the perf tool (“perf c2c”)
- Expect it to be available in RHEL7.2

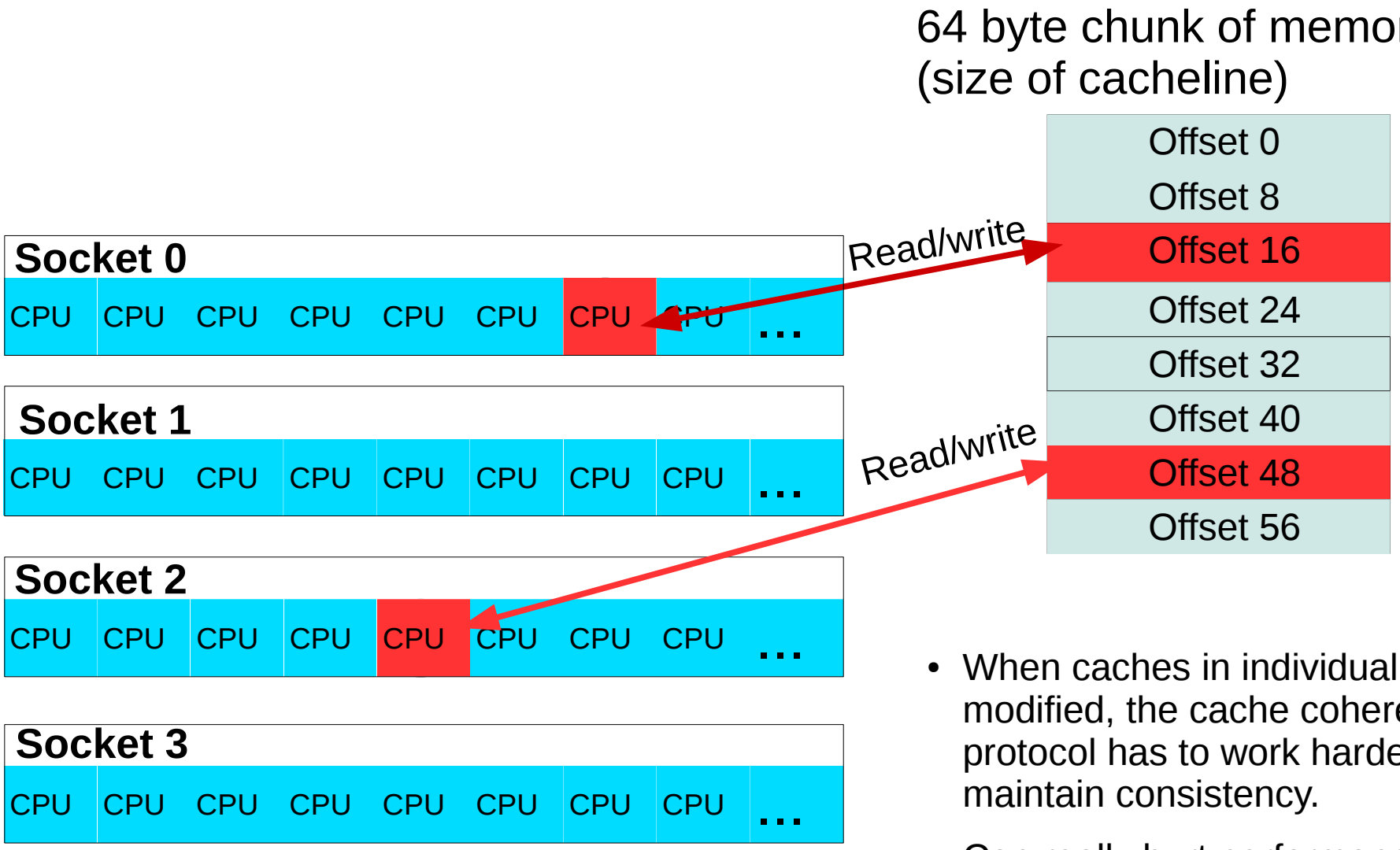
Resolving a memory access



Resolving a memory access



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)