RED HAT SUMMIT

Performance Analysis and Tuning – Part I

D. John Shakshober (Shak) – Director Performance Engineering

Larry Woodman - Senior Consulting Engineer / Kernel VM

Joe Mario - Senior Principal Performance Engineer / RHEL / Net / Tools

Sanjay Rao – Principal Performance Engineer / Database

Agenda: Performance Analysis Tuning Part I

- Part I
  - RHEL Evolution 5->6->7 , Hybrid Clouds / OSE / OSP tuned / CVE
  - NonUniform Memory Access (NUMA)
  - What is NUMA, RHEL Architecture, Auto-NUMA-Balance
  - HugePages
    - Static, Transparent, variable sized 4K/2MB/1GB
  - Control Groups
- “Meet The Experts” - Free as in Soda/Beer/Wine

Agenda: Performance Analysis Tuning Part II

- Part II
  - Disk and Filesystem IO - Database Throughput-performance
  - Network Performance Latency-performance
    - Tuned w/ cpu_partition profile
  - System Performance/Tools
    - Perf, and Tuna, PCP
  - Realtime RHEL7, KVM-RT and NFV w/ DPDK
- “Meet The Experts” - Free as in Soda/Beer/Wine

Red Hat Enterprise Linux Performance Evolution (fix)

RHEL5

Hugepages
Static
Ktune – on/off

CPU Affinity (taskset)

NUMA Pinning (numactl)
Irqbalance

RHEL6

HugePages
Transparent
Tuned – choose profile

CPU Affinity (ts/numactl)
NUMAD – uerspace tool
Cgroups -
irqbalance – NUMA enhanced

RHEL7

Transparent Hugepages
Tuned – throughput-performance (default)
CPU Affinity (ts/numactl)
Autonuma-Balance
LXC – Container/Docker
irqbalance – NUMA enhanced

RH Cloud Suites

RHV – out-of-the-box
virt-host/guest
RH OSP – blueprints
Tuned, Numa pining
NIC – jumbo sriov
RH OpenShift v3
RH Sat 6
RH Cloud Forms

Tuned Overview

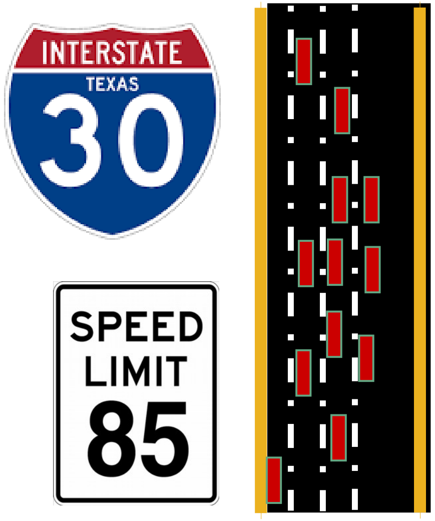
- Installed by default
- Auto-set Profiles
- Single config file
- Inheritance/Hooks
- bootloader/cmdline configs

- New Profiles since last ye
  - Realtime
  - NFV - cpu-partitioning
  - RHEL Atomic Host
  - OpenShift
  - Oracle

See `man tuned-profiles` for profile definitions

Performance Metrics ,

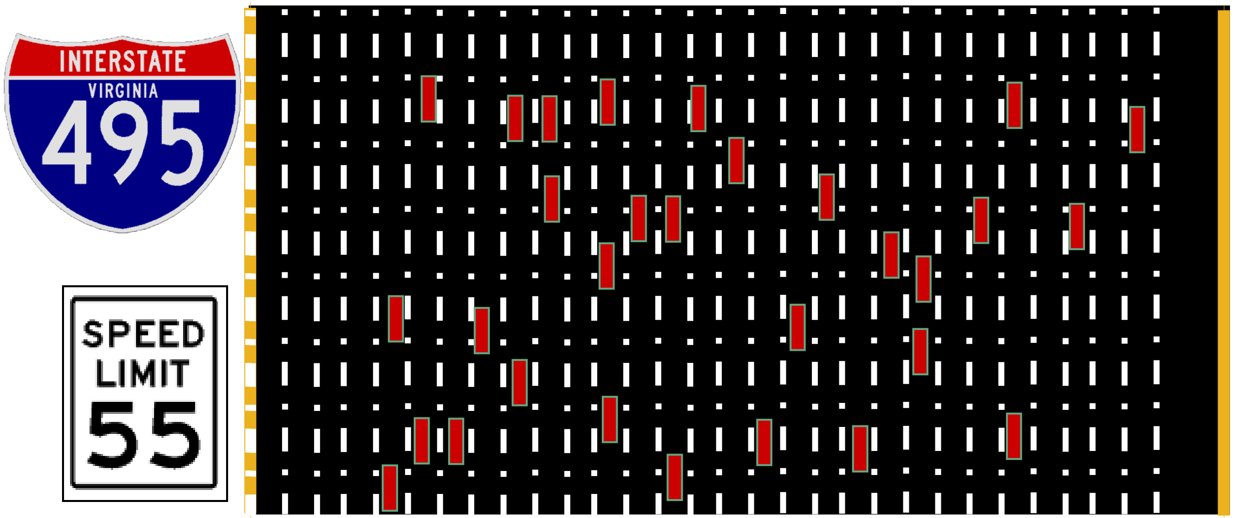
Latency==Speed



Latency – Speed Limit

- Ghz of CPU, Memory PCI
- Small transfers, disable aggregation – TCP nodelay
- Dataplane optimization DPDK

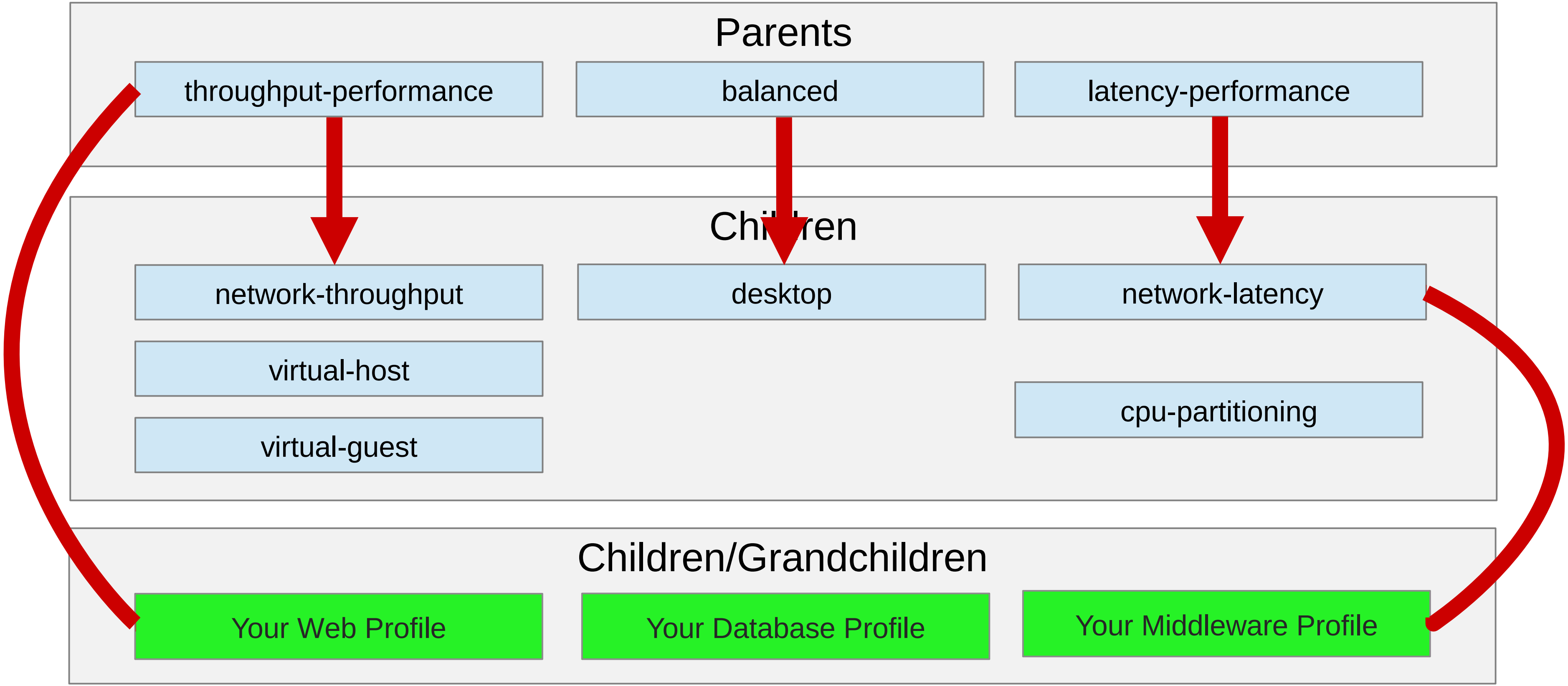
Throughput==Bandwidth



Throughput: Bandwidth: # lanes in Highway

- Width of data path / cachelines
- Bus Bandwidth, QPI links, PCI 1-2-3
- Network 1 / 10 / 40 Gb – aggregation, NAPI
- Fiberchannel 4/8/16, SSD, NVME Drivers

Tuned: Your Custom Profiles



Parents

throughput-performance

balanced

latency-performance

Children

network-throughput

virtual-host

virtual-guest

desktop

network-latency

cpu-partitioning

Children/Grandchildren

Your Web Profile

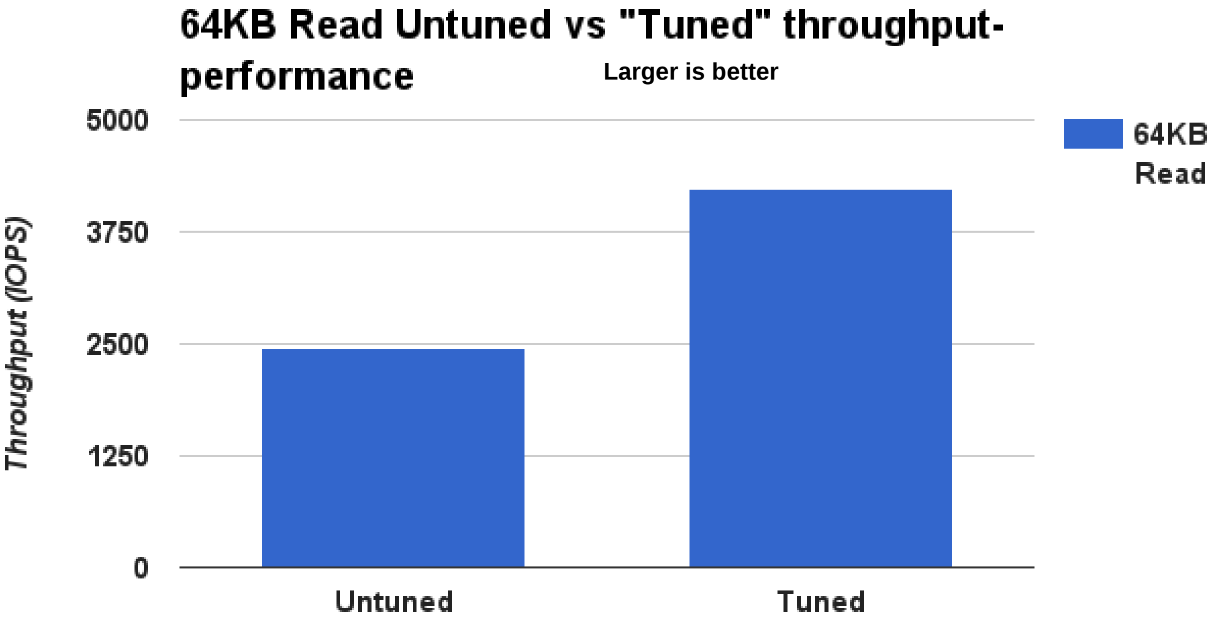
Your Database Profile

Your Middleware Profile

Tuned - Profiles

RHEL Desktop/Workstation	balanced
RHV Host, Guest	virtual-host/guest
Red Hat Storage	rhs-high-throughput, virt
RHEL Atomic	atomic-host, atomic-guest
RHEL Server/HPC	throughput-performance
RHV	virtual-host
OSP (compute node)	virtual-host
OCP – Open Shift	openshift-master, node
RHEL for Real Time	realtime
RHEL for Real Time KVM/NFV	realtime-virtual-host/guest
RHEL + SAP	sap / sap-hana
RHOP - NFV (compute node)	cpu-partitioning

Tuned: Storage Performance Boost: throughput-performance (default in RHEL7)



RHEL Security mitigation for Meltdown / Spectre

Spectre

- **Variant 1: Bounds check bypass**
    - Addressed through speculative load barriers (lfence/new nops).
    - Mitigation cannot **be disabled.**

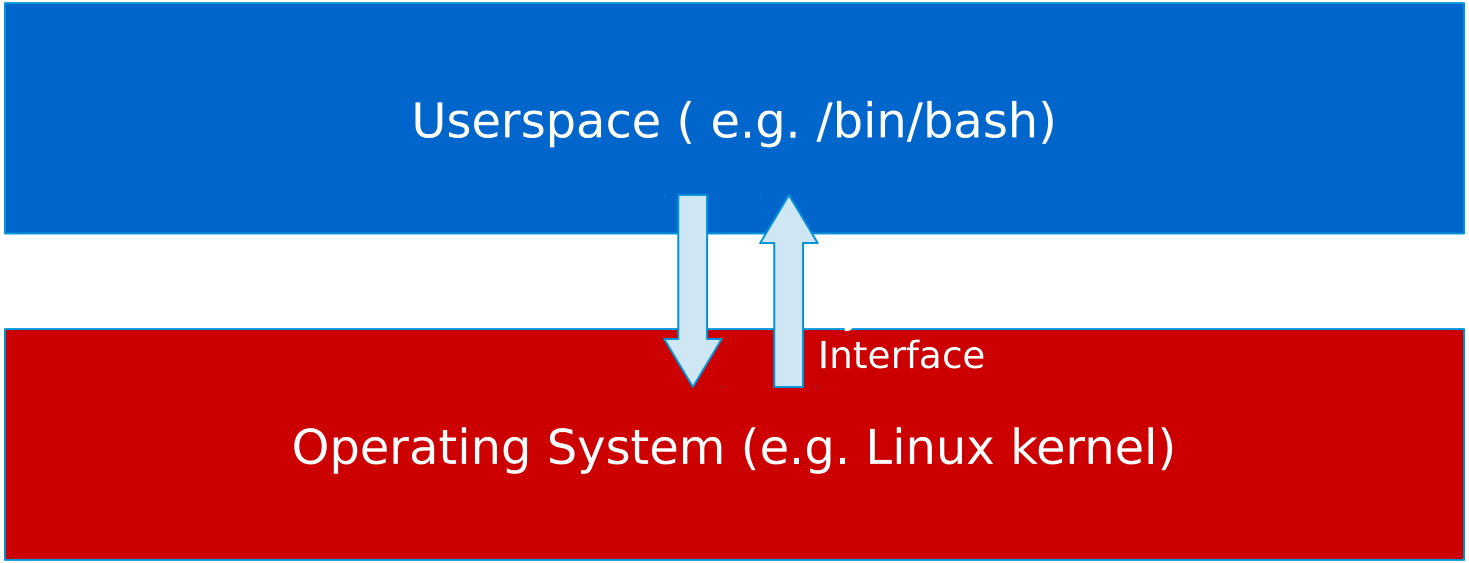
- **Variant 2: Indirect Branch Predictor poisoning**
    - Addressed through disabling the indirect branch predictor when running kernel code to avoid influence from application code.
    - Requires **microcode/millicode/firmware** updates from vendor.
    - Mitigation can be disabled, defaults to being enabled.

Meltdown

- **Variant 3: Rogue cache data load**
    - Addressed through Page Table Isolation (pti - preventing kernel data and VA/PA translations from being present in certain CPU structures).
    - Mitigation can be disabled, **defaults to being enabled.**

Spectre / Meltdown performance impact

func[user to kernel transitions & time in kernel]290139

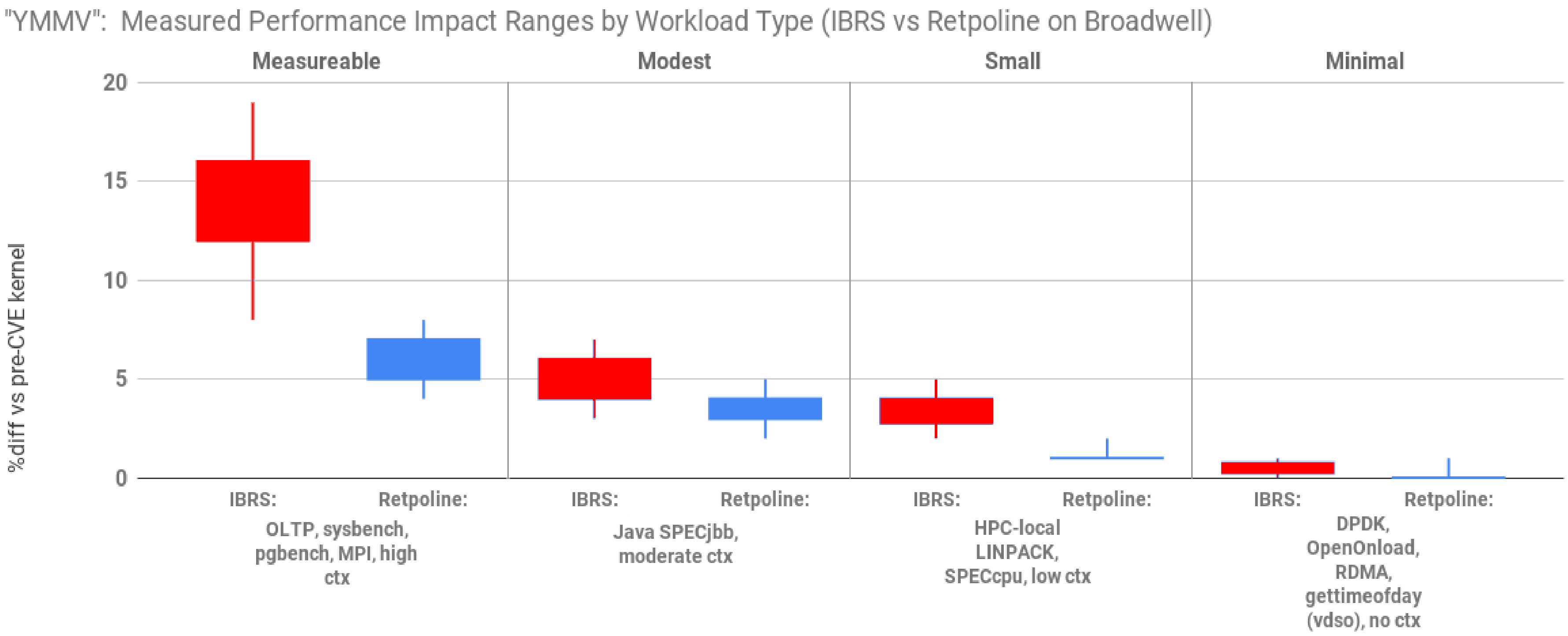


Userspace (e.g. /bin/bash)

Interface

Operating System (e.g. Linux kernel)

Spectre / Meltdown Impact
VARIES BY WORKLOAD



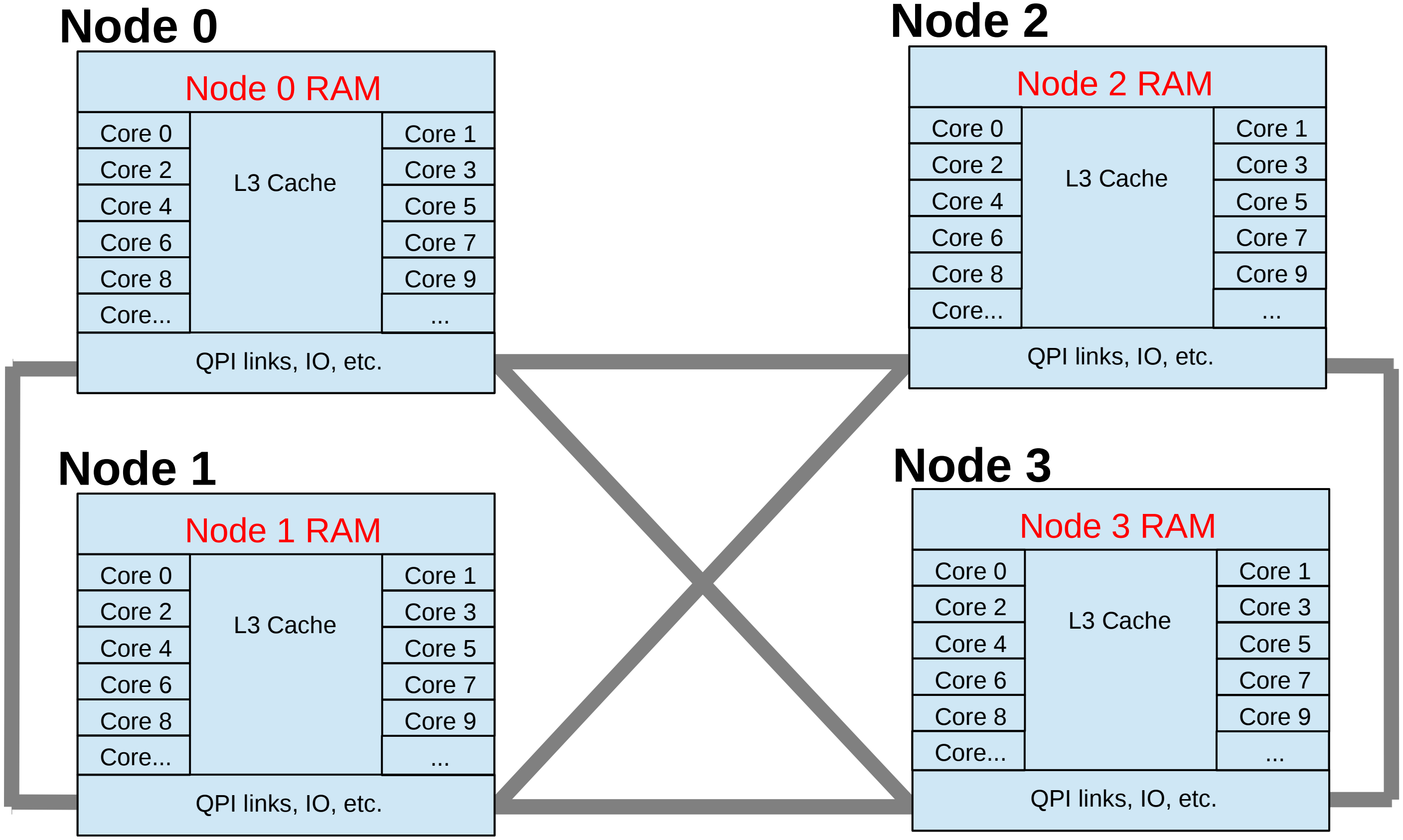
Spectre / Metldown Managing Perf Impact

- RHEL has transparent (thp) and static ***hugepages***
  - Reduces amount of TLB entries and thus total flush impact
- RHEL uses ***PCID*** support where possible to reduces impact of TLB flushes by tagging/tracking
- RHEL has ***runtime knobs*** to disable patches (no reboot)

```
echo 0 > /sys/kernel/debug/x86/pti_enabled

echo 0 > /sys/kernel/debug/x86/ibrs_enabled

echo 0 > /sys/kernel/debug/x86/retp_enabled
```

RHEL 6/7 Non-Uniform Memory Access (NUMA)

Typical Four-Node NUMA System



Node 0

Node 0 RAM

Core 0	L3 Cache	Core 1
Core 2		Core 3
Core 4		Core 5
Core 6		Core 7
Core 8		Core 9
Core...		...

QPI links, IO, etc.

Node 2

Node 2 RAM

Core 0	L3 Cache	Core 1
Core 2		Core 3
Core 4		Core 5
Core 6		Core 7
Core 8		Core 9
Core...		...

QPI links, IO, etc.

Node 1

Node 1 RAM

Core 0	L3 Cache	Core 1
Core 2		Core 3
Core 4		Core 5
Core 6		Core 7
Core 8		Core 9
Core...		...

QPI links, IO, etc.

Node 3

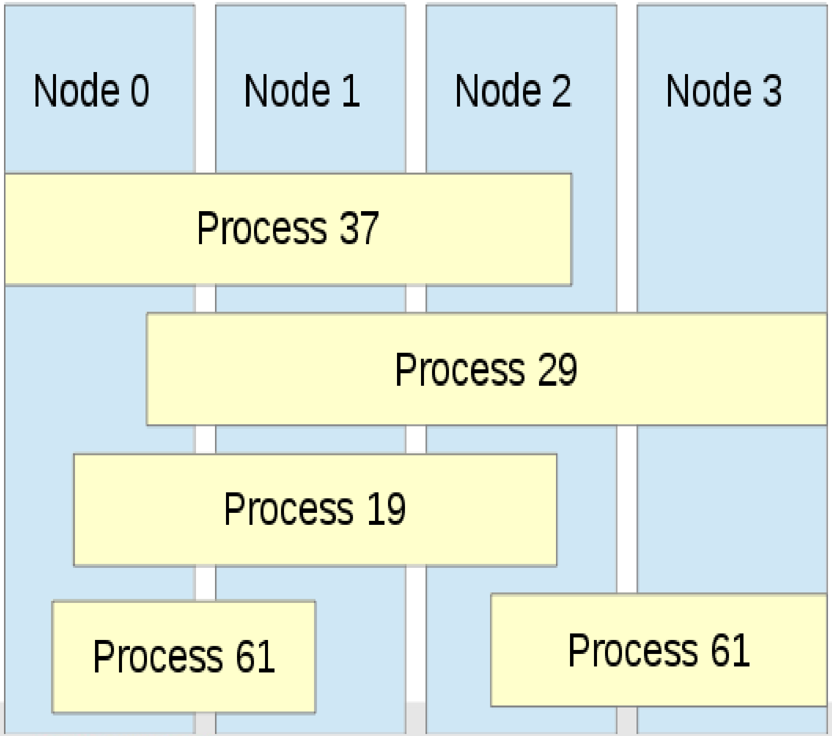
Node 3 RAM

Core 0	L3 Cache	Core 1
Core 2		Core 3
Core 4		Core 5
Core 6		Core 7
Core 8		Core 9
Core...		...

QPI links, IO, etc.

Four Node memory placement NUMA System

No NUMA management



Node 0

Node 1

Node 2

Node 3

Process 37

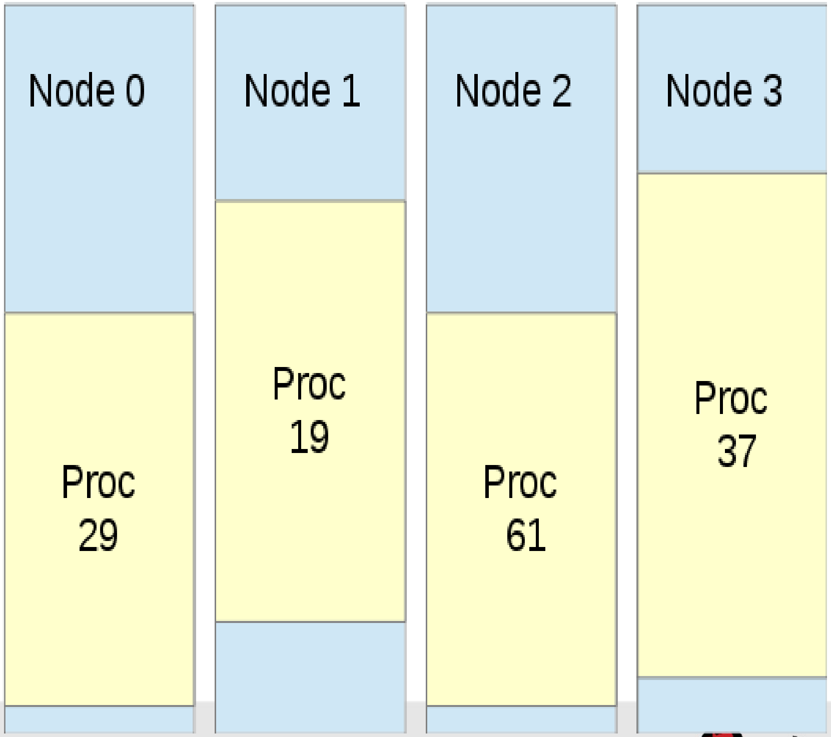
Process 29

Process 19

Process 61

Process 61

With NUMA management



Node 0

Node 1

Node 2

Node 3

Proc 29

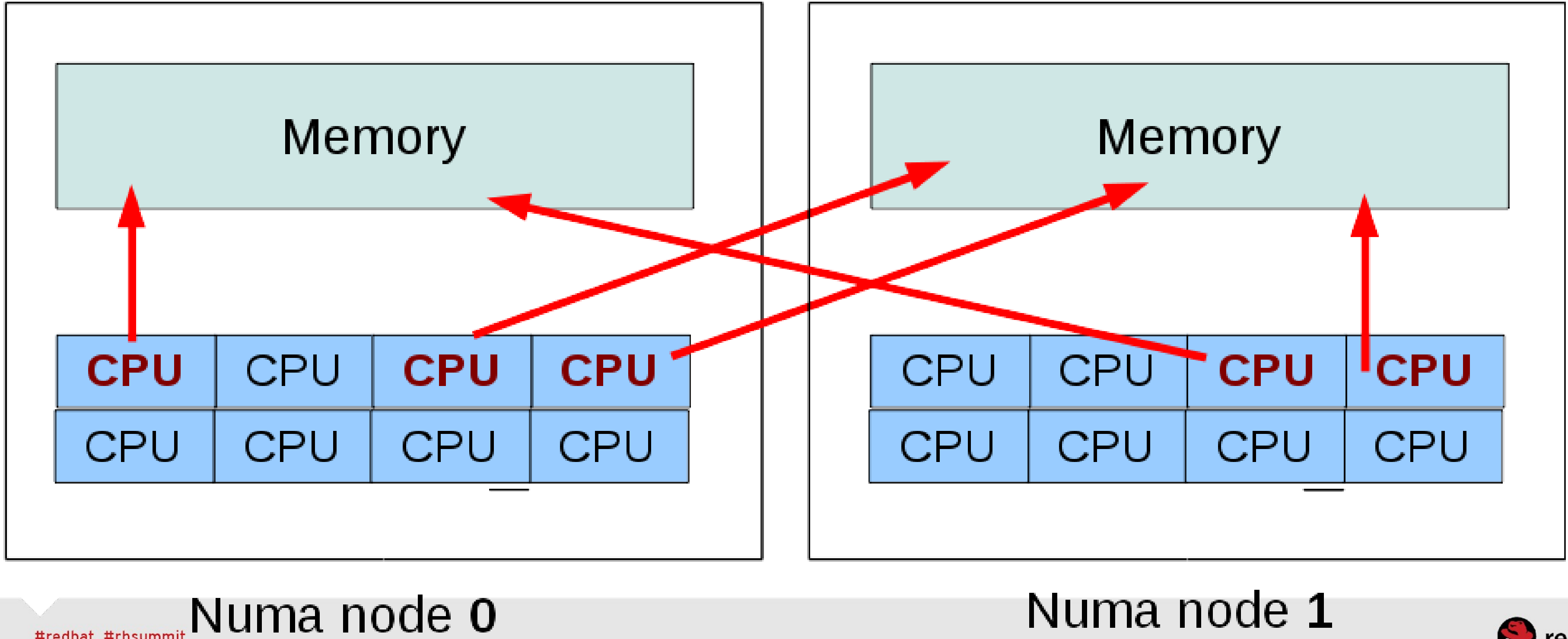
Proc 19

Proc 61

Proc 37

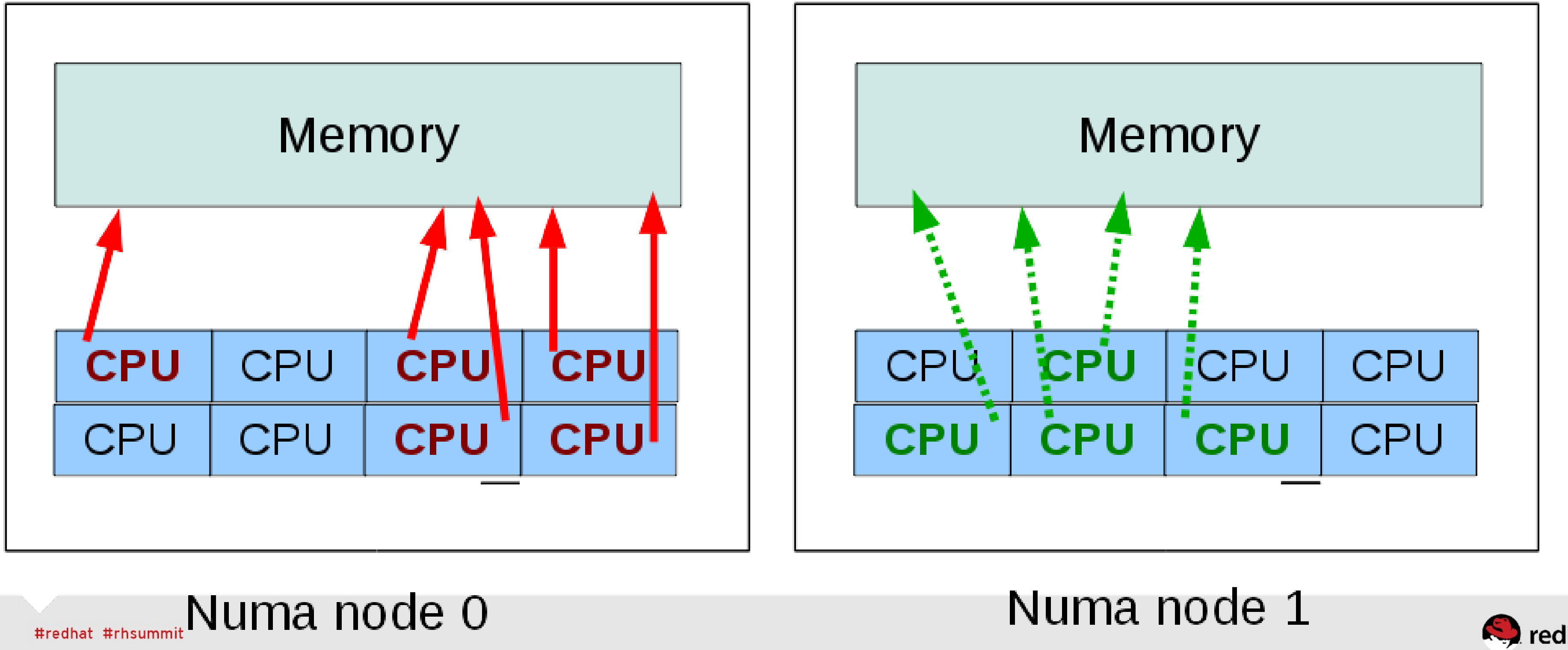
Non-optimal numa setup

Process 1 in **red**, 5 threads

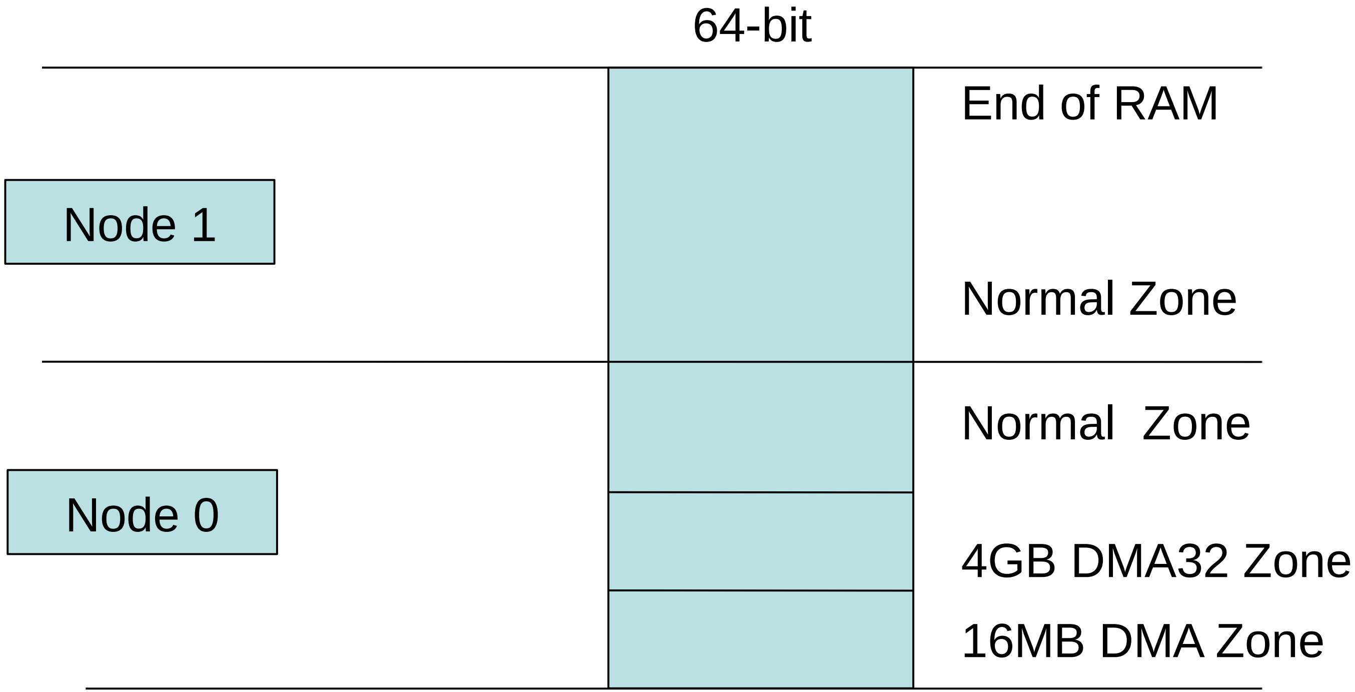


Optimal numa setup

Process 1 in **green**, 4 threads
Process 2 in **red**, 5 threads



NUMA Nodes and Zones



64-bit

Node 1

Node 0

End of RAM

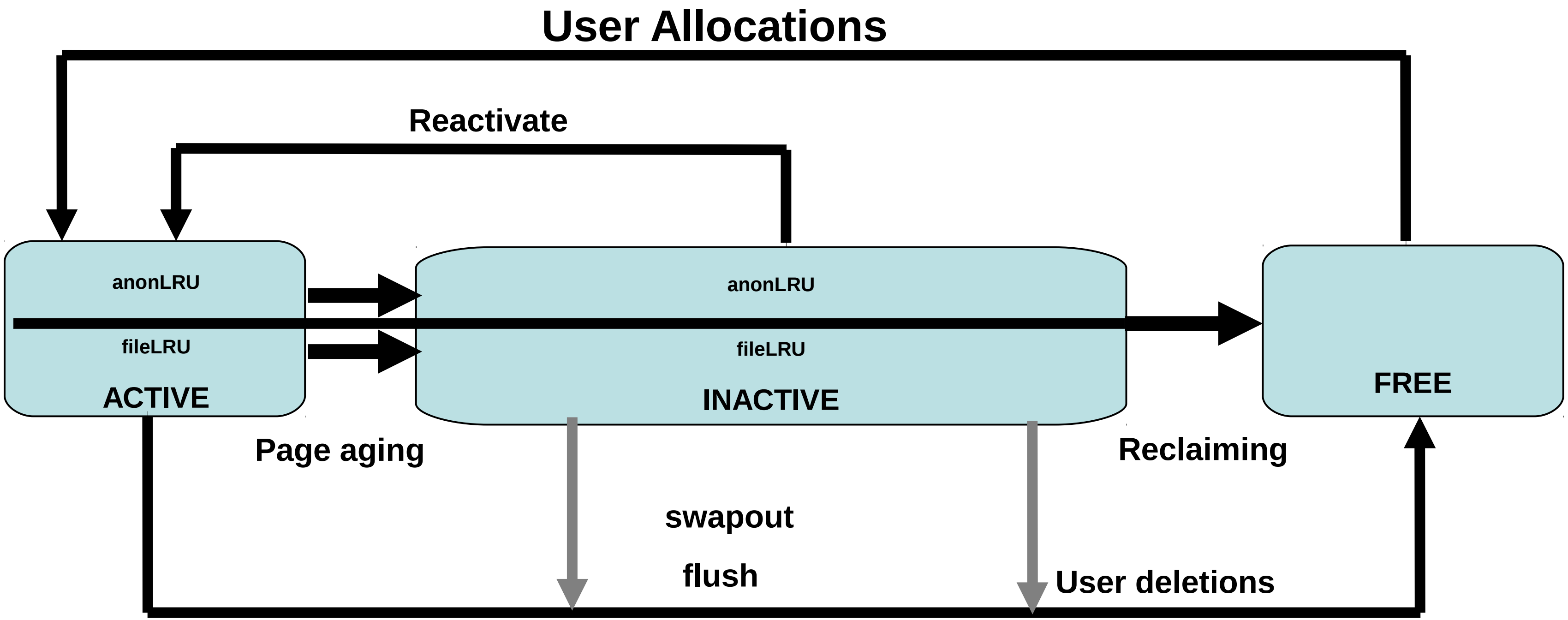
Normal Zone

Normal Zone

4GB DMA32 Zone

16MB DMA Zone

Per Node / Zone split LRU Paging Dynamics



Interaction between VM Tunables and NUMA

- Dependent on NUMA: Reclaim Ratios
  - /proc/sys/vm/swappiness
  - /proc/sys/vm/min_free_kbytes
  - /proc/sys/vm/zone_reclaim_mode

- Independent of NUMA: Reclaim Ratios
  - /proc/sys/vm/vfs_cache_pressure

- Writeback Parameters
  - /proc/sys/vm/dirty_background_ratio
  - /proc/sys/vm/dirty_ratio

- Readahead parameters
  - /sys/block/<bdev>/queue/read_ahead_kb

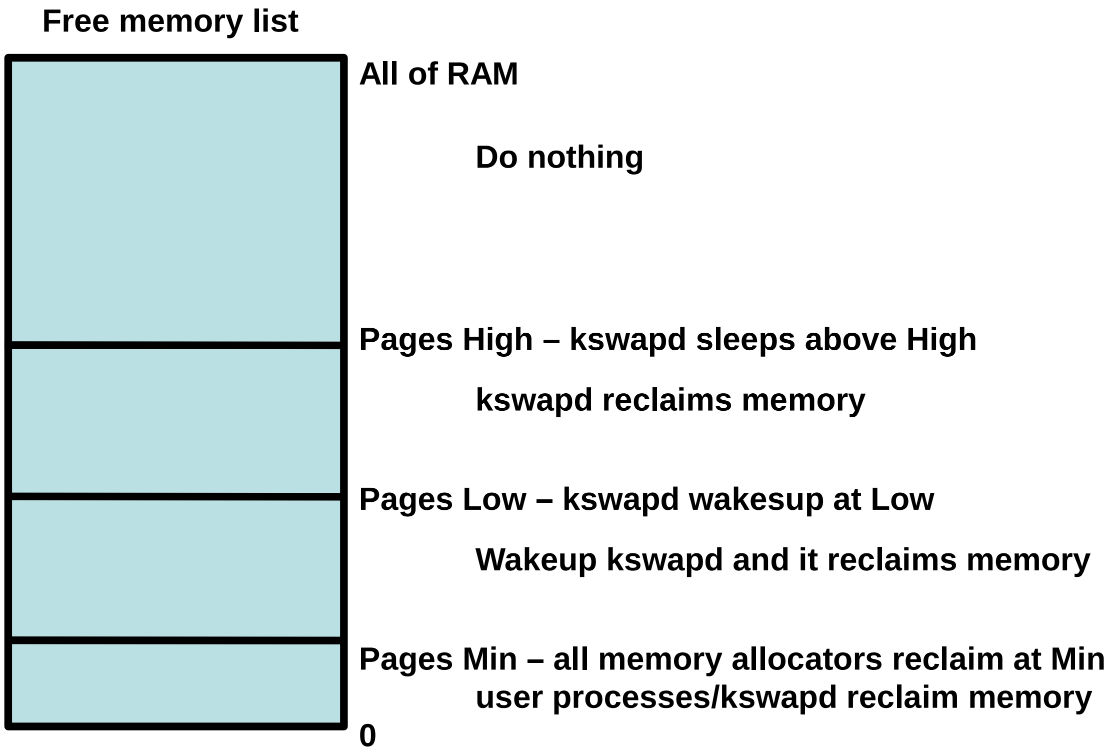
swappiness

- Controls how aggressively the system reclaims anonymous memory versus pagecache memory:

  - Anonymous memory – swapping and freeing
  - File pages – writing if dirty and freeing
  - System V shared memory – swapping and freeing

- Default is 60
- Decrease: more aggressive reclaiming of pagecache memory
- Increase: more aggressive swapping of anonymous memory
- Can effect Numa nodes differently.
- Tuning not as necessary on RHEL7 than RHEL6 and even less than RHEL5

Memory reclaim Watermarks



Free memory list

All of RAM

Do nothing

Pages High – kswapd sleeps above High

kswapd reclaims memory

Pages Low – kswapd wakesup at Low

Wakeup kswapd and it reclaims memory

Pages Min – all memory allocators reclaim at Min

user processes/kswapd reclaim memory

0

min_free_kbytes

Directly controls the page reclaim watermarks in KB
Distributed between the Numa nodes
Defaults are higher when THP is enabled

```
# cat /proc/sys/vm/min_free_kbytes
90100
----------------------------------------------------------
Node 0 DMA     min:80 low:100kB high:120kB
Node 0 DMA32 min:15312kB low:19140kB high:22968kB
Node 0 Normal min:29600kB low:37000kB high:44400kB
Node 1 Normal min:45108kB low:56384kB high:67660kB
----------------------------------------------------------

echo 180200 > /proc/sys/vm/min_free_kbytes
---------------------------------------------------------
Node 0 DMA     min:160kB low:200kB high:240kB
Node 0 DMA32 min:30624kB low:38280kB high:45936kB
Node 0 Normal min:59200kB low:74000kB high:88800kB
Node 1 Normal min:90216kB low:112768kB high:135320kB
----------------------------------------------------------
```

zone_reclaim_mode

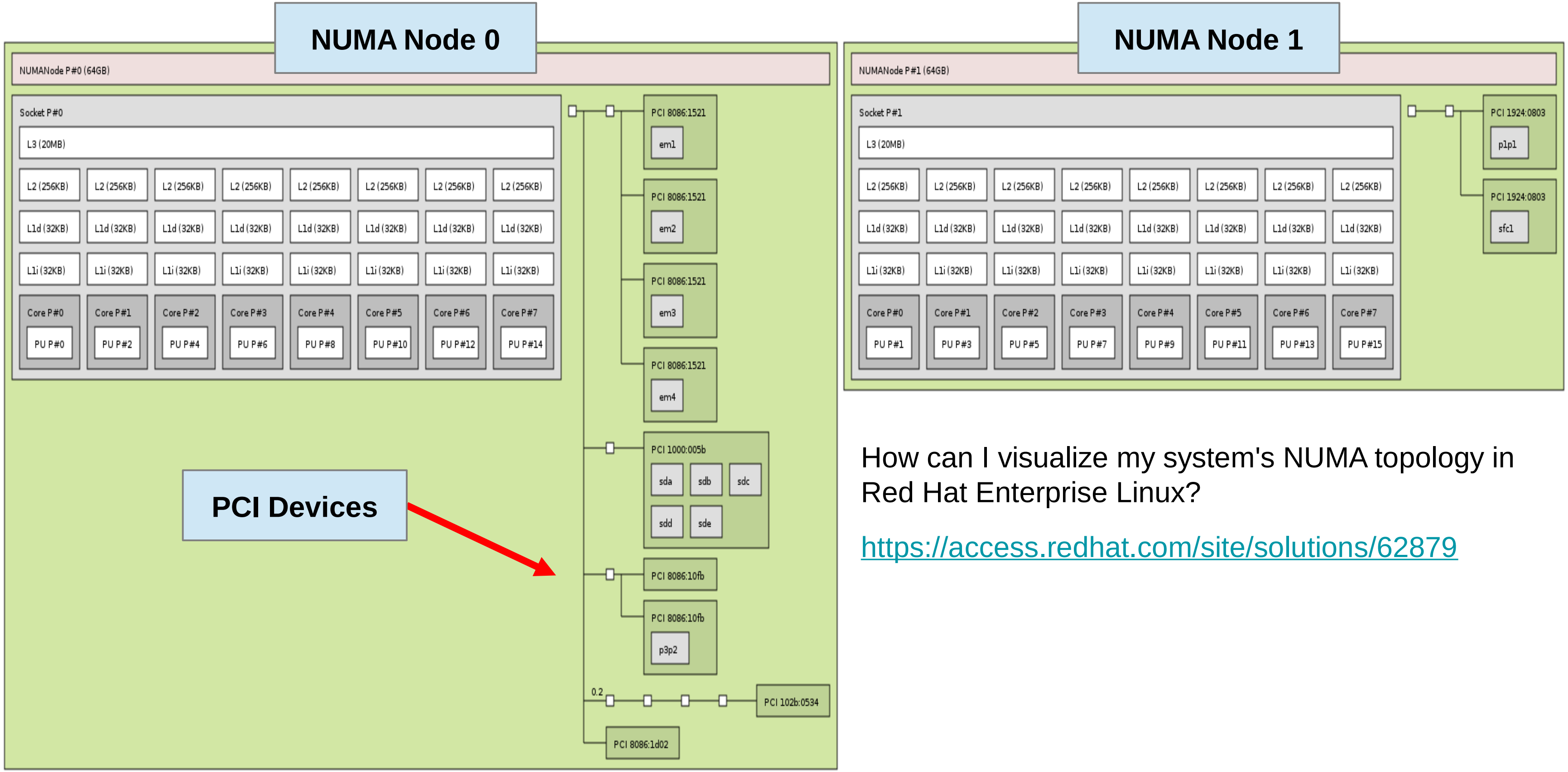
- Controls NUMA specific memory allocation policy

- To see current setting: cat /proc/sys/vm/zone_reclaim_mode
  - # echo 1 > /proc/sys/vm/zone_reclaim_mode
  - Reclaim memory from local node vs allocating from next node

  - #echo 0 > /proc/sys/vm/zone_reclaim_mode
  - Allocate from all nodes before reclaiming memory

- Default is set at boot time based on NUMA factor
- In Red Hat Enterprise Linux 6.6+ and 7+,
  - Default is usually 0 – because this is better for many applications

Visualize NUMA Topology: lstopo


NUMA Node 0
NUMA Node 1
PCI Devices
NUMANode P#0 (64GB)
Socket P#0
L3 (20MB)
L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB)
L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB)
L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB)
Core P#0 Core P#1 Core P#2 Core P#3 Core P#4 Core P#5 Core P#6 Core P#7
PU P#0 PU P#2 PU P#4 PU P#6 PU P#8 PU P#10 PU P#12 PU P#14
PCI 8086:1521
em1
PCI 8086:1521
em2
PCI 8086:1521
em3
PCI 8086:1521
em4
PCI 1000:005b
sda sdb sdc
sdd sde
PCI 8086:10fb
PCI 8086:10fb
p3p2
0.2
PCI 102b:0534
PCI 8086:1d02
NUMANode P#1 (64GB)
Socket P#1
L3 (20MB)
L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB) L2 (256KB)
L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB)
L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB)
Core P#0 Core P#1 Core P#2 Core P#3 Core P#4 Core P#5 Core P#6 Core P#7
PU P#1 PU P#3 PU P#5 PU P#7 PU P#9 PU P#11 PU P#13 PU P#15
PCI 1924:0803
p1p1
PCI 1924:0803
sfc1

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

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

Tools to display CPU and Memory (NUMA)

```
# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                40
On-line CPU(s) list:   0-39
Thread(s) per core:    1
Core(s) per socket:    10
CPU socket(s):         4
NUMA node(s):          4
 . . . .
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              30720K
NUMA node0 CPU(s):     0,4,8,12,16,20,24,28,32,36
NUMA node1 CPU(s):     2,6,10,14,18,22,26,30,34,38
NUMA node2 CPU(s):     1,5,9,13,17,21,25,29,33,37
NUMA node3 CPU(s):     3,7,11,15,19,23,27,31,35,39
```

cpu, core, socket, node info

The cpu numbers for each node

Tools to display CPU and Memory (NUMA)

```
# 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
```

cpus & memory for each node

Relative “node-to-node” latency costs.

Numactl

- The numactl command can launch commands with **static** NUMA memory and execution thread alignment
  - # numactl -m <NODES> -N <NODES> <Workload>
- Can specify devices of interest to process instead of explicit node list
- Numactl can interleave memory for large monolithic workloads
  - # numactl --interleave=all <Workload>

```
# numactl   -m 6-7 -N 6-7    numactl --show
policy: bind
preferred node: 6
physcpubind: 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78
79
cpubind: 6 7
nodebind: 6 7
membind: 6 7
```

```
# numactl   -m netdev:ens6f2 -N netdev:ens6f2    numactl --show
policy: bind
preferred node: 2
physcpubind: 20 21 22 23 24 25 26 27 28 29
cpubind: 2
nodebind: 2
membind: 2
```

```
# numactl   -m file:/data -N file:/data    numactl --show
policy: bind
preferred node: 0
physcpubind: 0 1 2 3 4 5 6 7 8 9
cpubind: 0
nodebind: 0
membind: 0
```

```
# numactl   --interleave=4-7  -N 4-7    numactl --show
policy: interleave
preferred node: 5 (interleave next)
interleavemask: 4 5 6 7
interleavenode: 5
physcpubind: 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58
59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
cpubind: 4 5 6 7
nodebind: 4 5 6 7
membind: 0 1 2 3 4 5 6 7
```

numastat shows need for NUMA management

```
# 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
```

unaligned

```
# 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
```

aligned

Techniques to control placement (cont):

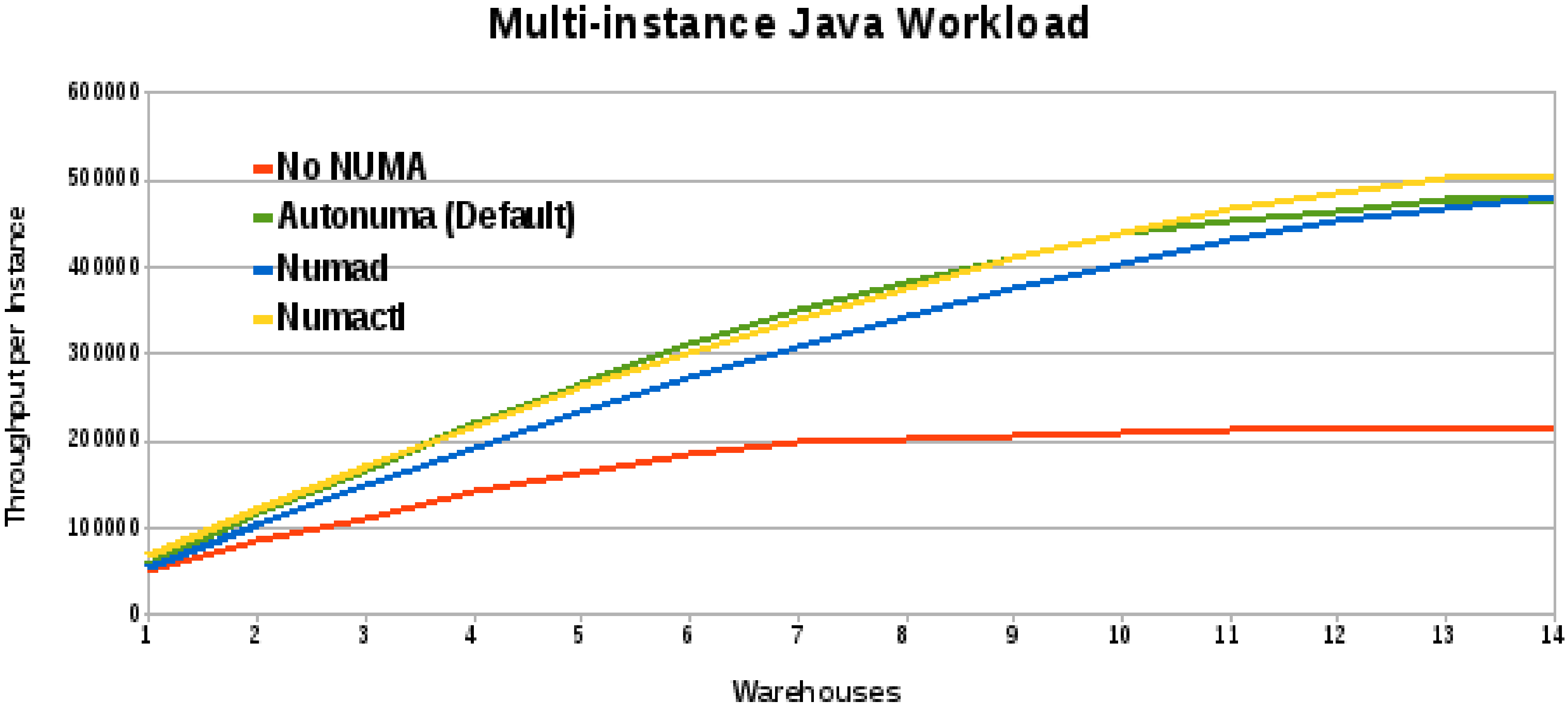
numad:

- User-mode daemon.
- Attempts to locate processes for efficient NUMA locality and affinity.
- Dynamically adjusting to changing system conditions.
- Available in RHEL 6 & 7.

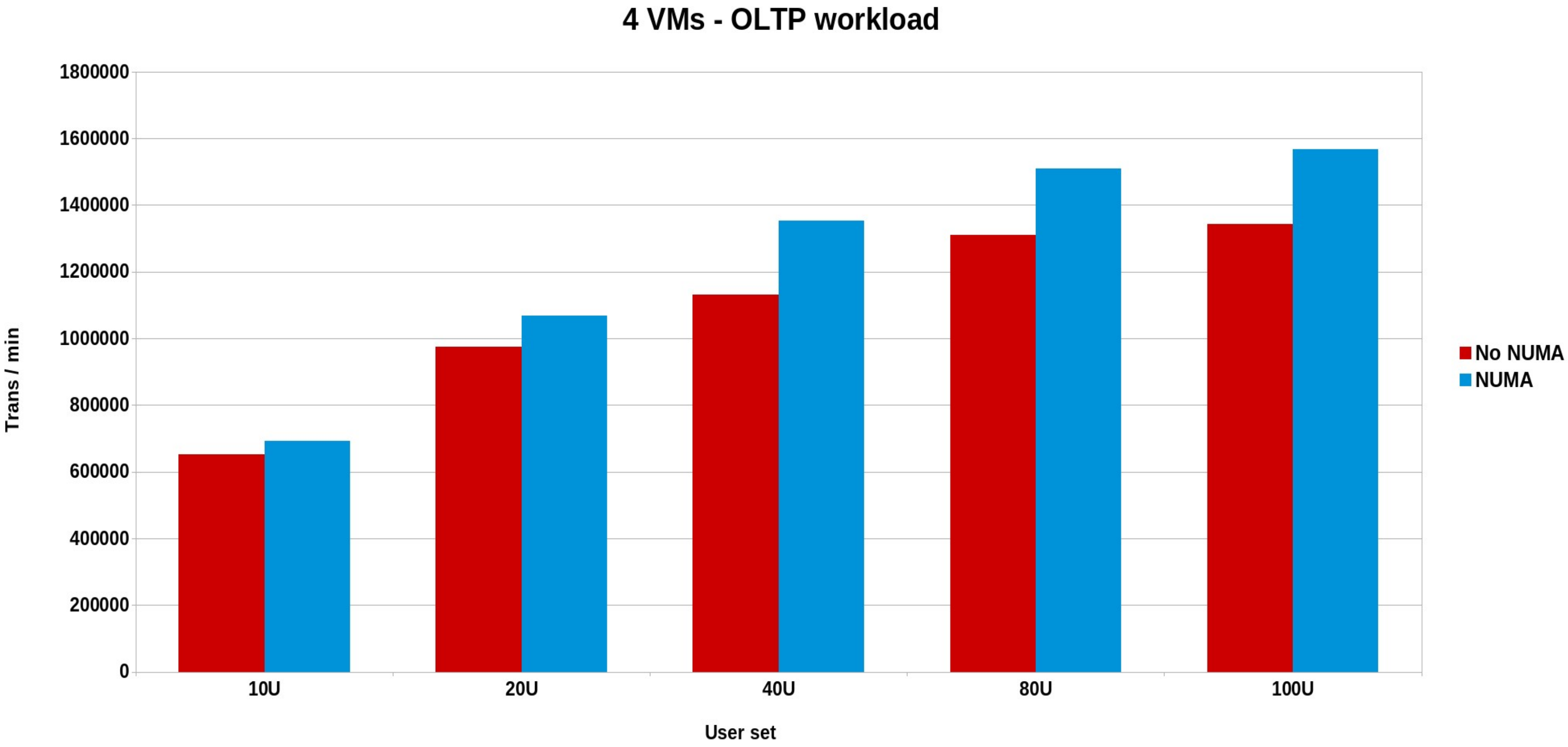
Auto-Numa-Balance kernel scheduler:

- Automatically run programs near their memory, and moves memory near the programs using it.
- Default enabled. Available in RHEL 7+
- Great video on how it works:
  - https://www.youtube.com/watch?v=mjVw_oe1hEA

Numa Multiple Java Workloads - bare-metal



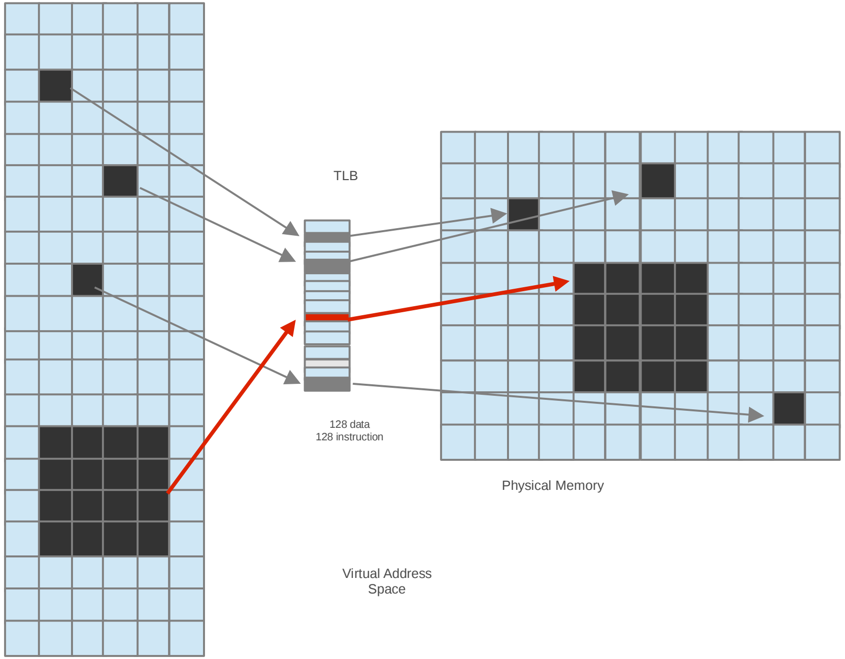
Numa with multiple database KVM VMs



RHEL VM HugePages

RHEL Hugepages/ VM Tuning

- Standard HugePages 2MB
  - Reserve/free via
    - /proc/sys/vm/nr_hugepages
    - /sys/devices/node/* /hugepages/*/nrhugepages
  - Used via hugetlbfs
- GB Hugepages 1GB
  - Reserved at boot time/no freeing
  - RHEL7 allows runtime allocation & freeing
  - Used via hugetlbfs
- Transparent HugePages 2MB
  - On by default via boot args or /sys
  - Used for anonymous memory



Transparent Hugepages

- Disable transparent_hugepages

```
#echo never > /sys/kernel/mm/transparent_hugepages=never
```

```
#time ./memory 15 0
real   0m12.434s
user   0m0.936s
sys    0m11.416s
```

```
# cat /proc/meminfo
MemTotal:       16331124 kB
AnonHugePages:  0 kB
```

- Boot argument: transparent_hugepages=always (enabled by default)
- `#echo always > /sys/kernel/mm/redhat_transparent_hugepage/enabled`

```
#time ./memory 15GB
real    0m7.024s
user    0m0.073s
sys     0m6.847s
```

```
#cat /proc/meminfo
MemTotal:       16331124 kB
AnonHugePages:  15590528 kB
```

SPEEDUP 12.4/7.0 = 1.77x, 56%

2MB standard Hugepages

```
# echo 2000 > /proc/sys/vm/nr_hugepages
# cat /proc/meminfo
MemTotal:       16331124 kB
MemFree:        11788608 kB

HugePages_Total:    2000
HugePages_Free:     2000
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB

# ./hugeshm 1000

# cat /proc/meminfo
MemTotal:       16331124 kB
MemFree:        11788608 kB

HugePages_Total:    2000
HugePages_Free:     1000
HugePages_Rsvd:     1000
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

Boot-time allocated 1GB Hugepages

- Boot arguments
  - default_hugepagesz=1G, hugepagesz=1G, hugepages=8

```
# cat /proc/meminfo | grep HugePages
HugePages_Total:        8
HugePages_Free:         8
HugePages_Rsvd:         0
HugePages_Surp:         0
```

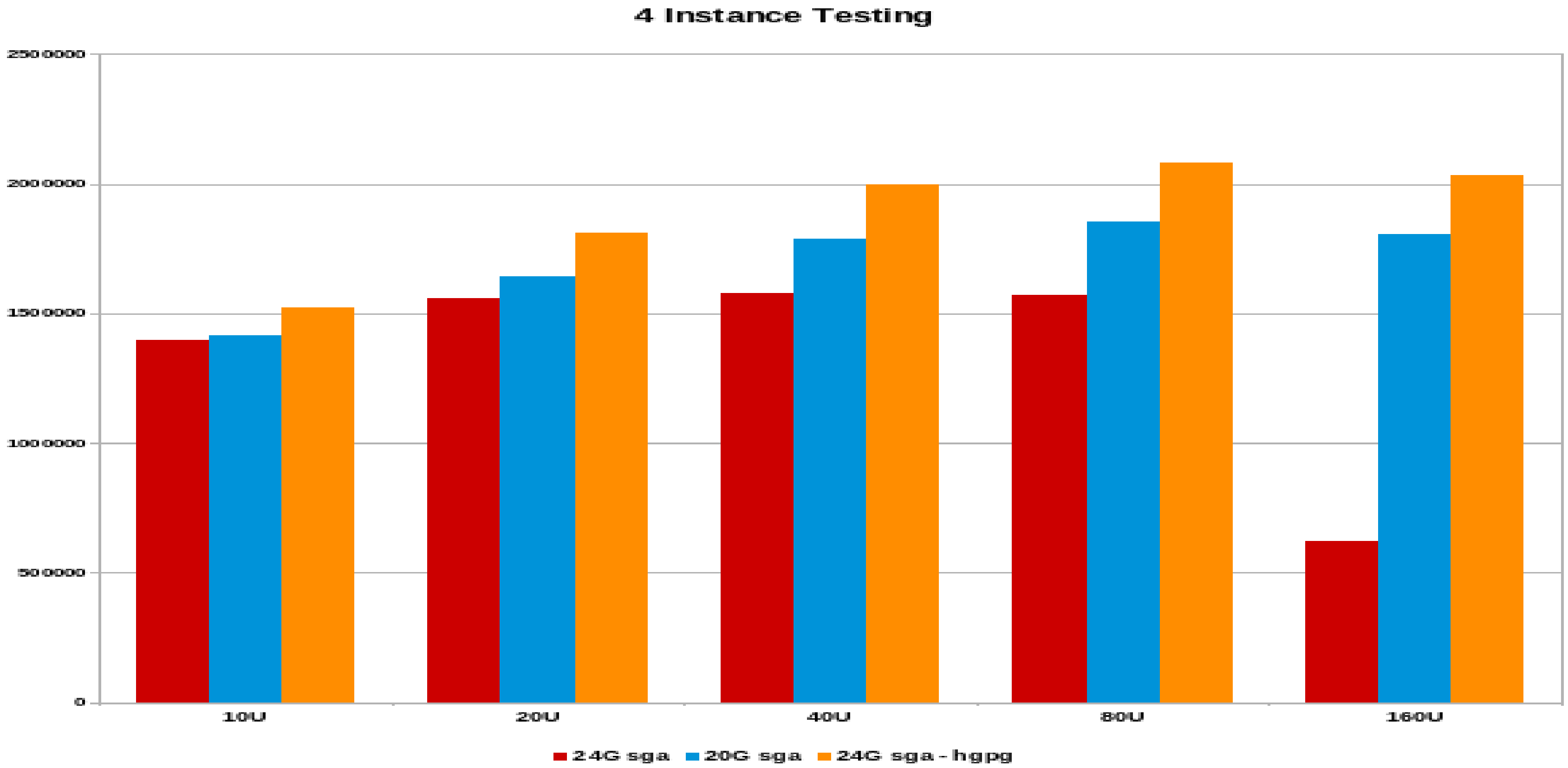
```
#mount -t hugetlbfs none /mnt
# ./mmapwrite /mnt/junk 33
writing 2097152 pages of random junk to file /mnt/junk
wrote 8589934592 bytes to file /mnt/junk
```

```
# cat /proc/meminfo | grep HugePages
HugePages_Total:        8
HugePages_Free:         0
HugePages_Rsvd:         0
HugePages_Surp:         0
```

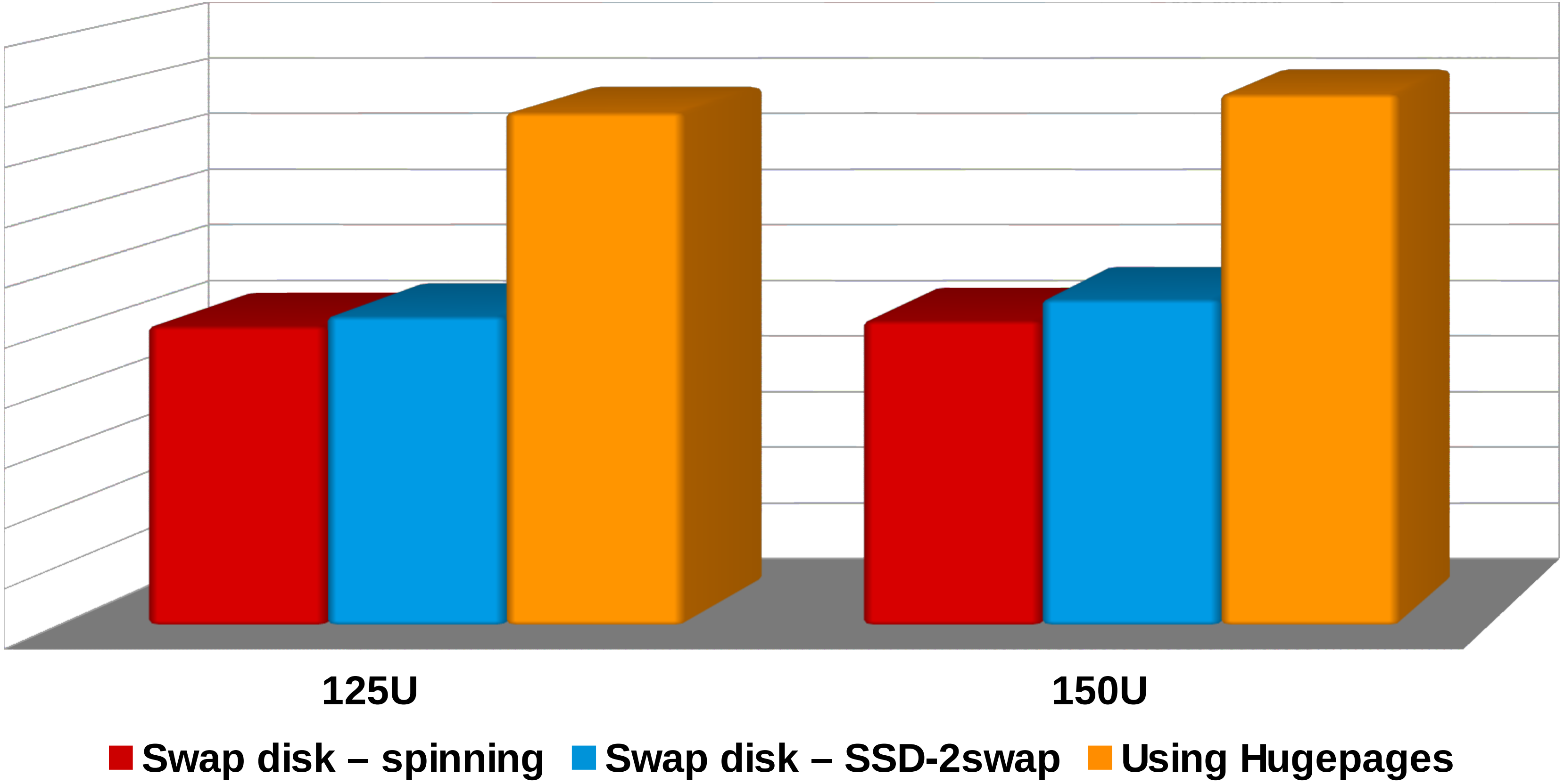
Hugepages - specific node allocation

```
# echo 0 > /proc/sys/vm/nr_hugepages
# cat /proc/meminfo | grep HugePages_Free
HugePages_Free:     0

# echo 1000 > /proc/sys/vm/nr_hugepages
# cat /proc/meminfo | grep HugePages_Free
HugePages_Free:     1000
# cat /sys/devices/system/node/node*/hugepages/hugepages-2048kB/nr_hugepages
500
500

# echo 0 > /proc/sys/vm/nr_hugepages
# echo 1000 > /sys/devices/system/node/node0/hugepages/hugepages-2048kB/nr_hugepages
# cat /proc/meminfo | grep HugePages_Free
HugePages_Free:     1000
# cat /sys/devices/system/node/node*/hugepages/hugepages-2048kB/nr_hugepages
1000
0
```

Memory Tuning – huge pages on Bare Metal



Avoid swapping - Use huge pages



RHEL Control Group - Cgroups

Cgroup default mount points

RHEL6

```
# cat /etc/cgconfig.conf

mount {
          cpuset          = /cgroup/cpuset;
          cpu     = /cgroup/cpu;
          cpuacct         = /cgroup/cpuacct;
          memory          = /cgroup/memory;
          devices         = /cgroup/devices;
          freezer         = /cgroup/freezer;
          net_cls         = /cgroup/net_cls;
          blkio   = /cgroup/blkio;
 }
```

RHEL7

/sys/fs/cgroup/

RHEL6

```
# ls -l /cgroup
drwxr-xr-x 2 root root 0 Jun 21 13:33 blkio
drwxr-xr-x 3 root root 0 Jun 21 13:33 cpu
drwxr-xr-x 3 root root 0 Jun 21 13:33 cpuacct
drwxr-xr-x 3 root root 0 Jun 21 13:33 cpuset
drwxr-xr-x 3 root root 0 Jun 21 13:33 devices
drwxr-xr-x 3 root root 0 Jun 21 13:33 freezer
drwxr-xr-x 3 root root 0 Jun 21 13:33 memory
drwxr-xr-x 2 root root 0 Jun 21 13:33 net_cls
```

RHEL7

```
#ls -l /sys/fs/cgroup/
drwxr-xr-x. 2 root root  0 Mar 20 16:40 blkio
drwxr-xr-x. 2 root root  0 Mar 20 16:40 cpu,cpuacct
drwxr-xr-x. 2 root root  0 Mar 20 16:40 cpuset
drwxr-xr-x. 2 root root  0 Mar 20 16:40 devices
drwxr-xr-x. 2 root root  0 Mar 20 16:40 freezer
drwxr-xr-x. 2 root root  0 Mar 20 16:40 hugetlb
drwxr-xr-x. 3 root root  0 Mar 20 16:40 memory
drwxr-xr-x. 2 root root  0 Mar 20 16:40 net_cls
drwxr-xr-x. 2 root root  0 Mar 20 16:40 perf_event
drwxr-xr-x. 4 root root  0 Mar 20 16:40 systemd
```

Cgroup how-to

Create a 2GB/4CPU subset of a 16GB/8CPU system

```
# numactl --hardware
# mount -t cgroup xxx /cgroups
# mkdir -p /cgroups/test
# cd /cgroups/test
# echo 0 > cpuset.mems
# echo 0-3 > cpuset.cpus
# echo 2G > memory.limit_in_bytes
# echo $$ > tasks
```

cgroups

```
# echo 0-3 > cpuset.cpus
# runmany 20MB 110procs &
# top -d 5
top - 12:24:13 up  1:36,  4 users,  load average: 22.70, 5.32, 1.79
Tasks: 315 total,  93 running, 222 sleeping,   0 stopped,   0 zombie
Cpu0  : 100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  : 100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  : 100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  : 100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu4  :  0.4%us,  0.6%sy,  0.0%ni, 98.8%id,  0.0%wa,  0.0%hi,  0.2%si,  0.0%st
Cpu5  :  0.4%us,  0.0%sy,  0.0%ni, 99.2%id,  0.0%wa,  0.0%hi,  0.4%si,  0.0%st
Cpu6  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu7  :  0.0%us,  0.0%sy,  0.0%ni, 99.8%id,  0.0%wa,  0.0%hi,  0.2%si,  0.0%st
```

Correct NUMA bindings

```
# echo 0 > cpuset.mems
# echo 0-3 > cpuset.cpus
# numastat
                          node0           node1
numa_hit                1648772          438778
numa_miss                 23459         2134520
local_node              1648648          423162
other_node                23583         2150136

# /common/lwoodman/code/memory 4G
faulting took 1.616062s
touching took 0.364937s

# numastat
                          node0           node1
numa_hit                2700423          439550
numa_miss                 23459         2134520
local_node              2700299          423934
other_node                23583         2150136
```

Incorrect NUMA bindings

```
# echo 1 > cpuset.mems
# echo 0-3 > cpuset.cpus
# numastat
                          node0           node1
numa_hit                1623318          434106
numa_miss                 23459         1082458
local_node              1623194          418490
other_node                23583         1098074

# /common/lwoodman/code/memory 4G
faulting took 1.976627s
touching took 0.454322s

# numastat
                          node0           node1
numa_hit                1623341          434147
numa_miss                 23459         2133738
local_node              1623217          418531
other_node                23583         2149354
```

cpu.shares default

```
# cat cpu.shares
1024
```

```
top - 10:04:19 up 13 days, 17:24, 11 users,  load average: 8.41, 8.31, 6.17

PID USER      PR  NI   VIRT   RES   SHR S %CPU %MEM     TIME
20104 root    20   0   4160   360   284 R  99.4  0.0  12:35.83 useless
20103 root    20   0   4160   356   284 R  91.4  0.0  12:34.78 useless
20105 root    20   0   4160   360   284 R  90.4  0.0  12:33.08 useless
20106 root    20   0   4160   360   284 R  88.4  0.0  12:32.81 useless
20102 root    20   0   4160   360   284 R  86.4  0.0  12:35.29 useless
20107 root    20   0   4160   356   284 R  85.4  0.0  12:33.51 useless
20110 root    20   0   4160   360   284 R  84.8  0.0  12:31.87 useless
20108 root    20   0   4160   360   284 R  82.1  0.0  12:30.55 useless
20410 root    20   0   4160   360   284 R  91.4  0.0   0:18.51 useful
```

cpu.shares throttled

```
# echo 10 > cpu.shares
```

```
top - 09:51:58 up 13 days, 17:11, 11 users,  load average: 7.14, 5.78, 3.09

PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME
20102 root    20   0    4160    360    284 R 100.0  0.0   0:17.45 useless
20103 root    20   0    4160    356    284 R 100.0  0.0   0:17.03 useless
20107 root    20   0    4160    356    284 R 100.0  0.0   0:15.57 useless
20104 root    20   0    4160    360    284 R  99.8  0.0   0:16.66 useless
20105 root    20   0    4160    360    284 R  99.8  0.0   0:16.31 useless
20108 root    20   0    4160    360    284 R  99.8  0.0   0:15.19 useless
20110 root    20   0    4160    360    284 R  99.4  0.0   0:14.74 useless
20106 root    20   0    4160    360    284 R  99.1  0.0   0:15.87 useless
20111 root    20   0    4160    356    284 R   1.0  0.0   0:00.08 useful
```

cpu.cfs_quota_us unlimited

```
# cat cpu.cfs_period_us
100000
# cat cpu.cfs_quota_us
-1
top - 10:11:33 up 13 days, 17:31, 11 users,  load average: 6.21, 7.78, 6.80

PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
20614 root     20   0    4160     360    284  R   100.0   0.0       0:30.77 useful
```

```
# echo 1000 > cpu.cfs_quota_us
```

```
top - 10:16:55 up 13 days, 17:36, 11 users,  load average: 0.07, 2.87, 4.93

PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
20645 root     20   0     4160    360     284 R     1.0     0.0        0:01.54 useful
```

Cgroup OOMkills

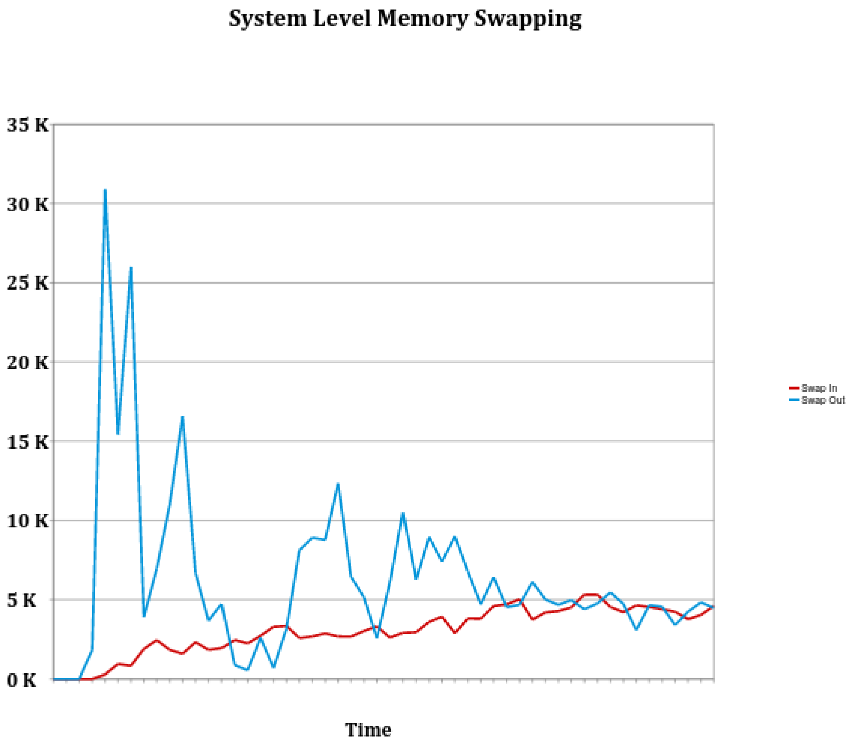
```
# mkdir -p /sys/fs/cgroup/memory/test
# echo 1G > /sys/fs/cgroup/memory/test/memory.limit_in_bytes
# echo 2G > /sys/fs/cgroup/memory/test/memory.memsw.limit_in_bytes
# echo $$ > /sys/fs/cgroup/memory/test/tasks

# ./memory 16G
size = 10485760000
touching 2560000 pages
Killed
# vmstat 1
...
 0  0  52224 1640116      0 3676924    0    0       0     0   202  487  0  0 100  0  0
 1  0  52224 1640116      0 3676924    0    0       0     0   162  316  0  0 100  0  0
 0  1 248532 587268       0 3676948   32 196312     32 196372   912  974  1  4 88   7  0
 0  1 406228 586572       0 3677308    0 157696      0 157704   624  696  0  1 87  11  0
 0  1 568532 585928       0 3676864    0 162304      0 162312   722 1039  0  2 87  11  0
 0  1 729300 584744       0 3676840    0 160768      0 160776   719 1161  0  2 87  11  0
 1  0 885972 585404       0 3677008    0 156844      0 156852   754 1225  0  2 88  10  0
 0  1 1042644 587128      0 3676784    0 156500      0 156508   747 1146  0  2 86  12  0
 0  1 1169708 587396      0 3676748    0 127064      4 127836   702 1429  0  2 88  10  0
 0  0  86648 1607092      0 3677020  144    0     148     0   491 1151  0  1 97   1  0
```

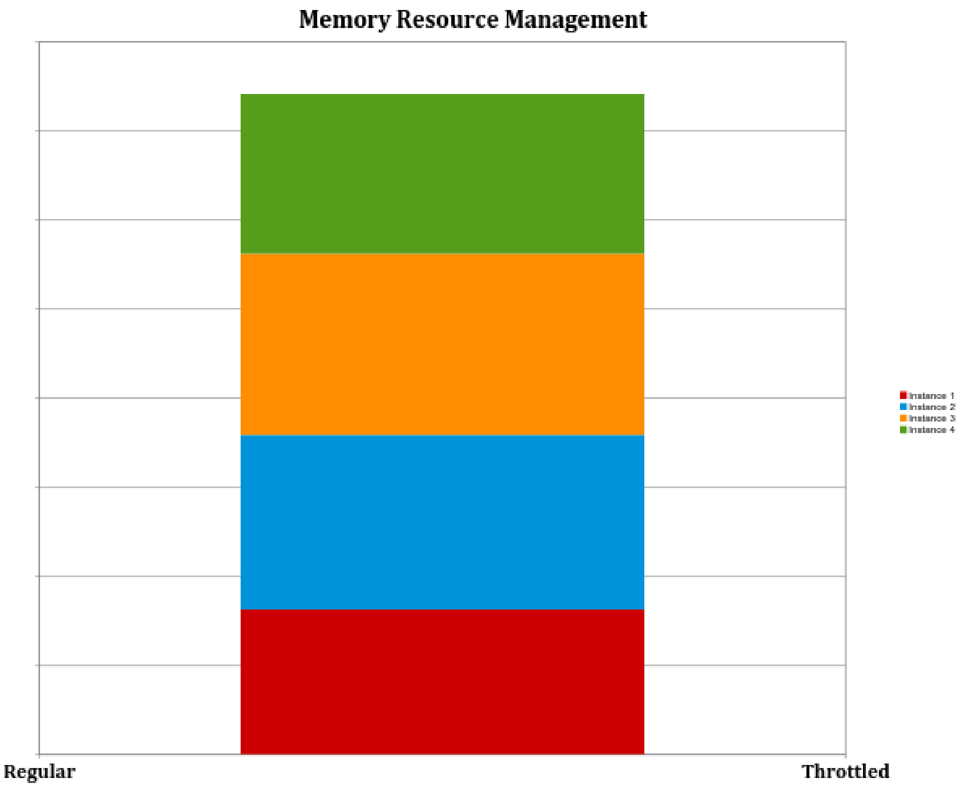
Cgroup OOMkills (continued)

```
# vmstat 1
...
 0  0  52224 1640116      0 3676924    0    0       0     0   202  487  0  0 100  0  0
 1  0  52224 1640116      0 3676924    0    0       0     0   162  316  0  0 100  0  0
 0  1 248532 587268       0 3676948   32 196312     32 196372  912  974  1  4 88   7  0
 0  1 406228 586572       0 3677308    0 157696      0 157704  624  696  0  1 87  11  0
 0  1 568532 585928       0 3676864    0 162304      0 162312  722 1039  0  2 87  11  0
 0  1 729300 584744       0 3676840    0 160768      0 160776  719 1161  0  2 87  11  0
 1  0 885972 585404       0 3677008    0 156844      0 156852  754 1225  0  2 88  10  0
 0  1 1042644 587128      0 3676784    0 156500      0 156508  747 1146  0  2 86  12  0
 0  1 1169708 587396      0 3676748    0 127064      4 127836  702 1429  0  2 88  10  0
 0  0  86648 1607092      0 3677020  144    0    148     0   491 1151  0  1 97   1  0
...
```

```
# dmesg
...
[506858.413341] Task in /test killed as a result of limit of /test
[506858.413342] memory: usage 1048460kB, limit 1048576kB, failcnt 295377
[506858.413343] memory+swap: usage 2097152kB, limit 2097152kB, failcnt 74
[506858.413344] kmem: usage 0kB, limit 9007199254740991kB, failcnt 0
[506858.413345] Memory cgroup stats for /test: cache:0KB rss:1048460KB rss_huge:10240KB
mapped_file:0KB swap:1048692KB inactive_anon:524372KB active_anon:524084KB inactive_file:0KB
active_file:0KB unevictable:0KB
```

Cgroup – Application Isolation





Even though one application does not have resources and starts swapping, other applications are not affected

Summary - Red Hat Enterprise Linux NUMA

- **RHEL6 – NUMAD - With Red Hat Enterprise Linux**
  - NUMAD can significantly improve performance and automate NUMA management on systems with server consolidation or replicated parallel workloads.

- **RHEL7, Auto-NUMA-Balance**
  - Works well for most applications out of the box!
  - Use NUMAstat and NUMActl tools to measure and/or fine control your application on RHEL.
  - Use HugePages for wired-down shared-memory (DB/Java), 2MB or 1GB

- **Q+A at “Meet The Experts” - Free as in Soda/Beer/Wine**

Performance Whitepapers

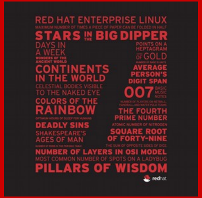
- Performance Tuning of Satellite 6.1 and Capsules https://access.redhat.com/articles/2356131
- OpenShift v3 Scaling, Performance and Capacity Planning https://access.redhat.com/articles/2191731
- Performance and Scaling your RHEL OSP 7 Cloud https://access.redhat.com/articles/2165131
- RHEL OSP 7: Cinder Volume Performance on RHCS 1.3 (Ceph) https://access.redhat.com/articles/2061493
- RHGS 3.1 Performance Brief (Gluster) https://access.redhat.com/articles/1982243

- Red Hat Performance Tuning Guide
- Red Hat Low Latency Tuning Guide
- Red Hat Virtualization Tuning Guide
- RHEL Blog / Developer Blog

LEARN MORE ABOUT IT OPTIMIZATION AT THE RED HAT BOOTH

Location: Booth #511, Moscone West

View technical demos, interact with our technology experts, get answers to your most pressing questions, and acquire some of our best shirts and stickers!


RED HAT ENTERPRISE LINUX
MAXIMUM NUMBER OF TIMES A PIECE OF PAPER CAN BE FOLDED IN HALF
STARS IN THE BIG DIPPER
DAYS IN A WEEK
WONDERS OF THE ANCIENT WORLD
CONTINENTS IN THE WORLD
CELESTIAL BODIES VISIBLE TO THE NAKED EYE
COLORS OF THE RAINBOW
OPTIMUM HOURS OF SLEEP FOR HUMANS
DEADLY SINS
SHAKESPEARE'S AGES OF MAN
NUMBER OF ROWS IN THE PERIODIC TABLE
POINTS ON A HEPTAGRAM
OF GOLD
NUMBER OF BASE SI UNITS
AVERAGE PERSON'S DIGIT SPAN
007
BASIC MUSIC NOTES
NUMBER OF PLAYERS ON NETBALL, HANDBALL, AND WATER POLO TEAMS
THE FOURTH PRIME NUMBER
ATOMIC NUMBER OF NITROGEN
SQUARE ROOT OF FORTY-NINE
THE SUM OF OPPOSITE SIDES OF DICE
NUMBER OF LAYERS IN OSI MODEL
MOST COMMON NUMBER OF SPOTS ON A LADYBUG
PILLARS OF WISDOM
redhat

RED HAT SUMMIT

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos

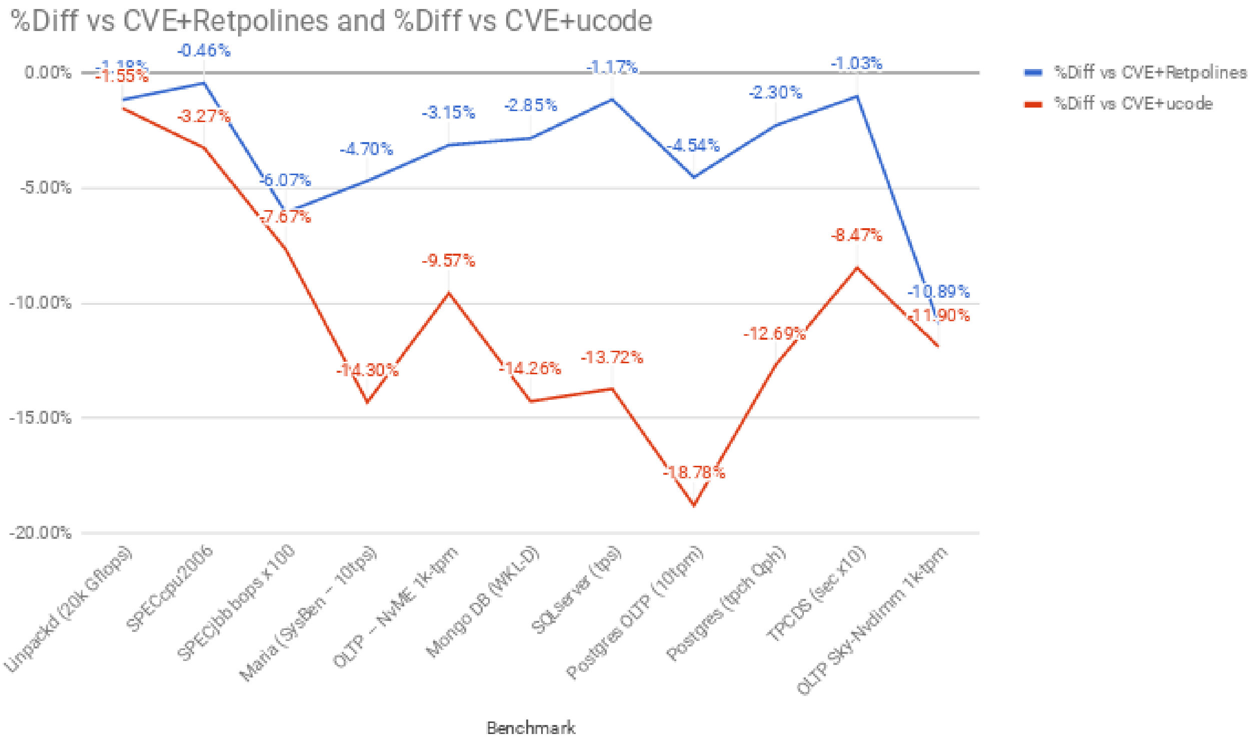
RED HAT SUMMIT

LEARN. NETWORK.
EXPERIENCE
OPEN SOURCE.

Spectre and Meltdown Application Perf Impact (kbase article - https://access.redhat.com/articles/3307751)

- Measureable: Ucode only 8-19%, **updated with retpoline (4-10%)**
  - Highly cached random memory, with buffered I/O, OLTP database workloads, and benchmarks with high kernel-to-user space transitions are impacted now within 10%. Scale out HPC MPI environments with significant message passing fall into this category. Examples include OLTP Workloads (tpc), sysbench, pgbench, tpm.
- Modest: Ucode only 3-7%, **updated with retpoline (2-5%)**
  - Database analytics, Decision Support System (DSS), and Java VMs are impacted less than the “Measurable” category. These applications may have significant sequential disk or network traffic, but kernel/device drivers are able to aggregate requests to moderate level of kernel-to-user transitions. Examples include SPECjbb2005, Queries/Hour and overall analytic timing (sec).
- Small: Ucode only 2-5%, **updated with retpoline (1-3%)**
  - HPC (High Performance Computing single system) Single system CPU-intensive workloads are affected the least with only 1-3% performance impact because jobs run mostly in user space and are scheduled using cpu-pinning or numa-control. Examples:Linpack NxN on x86 and SPECcpu2006.

Spectre / Meltdown Application Perf Impact in RHEL7.4z



RHEL Performance Workload Coverage

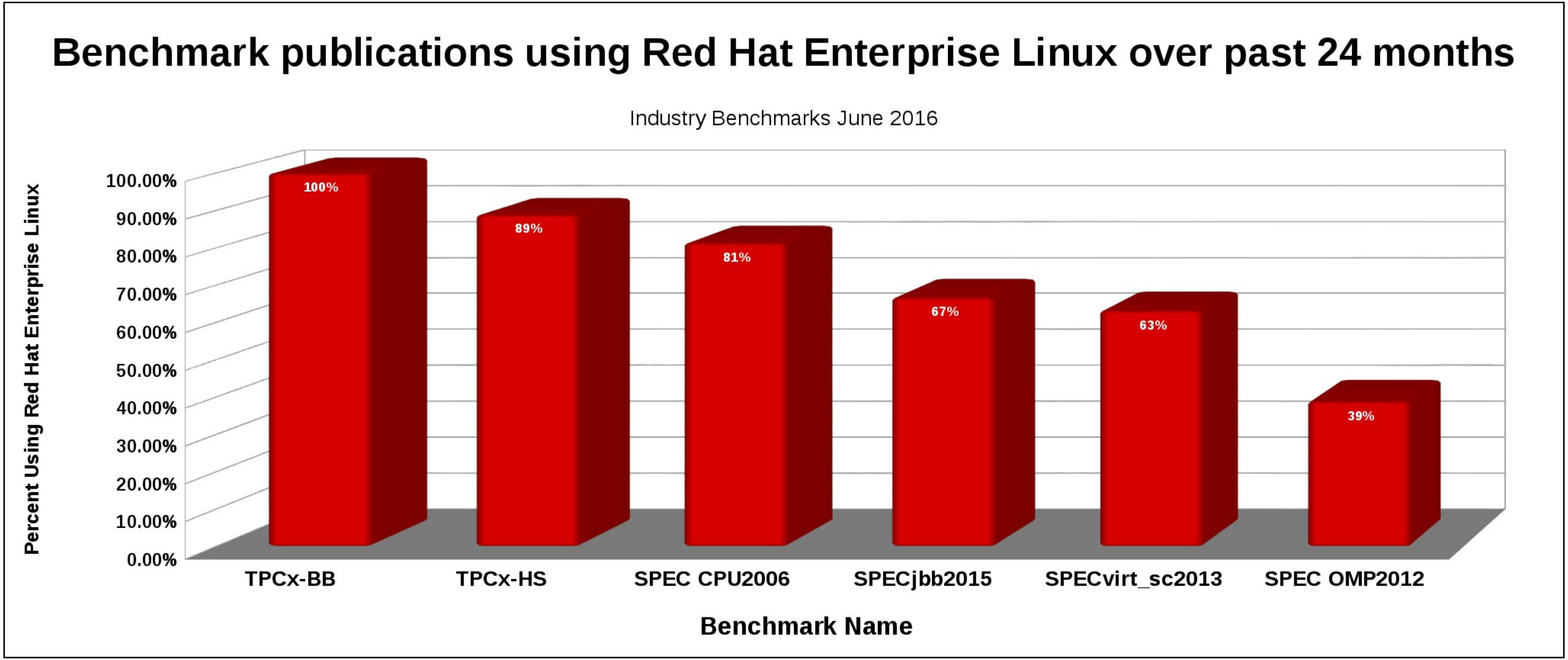
(bare metal, KVM virt w/ RHEV and/or OSP, LXC Kube/OSEand Industry Standard Benchmarks)

- **MicroBenchmarks – 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, Atomic Containers
  - White box AMD/Intel, with our OEM partners

- **Application Performance**
  - Linpack MPI, HPC workloads
  - AIM 7 – shared, filesystem, db, compute
  - Database: DB2, Oracle 11/12, Sybase 15.x , MySQL, MariaDB, Postgrs, MongoDB
  - OLTP – TPC-C, TPC-VMS
  - DSS – TPC-H/xDS
  - Big Data – TPCx-HS, Bigbench
  - SPEC cpu, jbb, sfs, virt, cloud
  - SAP – SLCS, SD
  - STAC = FSI (STAC-N)
  - SAS mixed Analytic, SAS grid (gfs2)

RHEL / Intel Benchmarks Broadwell EP/EX

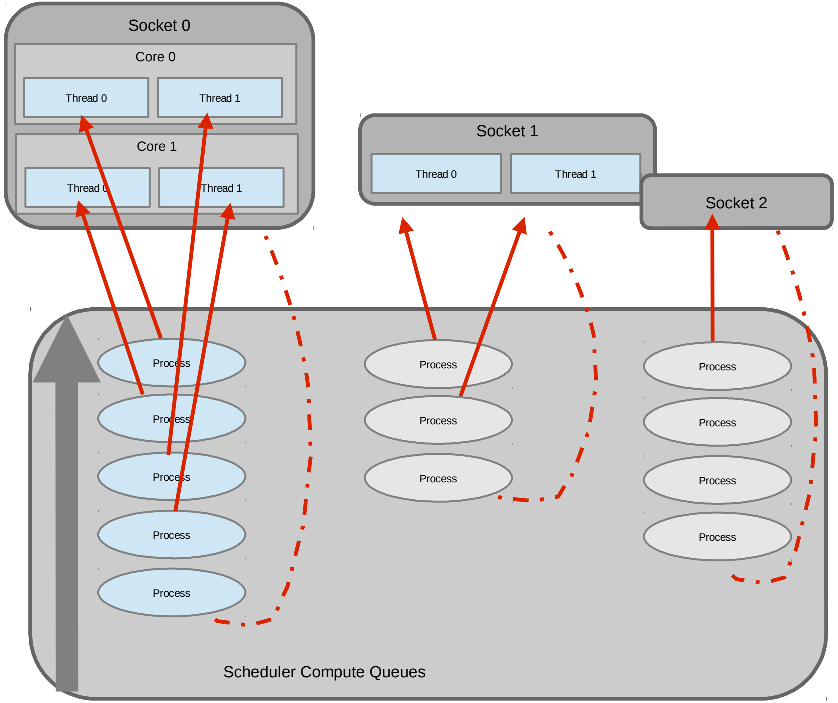
(http://rhelblog.redhat.com/2016/06/06/red-hat-delivers-high-performance-on-critical-enterprise-workloads-with-the-latest-intel-xeon-e7-v4-processor-family/)



RHEL CFS Scheduler

RHEL Scheduler Tunables

- Implements multiple red/black trees as
- run queues for sockets and cores (as opposed to one run queue per processor or per system)
-
- RHEL tunables
  - sched_min_granularity_ns
  - sched_wakeup_granularity_ns
  - sched_migration_cost
  - sched_child_runs_first
  - sched_latency_ns
-



Finer Grained Scheduler Tuning

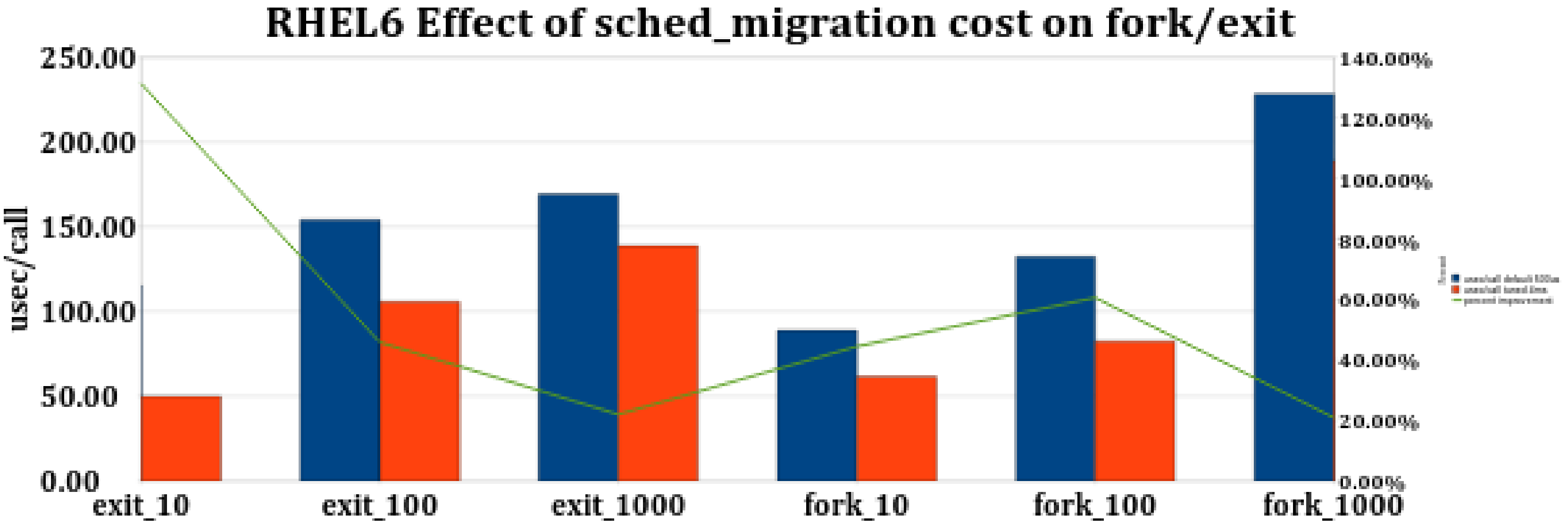
- RHEL6/7 Tuned-adm will increase quantum on par with RHEL5
  - echo 10000000 > /proc/sys/kernel/sched_min_granularity_ns
    - Minimal preemption granularity for CPU bound tasks.
    - See sched_latency_ns for details. The default value is 4000000 (ns).
  - echo 15000000 > /proc/sys/kernel/sched_wakeup_granularity_ns
    - The wake-up preemption granularity.
    - Increasing this variable reduces wake-up preemption, reducing disturbance of compute bound tasks.
    - Decreasing it improves wake-up latency and throughput for latency critical tasks, particularly when a short duty cycle load component must compete with CPU bound components. The default value is 5000000 (ns).

Load Balancing

- Scheduler tries to keep all CPUs busy by moving tasks form overloaded CPUs to idle CPUs
- Detect using “perf stat”, look for excessive “migrations”
- **/proc/sys/kernel/sched_migration_cost_ns**
  - Amount of time after the last execution that a task is considered to be “cache hot” in migration decisions. A “hot” task is less likely to be migrated, so increasing this variable reduces task migrations. The default value is 500000 (ns).
  - If the CPU idle time is higher than expected when there are runnable processes, try reducing this value. If tasks bounce between CPUs or nodes too often, try increasing it.
- Rule of thumb – increase by **2-10x** to reduce load balancing (tuned does this)
- Use 10x on large systems when many CGROUPs are actively used (ex: RHEV/ KVM/RHOS)

fork() behavior

- sched_child_runs_first
  - Controls whether parent or child runs first
  - Default is 0: parent continues before children run.
  - Default is different than RHEL5


RHEL6 Effect of sched_migration cost on fork/exit
usec/call
exit_10
exit_100
exit_1000
fork_10
fork_100
fork_1000