



Performance Analysis and Tuning

Larry Woodman - Senior Consulting Engineer / Kernel VM

Joe Mario - Senior Principal Performance Engineer / RHEL / NUMA

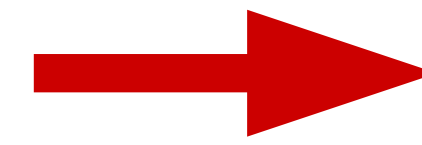
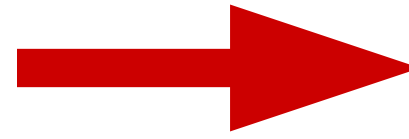
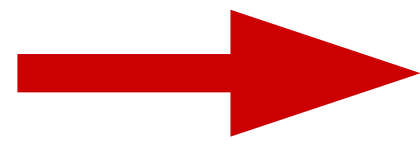
Agenda: Performance Analysis Tuning Part I

- **RHEL Evolution 5->6->7 – Hybrid Clouds Atomic / OSE / RHOP**
- **NonUniform Memory Access (NUMA)**
 - **What is NUMA, RHEL Architecture, Auto-NUMA-Balance**
- **Network Performance and Latency-performance noHZ_full**
- **System Performance/Tools**
 - **Tuned, Perf**

Agenda: Performance Analysis Tuning Part II

- **Scheduler tunables – lock change (MCS)**
- **Cgroups cpuset, memory, network and IO**
 - **Use to prevent IO from consuming 95% of memory**
 - **Used by RHEV w/ KVM and OSE w/ Atomic**
- **HugePages**
 - **Static, Transparent, variable sized 4K/2MB/1GB**
- **Disk and Filesystem IO - Throughput-performance - RHS / Cloud**

Red Hat Enterprise Linux Performance Evolution



RHEL5

Static Hugepages

Ktune – on/off

CPU Affinity
(taskset)

NUMA Pinning
(numactl)

Irqbalance

RHEL6

Transparent HugePage

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

RHEV – out-of-the-box
virt-host/guest

RHEL OSP – blueprints

Tuned, Numa pinning
NIC – jumbo sriov

RHEL Atomic
Host/Atomic Enterprise

RH OpenShift v3

Cloud Forms

RHEL Performance Workload Coverage

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

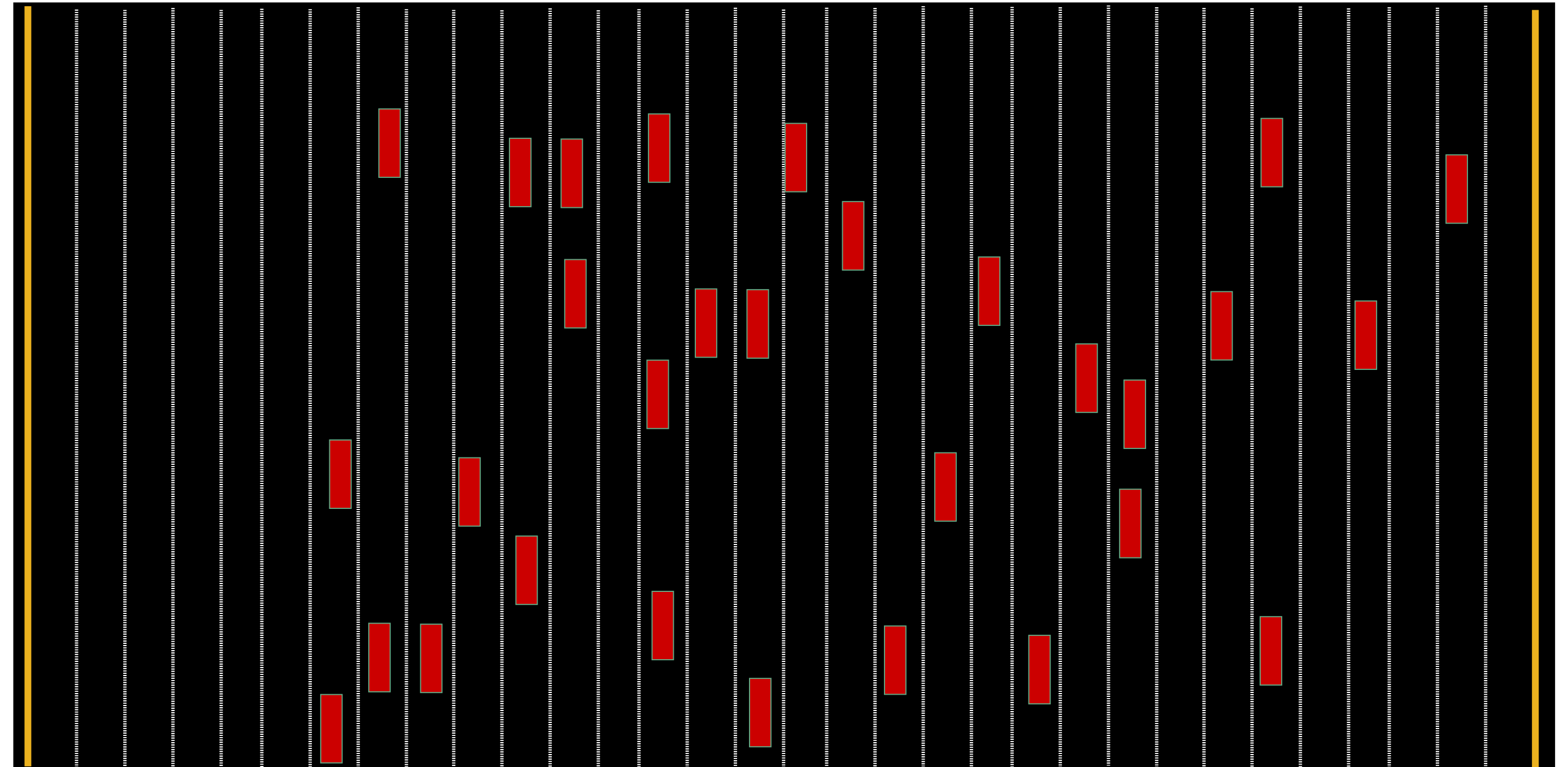
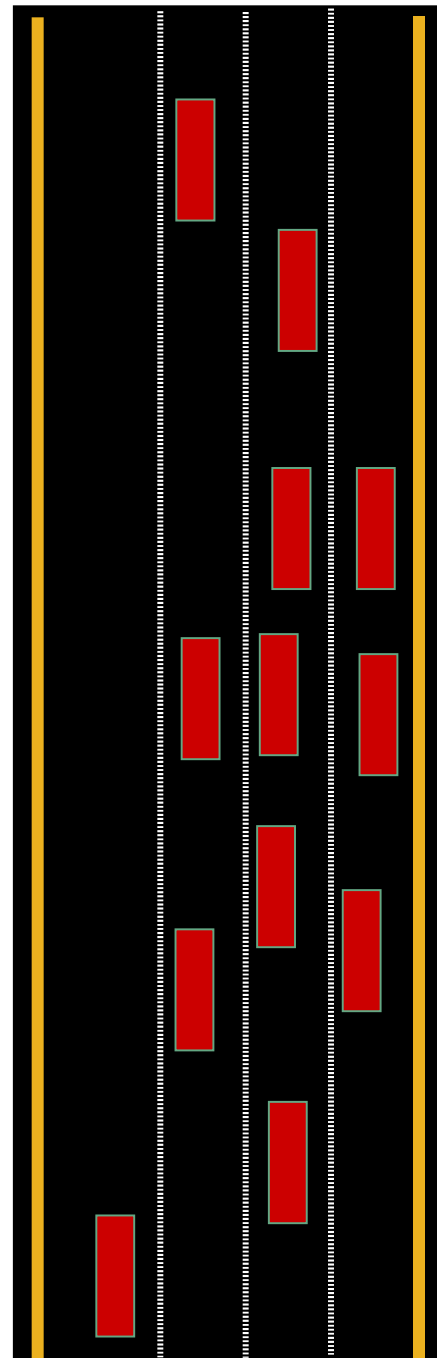
Benchmarks – code path coverage

- CPU – linpack, Imbench
- Memory – Imbench, 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, Postgres, 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)

Performance Metrics - Latency==Speed - Throughput==Bandwidth



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

- 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

tuned is a tool to dynamically tune
Red Hat Enterprise Linux.

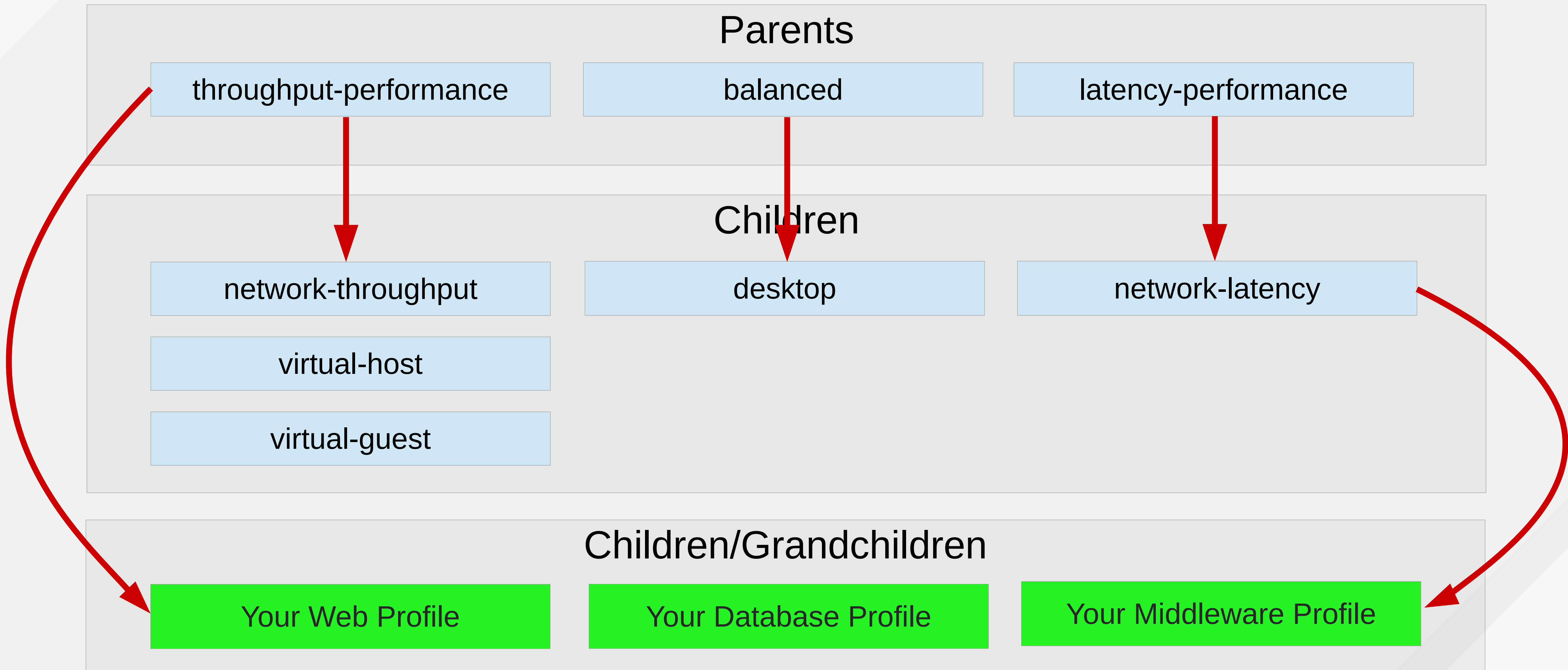
You could improve workload performance
by applying one of the predefined profiles or
use those that you've written yourself

Tuned Overview

- Installed by default
- Auto-set Profiles
- Single config file
- Inheritance/Hooks
- bootloader/cmdline configs
- New Profiles since last year
 - Realtime
 - NFV
 - RHEL Atomic Host
 - OpenShift
 - Oracle

See `man tuned-profiles` for profile definitions

Tuned: Your Custom Profiles



Tuned Profile Examples

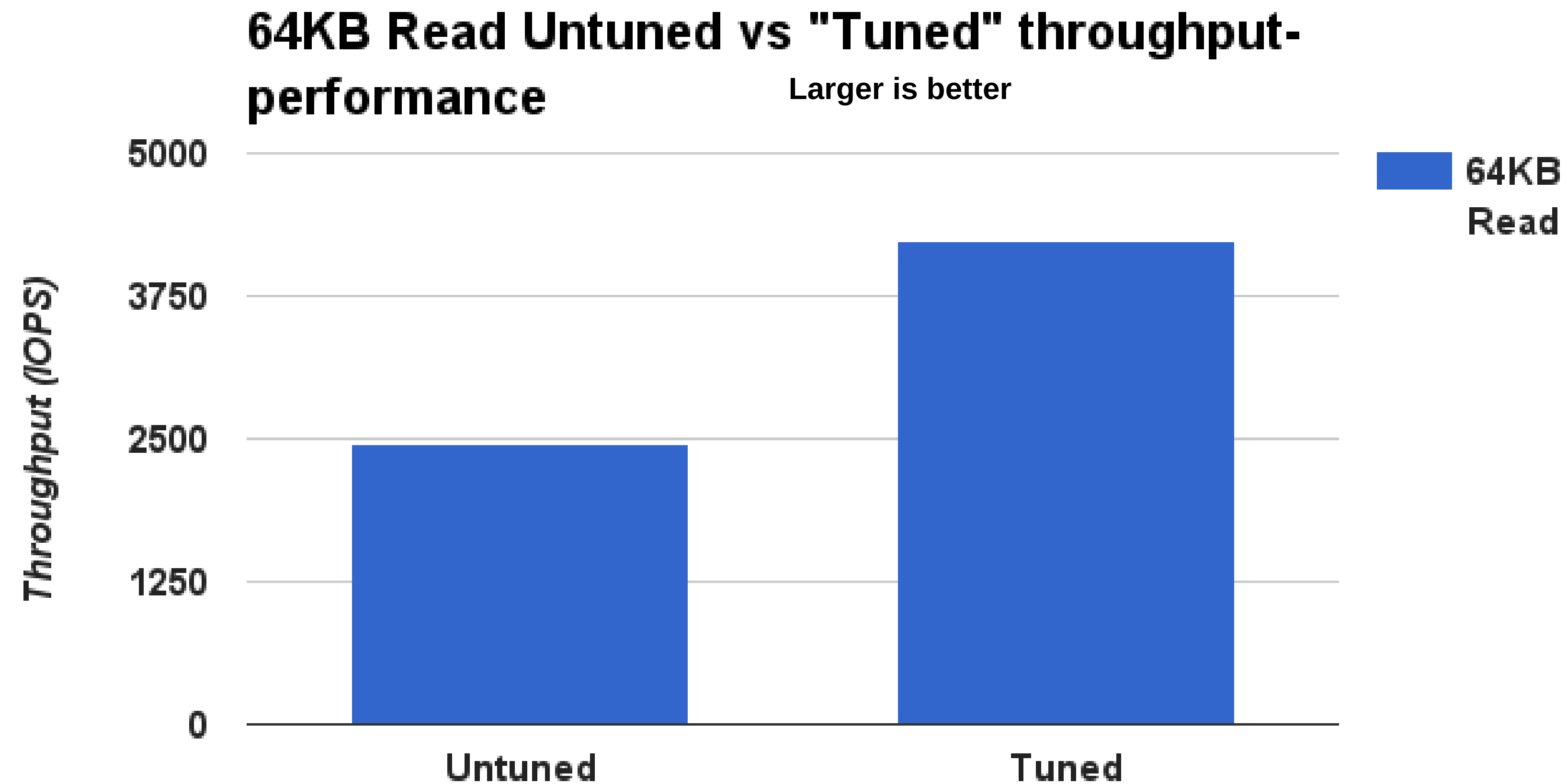
throughput-performance

```
governor=performance
energy_perf_bias=performance
min_perf_pct=100
transparent_hugepages=always
readahead=>4096
sched_min_granularity_ns = 10000000
sched_wakeup_granularity_ns = 15000000
vm.dirty_ratio = 40
vm.dirty_background_ratio = 10
vm.swappiness=10
```

latency-performance

```
force_latency=1
governor=performance
energy_perf_bias=performance
min_perf_pct=100
kernel.sched_min_granularity_ns=10000000
vm.dirty_ratio=10
vm.dirty_background_ratio=3
vm.swappiness=10
kernel.sched_migration_cost_ns=5000000
```

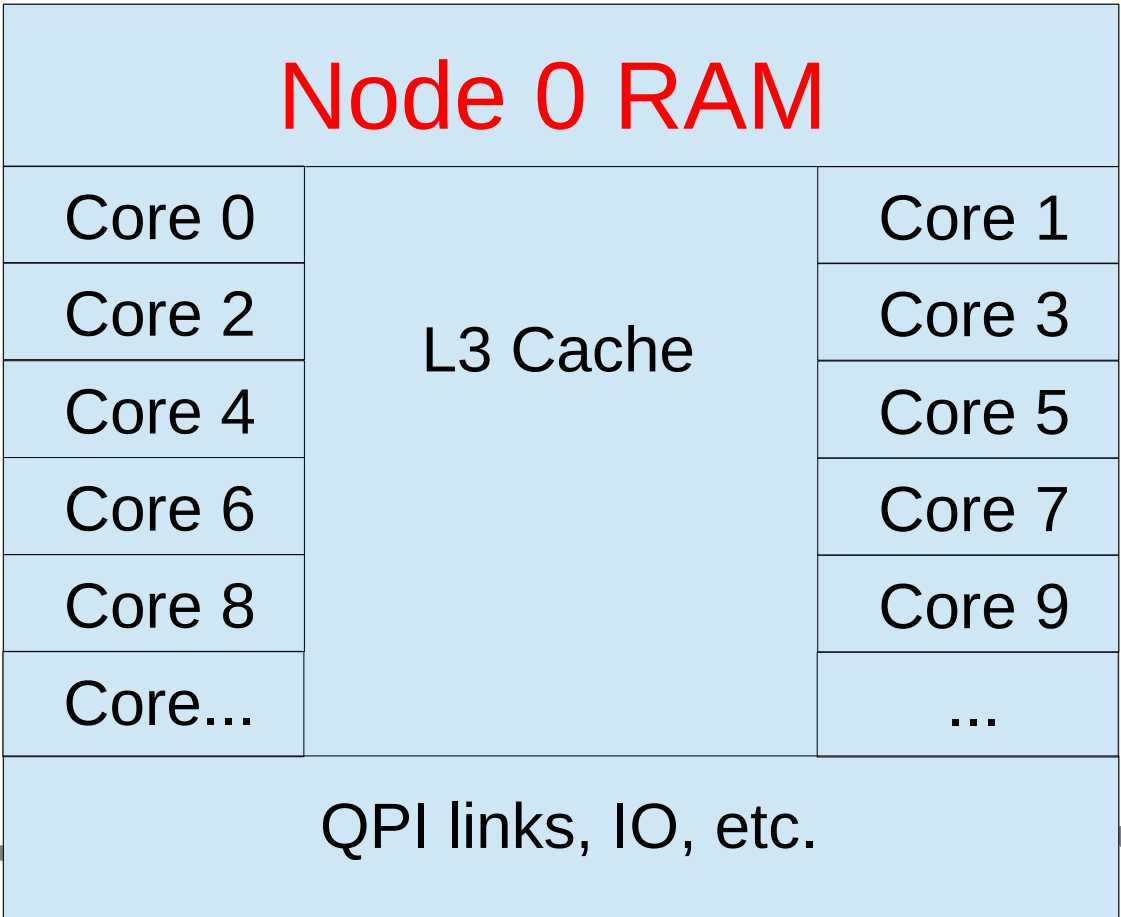
Tuned: Storage Performance Boost: throughput-performance (default in RHEL7)



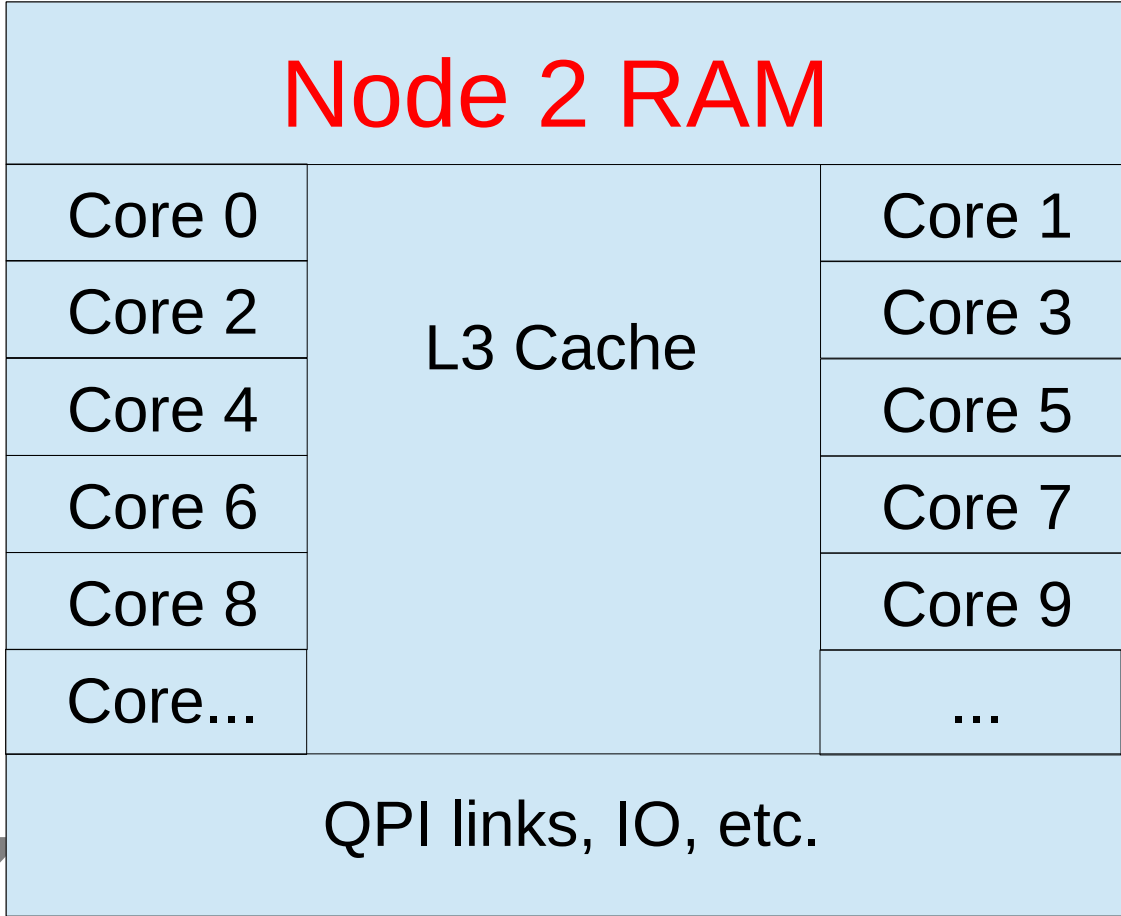
RHEL 6/7 Non-Uniform Memory Access (NUMA)

Typical Four-Node NUMA System

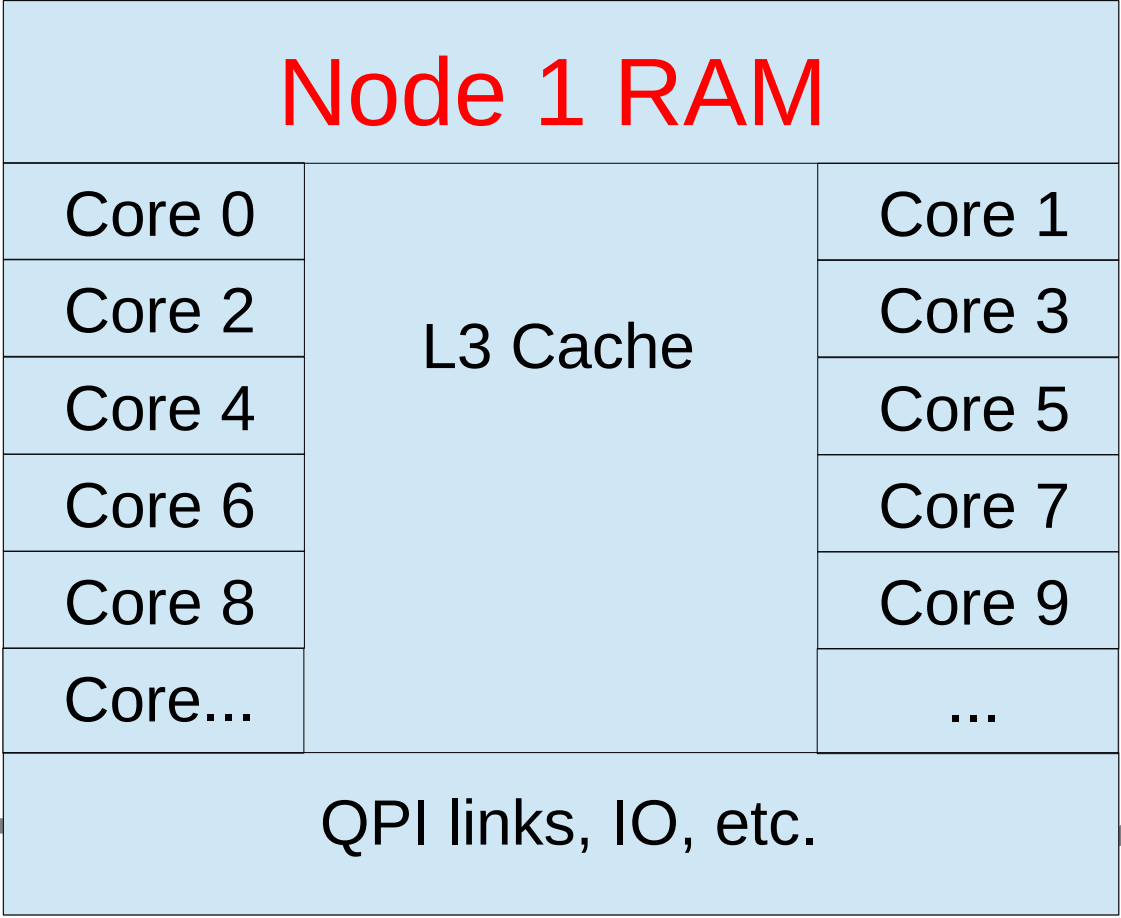
Node 0



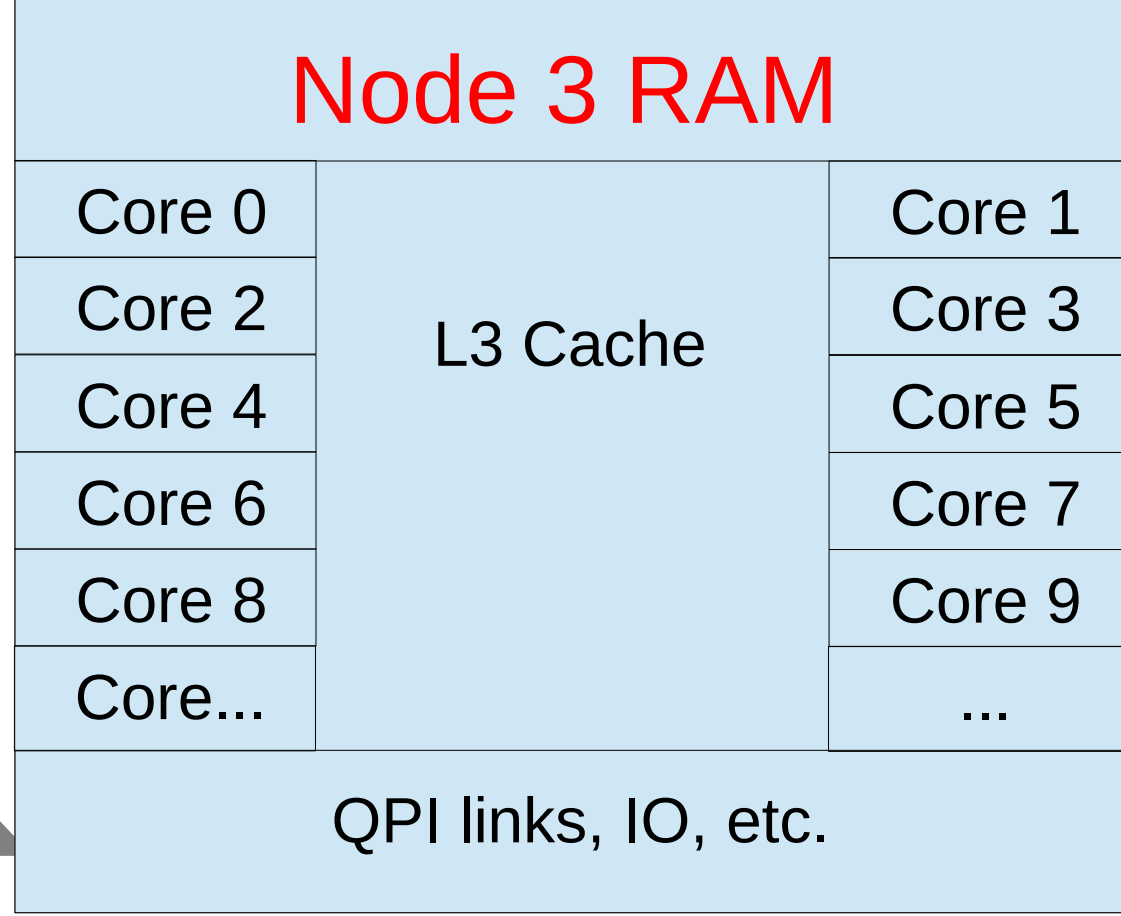
Node 2



Node 1

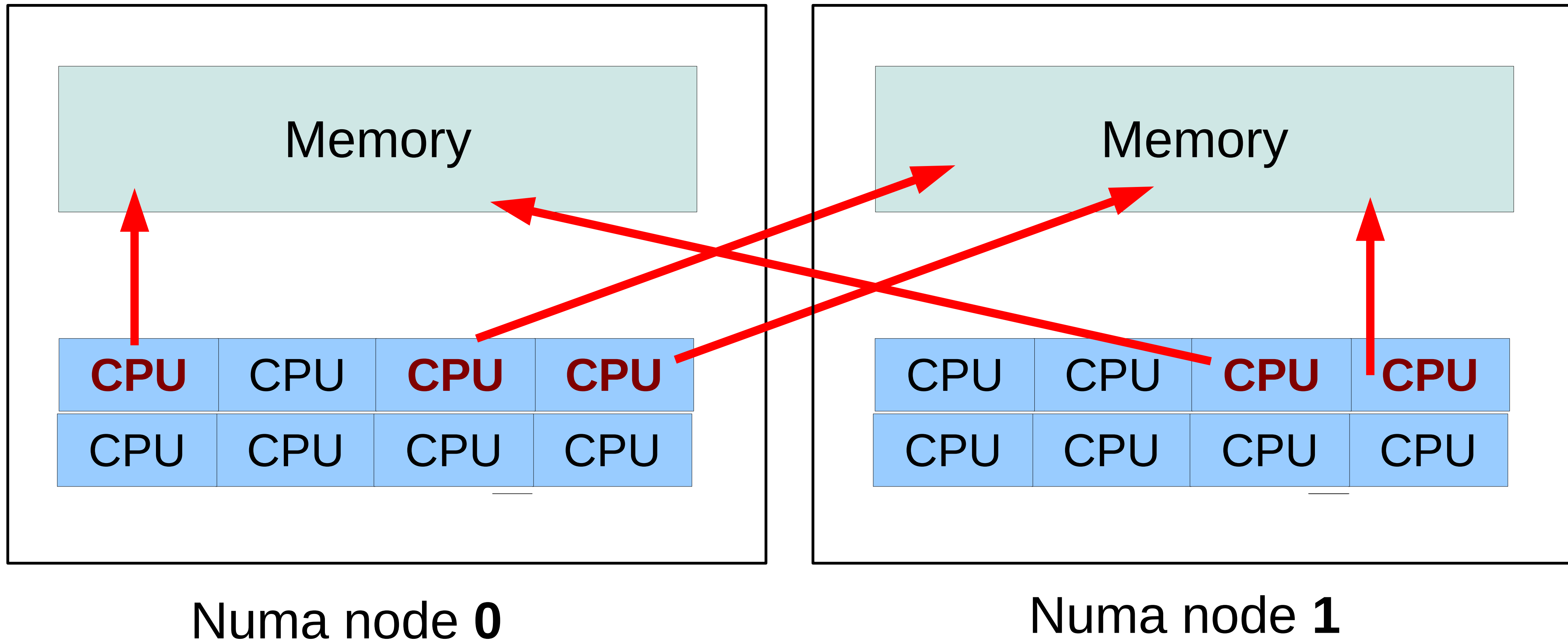


Node 3



Non-optimal numa setup

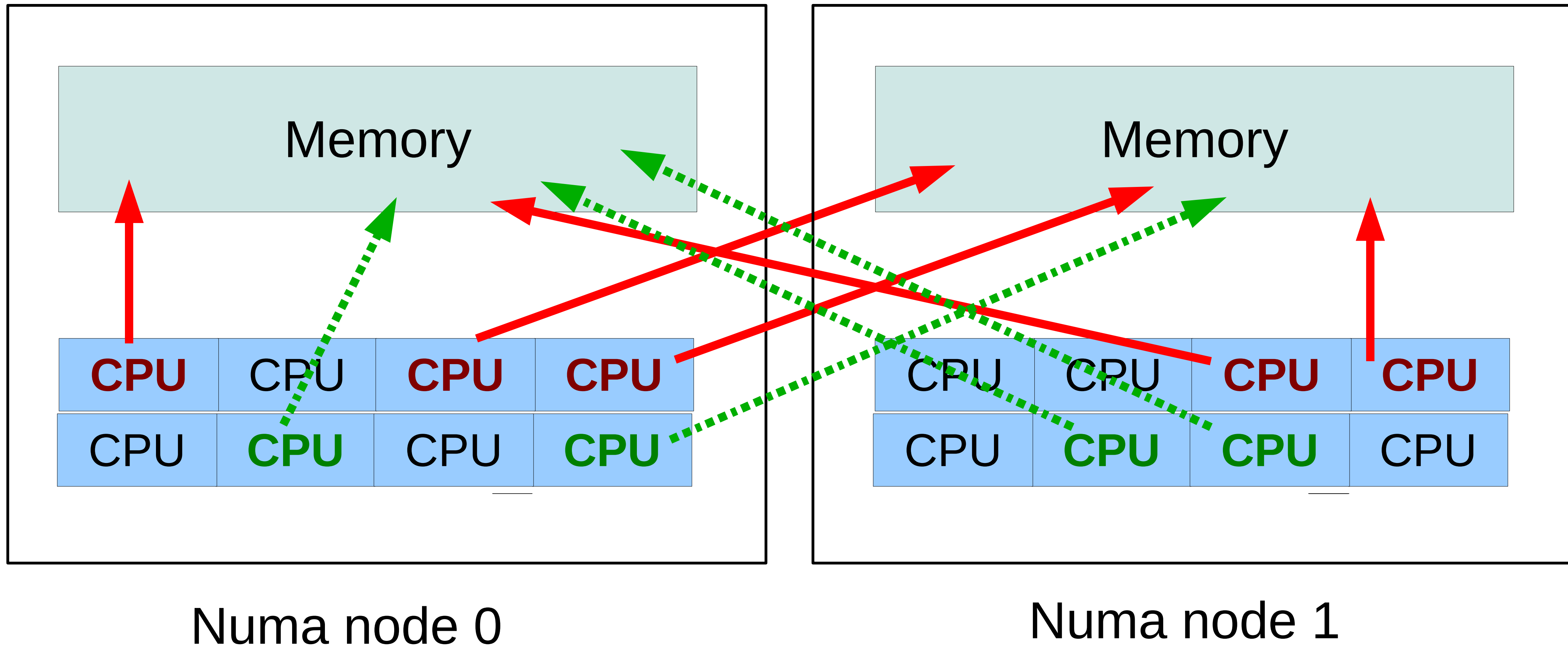
Process 1 in red, 5 threads



Add in more processes: non-optimal

Process 1 in red, 5 threads

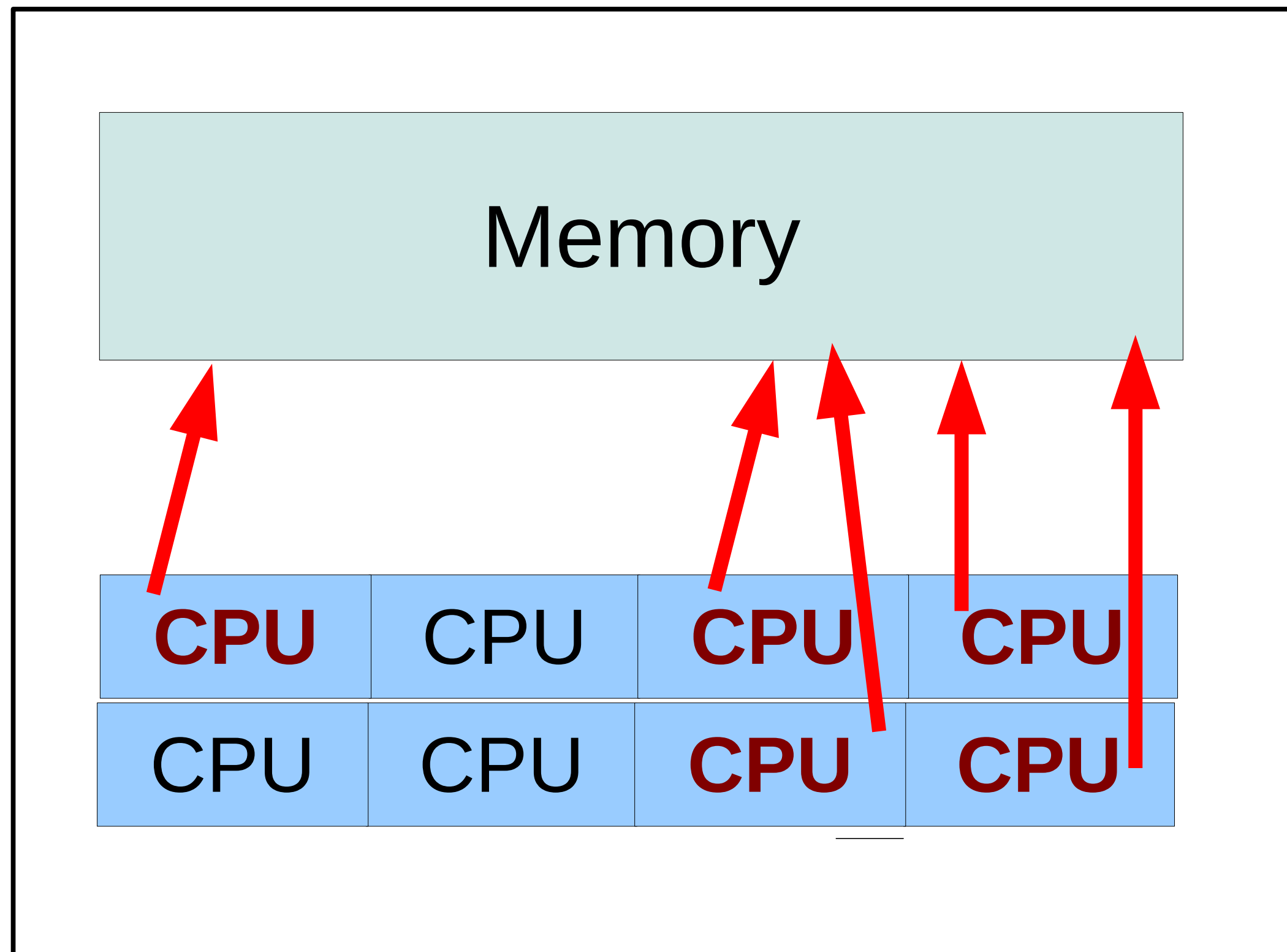
Process 2 in green, 4 threads.



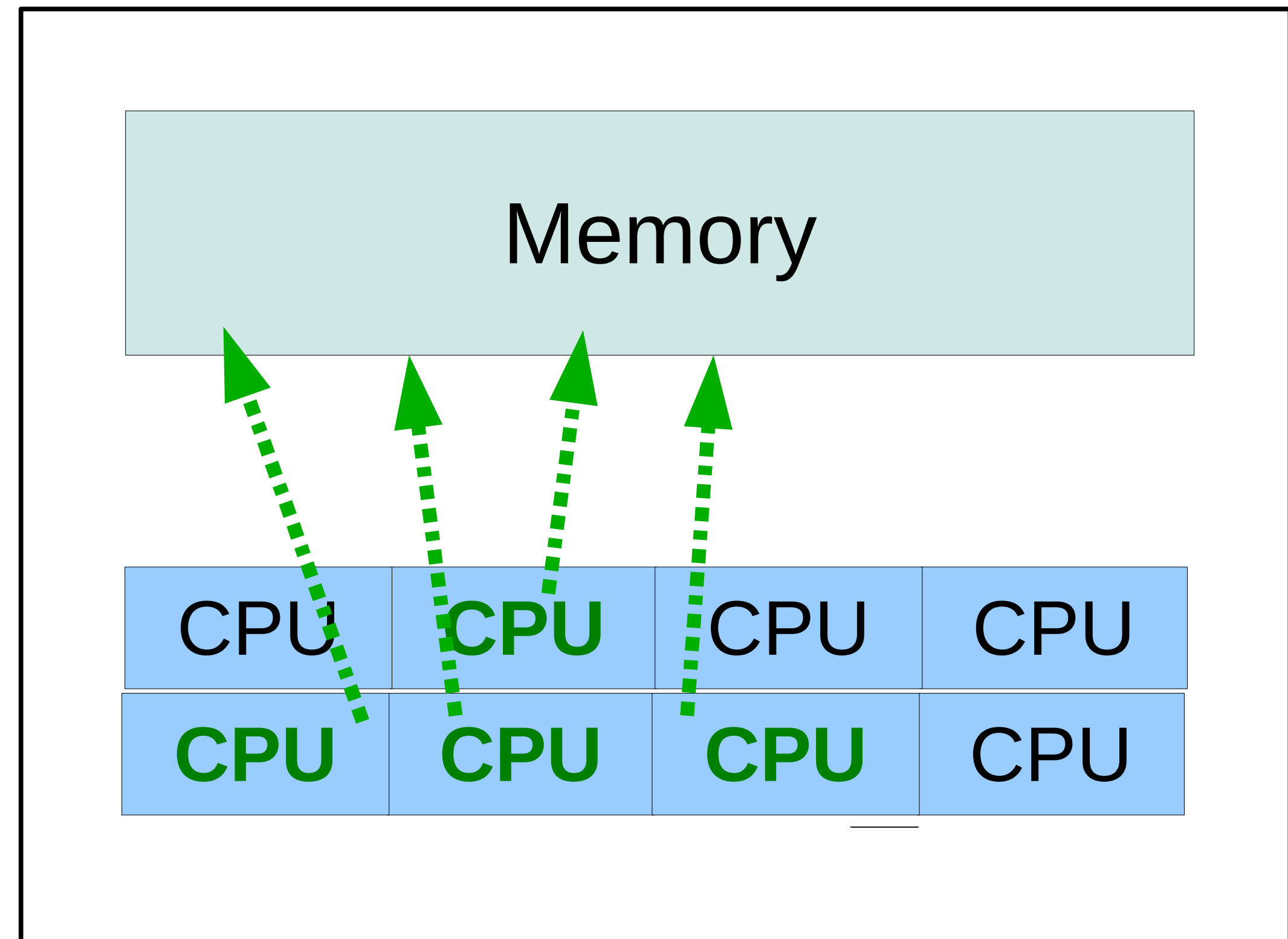
Optimal numa setup

Process 1 in green, 4 threads

Process 2 in red, 5 threads



Numa node 0



Numa node 1

Are my processes doing that?

- Variety of commands available to help:
 - lscpu
 - numactl
 - lstopo
 - numastat
 - ps
 - top

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.

Visualize CPUs via lstopo (hwloc-gui rpm)

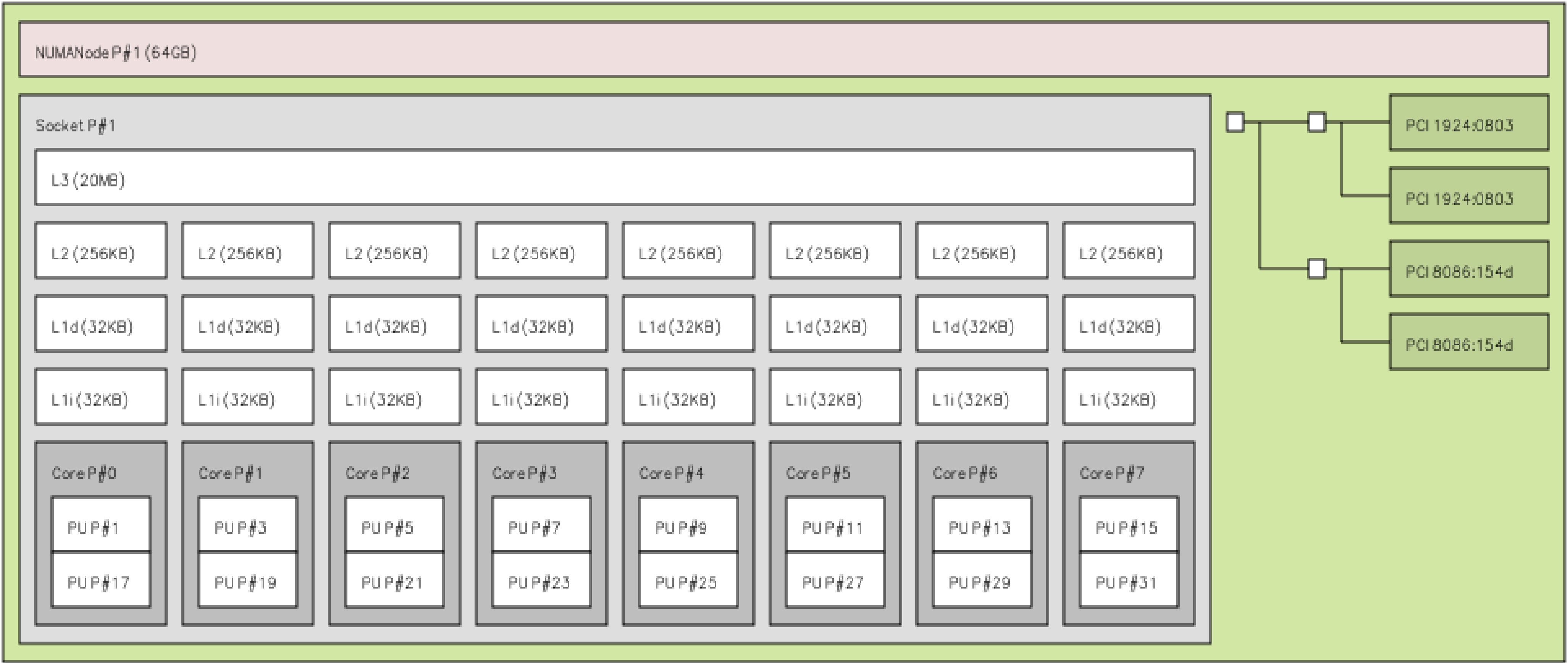
lstopo

NUMA

CACHE

cores
cpus
HT

PCIe



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

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

unaligned

aligned

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
```

What about my processes and threads?

Two ways to see “where it last ran”.

1) *ps -T -o pid,tid,psr,comm <pid>*

```
# ps -T -o pid,tid,psr,comm `pidof pig`
```

PID	TID	PSR	COMMAND
3175391	3175391	73	pig
3175391	3175392	1	pig
3175391	3175393	25	pig
3175391	3175394	49	pig
3175391	3175395	74	pig
3175391	3175396	2	pig
3175391	3175397	26	pig
3175391	3175398	50	pig
3175391	3175399	75	pig
3175391	3175400	3	pig

"Last Ran CPU" column

2) Run “**top**”, then enter “**f**”, then select “**Last used cpu**” field

Techniques to control placement:

numactl:

- Control NUMA policy for processes or shared memory:

taskset:

- Retrieve or set a process's CPU affinity

sched_getaffinity(), sched_setaffinity()

- for process affinity from within program

mbind(), get_mempolicy(), set_mempolicy()

- set default NUMA memory policy for a process children.

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.

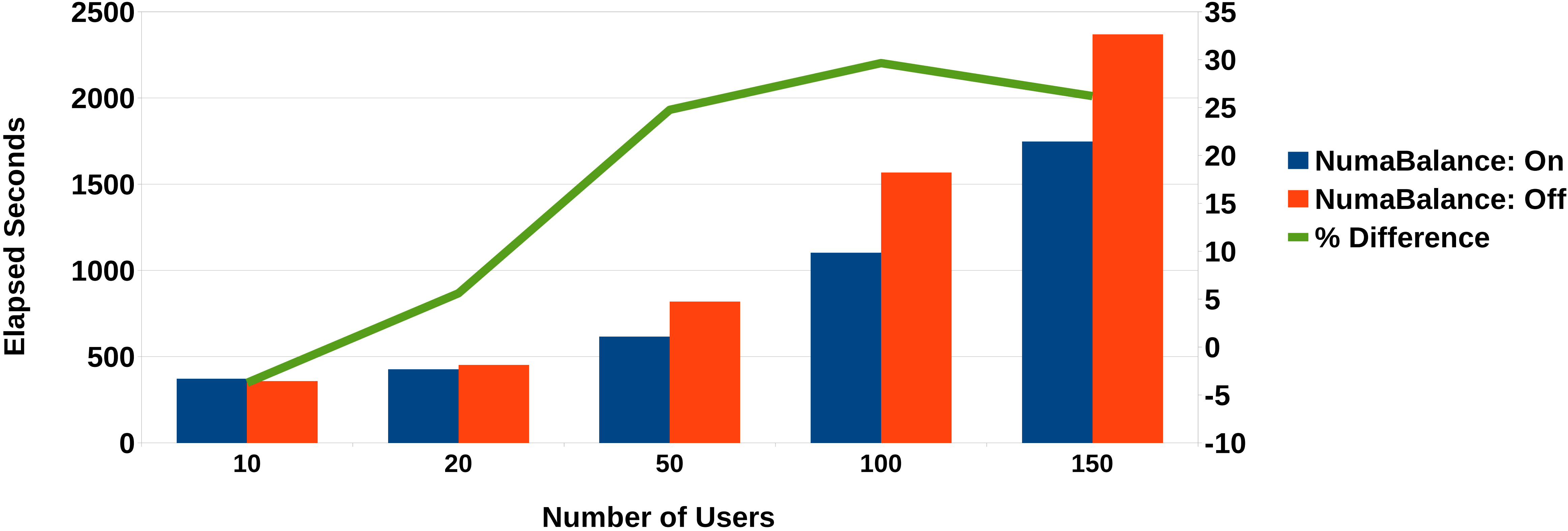
Auto-Numa-Balance kernel scheduler:

- 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

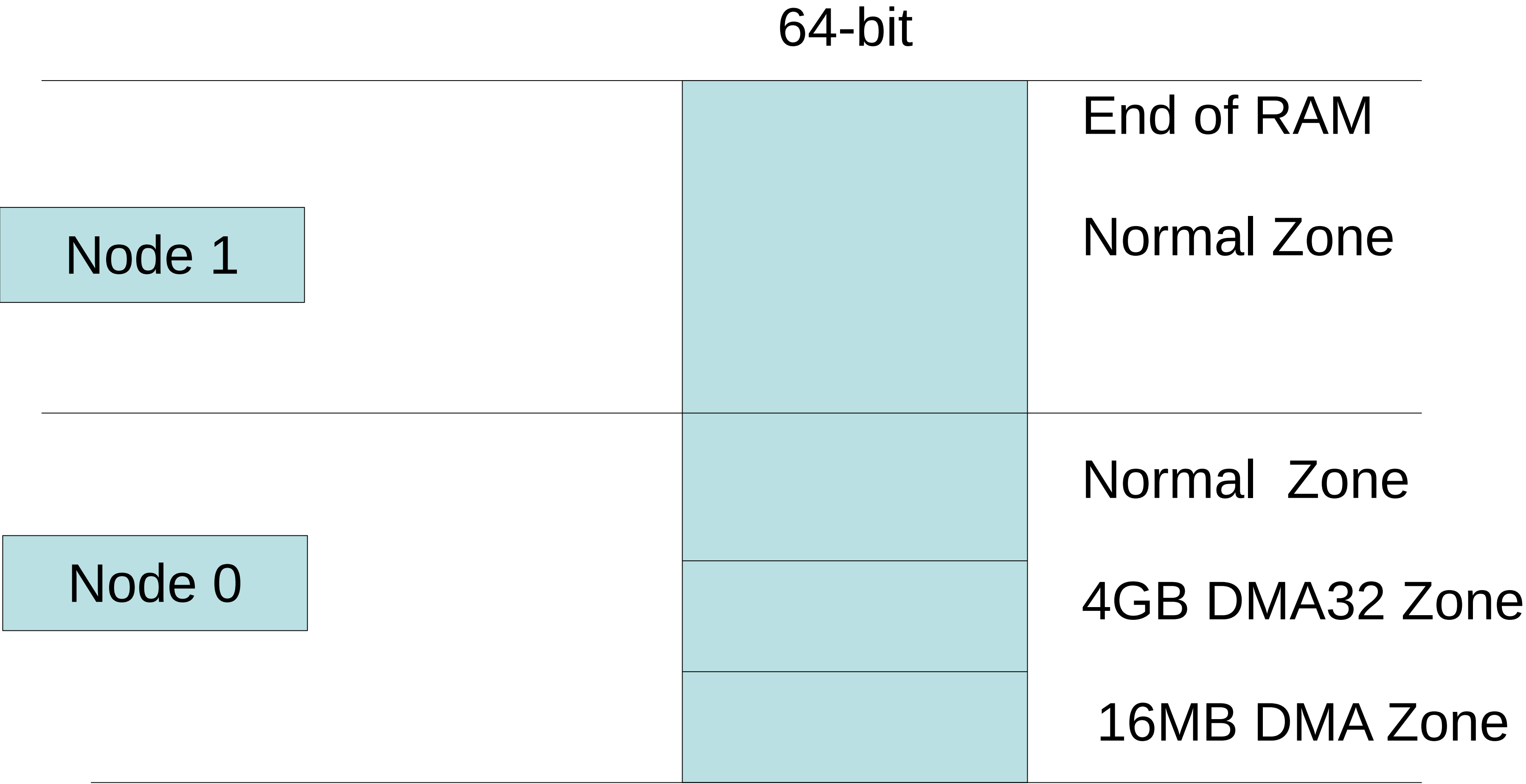
Early SAP HANA benefit from Auto-Numa-Balance

25+% gain. [Recent HANA numa-aware binary closed gap.]

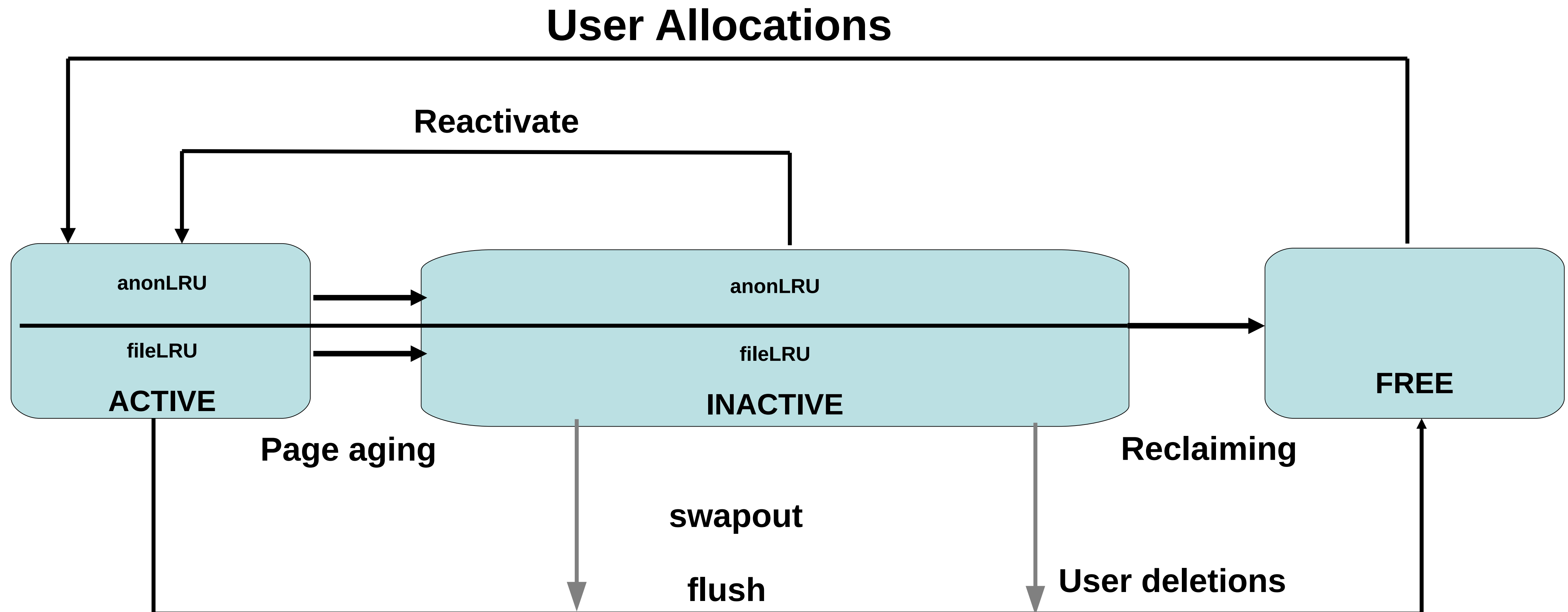
benchBWEMLSim - MultiProvider QueryRuntime
(LOWER==BETTER)



NUMA Nodes and Zones



Per Node / Zone split LRU Paging Dynamics



Interaction between VM Tunables and NUMA

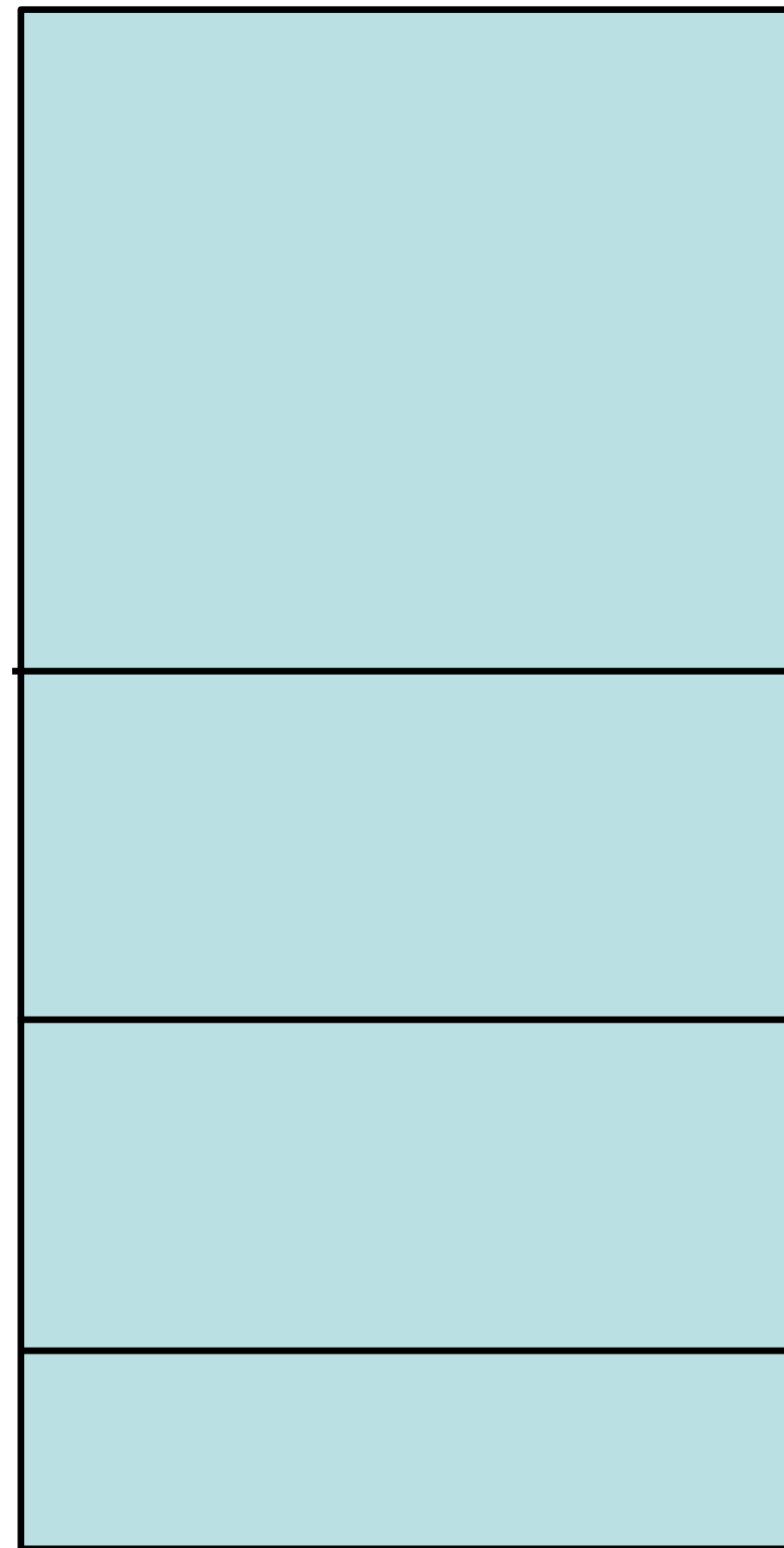
- **Dependent on NUMA: Reclaim Ratios**
 - `/proc/sys/vm/swappiness`
 - `/proc/sys/vm/min_free_kbytes`
 - `/proc/sys/vm/zone_reclaim_mode`
- **Independent of NUMA: Reclaim Ratios**
 - `/proc/sys/vm/vfs_cache_pressure`
- **Writeback Parameters**
 - `/proc/sys/vm/dirty_background_ratio`
 - `/proc/sys/vm/dirty_ratio`
- **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
```

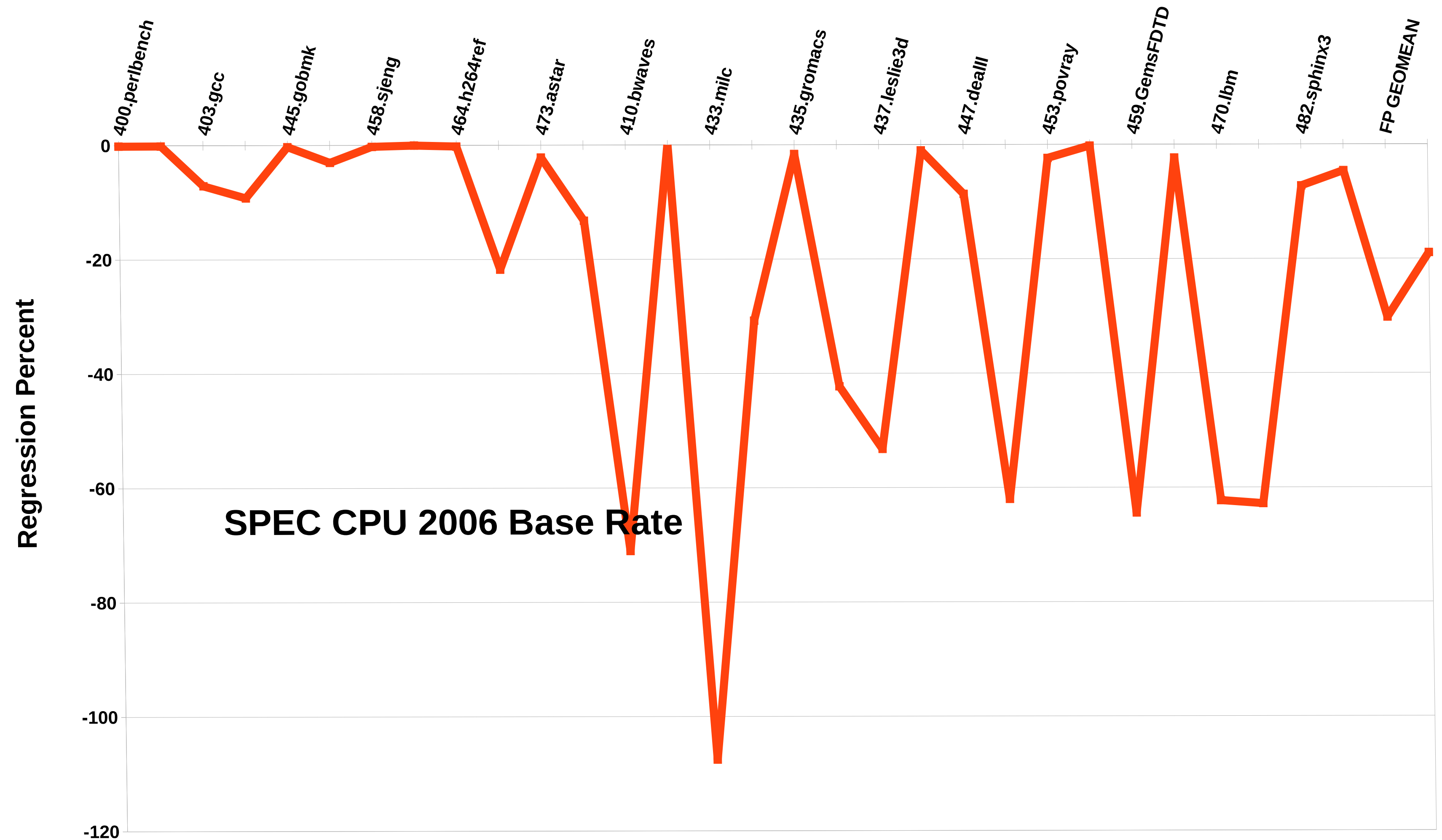
```
-----2
Node 0 DMA      min:160kB low:200kB high:240kB
Node 0 DMA32    min:30624kB low:38280kB high:45936kB
Node 0 Normal   min:59200kB low:74000kB high:88800kB
Node 1 Normal   min:90216kB low:112768kB high:135320kB
-----
```

zone_reclaim_mode

- Controls NUMA specific memory allocation policy
- To see current setting: `cat /proc/sys/vm/zone_reclaim_mode`
 - Turn ON: `echo 1 > /proc/sys/vm/zone_reclaim_mode`
 - Reclaim memory from local node rather than allocating from next node
 - Turn OFF: `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+, the default is usually OFF – because this is better for many applications

zone_reclaim_mode (continued)

- Low-memory SPEC CPU loses huge performance with wrong zone reclaim mode setting! Several benchmarks off more than 40%.
- (BTW, Don't run SPEC CPU with low memory!!)



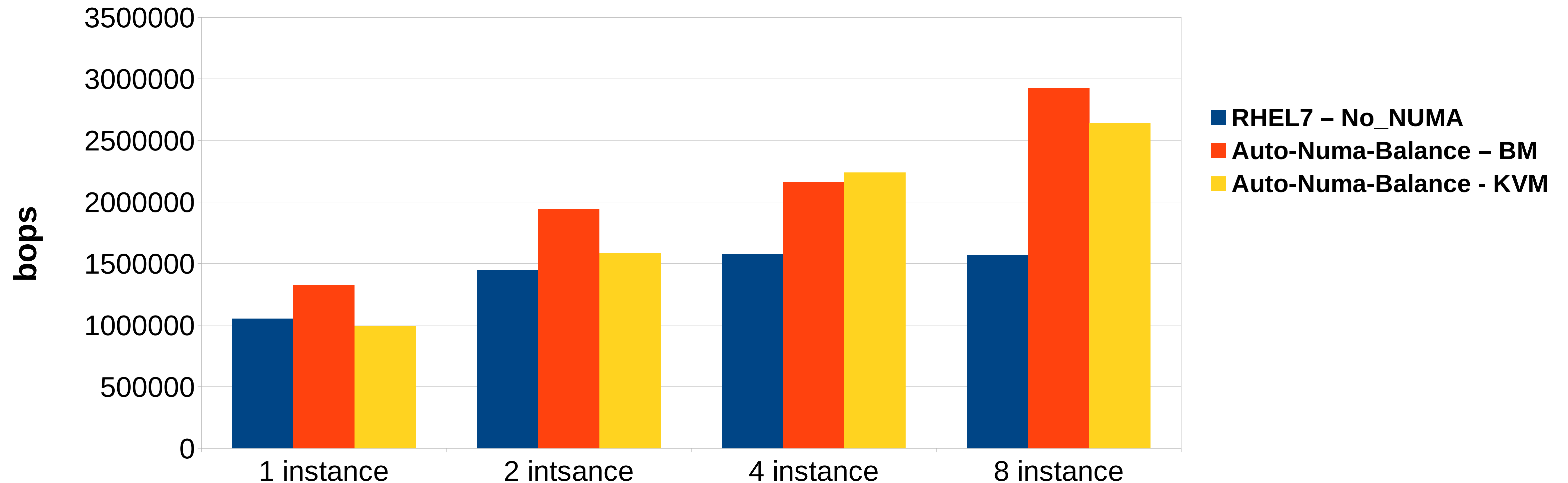
NUMA tuning for KVM / Atomic is the same!

- Best performance is achieved if the size of the guest/container can fit into a single NUMA node.
 - In RHEL7, auto-numa kernel scheduler will try to move guest to one node.
- Great doc with numerous examples: See the NUMA chapter in:
[Red Hat Virtualization Tuning and Optimization Guide](#)

NUMA Performance – SPECjbb2005 on DL980 Westmere EX

RHEL7 Auto-Numa-Balance SPECjbb2005 multi-instance - bare metal + kvm

8 socket, 80 cpu, 1TB mem

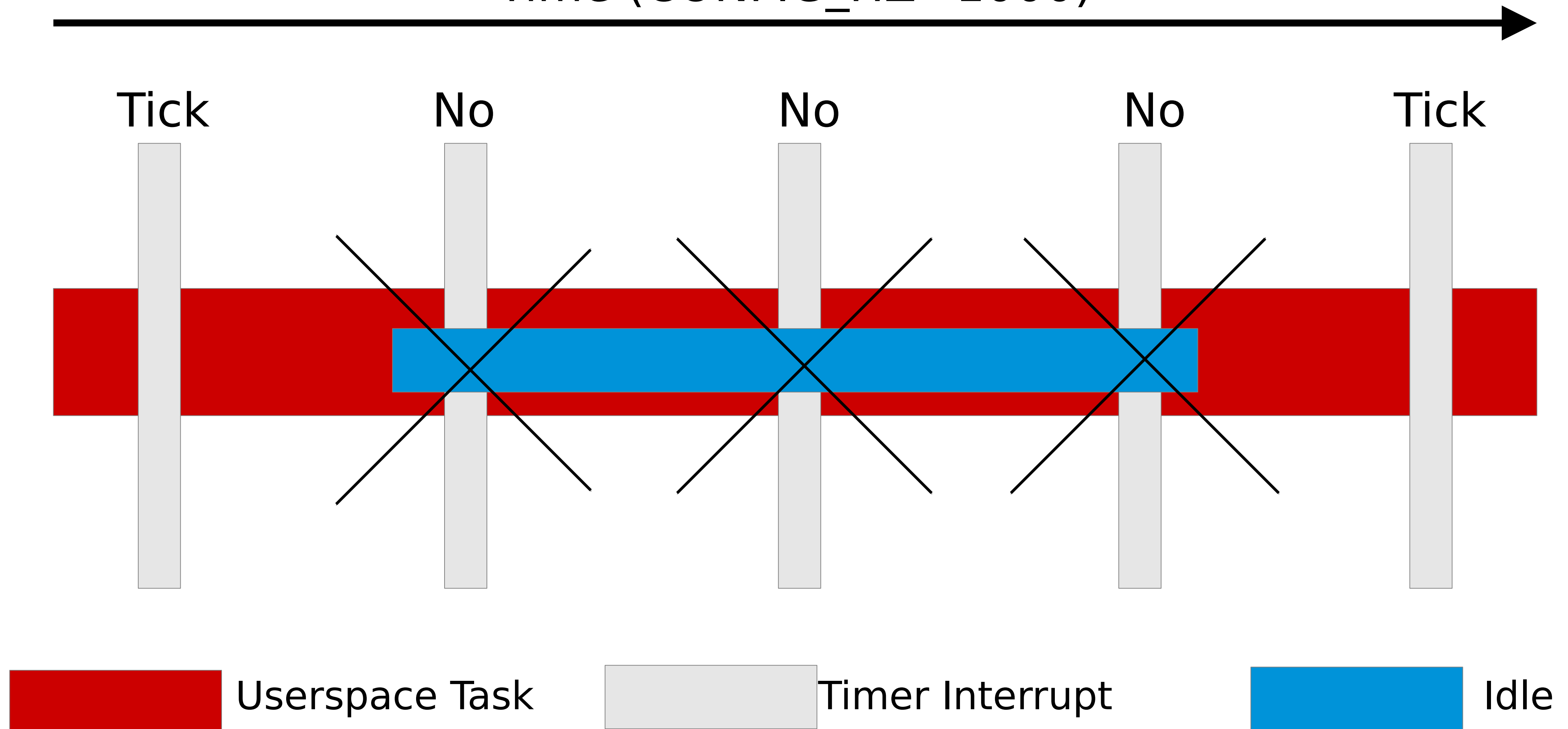


RHEL7 nohz_full

- 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

Time (CONFIG_HZ=1000)



Performance Tools - 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

```
Samples: 10K of event 'cycles', Event count (approx.): 5973713325
34.35%    httpd [kernel.kallsyms] [k] avtab_search_node
12.70%    httpd [kernel.kallsyms] [k]  _spin_lock
 8.61%    httpd [kernel.kallsyms] [k]  tg_load_down
 7.42%    httpd [kernel.kallsyms] [k]  _spin_lock_irq
 5.79%      init [kernel.kallsyms] [k]  intel_idle
 3.92%    httpd [kernel.kallsyms] [k]  _spin_lock_irqsave
 1.75%    httpd [kernel.kallsyms] [k]  sidtab_search_core
 1.74%    httpd [kernel.kallsyms] [k]  load_balance_fair
 1.18%    httpd [kernel.kallsyms] [k]  tg_nop
 1.13%      init [kernel.kallsyms] [k]  _spin_lock
```



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 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

perf diff / sched

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            [.] 0x000000000000064968
   4.32%    +6.43%  libdbus-1.so.3.7.4  [.] 0x00000000000029258
   4.06%    +4.72%  dbus-daemon        [.] 0x00000000000014a6e
   3.79%   -3.79%  libglib-2.0.so.0.3600.3  [.] 0x00000000000088d6a
   3.72%    +0.25%  [kernel.kallsyms]  [k] seq_list_start
```

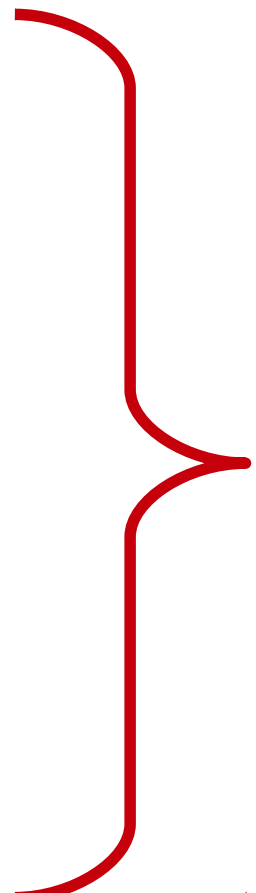
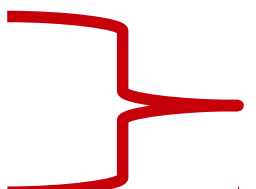
grep for something interesting, maybe to see what numabalance is doing ?

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

False Sharing

- Different threads sharing common data struct
- Different processes sharing common shared memory.

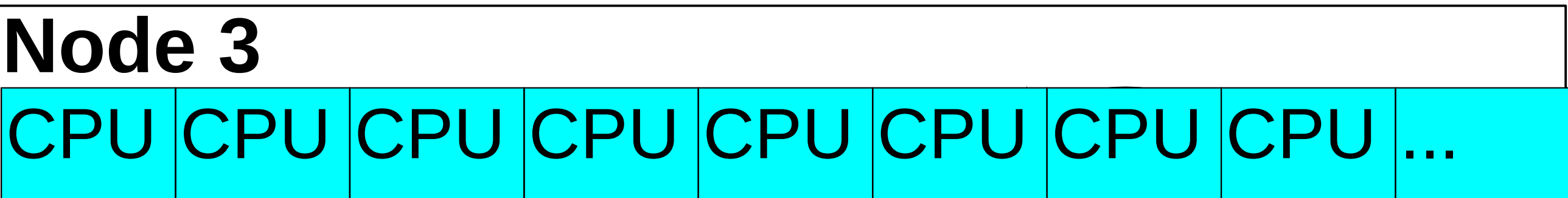
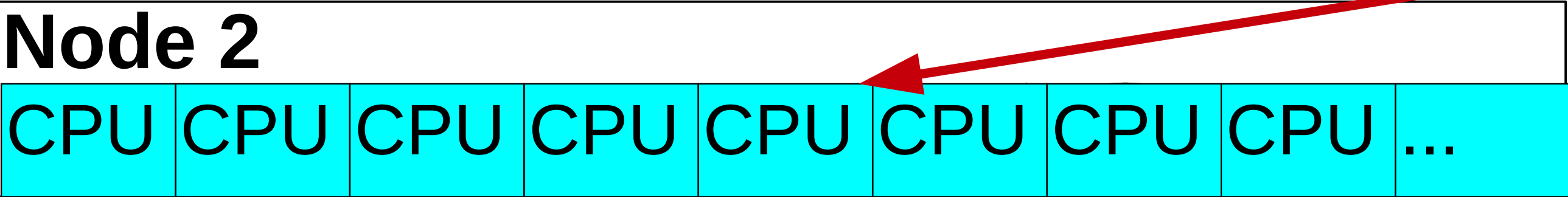
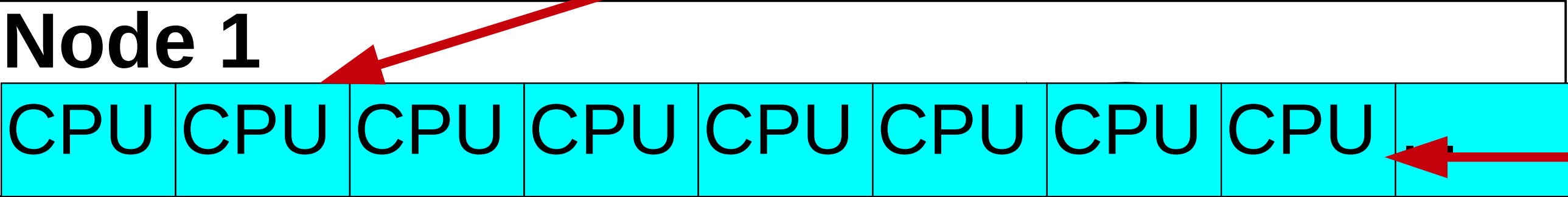
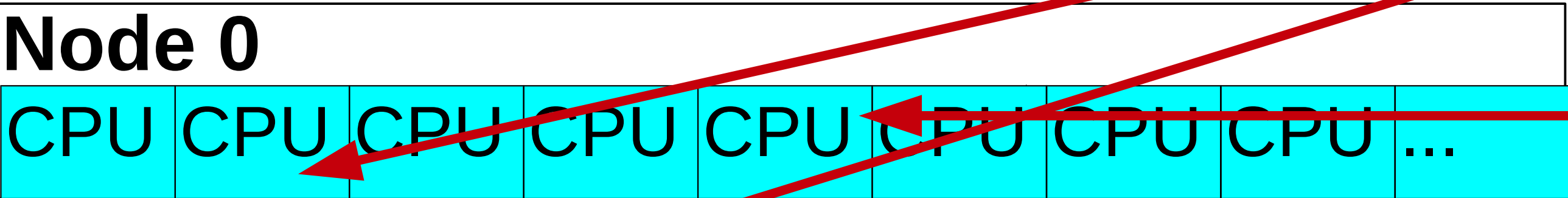
Ex: Two hotly contended data items sharing a 64-byte cacheline.

int a;	0	 <i>Hot pthread mutex</i>
pthread_mutex_t mutex1;	8	
	16	
	24	
	32	
	40	 <i>Hot variable</i>
long b;	48	
long sequence_cnt;	56	

Gets you contention like this:

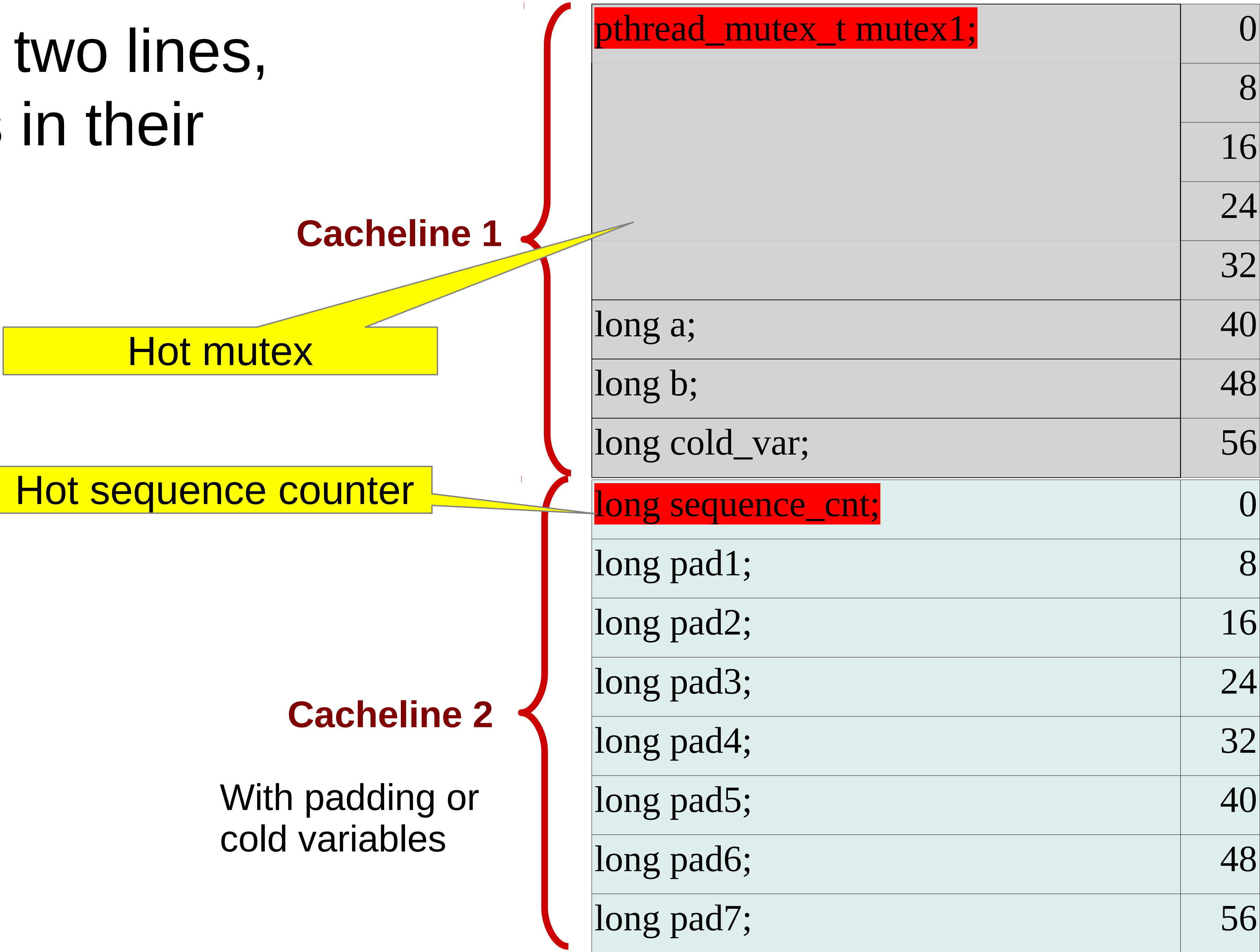
- Can be quite painful

64 byte cache line



int a;	offset	0
mutex	offset	8
mutex	offset	16
mutex	offset	24
mutex	offset	32
mutex	offset	40
long b;	offset	48
long seq_cnt;	offset	56

Split it up into two lines,
with hot items in their
own lines:



Future Red Hat update to perf: “c2c data sharing” tool

=====									
Cache									CPU
#	Refs	Stores	Data Address	Pid	Tid	Inst Address	Symbol	Object	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 who is contributing to the false sharing:

- 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)
- Disassemble the binary to find the ip, and track back to the sources.

Summary - Red Hat Enterprise Linux NUMA

- RHEL6 – NUMAD - With Red Hat Enterprise Linux 6.5
 - 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.
- App Developers – use perf to check for false sharing, advise padding

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

Performance Utility Summary

Supportability

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

#redhat #rhsummit

NUMA

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

Power/Tuning

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

Networking

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

Storage

- blktrace
- iotop
- iostat

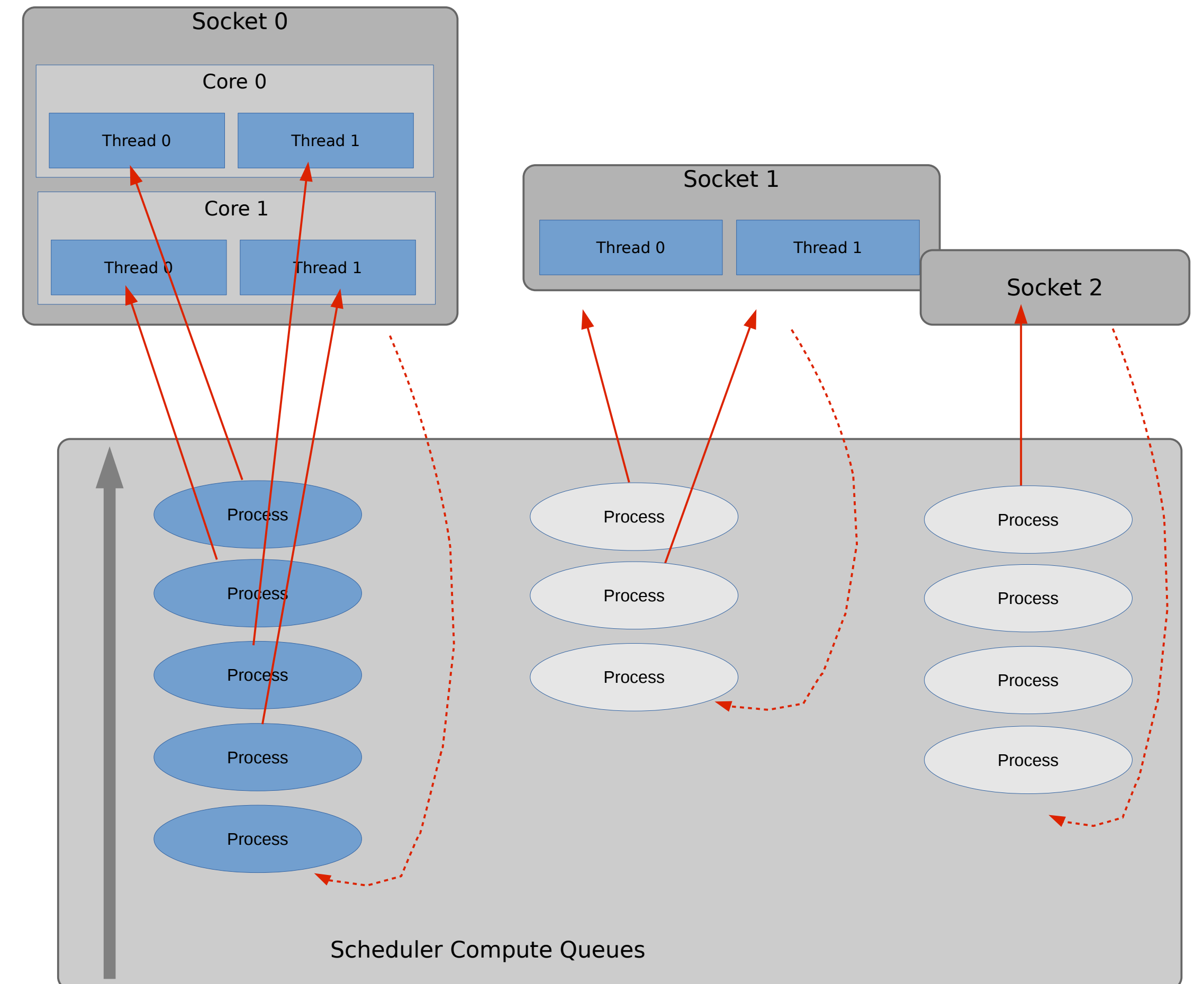
CFS Scheduler

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 from overloaded CPUs to idle CPUs
- Detect using “perf stat”, look for excessive “migrations”
- **/proc/sys/kernel/sched_migration_cost**
 - 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)

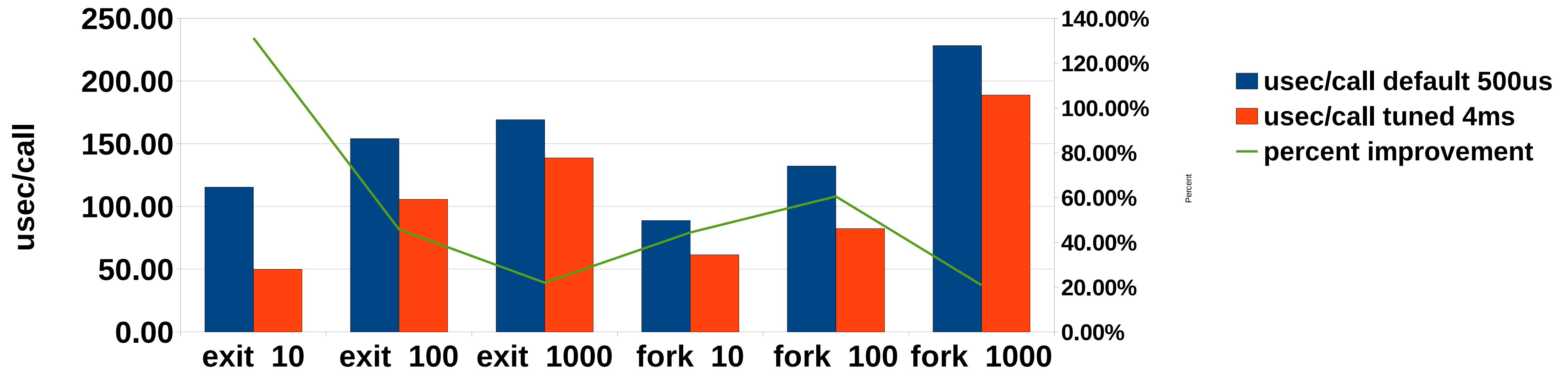
fork() behavior

`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

Intel Westmere EP 24cpu/12core, 24 GB mem



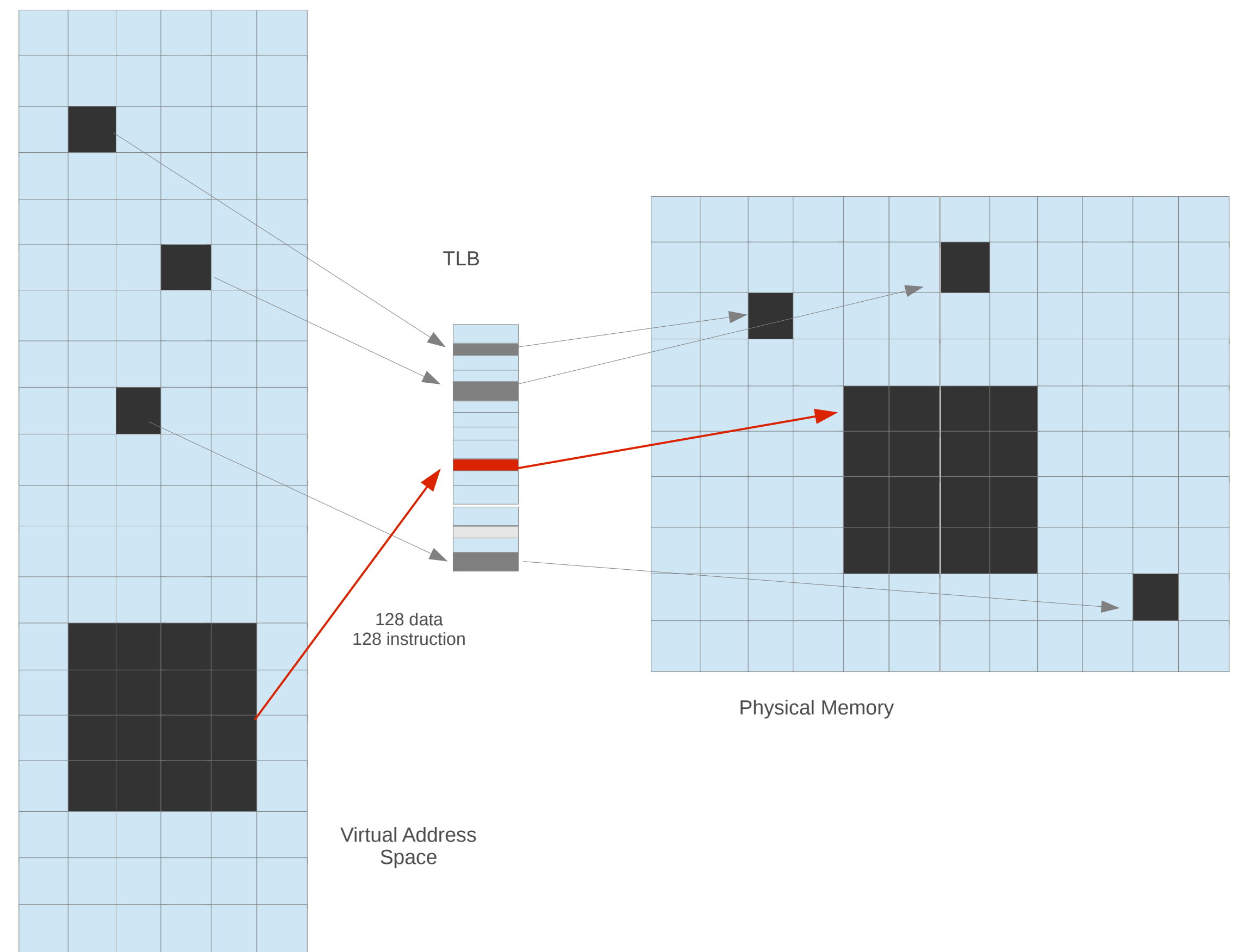
RHEL7.4 Core Kernel Features

- Deadline Scheduler – allows process to define when it must run.
- Persistent Memory – supports huge amounts of non-volatile RAM
- Qspinlock – prevents cacheline contention that can result in massive locking contention
- RWsemaphore locking – performance improvement to kernel read/write semaphores
- KASLR – kernel address space randomization provide better security

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
```


2MB Hugepages - specific node allocation

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

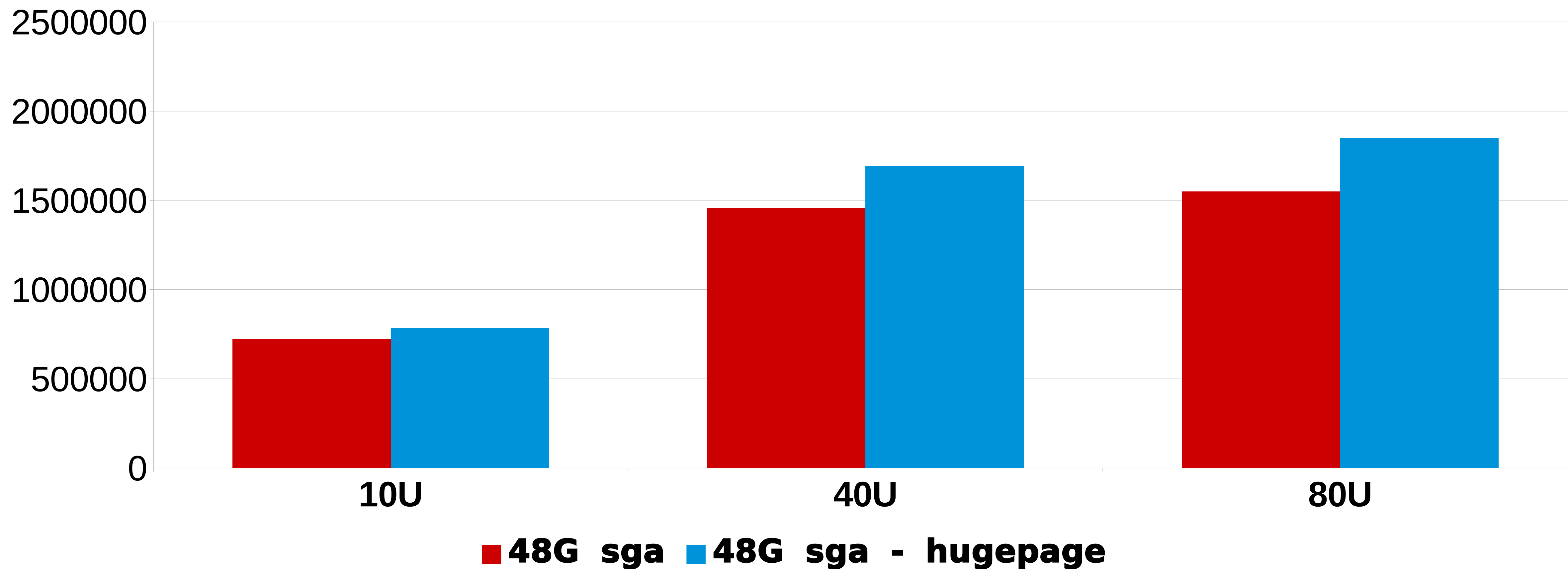
```
# echo 1000 > /proc/sys/vm/nr_hugepages
# cat /proc/meminfo | grep HugePages_Free
HugePages_Free: 1000
# cat /sys/devices/system/node/node*/hugepages/hugepages-2048kB/nr_hugepages
500
500
```

```
# 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
```

Database Performance OLTP (Higher = Better)

huge pages on Bare Metal

Oracle OLTP tran/min



The effect of hugepages are more pronounced as system drive to saturation

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
```

Dynamic per-node allocation/deallocation of 1GB Hugepages

```
# cat /sys/devices/system/node/node*/hugepages/hugepages-1048576kB/nr_hugepages
0
0
```

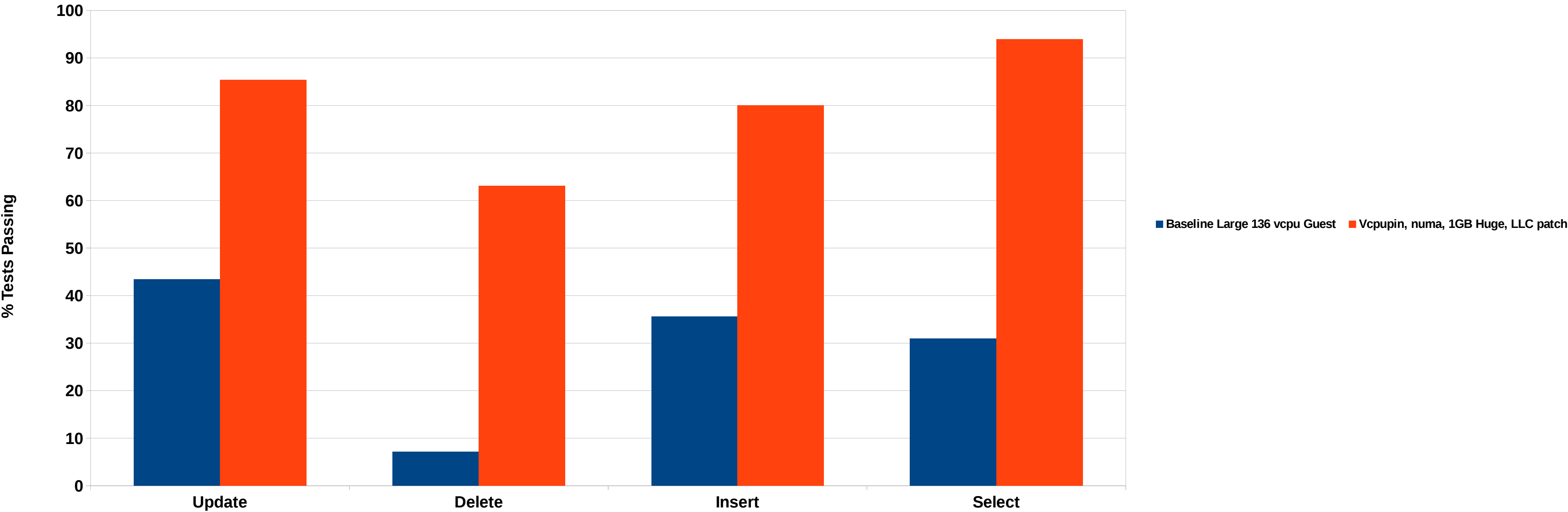
```
# echo 8 > /sys/devices/system/node/node0/hugepages/hugepages-1048576kB/nr_hugepages
# cat /proc/meminfo | grep HugePages_Free
HugePages_Free:      8
# cat /sys/devices/system/node/node*/hugepages/hugepages-1048576kB/nr_hugepages
8
0
```

```
# echo 0 > /sys/devices/system/node/node0/hugepages/hugepages-1048576kB/nr_hugepages
# cat /proc/meminfo | grep HugePages_Free
HugePages_Free:      0
# cat /sys/devices/system/node/node*/hugepages/hugepages-1048576kB/nr_hugepages
0
0
```

SAP Performance w/ Hana

RHEL7.3 RHV4.x - SAP HANA OLTP Cert Series - Baseline vs. Tuned Guest

Intel Haswell EX, 144 CPU, 4 socket, 512 GB memory, 2 PCI NVME cards



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/

```
# 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

Incorrect 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

```
# 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

cpu.shares throttled

echo 10 > cpu.shares

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

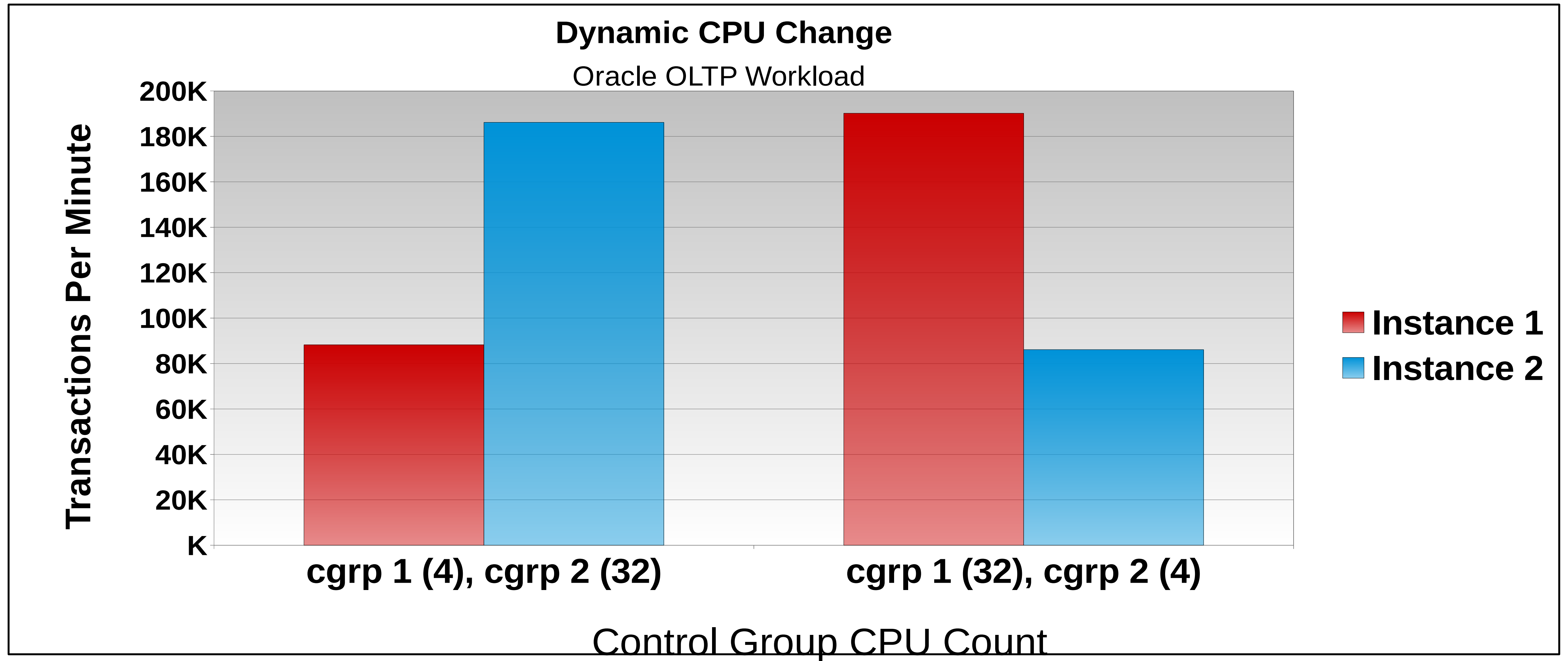
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME
20104	root	20	0	4160	360	284	R	99.4	0.0	12:35.83 useless
20103	root	20	0	4160	356	284	R	91.4	0.0	12:34.78 useless
20105	root	20	0	4160	360	284	R	90.4	0.0	12:33.08 useless
20106	root	20	0	4160	360	284	R	88.4	0.0	12:32.81 useless
20102	root	20	0	4160	360	284	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

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



C-group Dynamic resource control



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

...

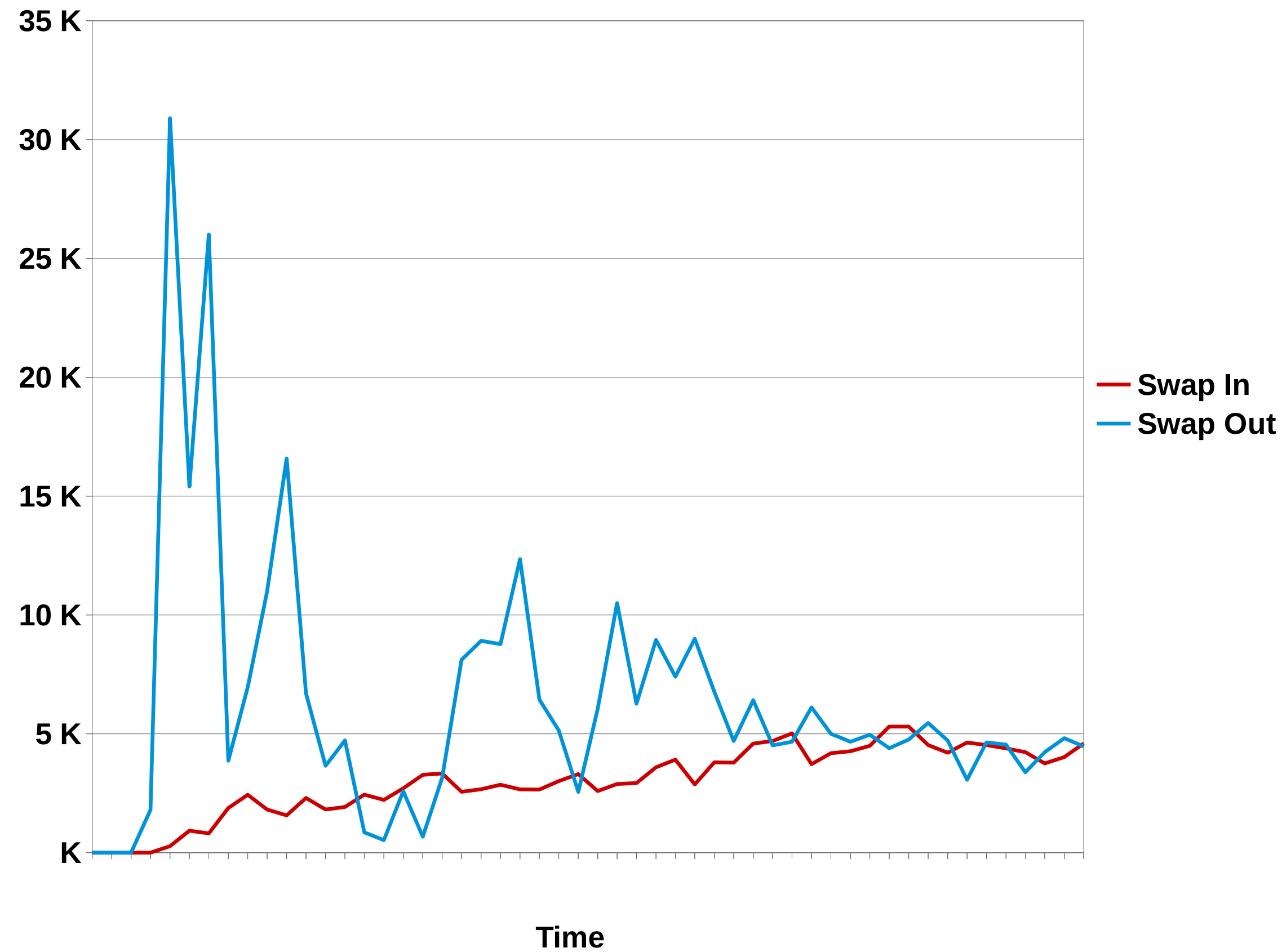
dmesg

...

```
[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
```

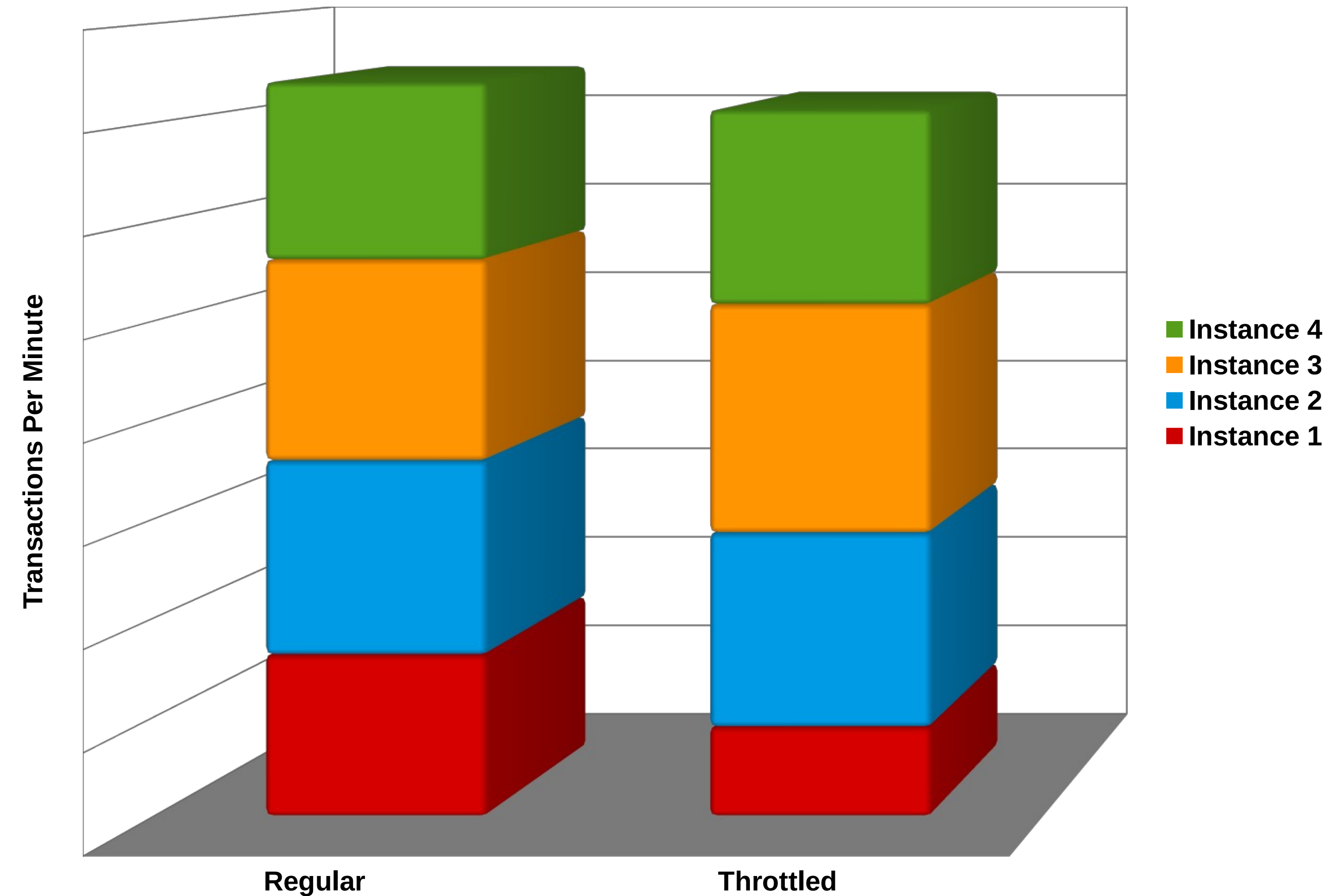
Cgroup – Application Isolation

System Level Memory Swapping



Memory Resource Management

Oracle OLTP Workload



Even though the “RED” application does not have resources and starts swapping, The other applications are not affected.

Disk IO

I/O Tuning – Understanding I/O Elevators

- Deadline – new RHEL7 default for all profiles
 - Two queues per device, one for read and one for writes
 - I/Os dispatched based on time spent in queue
- CFQ – used for system disks off SATA/SAS controllers
 - Per process queue
 - Each process queue gets fixed time slice (based on process priority)
- NOOP – used for high-end SSDs (Fusion IO etc)
 - FIFO
 - Simple I/O Merging
 - Lowest CPU Cost

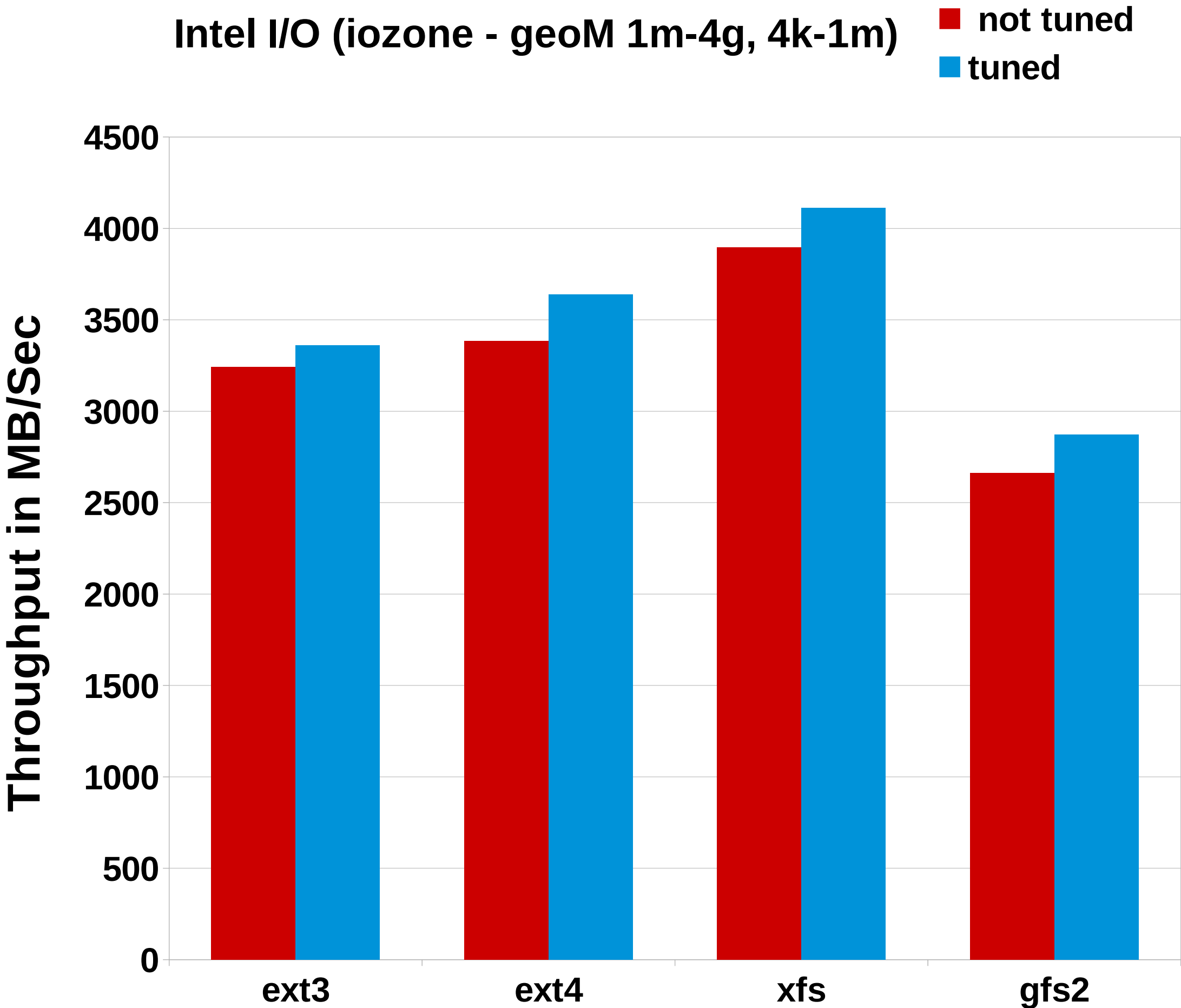
Tuned: Profile throughput-performance (RHEL7 default)

throughput-performance

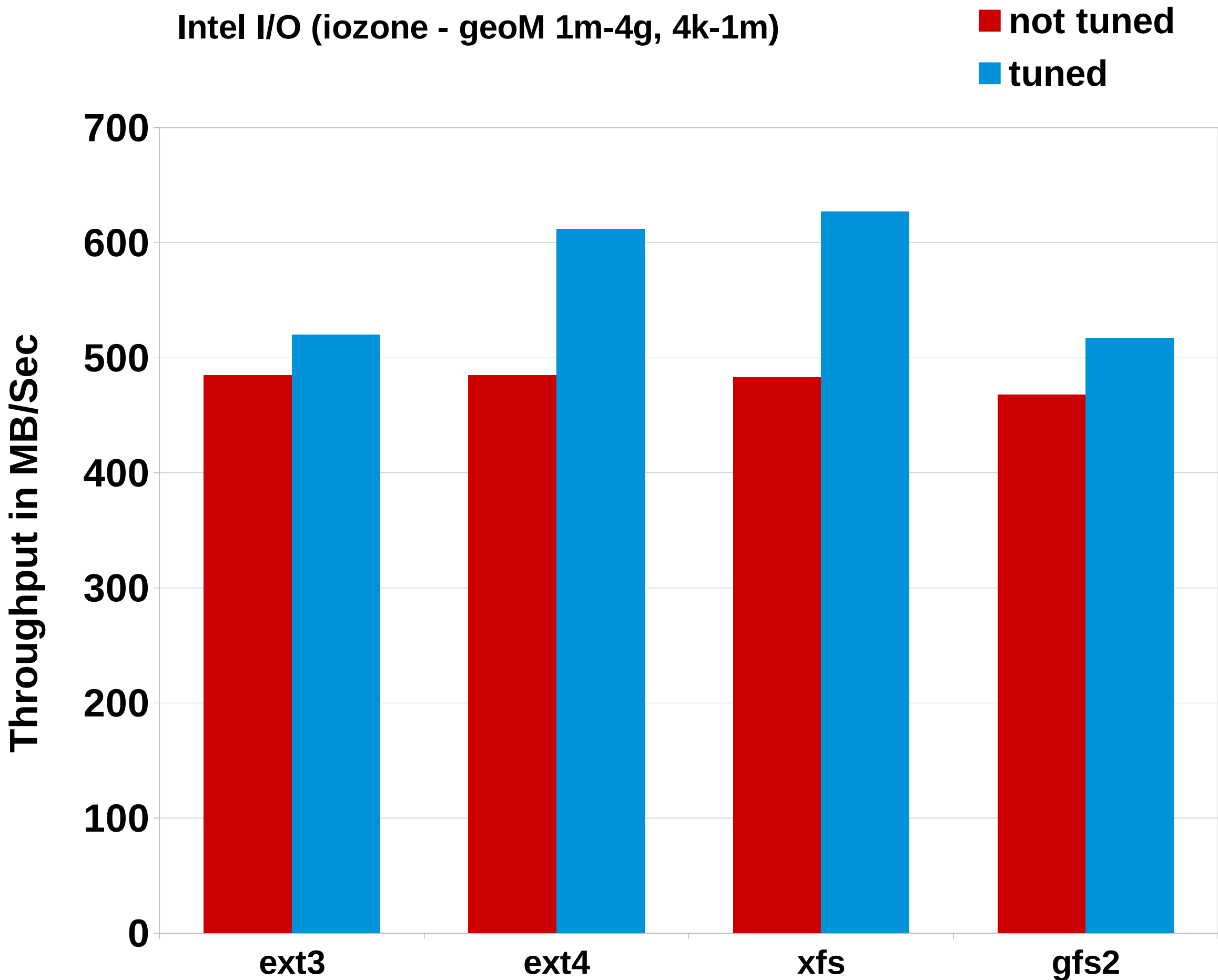
```
governor=performance
energy_perf_bias=performance
min_perf_pct=100
readahead=4096
kernel.sched_min_granularity_ns = 10000000
kernel.sched_wakeup_granularity_ns = 15000000
vm.dirty_background_ratio = 10
vm.swappiness=10
```

iozone Performance effect of “tuned” EXT4/XFS/GFS

RHEL 7.1 3.10.0-253 File System In Cache Performance



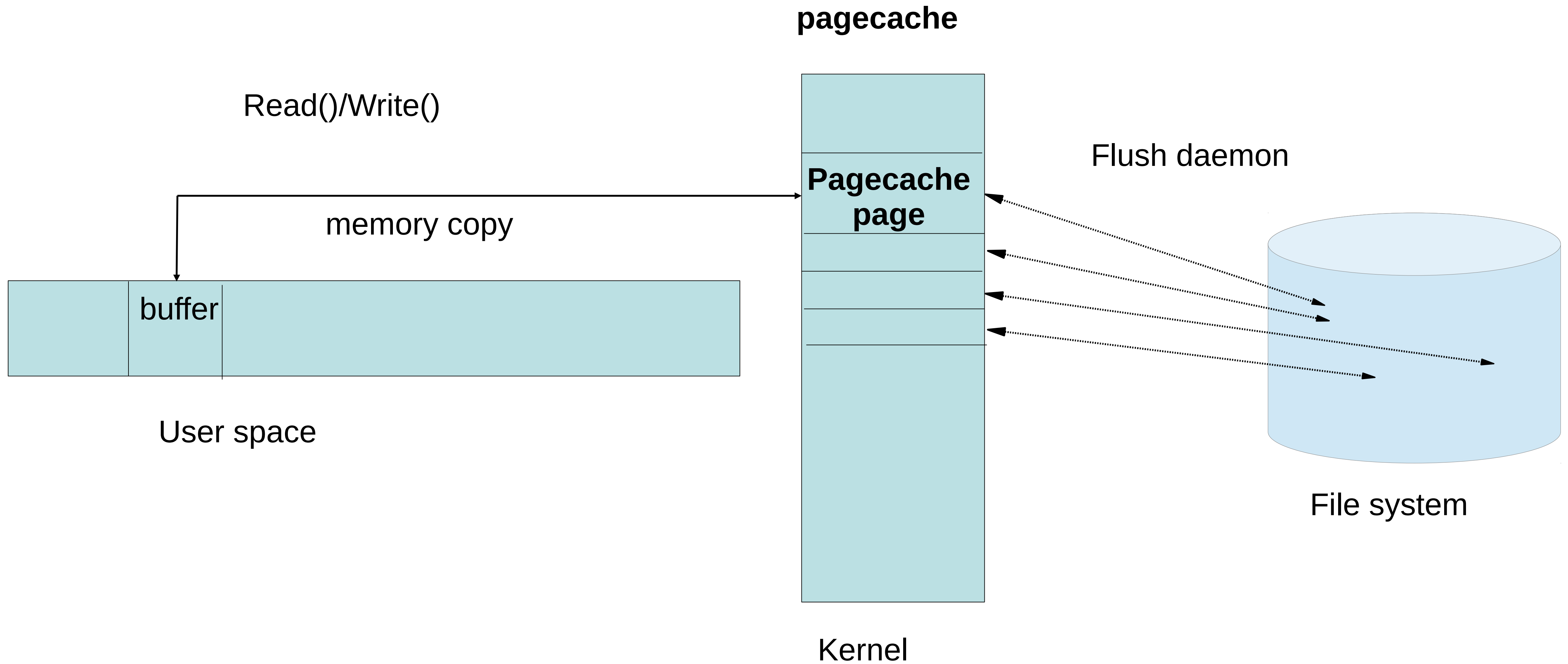
RHEL7 3.10.0-253 File System Out of Cache Performance



Tuning Memory – **Flushing Caches**

- Drop unused Cache – to control pagecache dynamically
 - ✓ Frees most pagecache memory
 - ✓ File cache
 - ✗ If the DB uses cache, may notice slowdown
- NOTE: Use for benchmark environments.
- **Free pagecache**
 - # sync; echo 1 > /proc/sys/vm/drop_caches
- **Free slabcache**
 - # sync; echo 2 > /proc/sys/vm/drop_caches
- **Free pagecache and slabcache**
 - # sync; echo 3 > /proc/sys/vm/drop_caches

Per file system flush daemon



Virtual Memory Manager (VM) Tunables

- **Reclaim Ratios**

- /proc/sys/vm/swappiness
- /proc/sys/vm/vfs_cache_pressure
- /proc/sys/vm/min_free_kbytes

- **Writeback Parameters**

- /proc/sys/vm/dirty_background_ratio
- /proc/sys/vm/dirty_ratio

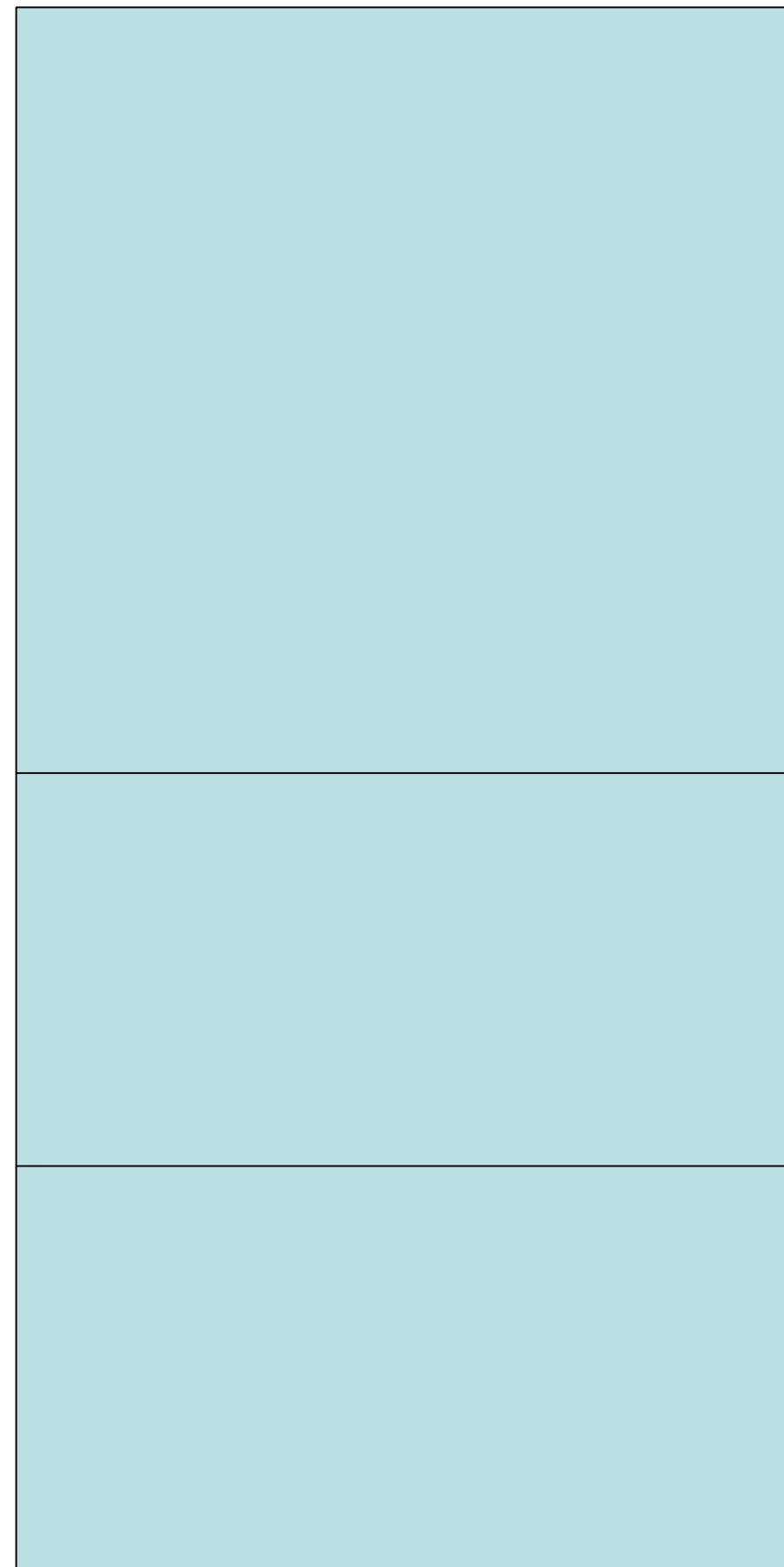
- **Readahead parameters**

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

dirty_background_ratio, dirty_background_bytes

- Controls when dirty pagecache memory starts getting written.
- Default is 10%
- Lower
 - flushing starts earlier
 - less dirty pagecache and smaller IO streams
- Higher
 - flushing starts later
 - more dirty pagecache and larger IO streams
- **dirty_background_bytes over-rides when you want < 1%**

dirty_ratio and dirty_background_ratio



100% of pagecache RAM dirty

flushd and write()'ng processes write dirty buffers

dirty_ratio(20% of RAM dirty) – processes start synchronous writes

flushd writes dirty buffers in background

dirty_background_ratio(10% of RAM dirty) – wakeup flushd

do_nothing

0% of pagecache RAM dirty

RHEL7 Performance Tuning Summary

- **Use “Tuned”, “NumaD” in RHEL6 and RHEL7**
 - Power savings mode (performance), cstates locked (latency profiles)
 - Transparent Hugepages for anon memory (monitor it)
 - numabalance – Multi-instance, consider “NumaD”
- **Manually Tune**
 - NUMA – via numactl, monitor numastat -c pid
 - Huge Pages – static hugepages 2M or 1GB for pinned shared-memory
 - Managing VM, dirty ratio and swappiness tuning
- **Q+A in “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](#)

The logo consists of a white speech bubble shape pointing downwards, containing the text "RED HAT" in a smaller, bold, red sans-serif font, and "SUMMIT" in a larger, bold, red sans-serif font below it.

RED HAT
SUMMIT

**LEARN. NETWORK.
EXPERIENCE
OPEN SOURCE.**