RED HAT SUMMIT

Performance Analysis and Tuning Red Hat Enterprise Linux Part 1 (what's new in RHEL8)

D. John Shakshober
Sr Distinguished Eng
Tech Director RH Perf+Scale

Joe Mario
Sr Principal Perf+Scale Eng
Kernel, Network, low latency, perf tools

Larry Woodman
Sr Distinguished Eng
R H Kernel Eng

Sanjay Rao
Sr Principal Perf+Scale Eng
Database, IO performance

8 May, 2019

Agenda: Performance Analysis Tuning Part I+II

- **RHEL Evolution 5->6->7-8 , What’s new for perf in RHEL8!**
  - Tuned and Perf Lab results
  - IO and Network Improvements
- **Disk IO**
  - Database / File system improvements w/ RHEL8
- **RHEL Memory management**
  - 5 Level Page Tables
  - NvDIMM arch and early certification / prelimin perf
  - NonUniform Memory Access (NUMA)
  - HugePages
- **Part II Meet The Experts Room 150 - all above + additional topics**
  - Low Latency Network (cpu_partitioning tuned)
  - XDP, eBPF, cgroup V2 (tech preview)
  - Perf tool, tuna, PCP copilot … etc

RHEL Performance Evolution

RHEL5	RHEL6	RHEL7	RHEL8
Static Hugepages CPU Sets Ktune on/off CPU Affinity (taskset) NUMA Pinning (numactl) irqbalance	Transparent Hugepages Tuned - Choose Profile NUMAD - userspace cgroups irqbalance - NUMA enhanced	Tuned - throughput-performance (default) Automatic NUMA-balancing Containers/OCI - CRI-O (podman) irqbalance - NUMA enhanced	5 level PTEs (THP cont) Tuned: Throughput/ Lat - SSD/Nvdimm Multi-Arch: Intel/ AMD/ ARM/ Power Networking: XDP and eBPF Acceleration GPU/FPGA/Offloads

RHEL tuned parameters that effect performance (sysctls)

CPU Scheduler tunables

Throughput Performance
Scheduler quantum (default 4/10 ms,-> 10/15 ms)

- kernel.sched_min_granularity_ns=10000000
- kernel_sched_wakeup _granularity_ns = 15000000

Weight function on how often to migrate - 5ms -> 50ms

- kernel.sched_migration_cost_ns=50000000

Latency Performance tuning

- Decrease quantum above to 4 /10 ms

Adjust power management - BIOS OS controlled

- pstates - governor=performance
- energy_perf_bias=performance
- cstate - force_latency=1

Disable scaning tools for better determinism

- Disable numa balance
  - kernel.numa_balancing = 0
- Disable Transparent HugePages
  - mm.redhat_transparent_hugepage never

VM Tunables

Reclaim Ratios

- vm.swappiness
- vm.vfs_cache_pressure
- vm.min_free_kbytes

Writeback Parameters 30/10 -> 10/3

- vm.dirty_background_ratio
- vm.dirty_ratio

Readahead parameters per device 512-> 4k

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

Non-Uniform Memory Access (NUMA) Hugepages

Auto numa balancing at scheduling time

- kernel.numa_balancing = 1
- Adjust numa scan interval 1000 ms -> 100 ms
- vm.zone_reclaim_mode = 1 (reclaim local node vs spill)

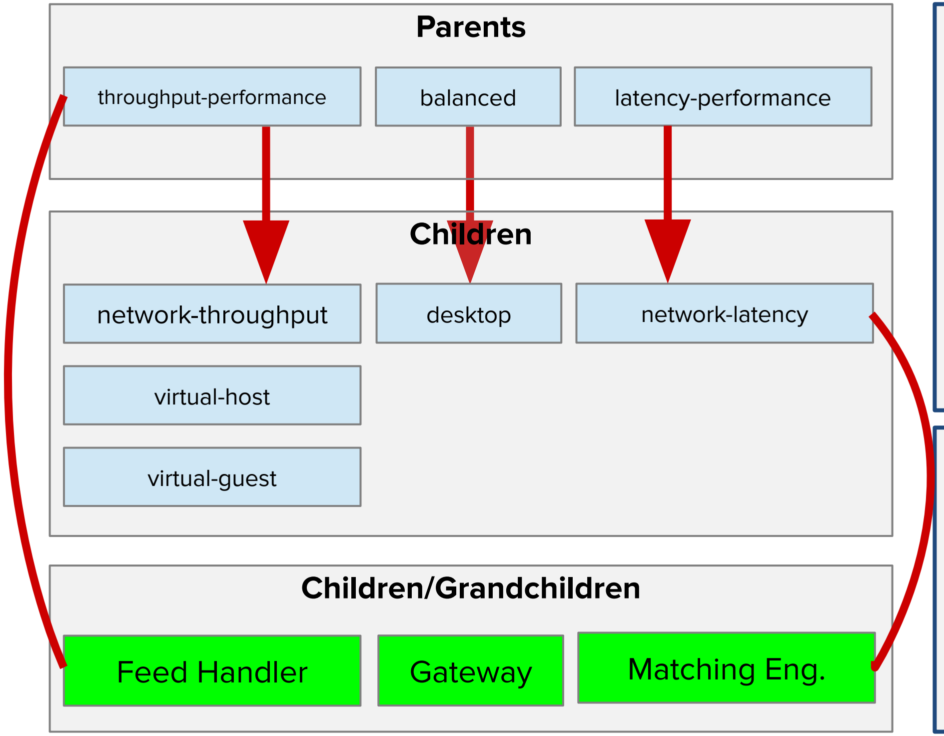
Transparent HugePages

- mm.redhat_transparent_hugepage enabled

Tuned Profiles throughout Red Hat's Product Line

Product	Profile
RHEL7/8 Laptop/Workstation	**balanced**
RHEL7/8 Server/HPC	**throughput-performance**
RHEL7/8 KVM Host, Guest	**virtual-host/guest**
RHV/OSP	**virtual-host**
Red Hat Storage	**rhs-high-throughput**
RHEL OSP (compute node)	**Virtual-host/guest**
Open Shift Platform	**control-plane/node**
NFV / RT	**cpu_partitioning/rt**

Tuned *network-latency* Profile



latency-performance

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

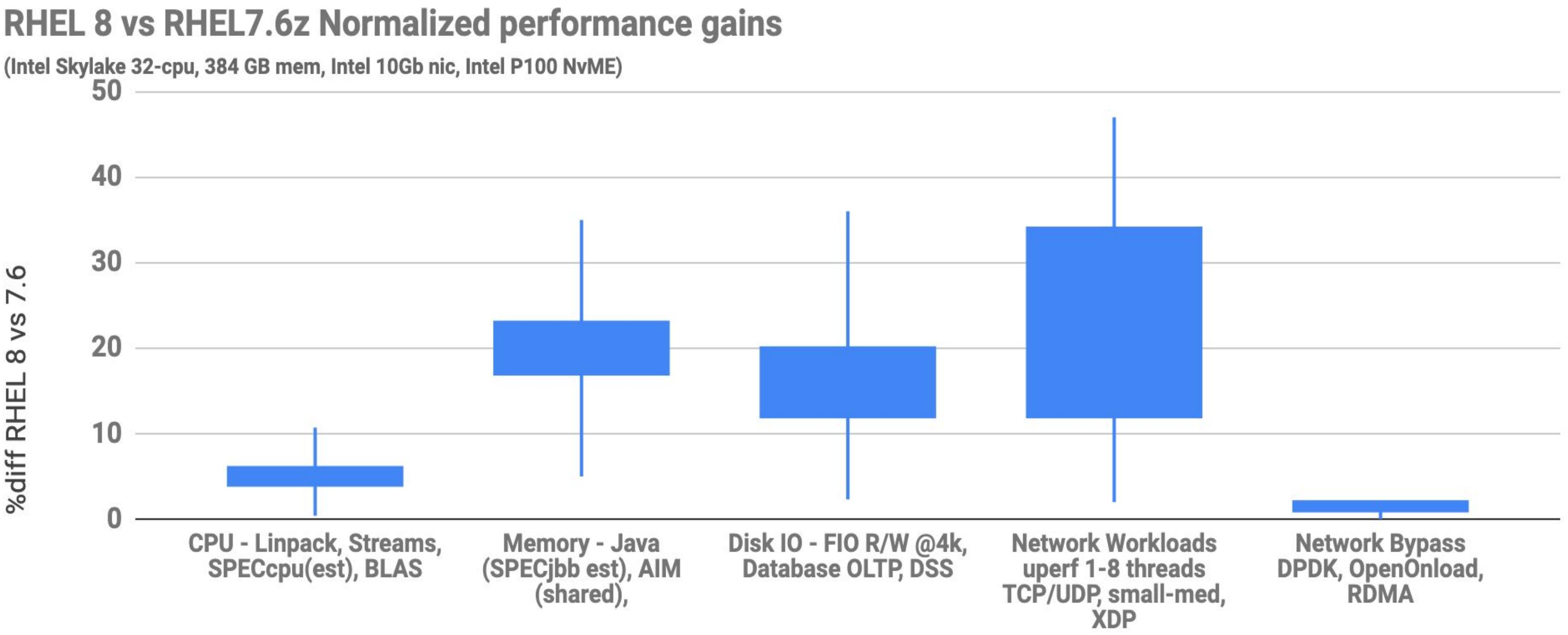
network-latency

include=latency-performance
transparent_hugepages=never
net.core.busy_read=50
net.core.busy_poll=50
net.ipv4.tcp_fastopen=3
kernel.numa_balancing=0

RHEL8 Performance Lab Results

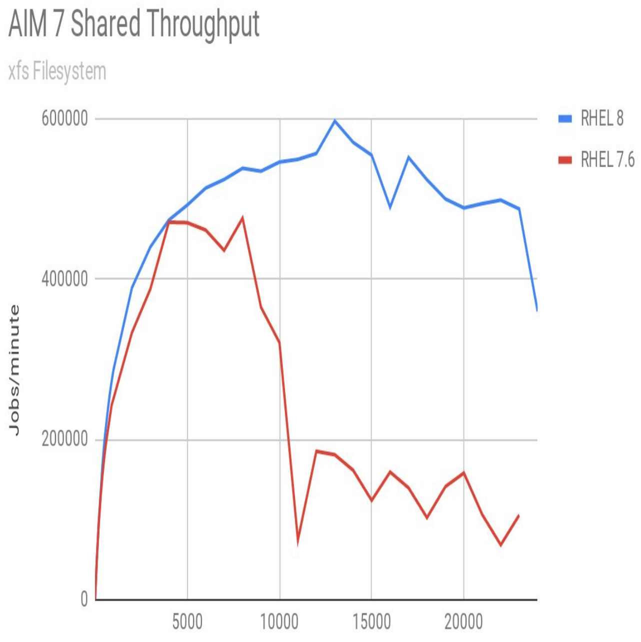
- Performance Testing at Red Hat
  - CPU
    - Intel - Haswell /Broadwells, SkyLakes, AMD EPYC, ARM
  - Memory - virtual memory
    - 512 Gb upto 24 TB (partner limits)
  - Networks
    - Intel, Mellanox, Solarflare 10, 25, 40, 100 Gb
  - Disk/Filesystem IO
    - xfs, ext4/3, gfs2, nfs, gluster, ceph
  - Security - CVE impacts, Retpoline for all Intel

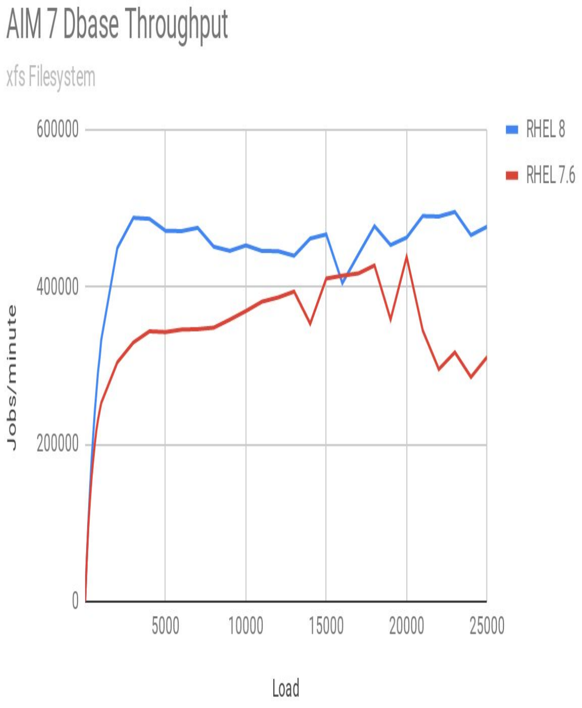
RHEL 8 vs RHEL 7 Workload Performance Gains

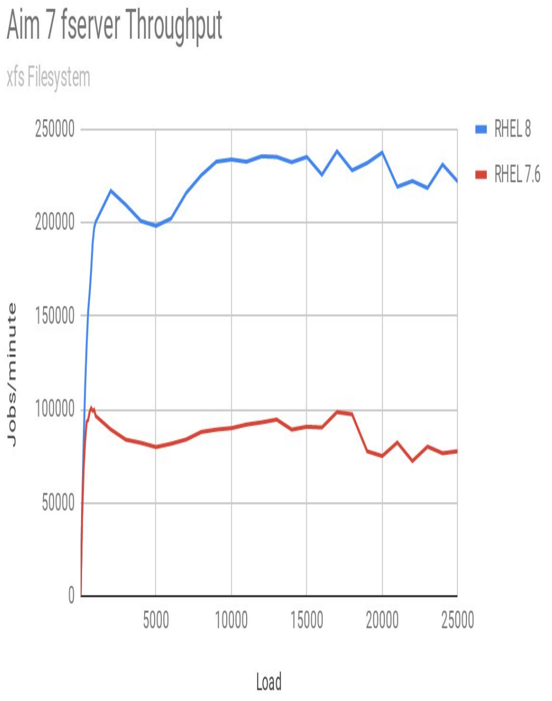


RHEL 8 Performance of AIM7 w/ different loads

AIM7 XFS - multiuser, throughput in jobs/min (Bigger==Better)







RHEL 8 Performance improvements w/ AIM7

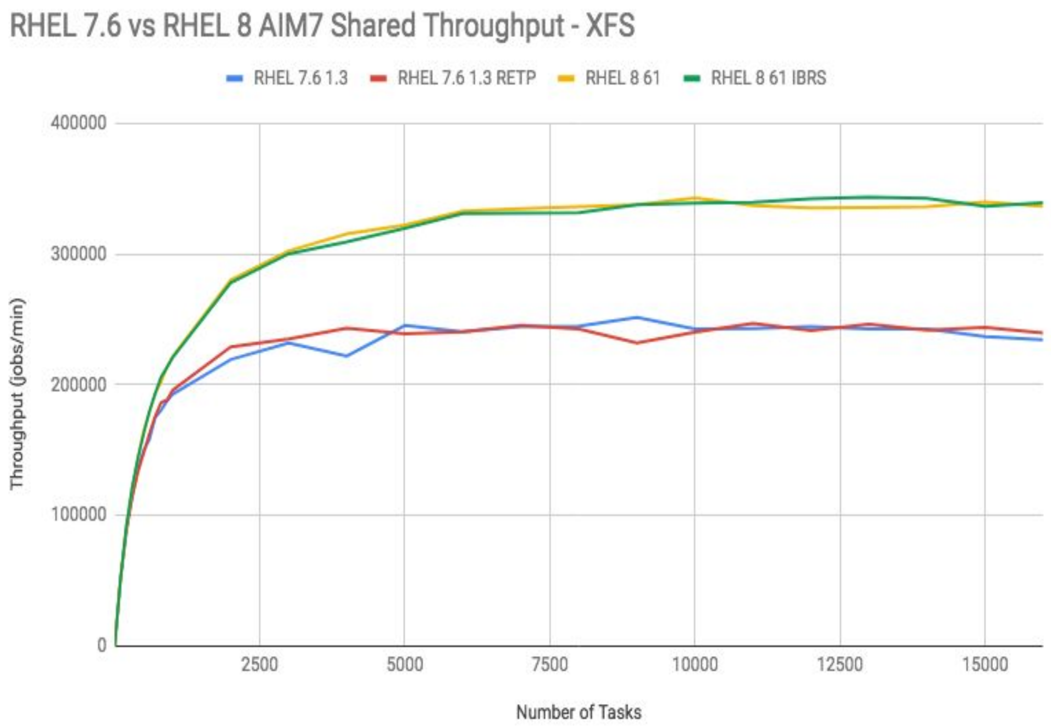
AIM7 Shared User Mix - multiuser benchmark, throughput in jobs/min +35.6%

RHEL 7.6, page fault stack not present.

raw_spin_unlock_irqrestore
_raw_spin_unlock_irqrestore
__wake_up
xlog_state_do_callback
xlog_state_done_syncing
xlog_iodone
xfs_buf_ioend
Xfs_buf_ioend_work

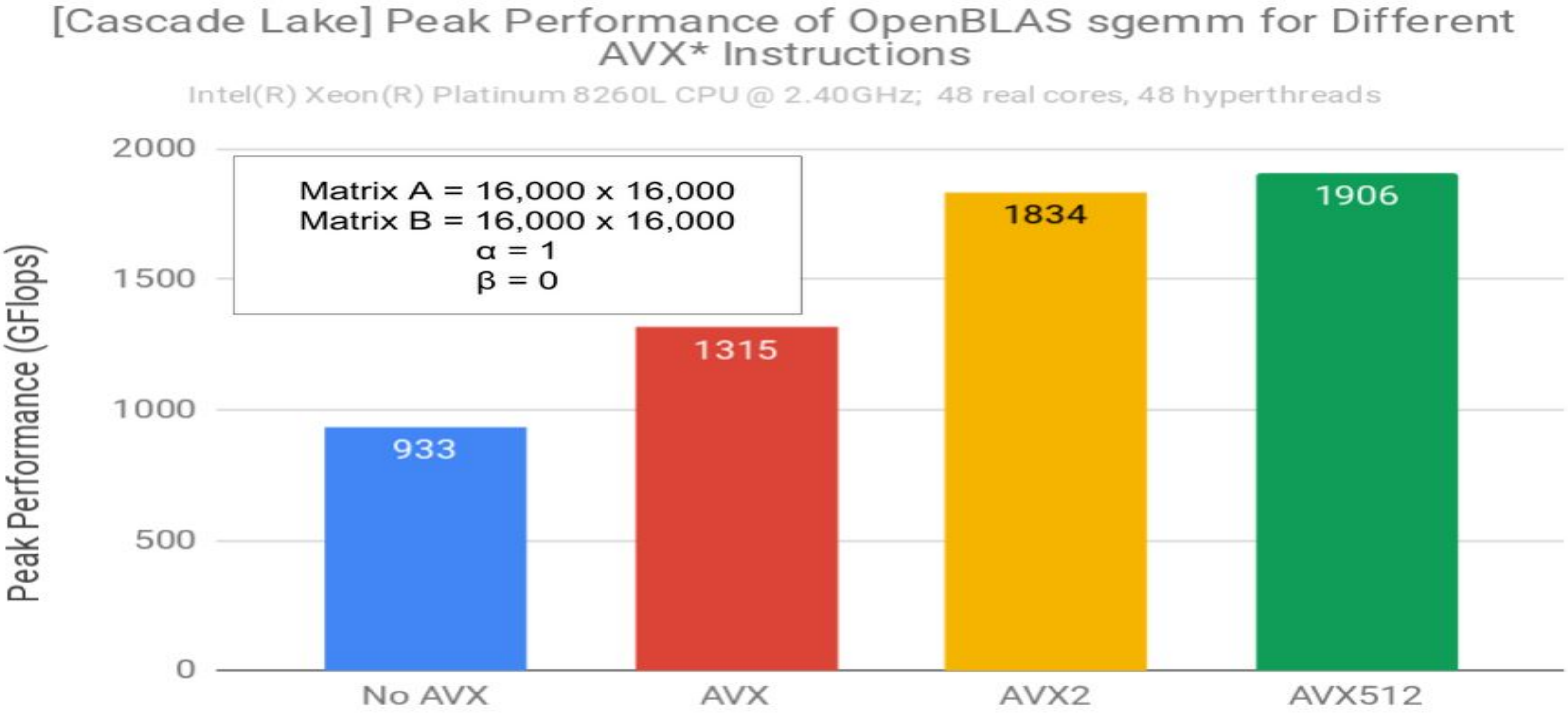
RHEL 8

filemap_map_pages+187
handle_pte_fault+2406
__handle_mm_fault+1066
handle_mm_fault+218
__do_page_fault+586
do_page_fault+50
page_fault+30



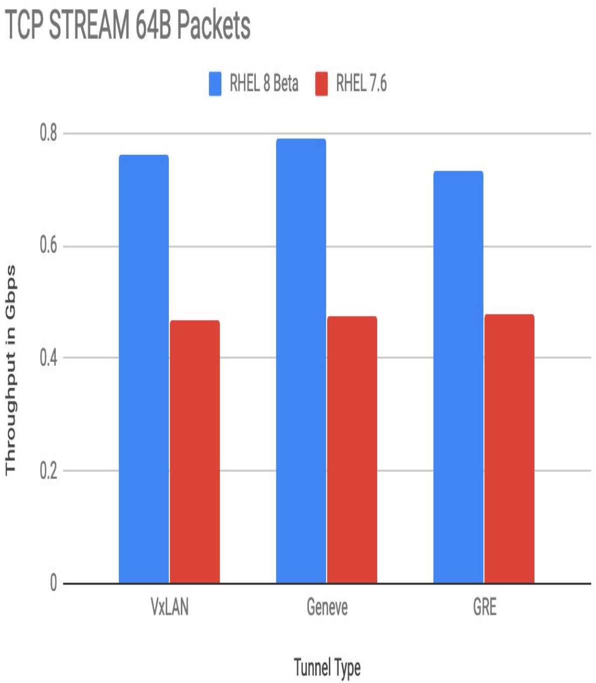
RHEL 8 Performance BLAS w/ AVX* Inst

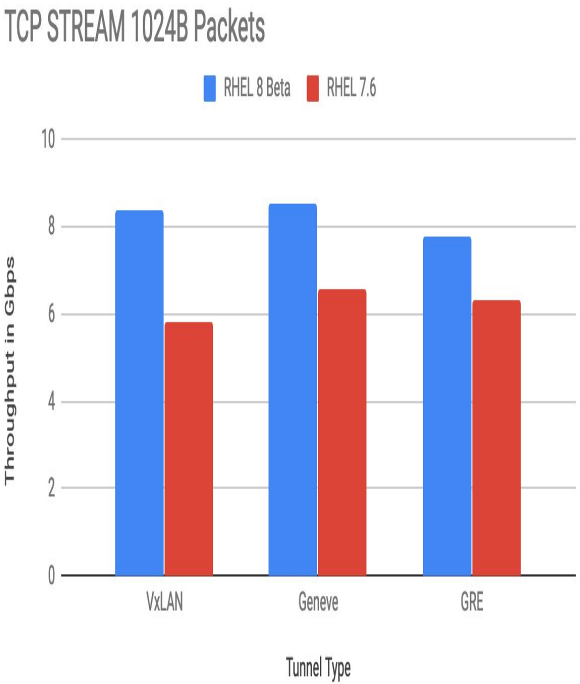
Intel 2nd generation of Intel® Xeon® Scalable processors

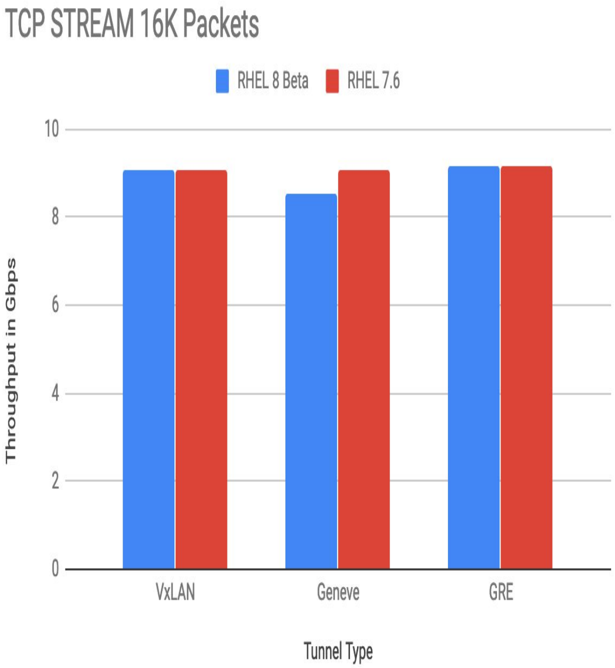


RHEL 8 Network Performance w/ uperf

Open Stack Control Plane Network Performance - 10 Gb Intel Nic

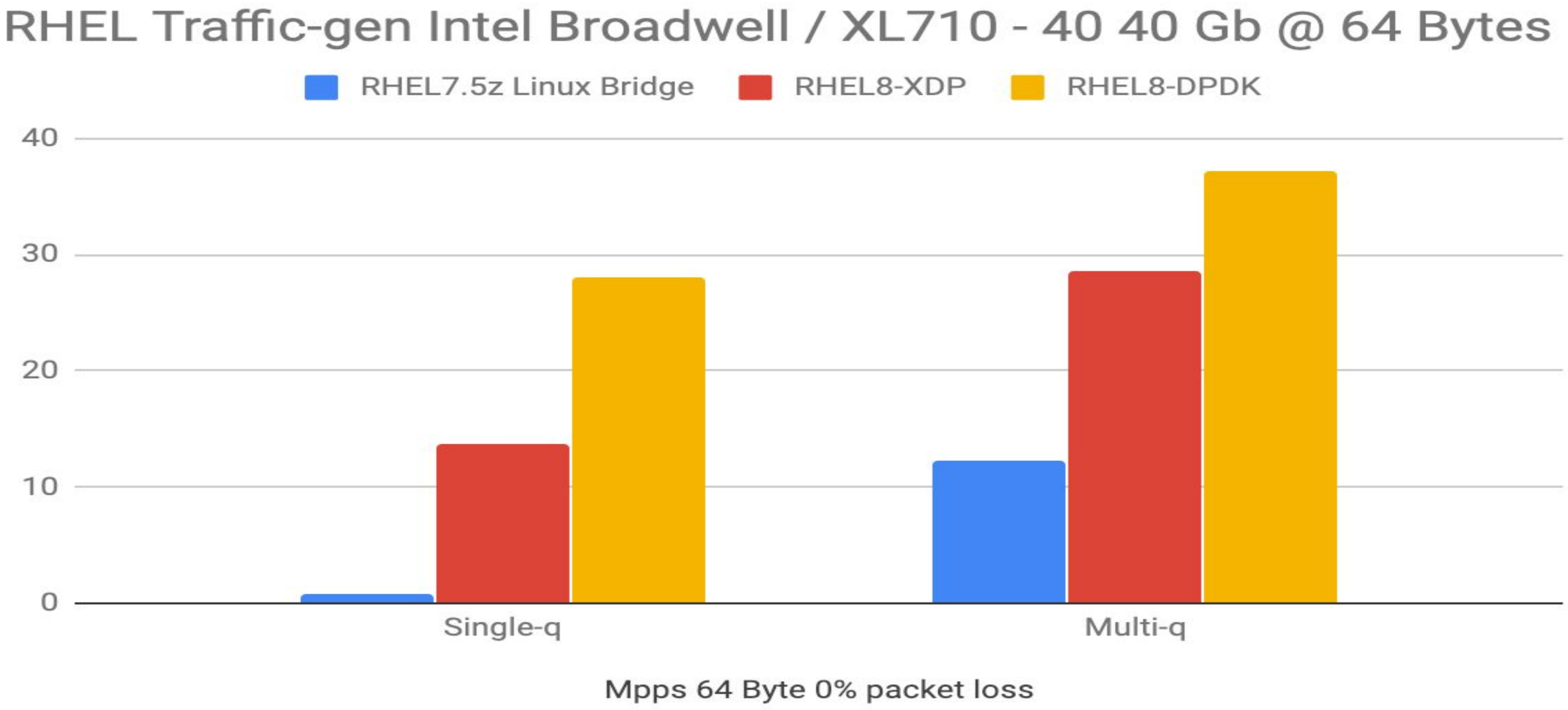






RHEL8 Tech Preview - XDP Performance

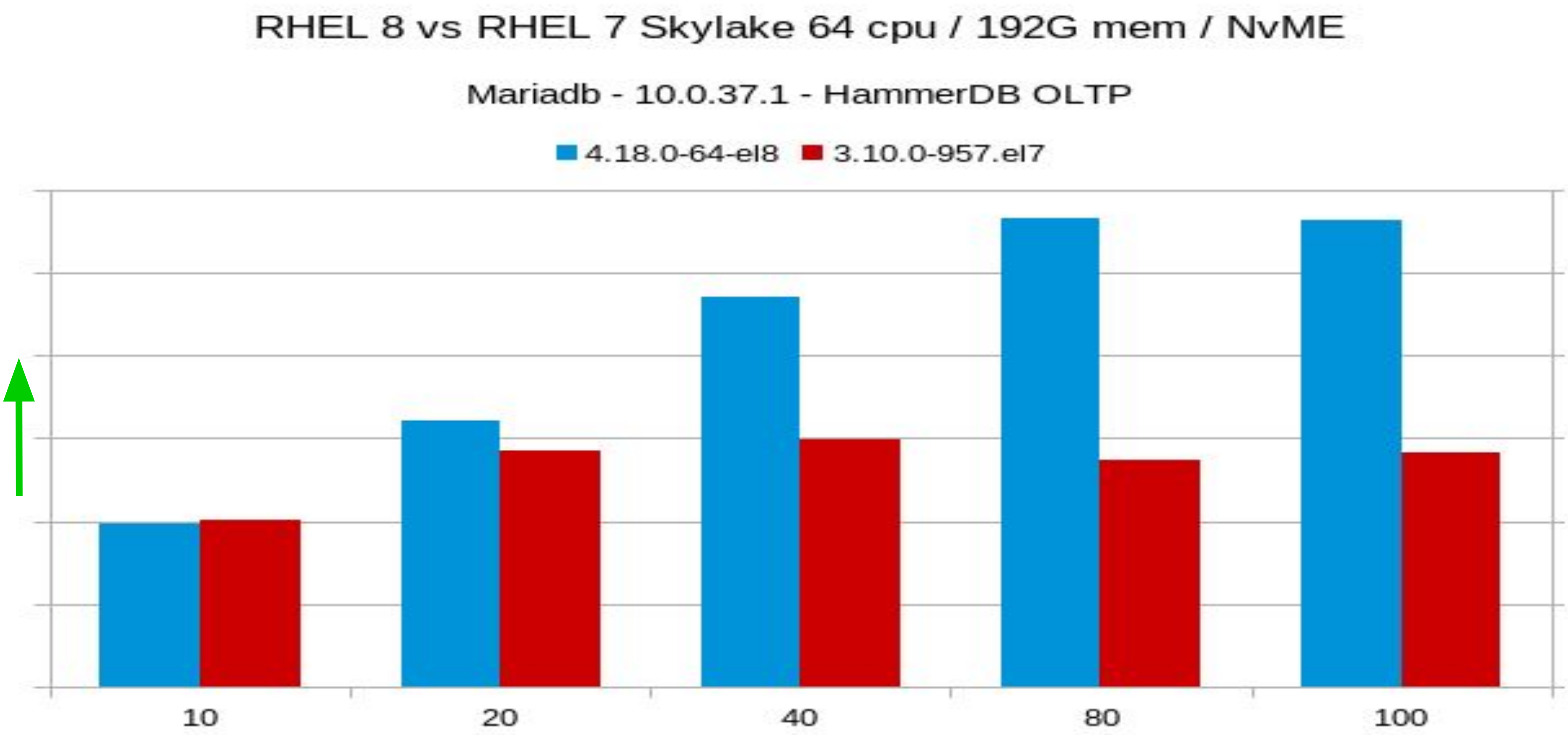
- **New Network Performance – TCP vs XDP vs DPDK in RHEL8 (2-20x gain)**



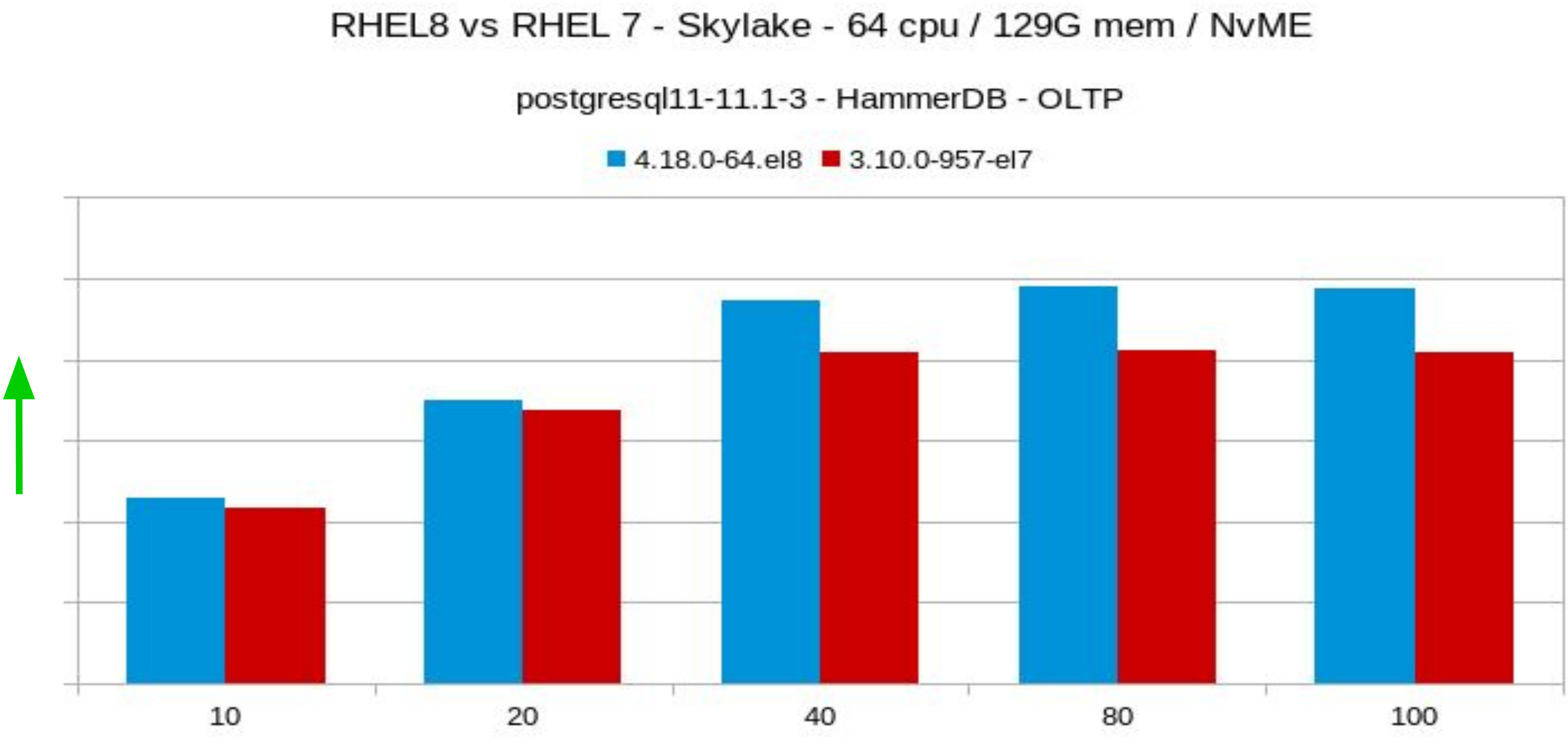
RHEL 8 - Database tuning tips

- **MariaDB**
    - Huge pages
        - Reduce TLB misses
        - For wiring down database pages
        - Prevent swapping
    - Lower dirty background ratio / Increase dirty ratio
        - To start early reclaim of dirty blocks
    - Size buffer pool based on user connections (or use connection pooling)
        - To prevent memory pressure
- **Postgres**
    - Use Huge pages
        - Reduce TLB misses
        - For wiring down database pages
        - Prevent swapping
    - Lower dirty background ratio / Increase dirty ratio
        - To start early reclaim of dirty blocks
    - Configure Shared buffers as well as effective cache size to avoid memory pressure

RHEL 8 Performance Open Source DBs



RHEL 8 Performance Open Source DBs

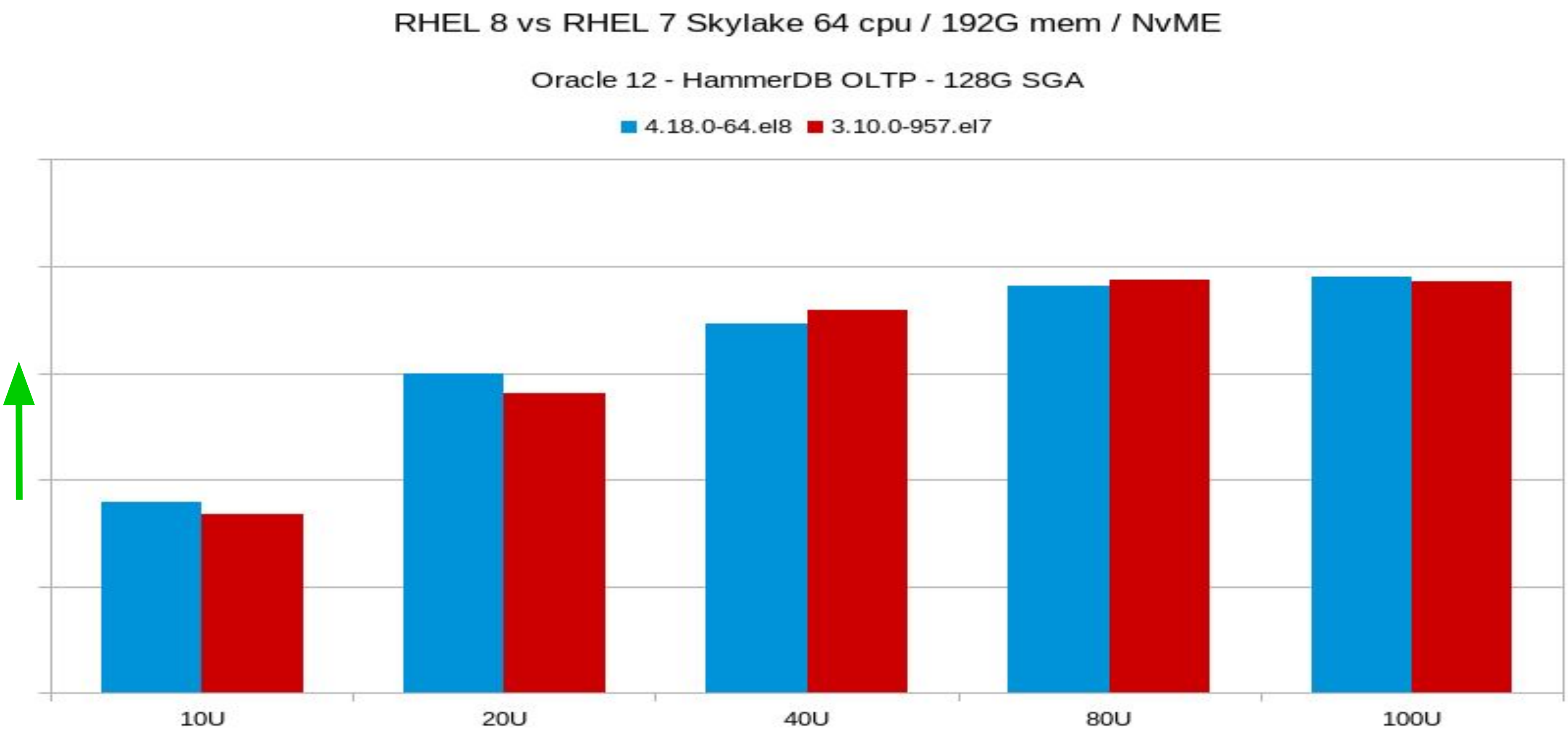


RHEL 8 - Database tuning tips

Oracle 12c

- Implement huge pages
  - Reduce TLB misses
  - For wiring down database pages
  - Prevent swapping
- Turn off Auto numa
  - To prevent conflict with Oracle NUMA optimization
- Turn of transparent huge pages
  - To reduce CPU overhead of THP scan
- Lower dirty background ratio
  - Start flushing dirty blocks and reclaim
- Increase dirty ratio
  - Delay the process of hitting dirty blocks threshold
- Use numa pinning in multiple instance environments (including listener process)
  - To take advantage of NUMA localization
- Size SGA based on user connections (or use connection pooling)
  - To prevent memory pressure

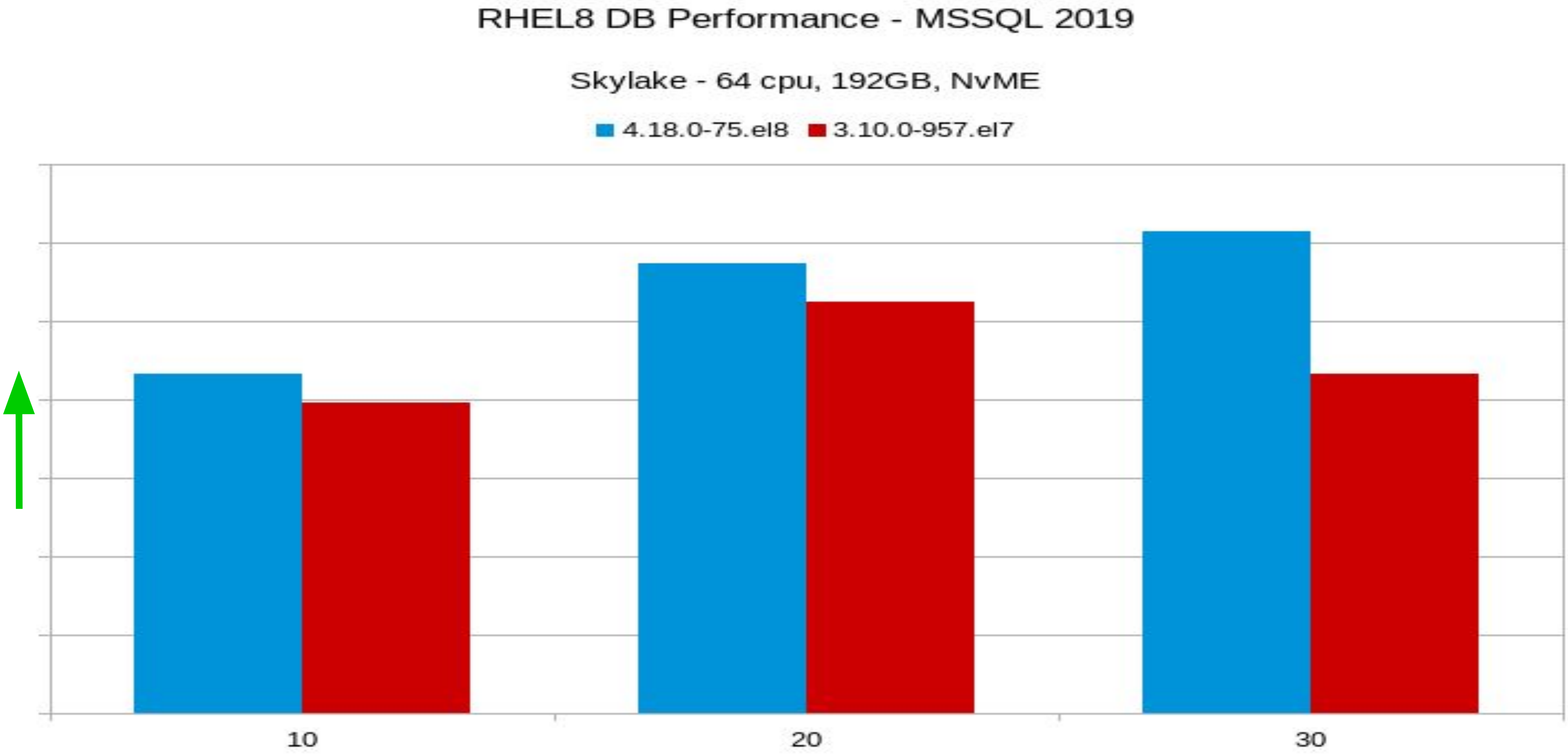
RHEL 8 Performance Legacy DB



New with SQL Server on Red Hat Enterprise Linux 8: Increased Performance

- Updates to the mssql tuned profile optimize tuning for decision support workloads
- New TCP/IP stack delivers increased performance and BBR congestion control
- Storage block devices now use multiqueue scheduling to make the best use of bandwidth available from modern flash-based storage devices
- XFS FUA enhancements for SQL Server - write request I/O traffic is reduced by ~50% for a SQL Server write-intensive workloads

RHEL 8 Performance DB Performance



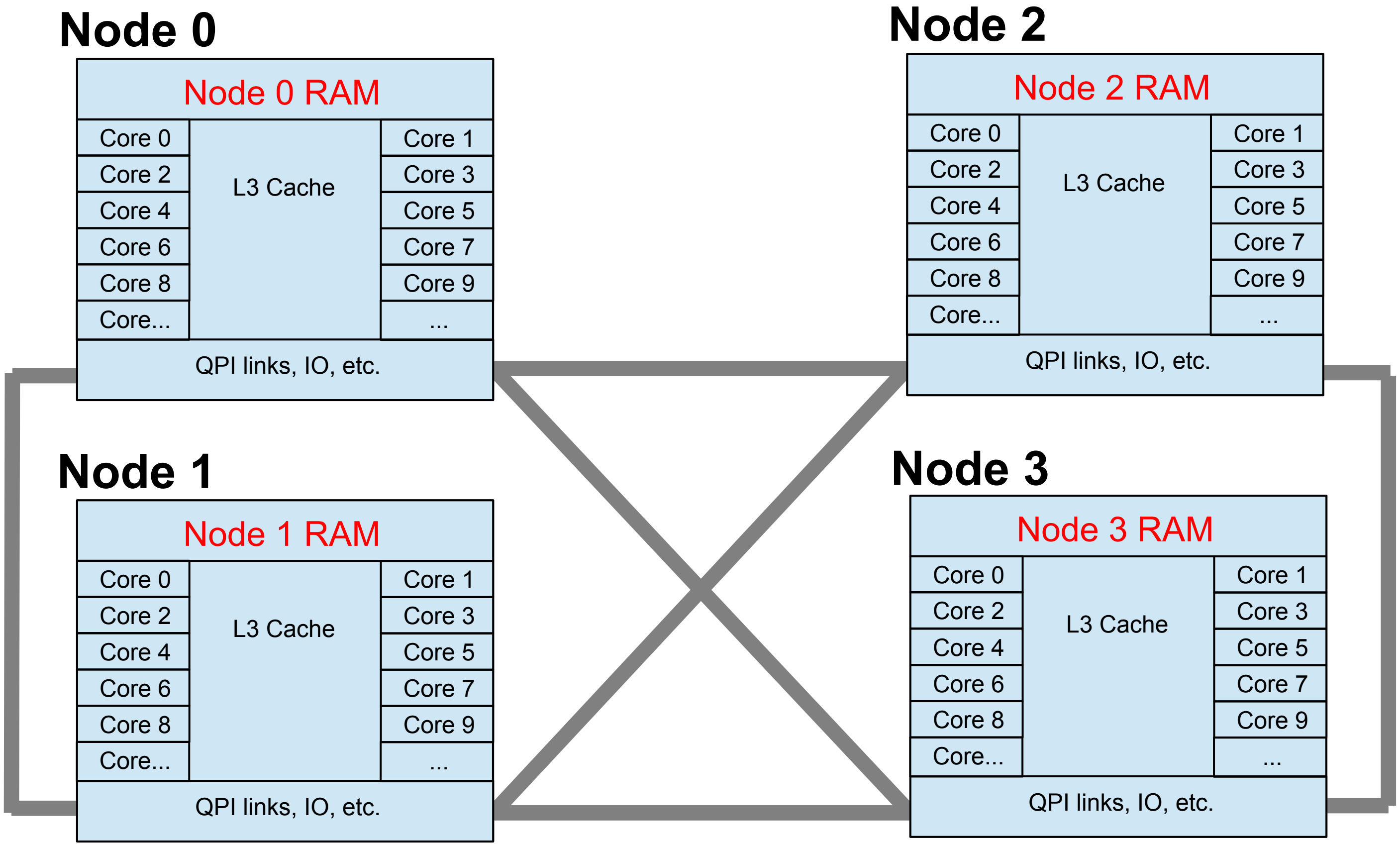
RHEL 8 Performance Summary

Performance Highlights:

- Microbenchmarks
  - Multiq SCSI - direct attached and fiberchannel, iozone, fio
  - Network – Netperf/Uperf (TCP/UDP) - improved sm/med packet
  - AIM multiuser (shared, db, fileserver) - lower syscall overhead, VM changes.
  - CVE impacts, use retpoline for spectre Intel (on Skylake vs IBRS)
- Databases – Oracle, MariaDB, Postgres, Mongo, SQLserver
  - Improvements in XFS journal / FUA opts
  - Virtual Memory (contention), algorithm for VM flushing dirty pages
- Java – SPECjbb, MAX bops 2005 and 2015
- SAP – ERP Sales and Distribution (SD bm) and Hana Analytics tpcds/bw loads
- SAS – Mixed Analytics (scale up), SAS Grid (cluster)

Numa and Memory Perf Tuning

Typical Four-Node NUMA System



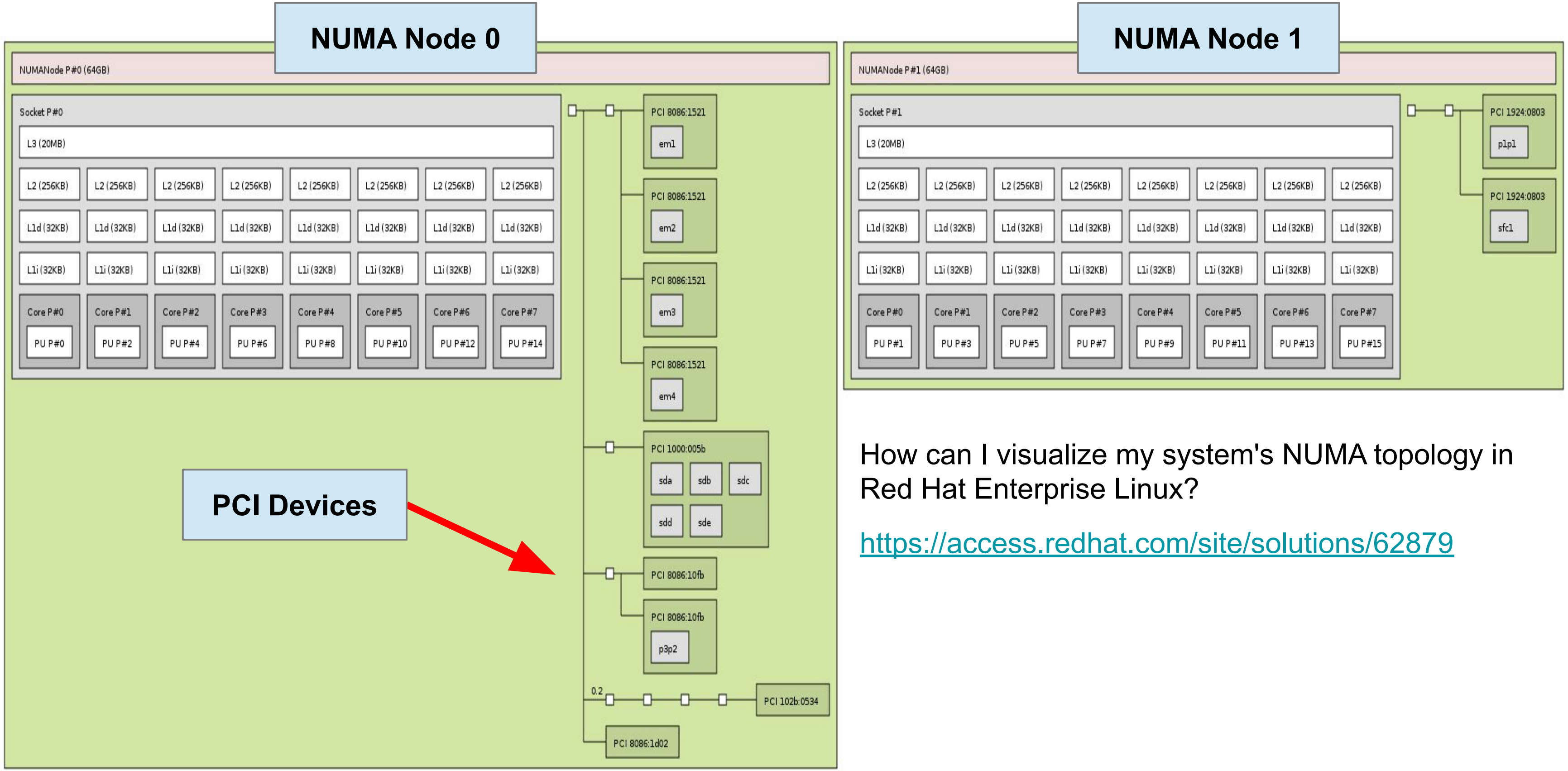
Tools to display CPU and Memory (NUMA)

```
# 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 NUMA Topology: lstopo



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

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

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

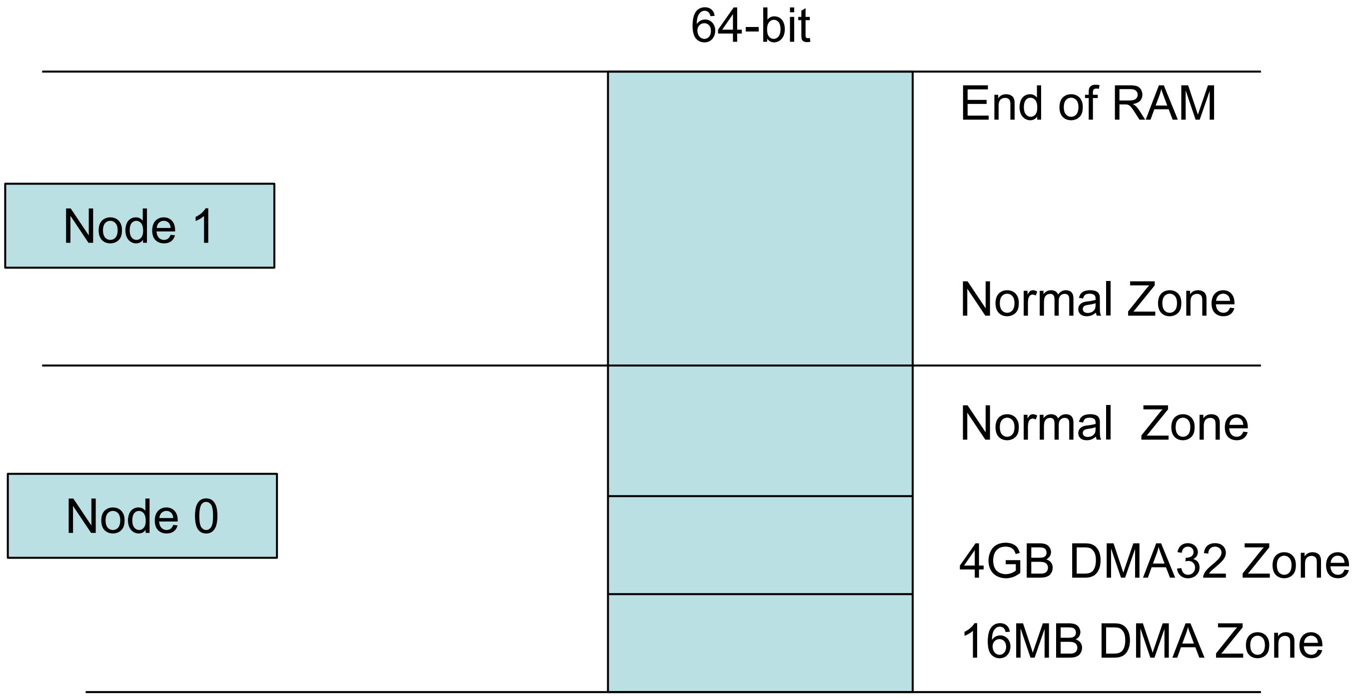
```
# numastat -c qemu  Per-node process memory usage (in Mbs)

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

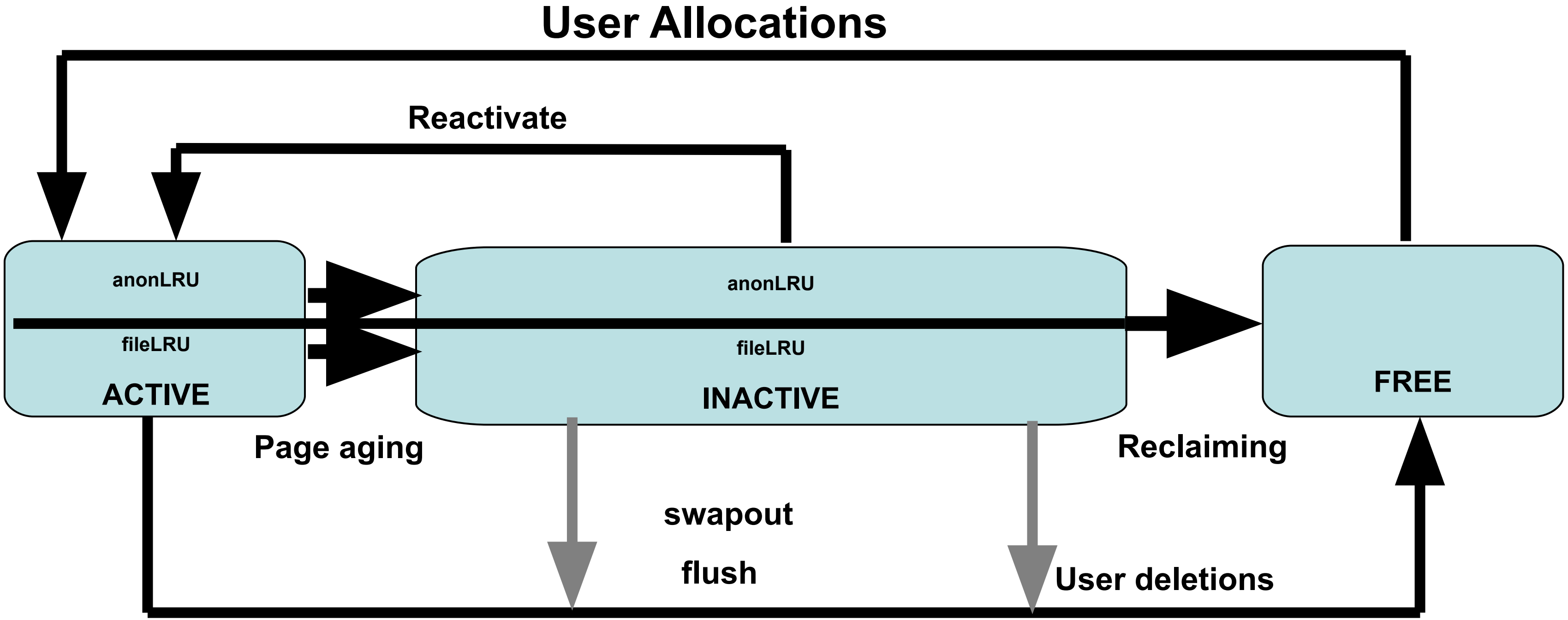
```
# numastat -c qemu

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

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

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

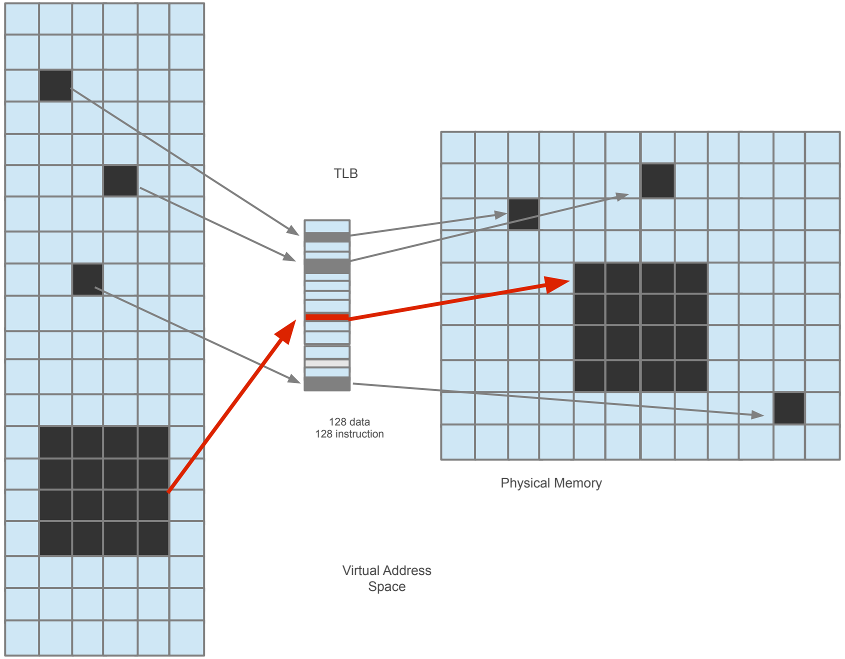
    - #echo 0 > /proc/sys/vm/zone_reclaim_mode
    - Allocate from all nodes before reclaiming memory

- 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

HugePages

Hugepages in RHEL

- X86_64 supports 3 page sizes:
  - 4KB, 2MB, 1GB
- 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&8 allows runtime allocation & freeing
  - Used via hugetlbfs
- Transparent HugePages 2MB
  - On by default via boot args or /sys
  - Used for anonymous memory



2MB standard and 1GB 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
```

```
hugepagesz=1G, hugepagesz=1G, hugepages=8

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

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

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

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
```

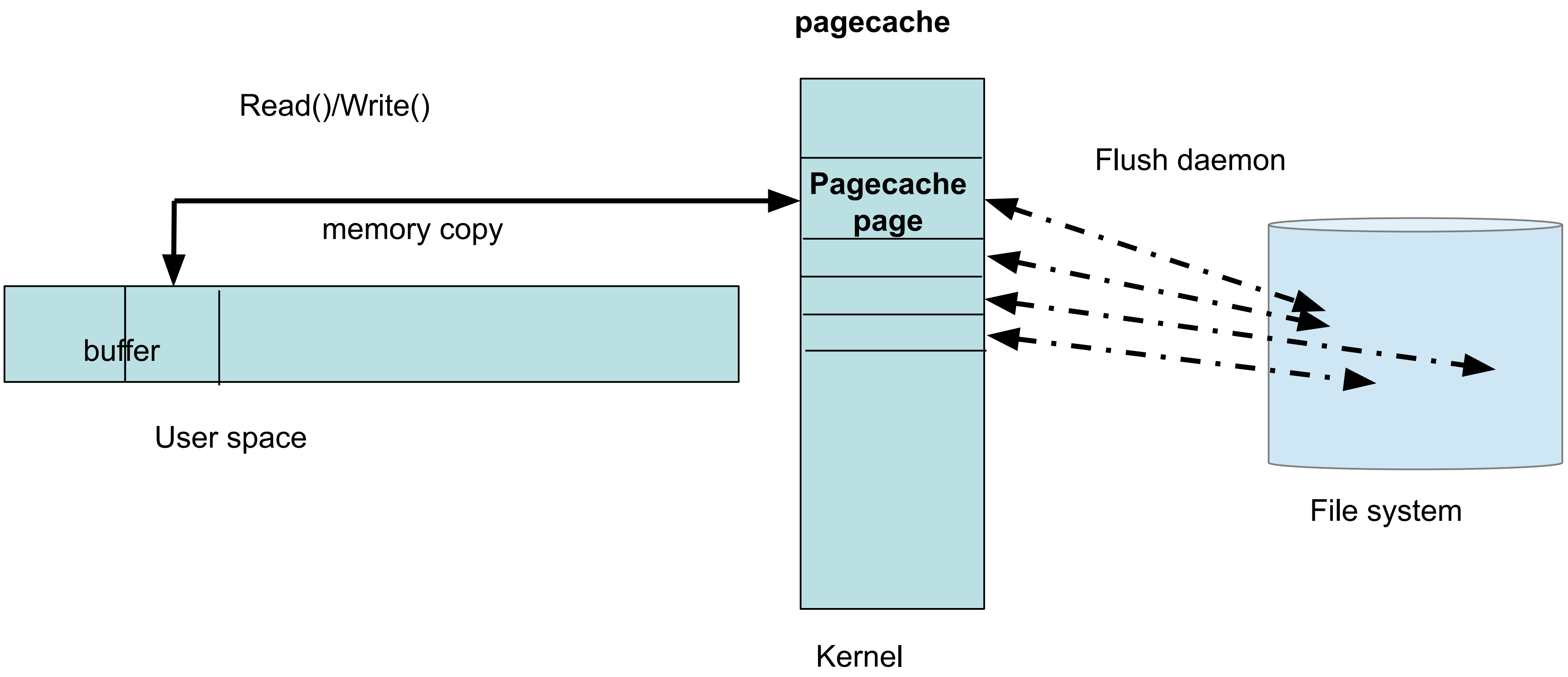
SPEEDUP 12.4/7.0 = 1.77x, 56%

RHEL Disk I/O and I/O Elevators

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

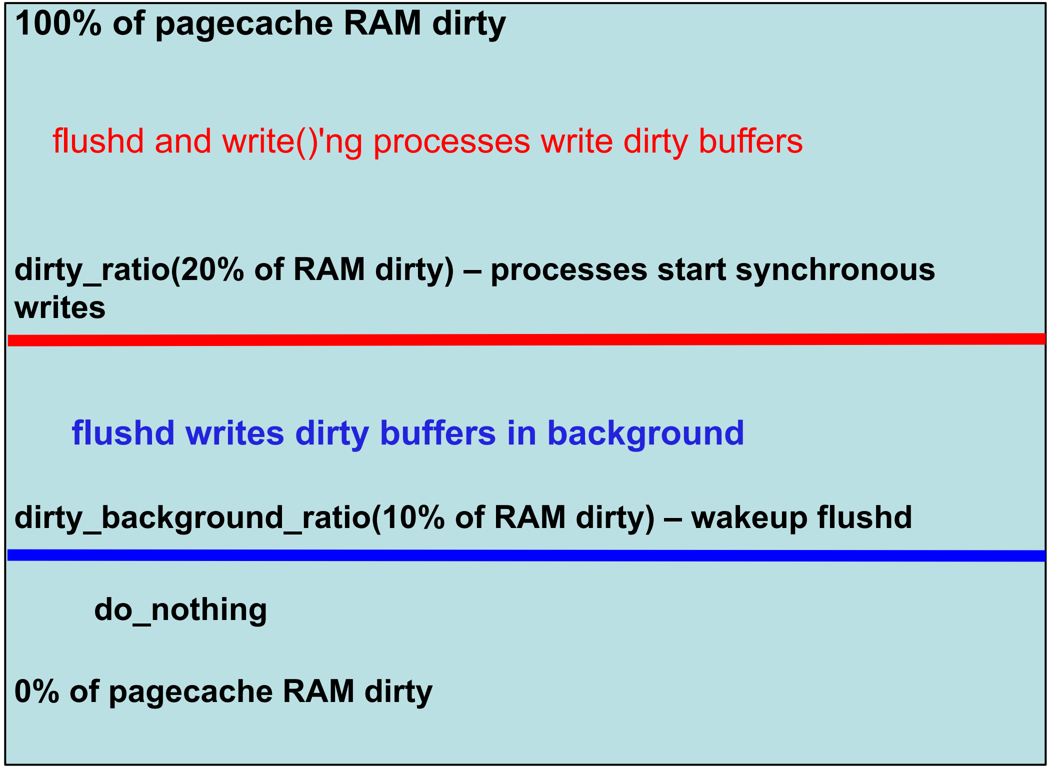


Virtual Memory Manager (VM) Tunables

- **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_ratio and dirty_background_ratio

pagecache



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

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

On very large memory systems, consider using more granularity by using

- **dirty_background_bytes**
- **dirty_bytes**

Tuning Memory – swappiness

- Not needed as much in RHEL7 & RHEL8
- Controls how aggressively the system reclaims “mapped” memory
- Default - 60%
- Decreasing: more aggressive reclaiming of unmapped pagecache memory, thereby delaying swapping
- Increasing: more aggressive swapping of mapped memory

- **Avoid swapping of database shared memory at all costs**

I/O Tuning – Database layout – vmstat

OLTP Workload Fibre Channel Storage

r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
62	19	5092	44894312	130704	76267048	0	0	8530	144255	35350	113257	43	4	45	9	0
3	20	5092	43670800	131216	77248544	0	0	6146	152650	29368	93373	33	3	53	11	0
21	27	5092	42975532	131620	77808736	0	0	2973	147526	20886	66140	20	2	65	13	0
7	20	5092	42555764	132012	78158840	0	0	2206	136012	19526	61452	17	2	69	12	0
25	18	5092	42002368	132536	78647472	0	0	2466	144191	20255	63366	19	2	67	11	0
4	21	5092	41469552	132944	79111672	0	0	2581	144470	21125	66029	22	2	65	11	0
1	32	5092	40814696	133368	79699200	0	0	2608	151518	21967	69841	23	2	64	11	0
4	17	5092	40046620	133804	80385232	0	0	2638	151933	23044	70294	24	2	64	10	0
17	14	5092	39499580	134204	80894864	0	0	2377	152805	23663	72655	25	2	62	10	0
35	14	5092	38910024	134596	81436952	0	0	2278	152864	24944	74231	27	2	61	9	0
20	13	5092	38313900	135032	81978544	0	0	2091	156207	24257	72968	26	2	62	10	0
1	14	5092	37831076	135528	82389120	0	0	1332	155549	19798	58195	20	2	67	11	0
23	24	5092	37430772	135936	82749040	0	0	1955	145791	19557	56133	18	2	66	14	0
34	12	5092	36864500	136396	83297184	0	0	1546	141385	19957	56894	19	2	67	13	0

OLTP Workload PCI SSD Storage

r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
77	0	6604	55179876	358888	66226960	0	0	7325	266895	70185	149686	90	7	4	0	0
77	1	6604	50630092	359288	70476248	0	0	6873	306900	70166	149804	88	7	5	0	0
76	3	6604	46031168	360132	74444776	0	0	5818	574286	77388	177454	88	8	4	0	0
81	1	6604	41510608	360512	78641480	0	0	4970	452939	75322	168464	89	7	3	0	0
82	3	6604	35358836	361012	84466256	0	0	4011	441042	74022	162443	88	7	4	0	0
81	3	6604	34991452	361892	84740008	0	0	2126	440876	73702	161618	88	7	5	0	0
79	1	6604	34939792	362296	84747016	0	0	2323	400324	73091	161592	90	6	3	0	0
79	0	6604	34879644	362992	84754016	0	0	2275	412631	73271	160766	89	6	4	0	0
76	2	6604	34844616	363396	84760976	0	0	2275	415777	73019	158614	89	6	4	0	0
61	3	6604	34808680	363828	84768016	0	0	2209	401522	72367	159100	89	6	4	0	0
77	1	6604	34781944	364180	84774992	0	0	2172	401966	73253	159064	90	6	4	0	0
54	4	6604	34724948	364772	84803456	0	0	3031	421299	72990	156224	89	6	4	0	0
80	2	6604	34701500	365500	84809072	0	0	2216	573246	76404	175922	88	7	5	1	0

Memory Stats

I/O Stats

CPU stats

Swap stats

Summary - RHEL Performance Tech/Tunables

- **RHEL6/7/8**
  - **Tuned** - apply profiles for throughput (default) vs latency
    - needed w/ more with advanced devices), per product (Open Shift OCP, Realtime RT, NFV cpu-part) vendors, (sap, sqlserver).
    - Adjust c-states, dirty-ratios, sched quantum/migration cost.
  - **NumaD/ AutoNUMA - With Red Hat Enterprise Linux**
    - AutoNUMA / NumaD can significantly improve performance for server consolidation or replicated parallel workloads.
  - **HugePages** wired-down, THP for vm's containers, DB/Java 2MB or 1GB
  - **Tools** - *stat, PCP, collectd, Perf (c-2-c), NUMAstat/ctl), tuna, pbench

  tools to measure and/or fine control your application on RHEL.

- **Q+A at "Meet The Experts" - Room 150**

RHEL tuned parameters that effect performance (sysctls)

CPU Scheduler tunables

Throughput Performance
Scheduler quantum (default 4/10 ms,-> 10/15 ms)

- kernel.sched_min_granularity_ns=10000000
- kernel_sched_wakeup _granularity_ns = 15000000

Weight function on how often to migrate - 5ms -> 50ms

- kernel.sched_migration_cost_ns=50000000

Latency Performance tuning

- Decrease quantum above to 4 /10 ms

Adjust power management - BIOS OS controlled

- pstates - governor=performance
- energy_perf_bias=performance
- cstate - force_latency=1

Disable scaning tools for better determinism

- Disable numa balance
    - kernel.numa_balancing = 0
- Disable Transparent HugePages
    - mm.redhat_transparent_hugepage never

VM Tunables

Reclaim Ratios

- vm.swappiness
- vm.vfs_cache_pressure
- vm.min_free_kbytes

Writeback Parameters 30/10 -> 10/3

- vm.dirty_background_ratio
- vm.dirty_ratio

Readahead parameters per device 512-> 4k

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

Non-Uniform Memory Access (NUMA) Hugepages

Auto numa balancing at scheduling time

- kernel.numa_balancing = 1
- Adjust numa scan interval 1000 ms -> 100 ms
- vm.zone_reclaim_mode = 1 (reclaim local node vs spill)

Transparent HugePages

- mm.redhat_transparent_hugepage enabled

Red Hat Performance Whitepapers

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

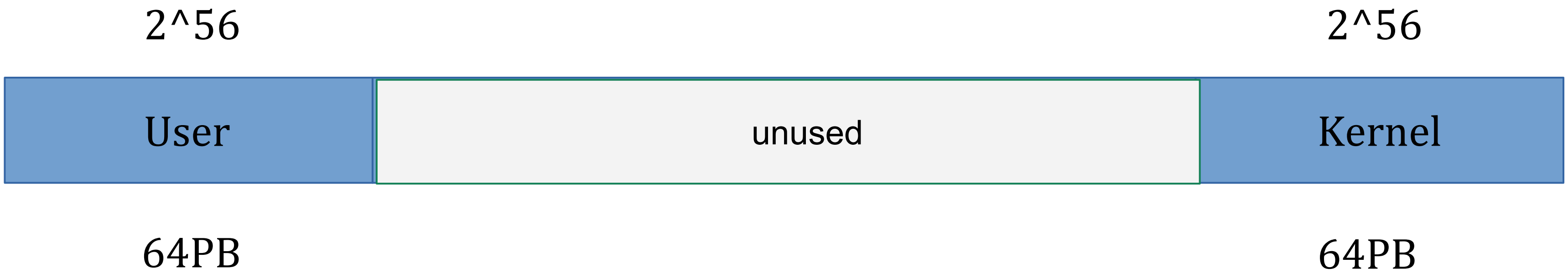
New to RHEL8:

X86_64 5-level page table/57-bit memory support

and

Persistent memory/NvDIMM support

57 bit address space
5-level page tables



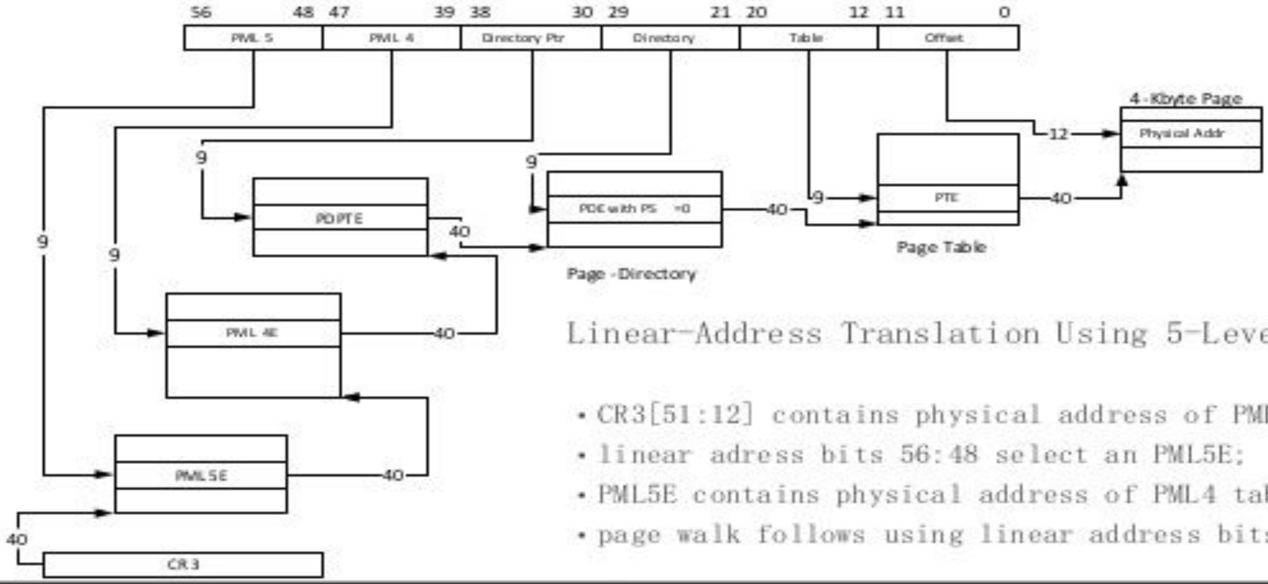
56 bit kernel address space

2^64 -2^56 | 2^64 - 2^55 | 2^64

Kmalloc: direct mapped RAM	Vmalloc: kernel virtual space
32PB/2^55(current HW limited to 4PB)	32PB/2^55

X86_64 5-level page table

5 level paging overview



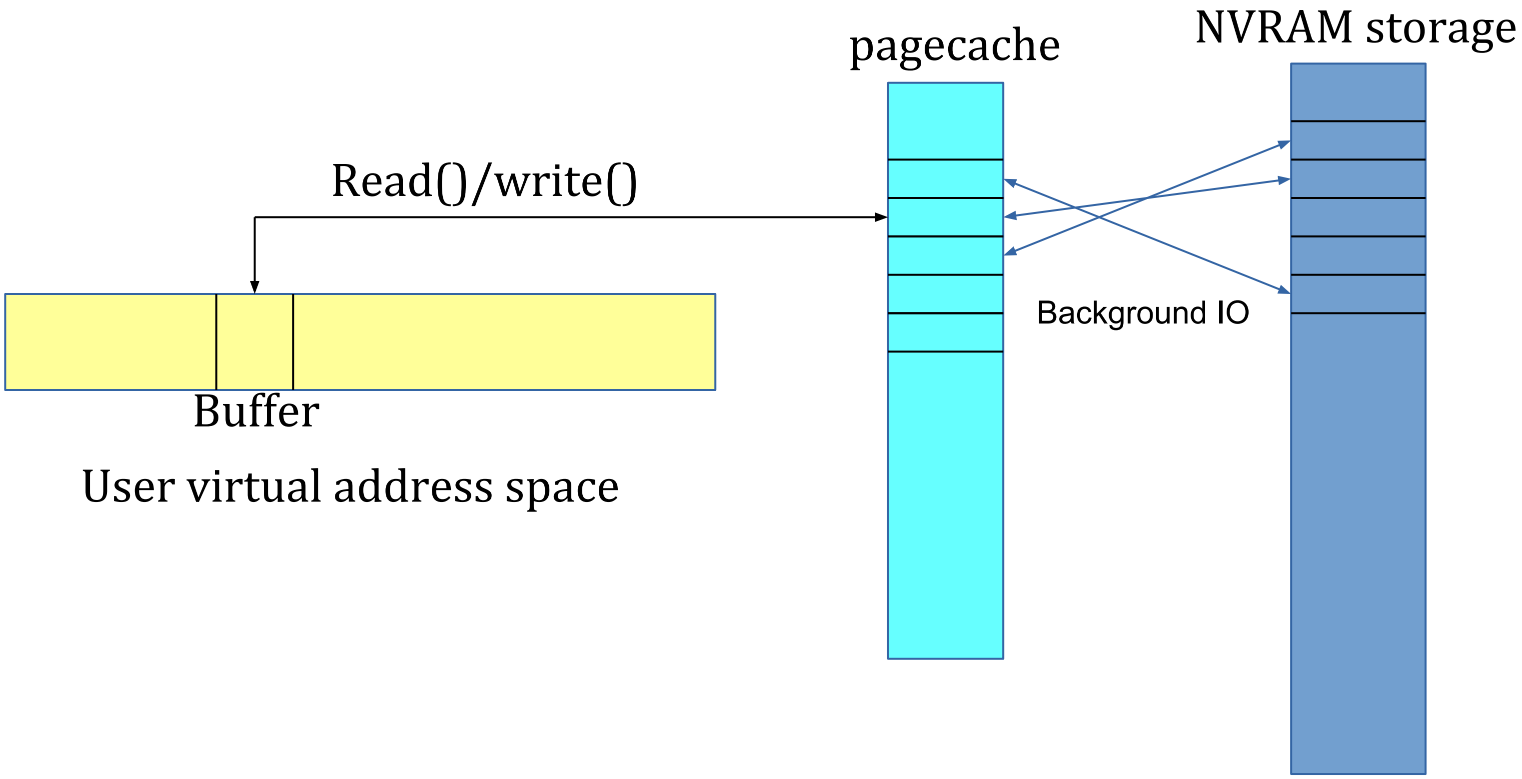
Linear-Address Translation Using 5-Level Paging

- CR3[51:12] contains physical address of PML5 table;
- linear adress bits 56:48 select an PML5E;
- PML5E contains physical address of PML4 table;
- page walk follows using linear address bits 47:0.

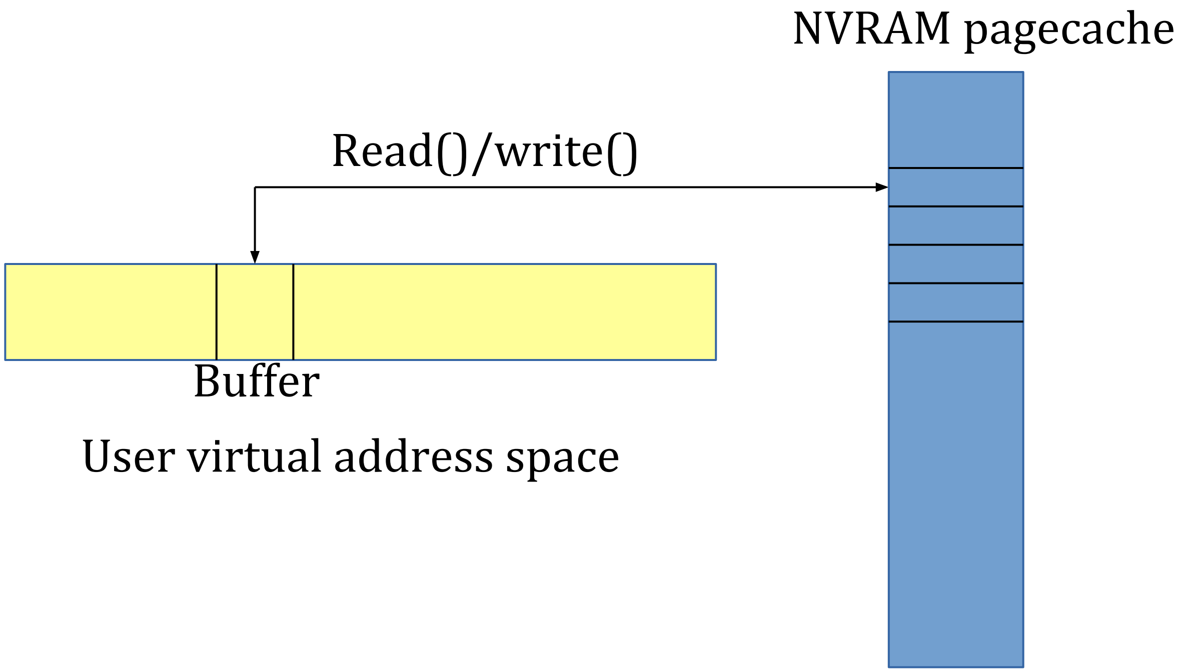
Persistent Memory/NvDIMM Support in RHEL

- Persistent memory is non-volatile memory NVDIMMs(aka NVRAM) that can be plugged into the DRAM slots.
  - Can/will be VERY large(need 5-page table support)
- NVRAM can not be accessed via the PCI interface like SSDs.
- NVRAM is accessed via the memory bus, its in the physical address space just like RAM
- NVRAM is primarily used for storage but can be configured as RAM(systems with NVDIMMs must also have DRAM).
  - Choosing if you want the NVDIMMs to be used as storage or RAM is controlled via BIOS settings.
  - In storage mode the DRAM is the system memory and the NVRAM is the storage.
  - In memory mode the NVDIMMs are the system memory and the DRAM is a cache for NVDIMMs.
- DAX – Direct Access File System: allows pages of NVRAM to be mapped directly in the pagecache.
  - Eliminates multiple copies of data
  - Reduces memory demand.
  - Eliminates need for pagecache write-back operations needed for disks and SSDs.

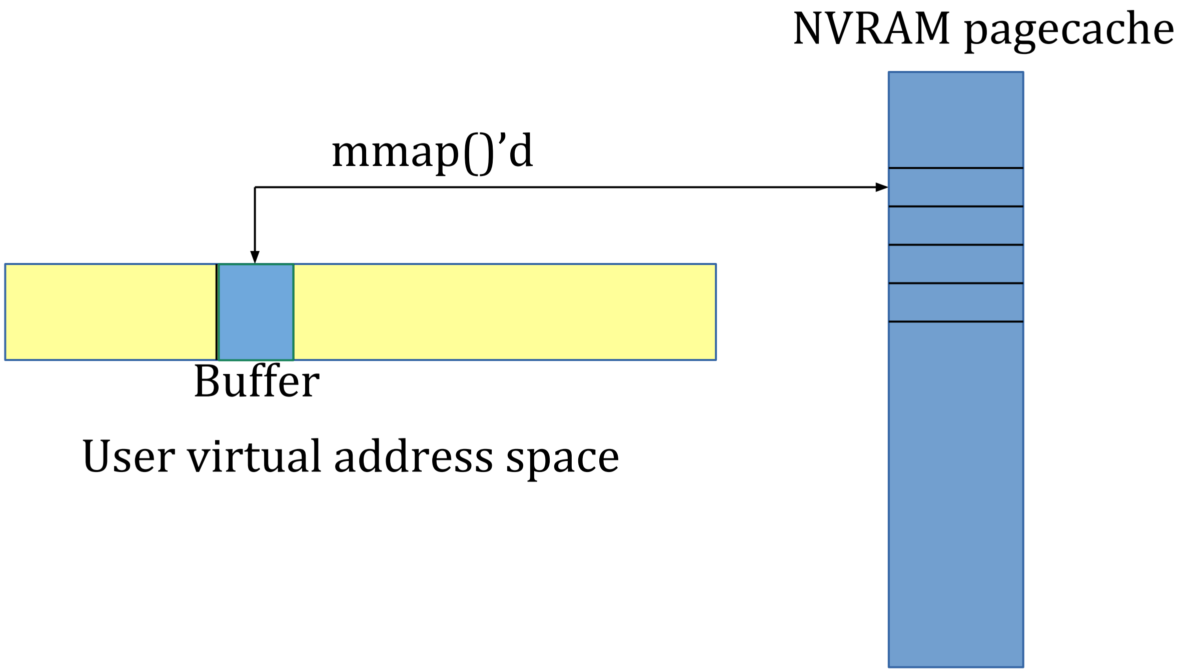
NVRAM as the storage device



DAX uses NVRAM for pagecache



DAX can map pagecache NVRAM into user buffer

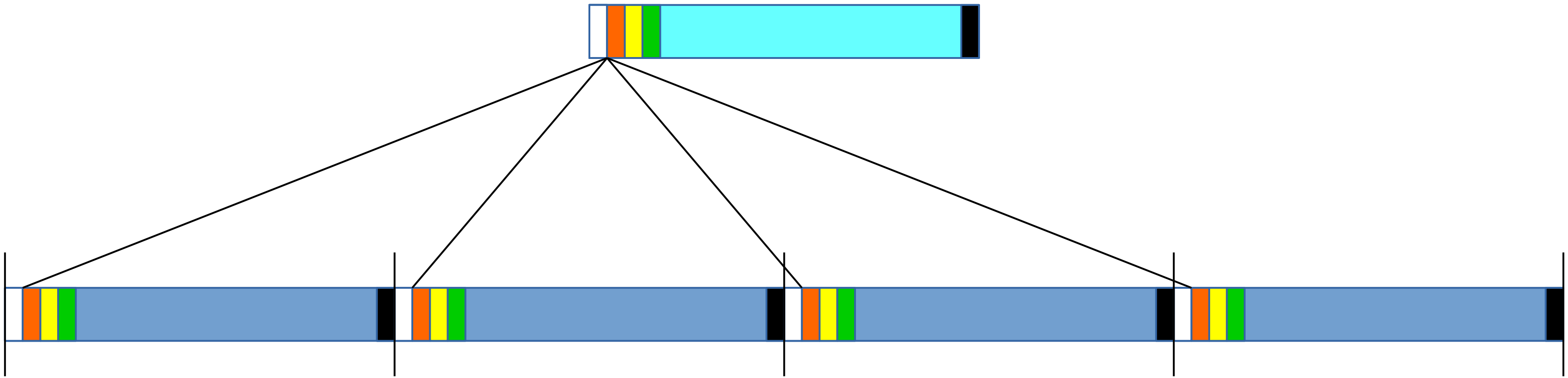


NVDIMM Memory mode

- NVRAM is the system RAM
- DRAM is used as a cache for the NVRAM.
  - A direct-mapped physical cache scheme is used in memory mode.
  - A page coloring algorithm must be used to optimize the NVRAM cache.
  - Memory references run at DRAM speed when working set is in DRAM cache
- NVRAM is typically 4 to 16 times the size of the DRAM.
- The DRAM speed and latency is orders of magnitude faster than the NVRAM.
- Expect a memory bandwidth slowdown when DRAM is too small.

Memory mode NVDIMM support

DRAM cache



Banks of NVDIMM Memory

Part II Meet the Experts - room 150

- Network Performance
  - Low Latency
    - Nohz_full
    - Cpu_partitioning
    - XDP + eBPF Denial of Service
- Perf Tools
  - perf c-2-c
  - tuna
- CVE impacts / tunable

RHEL-7 nohz_full option

Patchset Goal:

Stop interrupting userspace tasks
Move timekeeping to non-latency-sensitive cores

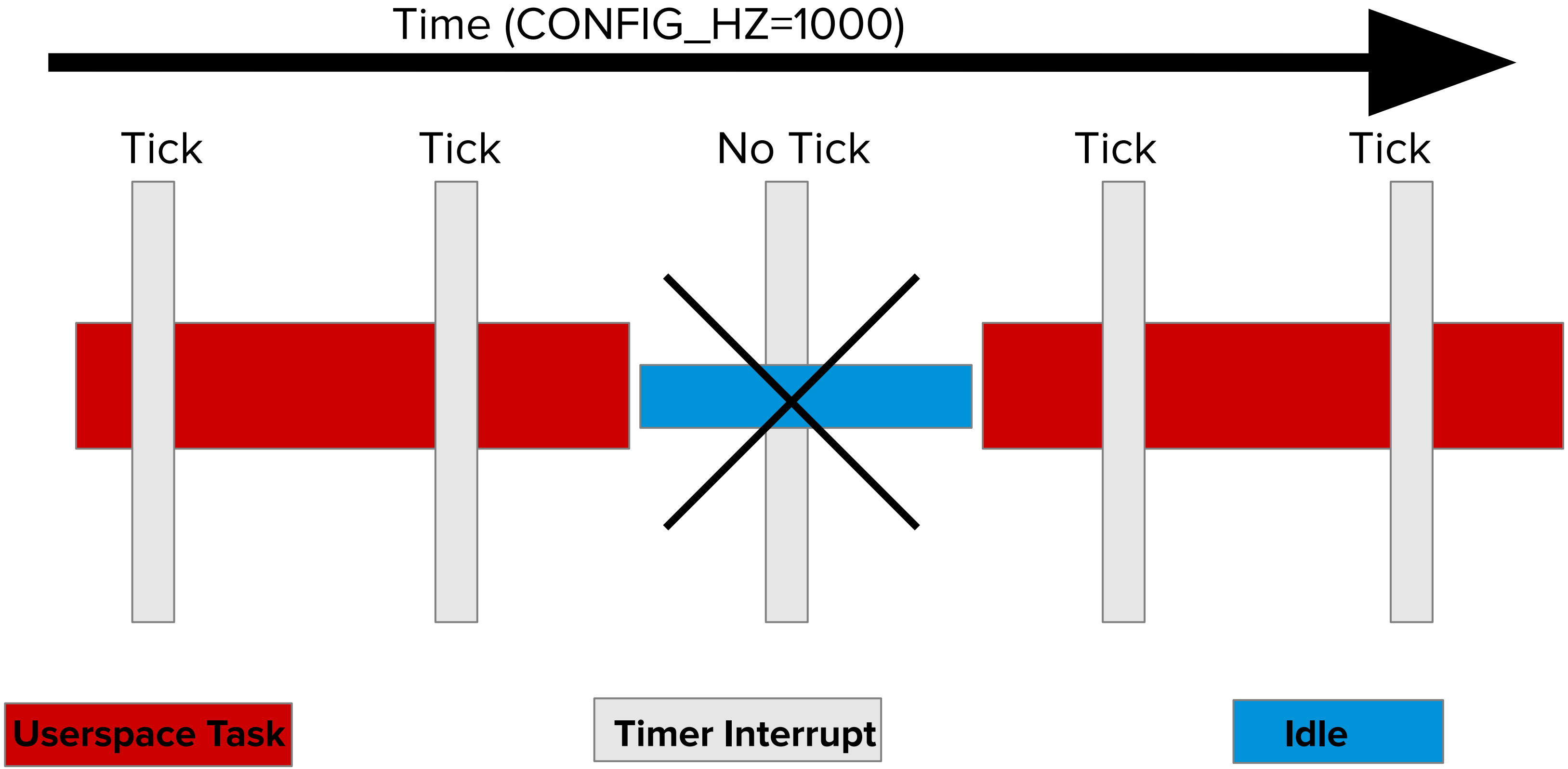
If nr_running=1, then scheduler/tick can avoid that core
Default disabled
Opt-in via nohz_full cmdline option

Kernel Ticks for:

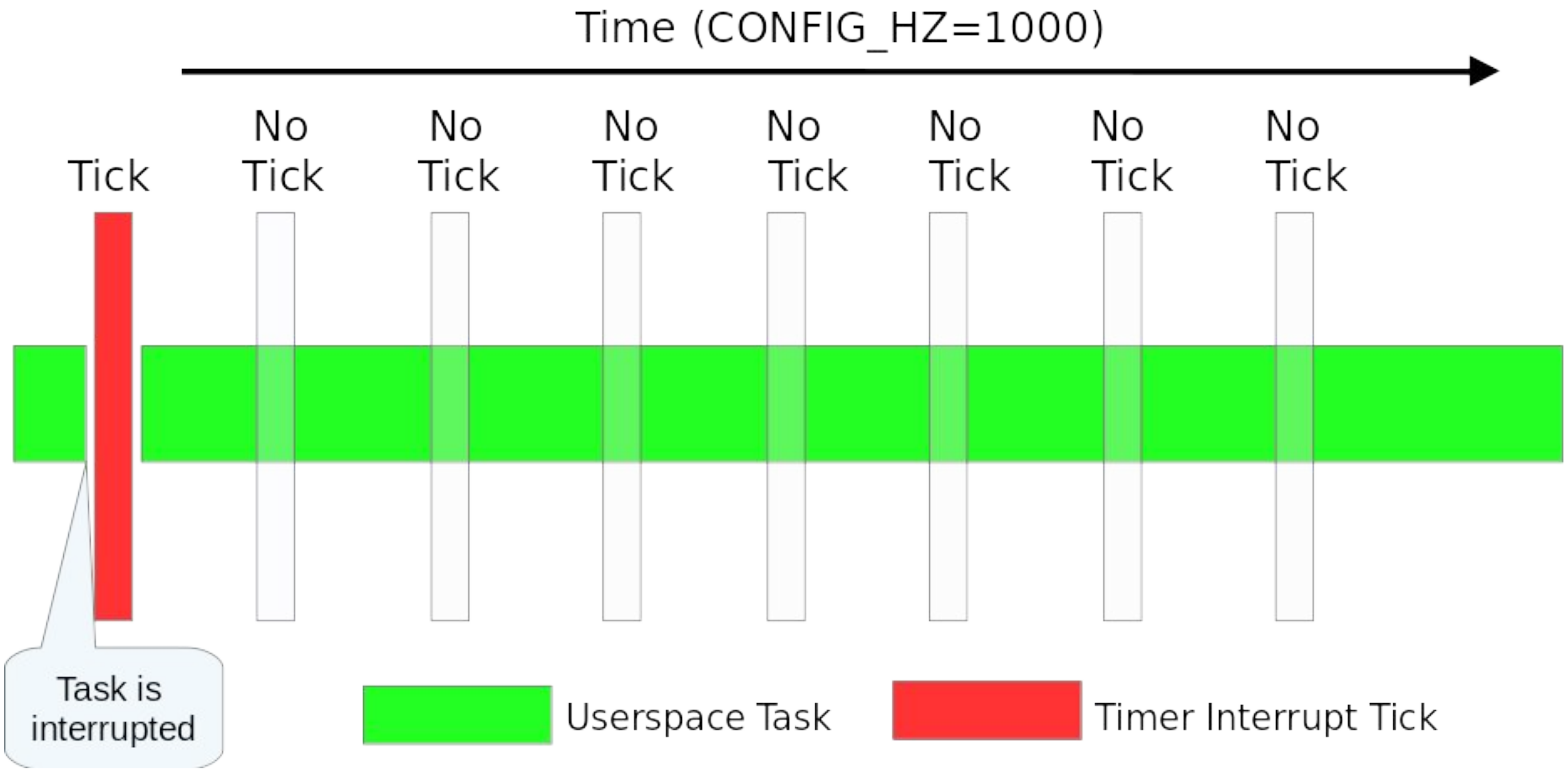
timekeeping (gettimeofday)
Scheduler load balancing
Memory statistics (vmstat)

RHEL6 Tickless

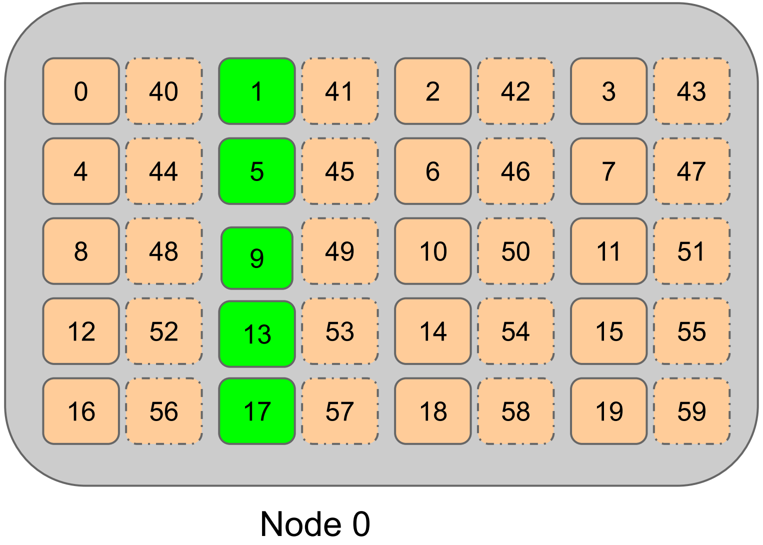
User tasks interupted 1000x/sec



RHEL 7 nohz_full



Isolcpus – the widely used "isolation" hammer.



Boot with "isolcpus=1,5,9,13,17"
Pin your application's individual threads to the isolated cores.
Life is good.

Isolcpus – no scheduler load balancing

Boot your system with “isolcpus=1,4,5,9,13,17”

Then run your multithreaded application:

taskset -c 1,4,5,9,13,17 my_low_latency_app

Result:

If you pin each thread to a cpu:

life is good.

Else

the entire application runs only on cpu 1.

“cpu-partitioning” tuned profile

For latency sensitive applications needing kernel scheduler load balancing.

Does all the “heavy lifting” for you.

1) Just edit *<u>/etc/tuned/cpu-partitioning-variables.conf</u>*

```
# Isolated CPUs with kernel load balancing:
isolated_cores=10-39
# Isolated CPUs without kernel load balancing:
no_balance_cores=2-9
```

1) Set the cpu-partitioning tuned profile.

 # tuned-adm profile cpu-partitioning

1) Then reboot!

Larry Woodman
Joe Mario

Cpu-partitioning – after reboot you have:

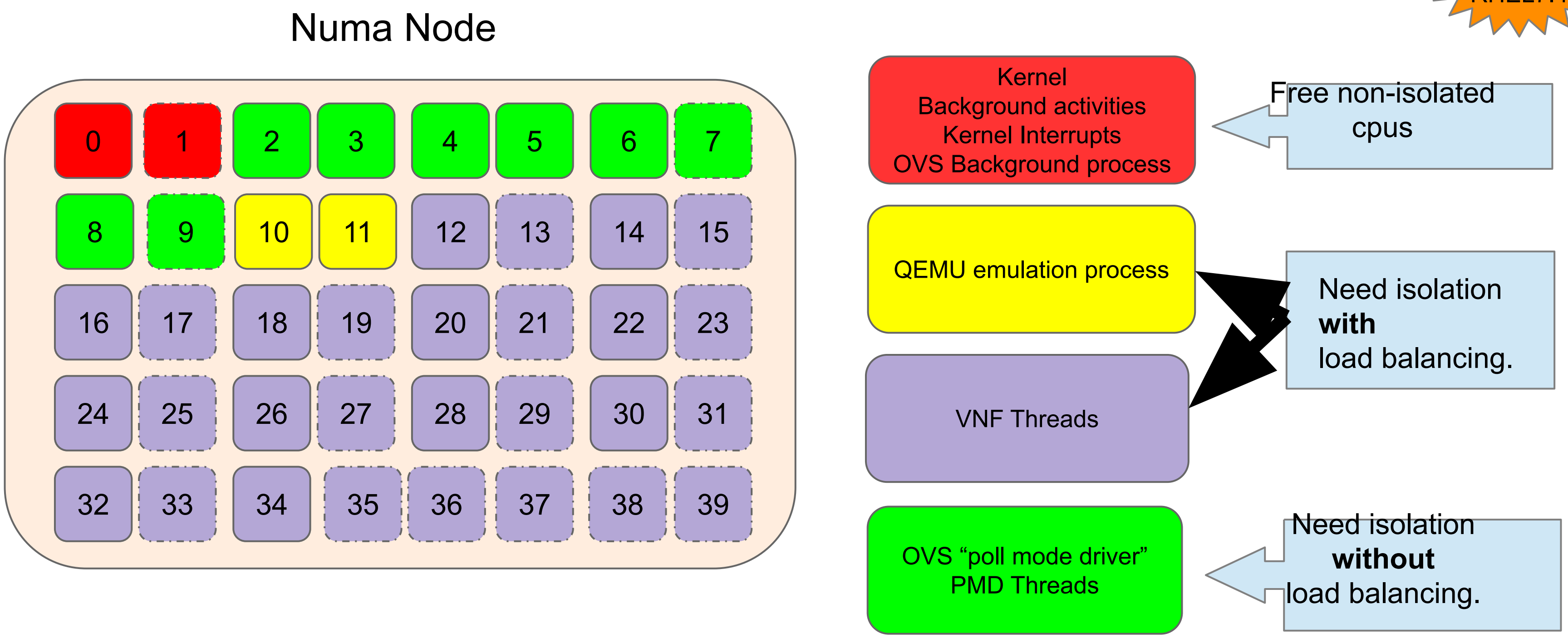
- Adds the following to the kernel boot line:

skew_tick=1
nohz=on
nohz_full=2-39
rcu_nocbs=2-39
tuned.non_isolcpus=0000000003
intel_pstate=disable
Nosoftlockup

- Moves all users tasks off the isolated cpus
  - Including all children of systemd (pid 1)
  - All future processes too, as default system cpu affinity is changed.

CPU Partitioning tuned profile
Simple, flexible low-latency cpu isolation tuning.

Available as of RHEL7.4



Cpu-partitioning – after reboot (continued):

- kernel.hung_task_timeout_secs = 600
- kernel.nmi_watchdog = 0
- vm.stat_interval = 10
- kernel.timer_migration = 1
- net.core.busy_read = 50
- net.core.busy_poll = 50
- kernel.numa_balancing = 0
- kernel.sched_min_granularity_ns = 10000000
- vm.dirty_ratio = 10
- vm.dirty_background_ratio = 3
- vm.swappiness = 10
- kernel.sched_migration_cost_ns = 5000000
- Disables Transparent Hugepages

RHEL8 eBPF Tech preview Denial Of Service (DoS)

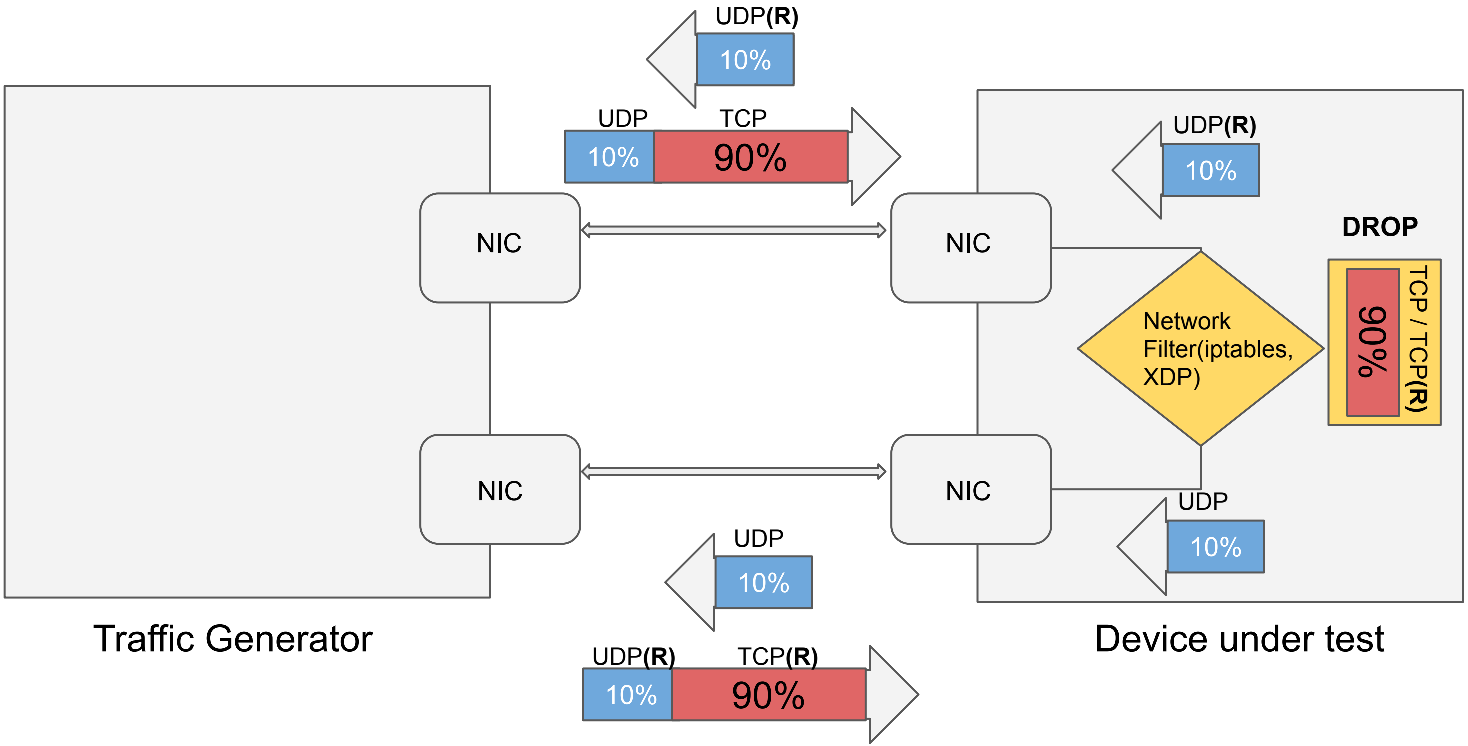
- The traffic flow is unidirectional from both interfaces.
- The packets are routed between the two DUT interfaces using kernel routing table and forwarded to the other traffic generator port respectively.
- A binary search is done to find the max packet rate till the test passes.
- The test is passed when:
  - **No TCP packet is received on both interfaces**
  - **0.002% of UDP packets drop threshold is maintained.**

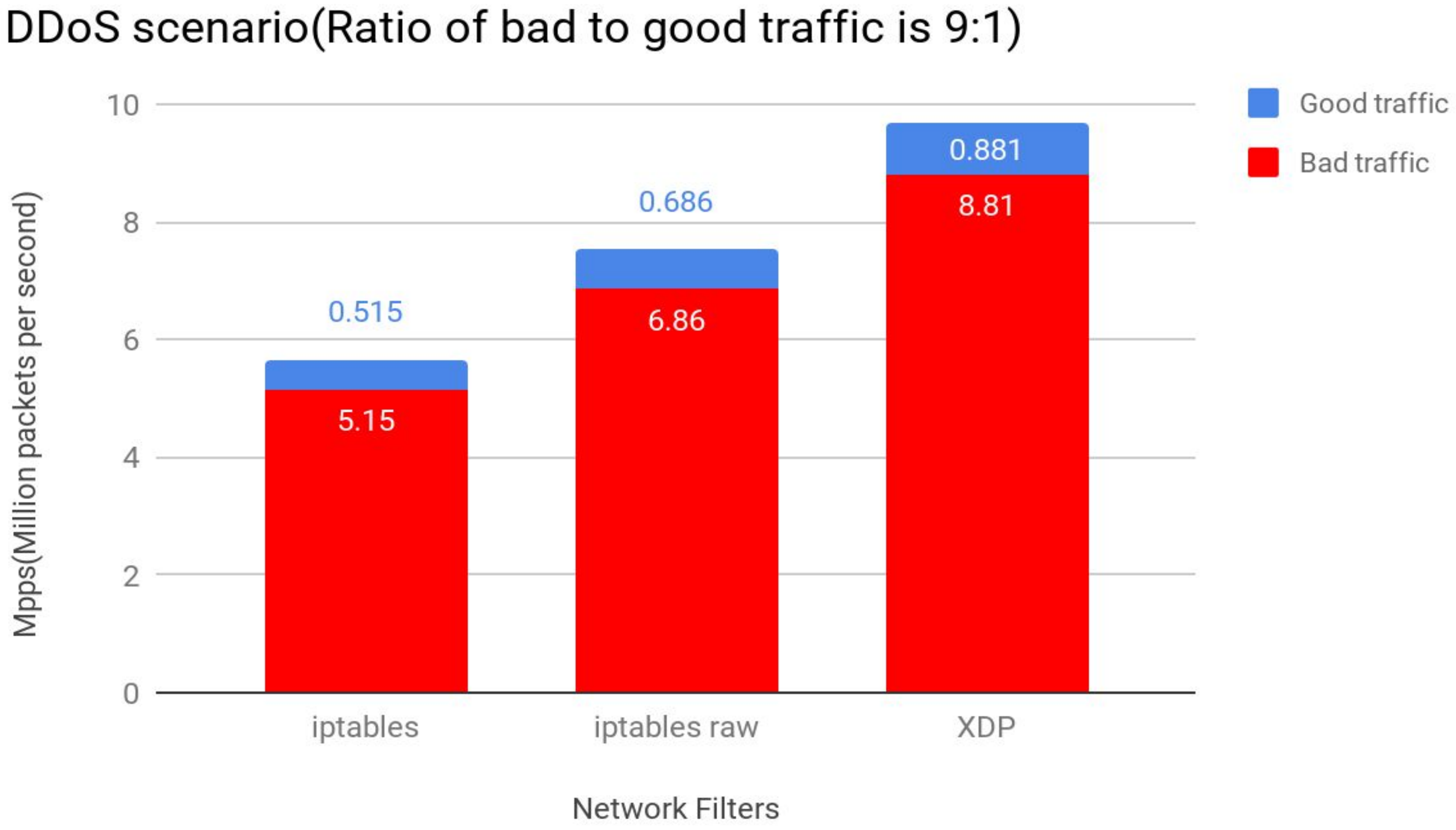
- Iptables filter and drops TCP port 80 packets:
  - Rules are added once in **filter** table and then in **raw** table for performance comparison
- For XDP, we are using xdp_ddos_blacklist[1] program which is loaded on both DUT interfaces and drops packets arriving on TCP port 80.

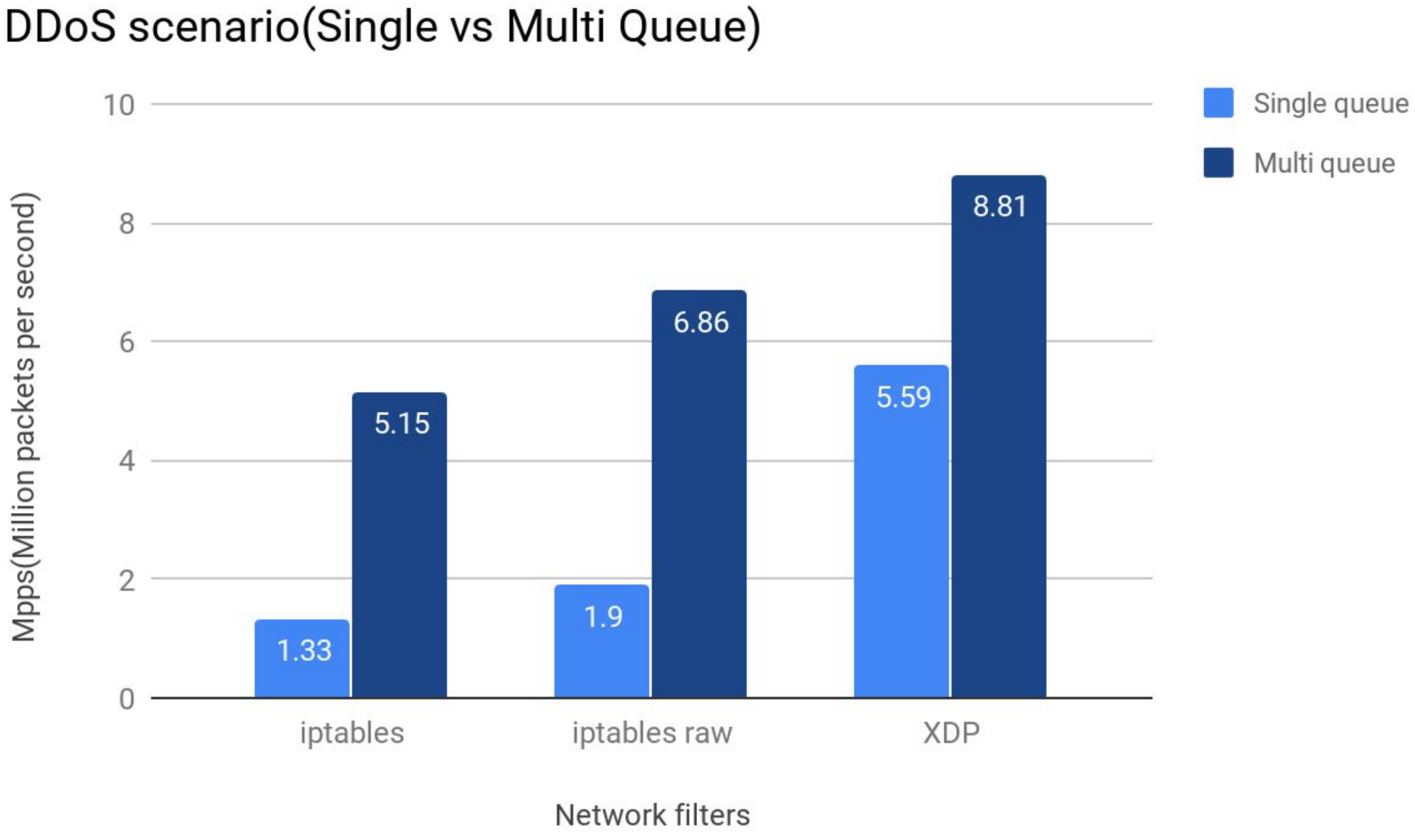
[1]: https://github.com/netoptimizer/prototype-kernel/blob/master/kernel/samples/bpf/xdp_ddos01_blacklist_kern.c

Test setup

*** (R): Reverse Direction**




DDoS scenario(Ratio of bad to good traffic is 9:1)
Mpps(Million packets per second)
10
8
6
4
2
0
Good traffic
Bad traffic
0.515
5.15
0.686
6.86
0.881
8.81
iptables
iptables raw
XDP
Network Filters



DDoS scenario(Single vs Multi Queue)
Single queue
Multi queue
Mpps(Million packets per second)
10
8
6
4
2
0
1.33
5.15
1.9
6.86
5.59
8.81
iptables
iptables raw
XDP
Network filters

perf c2c for cpu cacheline false sharing detection

Works on R8

Critical for:

- Shared memory applications
- Multi-threaded apps spanning multiple numa nodes

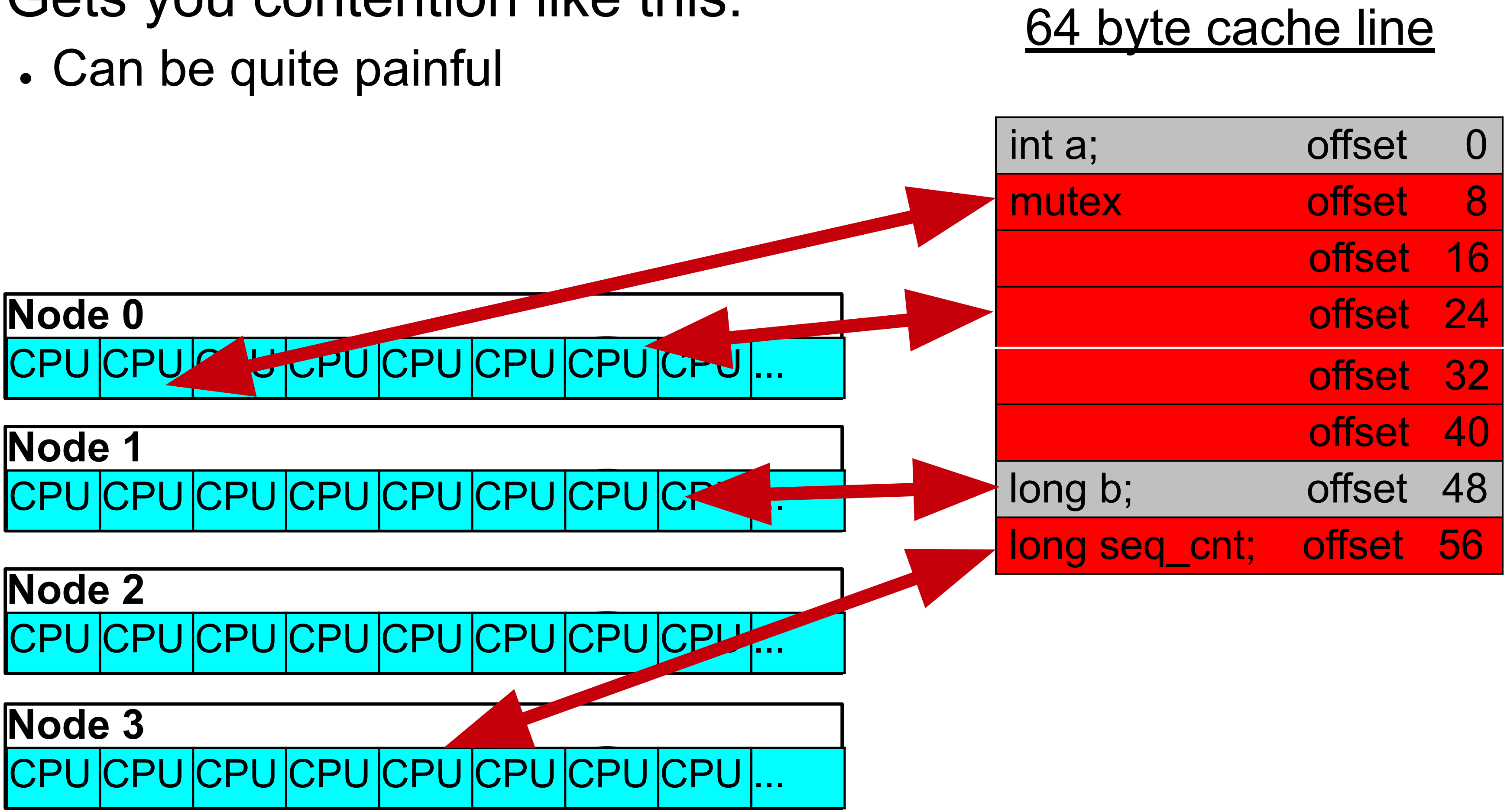
Shows everything needed to find false sharing:

- All readers and writers contending for hottest cachelines.
- The cpus and nodes they executed on.
- Process names, data addr, ip, pids, tids, src file and line number.
- Where hot variables are sharing cachelines, (like locks).
- Where hot structs are spanning cachelines, (like an unaligned mutex).

Detailed blog: https://joemario.github.io/blog/2016/09/01/c2c-blog/

Gets you contention like this:

- Can be quite painful



Where are my processes and threads running?

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
```

”*Last Ran CPU*” column

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

Are my threads and data aligned on same numa node?

Use perf (soon to report node & phys addr info where data resides)

perf mem record -- --sample-cpu foo_exe

perf mem report -F mem,cpu,dcacheline,snoop,symbol -s dcacheline --stdio

Tuna: **command line or gui**

Fine grained process view & control

- Adjust scheduler tunables, (sched policy, RT priority and CPU affinity)
- See results instantly
- Tune threads and IRQ handlers.
- Isolate CPU cores and sockets,

Examples:

Move an irq to cpu 5

 # tuna -c5 -q eth4-rx-4 –move

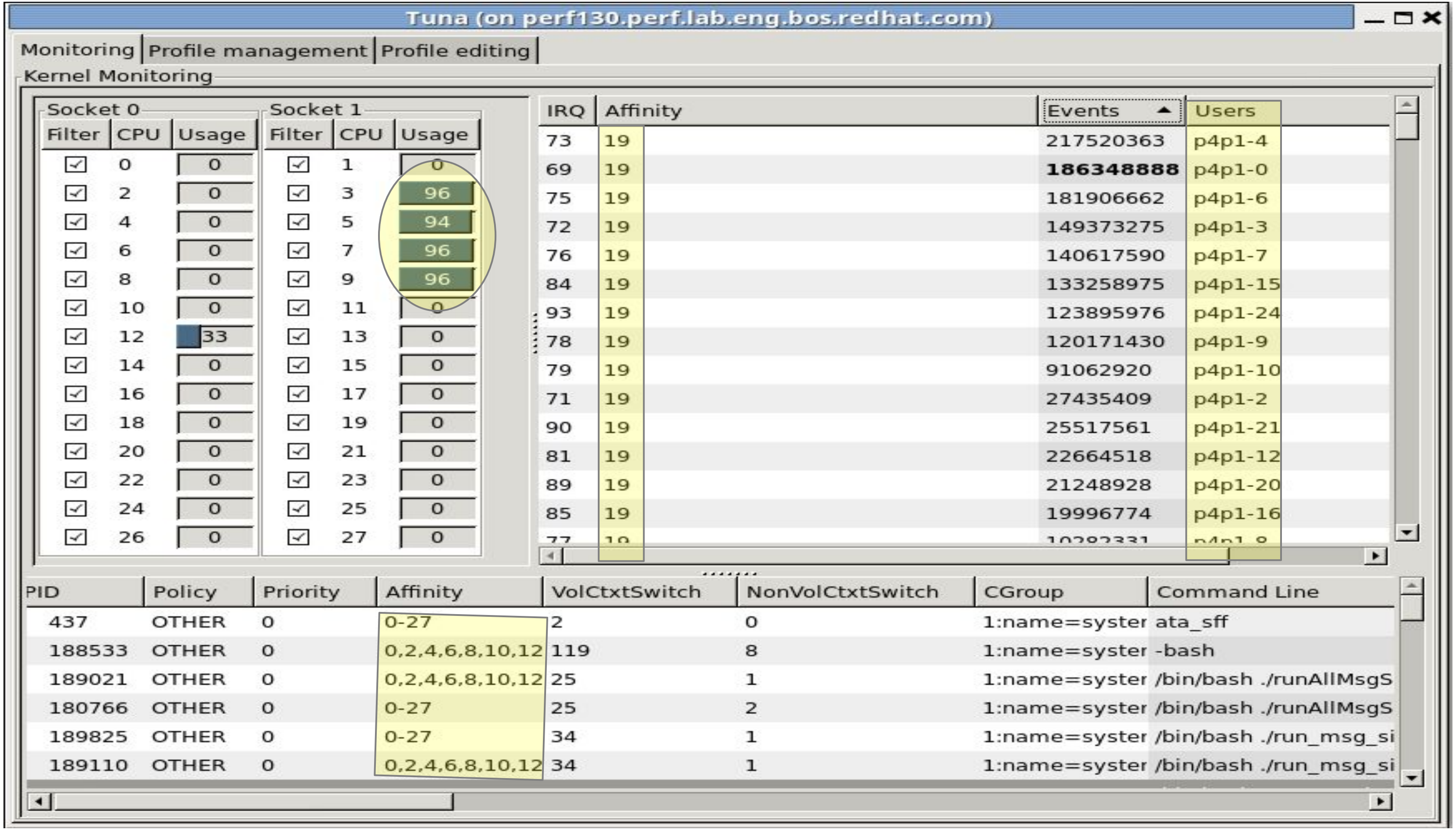
Move all irqs named “eth4*” away from numa node 1

 # tuna -S 1 -i -q ‘eth4’*

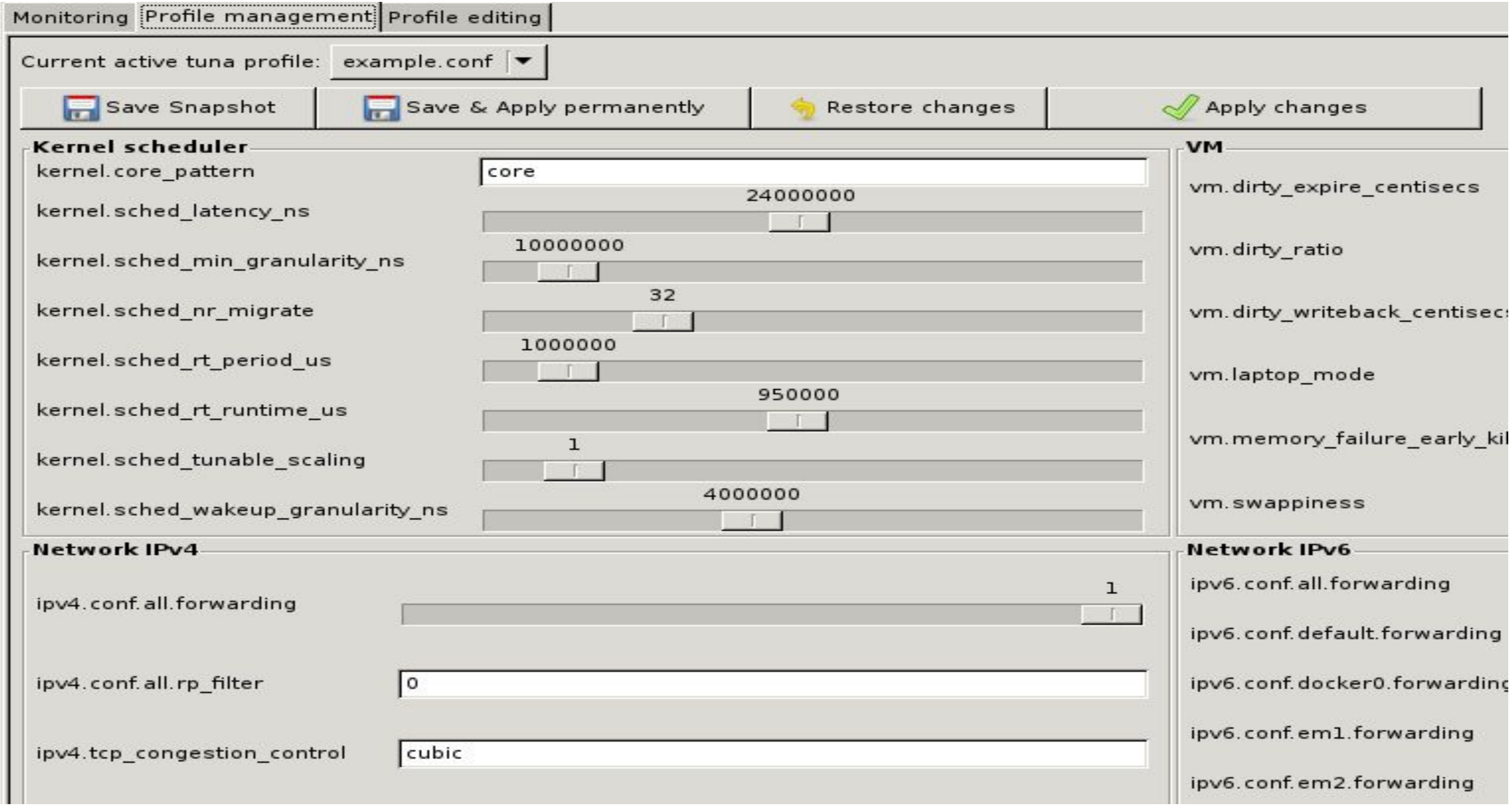
Move all rcu kernel threads to cpus 1 and 3

*# tuna -c1,3 -t ‘*rcu*’ --move*

Tuna example



Tuna GUI Capabilities Updated for RHEL7



CVE Performance overrides

To disable CVE on RHEL-{6,7,8}, add the following to the boot grub line

spectre_v2=off spec_store_bypass_disable=off nopti l1tf=off

Your resulting vulnerabilities files should then look something like these:

```
# grep . /sys/devices/system/cpu/vulnerabilities/*
```

/sys/devices/system/cpu/vulnerabilities/l**1tf:**Mitigation: PTE Inversion; **VMX: vulnerable**

/sys/devices/system/cpu/vulnerabilities/**meltdown:Vulnerable**

/sys/devices/system/cpu/vulnerabilities/**spec_store_bypass**:**Vulnerable**

/sys/devices/system/cpu/vulnerabilities/spectre_v1:Mitigation: __user pointer sanitization

/sys/devices/system/cpu/vulnerabilities/**spectre_v2:Vulnerable, IBPB: disabled, STIBP: disabled**

CVE Performance Defaults w/ SkyLake

```
# grep . /sys/devices/system/cpu/vulnerabilities/*
/sys/devices/system/cpu/vulnerabilities/l1tf:Mitigation: PTE
Inversion; VMX: conditional cache flushes, SMT vulnerable

/sys/devices/system/cpu/vulnerabilities/meltdown:Mitigation: PTI

/sys/devices/system/cpu/vulnerabilities/spec_store_bypass:Mitigation:
Speculative Store Bypass disabled via prctl and seccomp

/sys/devices/system/cpu/vulnerabilities/spectre_v1:Mitigation: __user
pointer sanitization

/sys/devices/system/cpu/vulnerabilities/spectre_v2:Mitigation: Full
generic retpoline, IBPB: conditional, IBRS_FW, STIBP: conditional,
RSB filling
```

RED HAT SUMMIT

THANK YOU

plus.google.com/+RedHat

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/redhat