Performance Analysis and Tuning – Part 1

D. John Shakshober (Shak) - Sr Consulting Eng / Director Performance Engineering
Larry Woodman - Senior Consulting Engineer / Kernel VM
Jeremy Eder - Principal Performance Engineering
Bill Gray - Principal Performance Engineer
Agenda: Performance Analysis Tuning Part I

- **Part I**
  - RHEL Evolution 5->6->7 – out-of-the-box tuned for Clouds - “tuned”
  - NonUniform Memory Access (NUMA)
  - Process Scheduler, Numa awareness, tunables
  - Transparent Hugepages, Static Hugepages 4K/2MB/1GB
  - Cgroups – the basis of Linux Containers / Atomic

- **Part II**
  - RHEL Atomic / Host, Tuning Optimized for Enterprise
  - Network Performance and Latency-performance
  - Disk and Filesystem IO - Throughput-performance
  - System Performance/Tools – perf, tuna, systemtap, performance-co-pilot

- **Performance Birds of the Feather (BoF) Wed 6-8 Room 206**
## Red Hat Enterprise Linux Performance Evolution

<table>
<thead>
<tr>
<th>RHEL5</th>
<th>RHEL6</th>
<th>RHEL7</th>
<th>RH Cloud Suites</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ktune – on/off</td>
<td>CPU Affinity (ts/numactl)</td>
<td>CPU Affinity (ts/numactl)</td>
<td>RHEL OSP – blueprints Tuned, Numa pinning</td>
</tr>
<tr>
<td>CPU Affinity</td>
<td>NUMAD – uerspace tool</td>
<td>Autonuma-Balance</td>
<td>NIC - jumbo SR-IOV</td>
</tr>
<tr>
<td>(taskset)</td>
<td></td>
<td></td>
<td>RHEL Atomic Host/Atomic Enterprise</td>
</tr>
<tr>
<td>NUMA Pinning</td>
<td></td>
<td></td>
<td>RH OpenShift v3</td>
</tr>
<tr>
<td>(numactl)</td>
<td></td>
<td></td>
<td>Cloud Forms</td>
</tr>
<tr>
<td>Irqbalance</td>
<td>Cgroups - irqbalance – NUMA enhanced</td>
<td>_irqbalance – NUMA enhanced</td>
<td></td>
</tr>
</tbody>
</table>

#redhat #rhsummit
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 – llnpack, 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
- 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)
## Red Hat / Intel Haswell EX Top Benchmark Results


<table>
<thead>
<tr>
<th>Significance</th>
<th>OEM Platform</th>
<th>Benchmark</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-socket world record</td>
<td>Dell PowerEdge R930 + HP ProLiant DL580 G9</td>
<td>SAP Sales &amp; Distribution (2-Tier) on Linux</td>
<td>RHEL 7.1</td>
</tr>
<tr>
<td>(Linux)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall world record</td>
<td>HP ProLiant DL580 G9</td>
<td>SPECfp_base2006</td>
<td>RHEL 7.1</td>
</tr>
<tr>
<td>2-socket world record</td>
<td>Fujitsu PRIMEQUEST 2800E2</td>
<td>SPECint_rate_base2006</td>
<td>RHEL 7.1</td>
</tr>
<tr>
<td>8-socket world record</td>
<td>Fujitsu PRIMEQUEST 2800E2</td>
<td>SPECint_rate_base2006</td>
<td>RHEL 6.6</td>
</tr>
<tr>
<td>(x86)</td>
<td>Huawei FusionServer RH8100 V3</td>
<td>SPECfp_rate_base2006</td>
<td>RHEL 7</td>
</tr>
<tr>
<td>8-socket world record</td>
<td>Huawei FusionServer RH8100 V3</td>
<td>SPECfp_rate_base2006</td>
<td>RHEL 7</td>
</tr>
<tr>
<td>4-socket record</td>
<td>Lenovo System x3850 X6</td>
<td>SPECvirt_sc2013</td>
<td>RHEL 6.6</td>
</tr>
</tbody>
</table>
RHEL / Intel Benchmark Haswell EX

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

Benchmark Publications Using Red Hat Enterprise Linux Over Past 24 Months

Industry Benchmarks February 2013 - February 2015
(As of March 2nd, 2015)

Percent Using Red Hat Enterprise Linux

SPEC CPU2006: 83%
SPECint_sc2013: 67%

World Record SAP SD 2-Tier Results

Highest two and four socket Linux results
(As of May 5, 2015)

Users

RHEL 7 (Intel E5 v3): 16,500
RHEL 7.1 (Intel E7 v3): 31,000
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 for RHEL Platforms
Overview and What's New
Tuned: Updates for RHEL7

• Re-written for maintainability and extensibility.
  – Configuration is now consolidated into a single tuned.conf file
  – Optional hook/callout capability
  – Adds concept of Inheritance (just like httpd.conf)

  – Profiles updated for RHEL7 features and characteristics
  – Added bash-completion :-)

#redhat #rhsummit
Tuned: Your Custom Profiles

Parents
- throughput-performance
- balanced
- latency-performance

Children
- network-throughput
- desktop
- network-latency
- virtual-host
- virtual-guest

Children/Grandchildren
- Your Web Profile
- Your Database Profile
- Your Middleware Profile
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 7.1 File System In Cache Performance
Intel I/O (iozone - geoM 1m-4g, 4k-1m)

- ext3
- ext4
- xfs
- gfs2

Throughput in MB/Sec

- not tuned
- tuned
RHEL 6/7 Non-Uniform Memory (NUMA)
Two-Socket SPEC CPU 2006 Rate History

- 2P int rate up 30x / 10 yrs
- Results track # Cores
- Scaling helped by NUMA topology which enables amazing RAM bandwidth (4 channels, 68 GB/s)
- 9 / 10 of recent leading x86 results achieved on Red Hat Enterprise Linux
Typical Four-Node NUMA System

Node 0
- Core 0
- Core 2
- Core 4
- Core 6
- Core 8
- Core...
- L3 Cache
- QPI links, IO, etc.

Node 1
- Core 0
- Core 2
- Core 4
- Core 6
- Core 8
- Core...
- L3 Cache
- QPI links, IO, etc.

Node 2
- Core 0
- Core 2
- Core 4
- Core 6
- Core 8
- Core...
- L3 Cache
- QPI links, IO, etc.

Node 3
- Core 0
- Core 2
- Core 4
- Core 6
- Core 8
- Core...
- L3 Cache
- QPI links, IO, etc.
What is NUMA: Non-Uniform Memory Access?

• Making bigger systems more scalable by distributing system memory near individual CPUs....
• Access to local memory is fast, more latency for remote memory
• Practically all multi-socket systems have NUMA
  • Most servers have 1 NUMA node / socket
  • Some AMD systems have 2 NUMA nodes / socket
• Sometimes optimal performance still requires manual tuning.
• Red Hat has been increasingly automating NUMA management!
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

# 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
Visualize CPUs via lstopo (hwloc-gui rpm)

# lstopo

NUMA
CACHE
HT

PCIe
Tips for Good NUMA Performance

• Never disable NUMA in the BIOS. Keep interleave memory OFF (which should be the system BIOS default)
  • Else OS will see only 1-NUMA node!!
• Understand basic operation and implications of NUMA
  • (e.g. per-node resources)
• Know your workload resource consumption attributes and access patterns. If possible, size parallel jobs to fit in NUMA nodes.
• Be aware of your system hardware NUMA topology.
• Use appropriate tuning if necessary for good performance.
Per NUMA-Node Resources

- CPUs, Caches, Memory
- Interrupt processing, IO / DMA capacity
- Memory zones (DMA & Normal zones)
- Page reclamation kernel thread (kswapd#)
- Lots of other kernel threads
- May need to check resource status per node (e.g. numastat -cm)
  - Because some resources are per node, you can have a node-local resource shortage even though overall system resources look OK!
NUMA Nodes and Zones

Node 0

- 16MB DMA Zone
- 4GB DMA32 Zone
- Normal Zone
- End of RAM

Node 1

- 64-bit
Per Node / Zone split LRU Paging Dynamics

- User Allocations
  - Reactivate
  - Page aging
  - Reclaiming
  - User deletions
  - swapout
  - flush
Interaction between VM Tunables and NUMA

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

- 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 wakes up at Low
  - Wakeup kswapd and it reclaims memory
- Pages Min – all memory allocators reclaim at Min
  - user processes/kswapd reclaim memory

0
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
- When set and node memory is exhausted:
  - Reclaim memory from local node rather than allocating from next node
  - Slower initial allocation, higher NUMA hit ratio
- When clear and node memory is exhausted:
  - Allocate from all nodes before reclaiming memory
  - Faster initial allocation, higher NUMA miss ratio
zone_reclaim_mode (continued)

• To see current setting: cat /proc/sys/vm/zone_reclaim_mode
• Turn ON: echo 1 > /proc/sys/vm/zone_reclaim_mode
• Turn OFF: echo 0 > /proc/sys/vm/zone_reclaim_mode
• 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
• This setting can make a big difference in NUMA performance!
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!!)
zone_reclaim_mode (continued)

• Is NUMA data locality more or less important than cache?
• For file servers or workloads that benefit from having their data cached, zone_reclaim_mode should be left disabled as the caching effect is likely to be more important than data locality.
• zone_reclaim may be enabled if it's known that the workload is partitioned such that each partition fits within a NUMA node and that accessing remote memory would cause a measurable performance reduction.
• Need to know workload resource attributes...
Know Your Workload and Resource Attributes

• Dedicated system or Server consolidation / replication
  • Large monolithic process (e.g. large in-memory database)
    • Workload consumes most of the system resources
    • Resource access patterns are global and unpredictable
  • Multiple processes using mostly local data (e.g. virtual guests)
    • Multiple workloads / threads consuming fractional subsets of system resources
    • Resource access patterns can be private, localized or contained
    • Ideally, these workloads / threads can be sized to fit within NUMA nodes!
• Leave zone_reclaim_mode OFF (and consider interleaved memory policy) for global, unpredictable accesses.
• Align CPUs, Memory, and Devices for workloads that can be localized to minimize latency, and isolated to avoid interference!
## NUMA Management Checklist

<table>
<thead>
<tr>
<th>Checklist</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐ Research NUMA Topology</td>
<td>✓ lscpu, lstopo, numactl --hardware</td>
</tr>
<tr>
<td>☐ Consider I/O devices and IRQ</td>
<td>✓ irqbalance, PCI Bus</td>
</tr>
<tr>
<td>☐ Virtualization?</td>
<td>✓ libvirt numatune</td>
</tr>
<tr>
<td>☐ Plan Resource Allocations</td>
<td>✓ NUMA nodes per workload</td>
</tr>
<tr>
<td>☐ Group Tasks and Resources</td>
<td>✓ numactl, cgroups</td>
</tr>
<tr>
<td>☐ Monitor NUMA memory</td>
<td>✓ numad, kernel NUMA balancing</td>
</tr>
<tr>
<td>☐ Monitor NUMA CPUs</td>
<td>✓ numastat -cm &lt;workload&gt;</td>
</tr>
<tr>
<td></td>
<td>✓ top (then press '2', or maybe '3')</td>
</tr>
</tbody>
</table>
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 <NODES> -N <NODES>  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
cpbind: 6 7
nodebind: 6 7
membind: 6 7

# numactl  --interleave=all  <Workload>
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
cpbind: 4 5 6 7
nodebind: 4 5 6 7
membind: 0 1 2 3 4 5 6 7
Numastat

• Enhanced by Red Hat (since Red Hat Enterprise Linux 6.4) with helpful and informative new memory display features.

• Numastat shows per-NUMA-node memory statistics for processes and the operating system.

• By default, numastat displays per-node kernel memory allocator hit and miss statistics.

• Any command line arguments to numastat will invoke enhanced behavior to show per-node distribution of memory.
numastat shows need for NUMA management

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

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

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

<table>
<thead>
<tr>
<th>PID</th>
<th>Node 0</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>10587 (qemu-kvm)</td>
<td>0</td>
<td>10723</td>
<td>5</td>
<td>0</td>
<td>10728</td>
</tr>
<tr>
<td>10629 (qemu-kvm)</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>10717</td>
<td>10722</td>
</tr>
<tr>
<td>10671 (qemu-kvm)</td>
<td>0</td>
<td>0</td>
<td>10726</td>
<td>0</td>
<td>10726</td>
</tr>
<tr>
<td>10713 (qemu-kvm)</td>
<td>10733</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>10738</td>
</tr>
<tr>
<td>Total</td>
<td>10733</td>
<td>10723</td>
<td>10740</td>
<td>10717</td>
<td>42913</td>
</tr>
</tbody>
</table>
```
Cgroup cpusets

• Another way of manually grouping and aligning a set of tasks, CPUs and associated memory
• Uses normal cgroup hierarchy of resource partitioning
• memory_migrate will cause memory to move
• Must enter TIDs separately

# mount -t cgroup -o cpuset cpuset /sys/fs/cgroup/cpuset
# cd /sys/fs/cgroup/cpuset
# mkdir my_cpuset
# cd my_cpuset
# echo 30-39 > cpuset.cpus
# echo 3 > cpuset.mems
# echo 1 > cpuset.memory_migrate
# echo <tid 1> tasks
# echo <tid 2> tasks
# echo <tid 3> tasks
# echo <tid 4> tasks
Correct NUMA bindings

```
# echo 0 > cpuset.mems
# echo 0-3 > cpuset.cpus
# numastat

numa_hit      1648772
numa_miss     23459
local_node   1648648
other_node   23583

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

Incorrect NUMA bindings

```
# echo 1 > cpuset.mems
# echo 0-3 > cpuset.cpus
# numastat

numa_hit      1623318
numa_miss     23459
local_node   1623194
other_node   23583

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

```bash
# numastat

numa_hit      2700423
numa_miss     23459
local_node   2700299
other_node   23583
```
KSM: Kernel Samepage Merging

• Ksmd allows oversubscription of resources by sharing memory pages between guest instances. Can save significant memory when running many similar guests.

• However, this is a performance talk....
  • Don't oversubscribe your resources if you want best performance!

• If you must, and multiple guests can fit in a single NUMA node, you might get some memory savings – without giving up NUMA isolation – by turning off ksm merging across NUMA nodes
  • # echo 0 > /sys/kernel/mm/merge_across_nodes
  • Turn ksmmd off: # echo 'KSM_ENABLED=0' > /etc/default/qemu-kvm
Numad and kernel NUMA balance align process memory and CPU threads within NUMA nodes

No NUMA management

With NUMA management
Automate NUMA with “numad” in RHEL6.4+

• An optional user-level CPU-and-memory-affinity management daemon to automatically improve NUMA performance

• Allocates CPU and NUMA memory resources to localize and isolate significant processes (e.g. KVM guests)

• Dynamically adjusts placement as loads change

• Maintains specified target utilization of node resources
  • Adjust default 85% with “-u <n>” to change node resource margin

• Pre-placement feature can be used by libvirt placement='auto'
  • <vcpu placement='auto'>2</vcpu>
  • <numatune> <memory mode='strict' placement='auto'/> </numatune>
Automate with RHEL7+ kernel NUMA balancing

• Periodically unmaps pages to see where memory is used
  • This sampling approach efficiently ignores unused memory.
• Moves task threads to NUMA nodes with their memory and moves accessed memory to NUMA nodes where threads are executing
• Lazy page migration efficiently moves memory in the background
• Much better than numad at fine grain (thread-level) control
• Enabled and active by default in Red Hat Enterprise Linux 7+
• Turn off: echo 0 > /proc/sys/kernel/numa_balancing
• Other tunables in /proc/sys/kernel/numa*, e.g. can adjust numa_balancing_scan_delay_ms to ignore short-lived processes. Normally don't need to change these.
RHEL 6.6 vs RHEL 7.1 SAP HANA Performance
25% gain due to Auto NUMA balancing (kernel.numa_balancing = 1)

benchBWEMLSim - MultiProvider QueryRuntime
(LOWER==BETTER)
NUMA Alignment Makes JBB 2005 2x Faster

SPEC JBB 2005 NUMA Comparison (16 Socket, 240 Cores, HP BL920 Gen8)

- No NUMA
- Autonuma (Default)
- Numad
- Autonuma + Numad
- Numad -u100 -H100
- Autonuma + Numad -u100 -H100
- Numactl
- No NUMA stddev
- Numactl stddev

Average Throughput per Instance (1/Node)

Standard Deviation across Instances
Summary - Red Hat Enterprise Linux Automates NUMA Management!

• With Red Hat Enterprise Linux 6.4+, careful use of numad can significantly improve performance and automate NUMA management on systems with server consolidation or replicated parallel workloads.

• With Red Hat Enterprise Linux 7+, most users will get good NUMA system memory management for most applications out of the box!

• Automated NUMA management is especially valuable in dynamic server environments with changing workload conditions.
Red Hat Enterprise Linux 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`
  – 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
Red Hat Enterprise Linux Page Sizes
RHEL Hugepages/ VM Tuning

- Standard HugePages 2MB
  - Reserve/free via
    - `/proc/sys/vm/nr_hugepages`
    - `/sys/devices/node/*/hugepages/*/nrhugupages`
  - 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
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
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

```bash
# 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
```
Transparent Hugepages

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

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

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

- Boot argument: transparent_hugepages=always (enabled by default)

```bash
# echo always > /sys/kernel/mm/redhat_transparent_hugepage/enabled
```

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

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

**SPEEDUP 12.4/7.0 = 1.77x, 56%**
Red Hat Enterprise Linux Cgroups
Resource Management using cgroups

Ability to manage large system resources effectively

• Control Group (Cgroups) for CPU/Memory/Network/Disk
• Benefit: guarantee Quality of Service & dynamic resource allocation
• Ideal for managing any multi-application environment
  • From back-ups to the Cloud
Cgroup default mount points

RHEL6

```bash
# 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

```bash
# ls -l /sys/fs/cgroup/
```

```
drwxr-xr-x. 2 root root 0 Mar 20 16:40 blkio
draxr-xr-x 2 root root 0 Mar 20 16:40 cpu,
draxr-xr-x 2 root root 0 Mar 20 16:40 cpuacct
draxr-xr-x 2 root root 0 Mar 20 16:40 cpuset
draxr-xr-x 2 root root 0 Mar 20 16:40 devices
draxr-xr-x 2 root root 0 Mar 20 16:40 freezer
draxr-xr-x 2 root root 0 Mar 20 16:40 hugetlb
draxr-xr-x. 2 root root 0 Mar 20 16:40 memory
draxr-xr-x. 2 root root 0 Mar 20 16:40 net_cls
draxr-xr-x. 2 root root 0 Mar 20 16:40 perf_event
draxr-xr-x. 4 root root 0 Mar 20 16:40 systemd
```
Cgroup how-to

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

```bash
# 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

<table>
<thead>
<tr>
<th></th>
<th>node0</th>
<th>node1</th>
</tr>
</thead>
<tbody>
<tr>
<td>numa_hit</td>
<td>1648772</td>
<td>438778</td>
</tr>
<tr>
<td>numa_miss</td>
<td>23459</td>
<td>2134520</td>
</tr>
<tr>
<td>local_node</td>
<td>1648648</td>
<td>423162</td>
</tr>
<tr>
<td>other_node</td>
<td>23583</td>
<td>2150136</td>
</tr>
</tbody>
</table>

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

# numastat

<table>
<thead>
<tr>
<th></th>
<th>node0</th>
<th>node1</th>
</tr>
</thead>
<tbody>
<tr>
<td>numa_hit</td>
<td>2700423</td>
<td>439550</td>
</tr>
<tr>
<td>numa_miss</td>
<td>23459</td>
<td>2134520</td>
</tr>
<tr>
<td>local_node</td>
<td>2700299</td>
<td>423934</td>
</tr>
<tr>
<td>other_node</td>
<td>23583</td>
<td>2150136</td>
</tr>
</tbody>
</table>

Incorrect NUMA bindings

# echo 1 > cpuset.mems
# echo 0-3 > cpuset.cpus
# numastat

<table>
<thead>
<tr>
<th></th>
<th>node0</th>
<th>node1</th>
</tr>
</thead>
<tbody>
<tr>
<td>numa_hit</td>
<td>1623318</td>
<td>434106</td>
</tr>
<tr>
<td>numa_miss</td>
<td>23459</td>
<td>1082458</td>
</tr>
<tr>
<td>local_node</td>
<td>1623194</td>
<td>418490</td>
</tr>
<tr>
<td>other_node</td>
<td>23583</td>
<td>1098074</td>
</tr>
</tbody>
</table>

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

# numastat

<table>
<thead>
<tr>
<th></th>
<th>node0</th>
<th>node1</th>
</tr>
</thead>
<tbody>
<tr>
<td>numa_hit</td>
<td>1623341</td>
<td>434147</td>
</tr>
<tr>
<td>numa_miss</td>
<td>23459</td>
<td>2133738</td>
</tr>
<tr>
<td>local_node</td>
<td>1623217</td>
<td>418531</td>
</tr>
<tr>
<td>other_node</td>
<td>23583</td>
<td>2149354</td>
</tr>
</tbody>
</table>
### cpus.Shares default

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

### cpus.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

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

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

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>20102</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>4160</td>
<td>360</td>
<td>284</td>
<td>R</td>
<td>100.0</td>
<td>0.0</td>
<td>0:17.45 useless</td>
</tr>
<tr>
<td>20103</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>4160</td>
<td>356</td>
<td>284</td>
<td>R</td>
<td>100.0</td>
<td>0.0</td>
<td>0:17.03 useless</td>
</tr>
<tr>
<td>20107</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>4160</td>
<td>360</td>
<td>284</td>
<td>R</td>
<td>100.0</td>
<td>0.0</td>
<td>0:15.57 useless</td>
</tr>
<tr>
<td>20104</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>4160</td>
<td>360</td>
<td>284</td>
<td>R</td>
<td>99.8</td>
<td>0.0</td>
<td>0:16.66 useless</td>
</tr>
<tr>
<td>20105</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>4160</td>
<td>360</td>
<td>284</td>
<td>R</td>
<td>99.8</td>
<td>0.0</td>
<td>0:16.66 useless</td>
</tr>
<tr>
<td>20108</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>4160</td>
<td>360</td>
<td>284</td>
<td>R</td>
<td>99.8</td>
<td>0.0</td>
<td>0:15.19 useless</td>
</tr>
<tr>
<td>20110</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>4160</td>
<td>360</td>
<td>284</td>
<td>R</td>
<td>99.4</td>
<td>0.0</td>
<td>0:14.74 useless</td>
</tr>
<tr>
<td>20106</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>4160</td>
<td>360</td>
<td>284</td>
<td>R</td>
<td>99.1</td>
<td>0.0</td>
<td>0:15.87 useless</td>
</tr>
<tr>
<td>20111</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>4160</td>
<td>356</td>
<td>284</td>
<td>R</td>
<td>1.0</td>
<td>0.0</td>
<td>0:00.08 useful</td>
</tr>
</tbody>
</table>
C-group Dynamic resource control

![Dynamic CPU Change](image)

**Dynamic CPU Change**

**Oracle OLTP Workload**

Control Group CPU Count

<table>
<thead>
<tr>
<th>Transactions Per Minute</th>
</tr>
</thead>
<tbody>
<tr>
<td>200K</td>
</tr>
<tr>
<td>180K</td>
</tr>
<tr>
<td>160K</td>
</tr>
<tr>
<td>140K</td>
</tr>
<tr>
<td>120K</td>
</tr>
<tr>
<td>100K</td>
</tr>
<tr>
<td>80K</td>
</tr>
<tr>
<td>60K</td>
</tr>
<tr>
<td>40K</td>
</tr>
<tr>
<td>20K</td>
</tr>
</tbody>
</table>

- `cgrp 1 (4), cgrp 2 (32)`
- `cgrp 1 (32), cgrp 2 (4)`

**Instance 1**

**Instance 2**
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

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>20614</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>4160</td>
<td>360</td>
<td>284</td>
<td>R</td>
<td>100.0</td>
<td>0.0</td>
<td>0:30.77 use</td>
<td>useful</td>
</tr>
</tbody>
</table>

# 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

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>20645</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>4160</td>
<td>360</td>
<td>284</td>
<td>R</td>
<td>1.0</td>
<td>0.0</td>
<td>0:01.54 use</td>
<td>useful</td>
</tr>
</tbody>
</table>
Cgroup OOMKills

```bash
# 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 32 196312 32 196372 912 974 1 4 88 7 0
    0   1 406228 586572 0 157696 0 157704 624 696 0 1 87 11 0
    0   1 568532 585928 0 162304 0 162312 722 1039 0 2 87 11 0
    0   1 729300 584744 0 160768 0 160776 719 1161 0 2 87 11 0
    0   0  86648 1607092 0 3676784  0  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
```
RHEL7 Performance Tuning Summary

• Use “Tuned”, “NumaCTL”, “NumaD” in RHEL6 and RHEL7
  • Transparent Hugepages for anon memory (monitor it)
  • Scheduler – Auto-Numa-Balance – Multi-instance, consider “NumaD”
  • Scheduler – tuned profiles, load balance
  • Cgroup infrastructure for RHEL6, Atomic/ Docker for RHEL7

• Manually Tune
  • NUMA – via numactl, monitor numastat -c pid
  • Huge Pages – static hugepages for pinned shared-memory
  • Managing VM, dirty ratio and swappiness tuning
  • Use cgroups for further resource management control
## Performance Utility Summary

<table>
<thead>
<tr>
<th>Supportability</th>
<th>NUMA</th>
<th>Networking</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>redhat-support-tool</td>
<td>hwloc</td>
<td>dropwatch</td>
<td>blktrace</td>
</tr>
<tr>
<td>sos</td>
<td>Intel PCM</td>
<td>ethtool</td>
<td>iotop</td>
</tr>
<tr>
<td>kdump</td>
<td>numactl</td>
<td>netsniff-ng (EPEL6)</td>
<td>iostat</td>
</tr>
<tr>
<td>perf</td>
<td>numad</td>
<td>tcpdump</td>
<td></td>
</tr>
<tr>
<td>psmisc</td>
<td>numatop (01.org)</td>
<td>wireshark/tshark</td>
<td></td>
</tr>
<tr>
<td>strace</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sysstat</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>systemtap</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trace-cmd</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Util-linux-ng</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pcp</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Power/Tuning**

- cpupowerutils (R6)
- kernel-tools (R7)
- powertop
- tuna
- tuned
LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.
Performance Analysis and Tuning – Part 2

D. John Shakshober (Shak) - Sr Consulting Eng / Director Performance Engineering
Larry Woodman - Senior Consulting Engineer / Kernel VM
Jeremy Eder - Principal Performance Engineering
Bill Gray - Principal Performance Engineer
Agenda: Performance Analysis Tuning Part II

• Part I
  - RHEL Evolution 5->6->7 – out-of-the-box tuned for Clouds - “tuned”
  - NonUniform Memory Access (NUMA)
  - Cgroups – the basis of Linux Containers / Atomic
  - Process Scheduler, Numa awareness, tunables
  - Transparent Hugepages, Static Hugepages 4K/2MB/1GB

• Part II
  - RHEL Atomic / Platform, Tuning Optimized for Enterprise
  - Network Performance and Latency-performance, Real Time
  - Disk and Filesystem IO - Throughput-performance
  - Cloud Performance Topics OpenShift, OpenStack - NFV

• Performance Birds of the Feather (BoF) Wed 6-8 Room 206
RED HAT ENTERPRISE LINUX
ATOMIC HOST

MINIMAL, SECURE FOOTPRINT
- Minimal host provides “just enough” to support apps.

RAPID PROVISIONING
- Apps can be provisioned and started in milliseconds.

SIMPLIFIED MAINTENANCE
- Atomic updates are quick, reliable, and can be rolled back.

Atomic Tuned Profile Inheritance

Parents
- throughput-performance

Children
- atomic-host
- virtual-guest

Children/Grandchildren
- atomic-guest
Network Latency and Throughput

netperf Latency and Throughput
Higher is Better

%diff vs Bare Metal

TCP_RR | UDP_RR | TCP_STREAM | UDP_STREAM

Bare Metal | Container | KVM

#redhat #rhsummit
Large OLTP Database, BM vs Container vs KVM

Large OLTP Database (3 instances of 100 Users)
Higher is Better

- Bare Metal
- Containers
- KVM

Red Hat Company Confidential, NDA Required
Container performance across multiple workloads

Time to Complete Test Workload
Higher is Better

<table>
<thead>
<tr>
<th>Workload</th>
<th>RHEL7 bare metal</th>
<th>RHEL7 container</th>
<th>RHEL7 KVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculate primes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OLTP workload</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analytics App.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Network Performance
NFV
Realtime
Visualize CPUs via lstopo (hwloc-gui rpm)

# lstopo

NUMA

CACHE

HT

PCIe
Realtime, Realtime KVM/NFV Tuned Profiles

Parents
- latency-performance

Children
- network-latency

Children/Grandchildren
- realtime
- realtime-virtual-host
- realtime-virtual-guest
Networking Latency Performance – System setup

- Evaluate the 2 new tuned profiles for networking
- Disable unnecessary services, runlevel 3
  - Follow vendor guidelines for BIOS Tuning
  - Logical cores? Power Management? Turbo?
- In the OS, consider
  - Disabling filesystem journal
  - SSD/Memory Storage
  - Reducing writeback thresholds if your app does disk I/O
  - NIC Offloads favor throughput
RHEL 7.x Network Performance
Intel Haswell EP, 12-40Gb ports (6 cards)
40G Network Data/Tuned Networks

Network Throughput, Gbps

RHEL7.1, 12 x 40Gb NiCs
TCP_STREAM, 48 x 16KB, Bi-directional

Baseline

Tuned (NFV)

421 Gbps
Network Function Virtualization (NFV) Throughput and Packets/sec (RHEL7.x+DPDK)

NFV: Millions of Packets Per Second

RHEL7.x, L2 Forwarding, 12 x 40Gb NICs

<table>
<thead>
<tr>
<th></th>
<th>Packets Per Second ( Millions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KVM</td>
<td>208</td>
</tr>
<tr>
<td>Docker</td>
<td>215</td>
</tr>
<tr>
<td>Bare-metal</td>
<td>218</td>
</tr>
<tr>
<td>HW Maximum</td>
<td>225</td>
</tr>
</tbody>
</table>
421 Gbps
Kernel

208Mpps+
INTO KVM
DPDK

wow
such bandwidth
much packets

go RHEL
Scheduler Latency (cyclictest)

Cyclictest Latency

- Min
- Mean
- 99.9%
- Stddev
- Max

Remove maxes to zoom in
Realtime Scheduler Latency Jitter Plot

cyclictest -m -n -N -q -v -p95 -h60 -i 200 -D 1h

RHEL 7.2 (259)  
RHEL-RT 7.2 (237)  
RHEL-RT 7.2 (237+KVM)  

Latency (microseconds)  

0 1e+06 2e+06 3e+06 4e+06 5e+06 6e+06 7e+06 8e+06 9e+06 1e+07  
cycle  

10 Million Samples
OpenShift JMeter

Density impact on HTTP Response Time

RHEL7.1, docker-1.6.2, JMeter, 10Gb

# simultaneous active containers

Response Time (milliseconds)

0 10 20 30 40 50

4 10 22 41
RHEL7.1 + Solarflare OpenOnload Bare Metal / KVM / Containers

- Lower is better
- Alternative kernel-bypass mechanism to DPDK
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)
RHEL7 BUSY_POLL Socket Options

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

![Graph showing netperf TCP_RR and UDP_RR Transactions/sec comparison between Baseline and SO_BUSY_POLL options.](image-url)
### Tuned-Adm Profile: Throughput-Performance

```bash
# tuned-adm profile throughput-performance
# turbostat sleep 5
```

<table>
<thead>
<tr>
<th>Bzy_MHz</th>
<th>TSC_MHz</th>
<th>SMI</th>
<th>CPU%c1</th>
<th>CPU%c3</th>
<th>CPU%c6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1866</td>
<td>2600</td>
<td>0</td>
<td>0.22</td>
<td>0.01</td>
<td>99.71</td>
</tr>
</tbody>
</table>

### Tuned-Adm Profile: Network-Latency

```bash
# tuned-adm profile network-latency
# turbostat sleep 5
```

<table>
<thead>
<tr>
<th>Bzy_MHz</th>
<th>TSC_MHz</th>
<th>SMI</th>
<th>CPU%c1</th>
<th>CPU%c3</th>
<th>CPU%c6</th>
</tr>
</thead>
<tbody>
<tr>
<td>3108</td>
<td>2600</td>
<td>0</td>
<td>99.99</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Disk I/O in RHEL
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

Intel I/O (iozone - geoM 1m-4g, 4k-1m)

RHEL 7 3.10.0-253 File System Out of Cache Performance

Intel I/O (iozone - geoM 1m-4g, 4k-1m)
SAS Application on Standalone Systems

Picking a RHEL File System

RHEL 7 limits

**xfs**  most recommended
- Max file system size 500TB
- Max file size 100 TB
- Best performing

**ext4** recommended
- Max file system size 50 TB
- Max file size 16 TB

**ext3** not recommended
- Max file system size 16TB
- Max file size 2TB

SAS Mixed Analytics (RHEL6 vs RHEL7)
perf 32 (2 socket Nehalem) 8 x 48GB

<table>
<thead>
<tr>
<th>File system tested</th>
<th>TOTAL Time</th>
<th>System Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>xfs-rhel6</td>
<td>6.18</td>
<td>-4.05</td>
</tr>
<tr>
<td>ext3-rhel6</td>
<td>4.94</td>
<td>-4.05</td>
</tr>
<tr>
<td>ext4-rhel6</td>
<td>9.59</td>
<td>-4.05</td>
</tr>
</tbody>
</table>

#redhat #rhsummit
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

Read()/Write() → memory copy

Buffer → Page cache → Flush daemon

User space → Kernel → File system
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
RHEL Performance in Cloud
RHEL Platforms (RHEV/ OpenStack)

- **RHEV**
  - KVM Performance
  - SPECvirt leadership
    - Vcpu ping
    - NUMA binding
    - Hugepages 2MB + 1GB
  - Low Latency SR-IOV
  - High Throughput via
    - DPDK

- **RH OpenStack Platform**
  - Control Plane
    - Keystone
    - Database / HA
    - Messaging / HA
    - Ceilometer
  - Data Plane
    - Nova
    - Neutron / HA
    - Cinder/ Swift

- **OpenShift**
  - V3 LXC / containers

- **RH Storage**
  - Gluster file w/ RHEV /OSP
  - Ceph Block w Cinder
    - Block (librados)
    - w/ SSD block storage
SPECvirt_sc® 2013: RHEL 6/7 and KVM Post Industry Leading Results

http://www.spec.org/virt_sc2013/results/
RHEL OSP 5/6 – Architecture - https://access.redhat.com/articles/1321163
Performance Tuning RHEL OSP5/ R7

General tuning:

- Disable BIOS power
  - switch to OS controlled/prtg

- Tuned-adm
  - Default for OSP 4/5 has been “throughput-performance”
  - NFV - latency sensitive loads should consider
    - Alter cstate to contant 1 through the latency profiles
  - R6 - profile latency-performance
  - R7 - profile network-latency
Neutron Performance / Network Performance / VxLAN OVS

Guest to Guest: Guest MTU 8950; Tunnel MTU 9000

Throughput vs Message Size

- Single Stream, 8950 MTU
- 4x Concurrent Streams @ 8950 MTU
- Single Stream, 1450 MTU

#redhat #rhsummit
NUMA Pinning and Topology Awareness – RHEL OSP 6

Per NUMA Node
- Huge pages

NUMA Node 0

Socket

CPU core 0

CPU core 1

CPU core 2

CPU core 3

Virtual Machine

vCPU

vCPU

vCPU

vCPU
RHEL 7.x w/ OSP / Network - DPDK OVS

- **Configurations: BM / Atomic/KVM**
  - kernel
  - dpdk-lib
  - container: dpdk-app
  - vfio
  - hardware
    - Intel XL710

- **Boot options**
  - CPU cstate=1
  - Reserve 1GB hugepages
  - isolate CPUs

- **Current software versions tested:**
  - dpdk-2.0.0-6
  - pktgen-dpdk-2.8.4
  - openvswitch-2.3.90-10031.gitf097013a.3
RHEL7.x Network Function Virtualization (NFV)
Throughput and Packets/sec @ 64Bytes (RHEL7.x+DPDK)

- **Bare-Metal**
  - DPDK application runs right on bare-metal, forwarding packets between port pair in all XL710 adapters
  - Two cores per adapter used
  - 218 Million packets per second processed

- **Atomic Containers**
  - VFIO module loaded in host (bare-metal)
  - Docker container launched
  - Docker container includes only software to run DPDK app
  - Docker container given access to /dev/vfio devices
  - Two cores per adapter used, twelve cores total
  - 215 Million packets per second processed

- **KVM**
  - Six VMs created, 1 per network adapter
  - Each VM configured for optimal performance
    - NUMA-node local I/O, CPU, and memory
    - VM memory backed with 1GB memory pages
  - Each VM assigned a network adapter
    - (2 physical functions)
  - Under 5% performance impact using KVM
  - 208 Million packets per second processed

<table>
<thead>
<tr>
<th></th>
<th>Haswell EP,, 48p Intel 6-40Gb</th>
<th>1-port Mpps</th>
<th>Total Mpps</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHEL 7.beta</td>
<td>18.1</td>
<td></td>
<td>218</td>
</tr>
<tr>
<td>RHEL w/ Atomic 12 containers-PT</td>
<td>17.9</td>
<td></td>
<td>215</td>
</tr>
<tr>
<td>RHEL w/ KVM 6 Vms - PT</td>
<td>17.3</td>
<td></td>
<td>208</td>
</tr>
</tbody>
</table>
RHEL RHS Tuning w/ RHEV/RHEL OSP (tuned)

- Ceph and Gluster – (visit Summit talks)
- XFS mkfs -n size=8192, mount inode64, noatime
- RHS server: tuned-adm profile rhs-virtualization
  - Increase in readahead, lower dirty ratio's
- KVM host: tuned-adm profile virtual-host
  - Better response time shrink guest block device queue
    - /sys/block/vda/queue/nr_request (16 or 8)
  - Best sequential read throughput, raise VM read-ahead
    - /sys/block/vda/queue/read_ahead_kb (4096/8192)
Ceph Block Perf – scale 4 to 8 ceph servers
https://access.redhat.com/articles/1321163
SELinux

- Run representative set of benchmarks
- A:B comparison (SELinux enabled/disabled)
- Provide % difference between runs

**Overhead 0-5%.

Tests run:
- LINPACK, STREAMS, SPECJBB2005
- SPECJBB2015, IOZONE, Phoronix Apache Bench
- Oracle OLTP, Oracle OLTP+NFS, DB2
RHEL7 Performance Tuning Summary

• Use “Tuned”, “NumaD” and “Tuna” in RHEL6 and RHEL7
  - Power savings mode (performance), locked (latency)
  - Transparent Hugepages for anon memory (monitor it)
  - numabalance – Multi-instance, consider “NumaD”
  - Virtualization – virtio drivers, consider SR-IOV

• Manually Tune
  - NUMA – via numactl, monitor numastat -c pid
  - Huge Pages – static hugepages for pinned shared-memory
  - Managing VM, dirty ratio and swappiness tuning
  - Use cgroups for further resource management control
Upcoming Performance Talks

1) Wednesday, Jun 24, 4:50 PM: Performance tuning Red Hat Enterprise Linux Platform for databases, Ballroom A

2) Wednesday, Jun 24, 6:00 PM: Performance analysis & tuning: An interactive mixer, Room 202

3) Thursday, Jun 25, 10:40 AM: Red Hat Storage performance, Room 310

4) Thursday, Jun 25, 4:50 PM: Performance of OpenStack Cinder on Ceph, Room 310

5) Thursday, Jun 25, 4:50 PM: Cloud Architecture and Performance, Room 313

6) Friday, Jun 26, 9:45 AM: Containers versus virtualization, 302
## Performance Utility Summary

<table>
<thead>
<tr>
<th><strong>Supportability</strong></th>
<th><strong>NUMA</strong></th>
<th><strong>Networking</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• redhat-support-tool</td>
<td>• hwloc</td>
<td>• ss</td>
</tr>
<tr>
<td>• sos</td>
<td>• Intel PCM</td>
<td>• dropwatch</td>
</tr>
<tr>
<td>• kdump</td>
<td>• numactl</td>
<td>• ethtool</td>
</tr>
<tr>
<td>• perf</td>
<td>• numad</td>
<td>• netsniff-ng (ifpps)</td>
</tr>
<tr>
<td>• strace</td>
<td>• numatop (01.org)</td>
<td>• tcpdump</td>
</tr>
<tr>
<td>• sysstat</td>
<td><strong>Power/Tuning</strong></td>
<td>• wireshark/tshark</td>
</tr>
<tr>
<td>• systemtap</td>
<td>• kernel-tools</td>
<td><strong>Storage</strong></td>
</tr>
<tr>
<td>• trace-cmd</td>
<td>• powertop/cpupower</td>
<td>• blktrace</td>
</tr>
<tr>
<td>• util-linux-ng</td>
<td>• tuna</td>
<td>• iotop</td>
</tr>
<tr>
<td>• pcp</td>
<td>• tuned</td>
<td>• iostat</td>
</tr>
</tbody>
</table>
Helpful Links

• Official Red Hat Documentation
• Red Hat Low Latency Performance Tuning Guide
• Low Latency Performance Tuning for RHEL7
• How do I create my own tuned profile on RHEL7?
• RHEL7: Optimizing Memory System Performance
False cacheline sharing.
Add writer (in red) – slows down all readers.

Frequent writes. Causes cacheline thrashing, impacting performance.
LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.