Performance Analysis and Tuning Red Hat Enterprise Linux 6 and 7

D. John Shakshober
Director /Sr Consulting Engineer
Red Hat Performance Engineering
Performance Analysis of RHEL6/7

- Performance Tools and Tuned
- Performance Analysis Utilities
  - Perf, Pcp, Tuna
  - Out of the Box Experience
- Disk / NUMA Tuning
- Network / Low Lat / NFV
- RHEV KVM / Open Stack
- Atomic / Open Shift
Red Hat Performance Engineering

• Benchmarks – code path coverage
  • CPU – linpack, Imbench
  • Memory – Imbench, McCalpin Streams
  • Disk IO – iozone, aiostress – scsi, FC, iSCSI
  • Filesystem – IOzone, postmark– ext3/4, xfs, gfs2, gluster
    • Network – Netperf – 10 Gbit, 40 Gbit IB, PCI3
• Bare Metal, RHEL6/7 KVM, Atomic Container
• White box Intel/Arm/Power/AMD w/OEMs
# RHEL Performance Evolution

<table>
<thead>
<tr>
<th>RHEL5</th>
<th>RHEL6</th>
<th>RHEL7</th>
<th>RH Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Hugepages</td>
<td>Transparent Hugepages</td>
<td>Tuned - throughput-performance (default)</td>
<td>RHEV tuned profile</td>
</tr>
<tr>
<td>CPU Sets</td>
<td>Tuned - Choose Profile</td>
<td>Automatic NUMA-balancing</td>
<td>RHOP Tuned, NUMA, SR-IOV</td>
</tr>
<tr>
<td>Ktune on/off</td>
<td>NUMAD - userspace</td>
<td>RHEL Realtime</td>
<td>RHEL Atomic Host</td>
</tr>
<tr>
<td>CPU Affinity (taskset)</td>
<td>cgroups</td>
<td></td>
<td>OpenShift v3</td>
</tr>
<tr>
<td>NUMA Pinning (numactl)</td>
<td>irqbalance - NUMA enhanced</td>
<td></td>
<td>CloudForms</td>
</tr>
<tr>
<td>irqbalance</td>
<td></td>
<td></td>
<td></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)

- SPEC CPU2006: 83%
- SPECvirt_sc2013: 67%

World Record SAP SD 2-Tier Results

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

- RHEL 7 (Intel E5 v3): 16,500 Users
- RHEL 7.1 (Intel E7 v3): 31,000 Users
Subsystem Analysis
Subsystem Analysis: CPU

```
# mpstat -P 1 1
04:27:56 PM  CPU   %usr  %nice %sys %iowait  %irq  %soft  %steal  %guest  %gnice  %idle
04:27:57 PM  1   0.00   0.00  0.00   0.00   0.00   0.00   0.00   0.00    0.00  100.00
04:27:58 PM  1  73.27   0.00  0.00   0.00   0.00   0.00   0.00   0.00    0.00   26.73
04:27:59 PM  1  100.00  0.00  0.00   0.00   0.00   0.00   0.00   0.00    0.00    0.00
04:28:00 PM  1  100.00  0.00  0.00   0.00   0.00   0.00   0.00   0.00    0.00    0.00
04:28:01 PM  1  100.00  0.00  0.00   0.00   0.00   0.00   0.00   0.00    0.00    0.00
04:28:02 PM  1  100.00  0.00  0.00   0.00   0.00   0.00   0.00   0.00    0.00    0.00
04:28:03 PM  1  26.00  0.00  1.00   0.00   0.00   0.00   0.00   0.00    0.00    73.00
04:28:04 PM  1   0.00   0.00  0.00   0.00   0.00   0.00   0.00   0.00    0.00  100.00
```
## Subsystem Analysis: Memory

```bash
# vmstat 1

procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
  r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
2  0      0 25554152 986896 13833932    0    0     0     0  307  266  0  1 99  0  0
1  0      0 24454408 986896 13833932    0    0     0     0 1286  548  0  4 96  0  0
1  0      0 23428300 986896 13833932    0    0     0     0 1288  508  0  4 96  0  0
1  0      0 22371768 986896 13833932    0    0     0     8 1120  150  0  4 96  0  0
1  0      0 21571872 986896 13833932    0    0     0     0 1162  305  1  4 96  0  0
1  0      0 21571872 986896 13833932    0    0     0     0 1045   70  4  0 96  0  0
1  0      0 21571872 986896 13833932    0    0     0     0 1045   70  4  0 96  0  0
0  0      0 25773696 986896 13833932    0    0     0    24 487 130  2  0 98  0  0
```
## Subsystem Analysis: Memory

**# 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><strong>Total</strong></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><strong>Total</strong></td>
<td>10733</td>
<td>10723</td>
<td>10740</td>
<td>10717</td>
<td>42913</td>
</tr>
</tbody>
</table>

*UNALIGNED*

*ALIGNED*
# numastat -mczs

Per-node system memory usage (in MBs):

<table>
<thead>
<tr>
<th></th>
<th>Node 0</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
<th>Node 4</th>
<th>Node 5</th>
<th>Node 6</th>
<th>Node 7</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemTotal</td>
<td>32766</td>
<td>32768</td>
<td>32768</td>
<td>32768</td>
<td>32768</td>
<td>32768</td>
<td>32752</td>
<td>262126</td>
<td></td>
</tr>
<tr>
<td>MemFree</td>
<td>31863</td>
<td>31965</td>
<td>32120</td>
<td>32086</td>
<td>32098</td>
<td>32080</td>
<td>32114</td>
<td>256388</td>
<td></td>
</tr>
<tr>
<td>MemUsed</td>
<td>903</td>
<td>803</td>
<td>648</td>
<td>682</td>
<td>670</td>
<td>688</td>
<td>654</td>
<td>690</td>
<td>5738</td>
</tr>
<tr>
<td>FilePages</td>
<td>11</td>
<td>26</td>
<td>8</td>
<td>37</td>
<td>21</td>
<td>18</td>
<td>9</td>
<td>45</td>
<td>176</td>
</tr>
<tr>
<td>Slab</td>
<td>25</td>
<td>16</td>
<td>7</td>
<td>10</td>
<td>12</td>
<td>36</td>
<td>10</td>
<td>10</td>
<td>126</td>
</tr>
<tr>
<td>Active</td>
<td>5</td>
<td>13</td>
<td>4</td>
<td>25</td>
<td>10</td>
<td>9</td>
<td>6</td>
<td>41</td>
<td>113</td>
</tr>
<tr>
<td>Active(file)</td>
<td>4</td>
<td>11</td>
<td>3</td>
<td>23</td>
<td>8</td>
<td>6</td>
<td>3</td>
<td>40</td>
<td>99</td>
</tr>
<tr>
<td>SUnreclaim</td>
<td>19</td>
<td>10</td>
<td>6</td>
<td>6</td>
<td>9</td>
<td>33</td>
<td>7</td>
<td>7</td>
<td>97</td>
</tr>
<tr>
<td>Inactive</td>
<td>7</td>
<td>15</td>
<td>4</td>
<td>14</td>
<td>12</td>
<td>12</td>
<td>6</td>
<td>6</td>
<td>76</td>
</tr>
<tr>
<td>Inactive(file)</td>
<td>7</td>
<td>15</td>
<td>4</td>
<td>14</td>
<td>12</td>
<td>12</td>
<td>6</td>
<td>6</td>
<td>76</td>
</tr>
<tr>
<td>SReclaimable</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>29</td>
</tr>
<tr>
<td>Active(anon)</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>AnonPages</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>Mapped</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>KernelStack</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>PageTables</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Shmem</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Inactive(anon)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
# iostat -N 1

Linux 3.10.0-229.el7.x86_64 (jerms-lab7.perf.lab.eng.rdu.redhat.com)
03/12/2015   _x86_64_ (24 CPU)

### avg-cpu:

<table>
<thead>
<tr>
<th></th>
<th>%user</th>
<th>%nice</th>
<th>%system</th>
<th>%iowait</th>
<th>%steal</th>
<th>%idle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.03</td>
<td>0.00</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
<td>99.95</td>
</tr>
</tbody>
</table>

### Device:

<table>
<thead>
<tr>
<th>Device</th>
<th>tps</th>
<th>kB_read/s</th>
<th>kB_wrtn/s</th>
<th>kB_read</th>
<th>kB_wrtn</th>
</tr>
</thead>
<tbody>
<tr>
<td>vda</td>
<td>5.42</td>
<td>17.19</td>
<td>59.87</td>
<td>19633224</td>
<td>68398872</td>
</tr>
<tr>
<td>vdb</td>
<td>0.46</td>
<td>0.98</td>
<td>5.48</td>
<td>1119751</td>
<td>6263272</td>
</tr>
<tr>
<td>vg0-swap</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1008</td>
<td>0</td>
</tr>
<tr>
<td>vg0-root</td>
<td>5.46</td>
<td>17.18</td>
<td>59.87</td>
<td>19625293</td>
<td>68396764</td>
</tr>
</tbody>
</table>

### avg-cpu:

<table>
<thead>
<tr>
<th></th>
<th>%user</th>
<th>%nice</th>
<th>%system</th>
<th>%iowait</th>
<th>%steal</th>
<th>%idle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.04</td>
<td>0.00</td>
<td>0.71</td>
<td>3.31</td>
<td>0.00</td>
<td>95.94</td>
</tr>
</tbody>
</table>

### Device:

<table>
<thead>
<tr>
<th>Device</th>
<th>tps</th>
<th>kB_read/s</th>
<th>kB_wrtn/s</th>
<th>kB_read</th>
<th>kB_wrtn</th>
</tr>
</thead>
<tbody>
<tr>
<td>vda</td>
<td>7043.00</td>
<td>0.00</td>
<td>12325.00</td>
<td>0</td>
<td>12326</td>
</tr>
<tr>
<td>vdb</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>vg0-swap</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>vg0-root</td>
<td>7042.00</td>
<td>0.00</td>
<td>12322.00</td>
<td>0</td>
<td>12322</td>
</tr>
</tbody>
</table>
# Subsystem Analysis: Network

```bash
# ifpps -d <device>
```

```
3.10.0-231.el7.bz1200859.x86_64, pip1 (sfc 10000Mbit/s link:yes), t=1000ms, cpus=5+1/16  
(consider to increase your sampling interval, e.g. -t 100000)

<table>
<thead>
<tr>
<th></th>
<th>rx:</th>
<th>tx:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>339.255 MiB/t</td>
<td>10.589 MiB/t</td>
<td>443006 pkts/t</td>
<td>91660 pkts/t</td>
<td>668 drops/t</td>
<td>0 drops/t</td>
</tr>
<tr>
<td></td>
<td>1633625 drops</td>
<td>0 drops</td>
<td>0 errors/t</td>
<td>0 errors/t</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sys:</td>
<td>94436 cs/t</td>
<td>1569 procs</td>
<td>2 running</td>
<td>0 iowait</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mem:</td>
<td>128728M total</td>
<td>4109M used</td>
<td>2062M active</td>
<td>521M inactive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>swap:</td>
<td>1023M total</td>
<td>0M used</td>
<td>0M cached</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpu13 +::</td>
<td>1.0% us/t</td>
<td>15.5% sy/t</td>
<td>83.5% id/t</td>
<td>0.0% iow/t</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpu15</td>
<td>::</td>
<td>1.0% us/t</td>
<td>13.5% sy/t</td>
<td>85.4% id/t</td>
<td>0.0% iow/t</td>
<td></td>
</tr>
<tr>
<td>cpu 1</td>
<td>::</td>
<td>1.0% us/t</td>
<td>12.4% sy/t</td>
<td>86.6% id/t</td>
<td>0.0% iow/t</td>
<td></td>
</tr>
<tr>
<td>cpu 3</td>
<td>::</td>
<td>1.0% us/t</td>
<td>12.1% sy/t</td>
<td>86.9% id/t</td>
<td>0.0% iow/t</td>
<td></td>
</tr>
<tr>
<td>cpu 5</td>
<td>::</td>
<td>2.0% us/t</td>
<td>10.2% sy/t</td>
<td>87.8% id/t</td>
<td>0.0% iow/t</td>
<td></td>
</tr>
<tr>
<td>cpu14 -::</td>
<td>0.0% us/</td>
<td>0.0% sy/</td>
<td>100.0% id/</td>
<td>0.0% iow/</td>
<td></td>
<td></td>
</tr>
<tr>
<td>avg:</td>
<td>0.0%</td>
<td>4.4%</td>
<td>95.6%</td>
<td>0.0%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpu13 +::</td>
<td>19658 irqs/t</td>
<td>19239 sirq rx/t</td>
<td>74 sirq tx/t</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpu15</td>
<td>::</td>
<td>14126 irqs/t</td>
<td>13910 sirq rx/t</td>
<td>57 sirq tx/t</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpu 9</td>
<td>::</td>
<td>12391 irqs/t</td>
<td>12364 sirq rx/t</td>
<td>27 sirq tx/t</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpu 1</td>
<td>::</td>
<td>11082 irqs/t</td>
<td>10996 sirq rx/t</td>
<td>33 sirq tx/t</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpu 5</td>
<td>::</td>
<td>9862 irqs/t</td>
<td>9829 sirq rx/t</td>
<td>68 sirq tx/t</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
# pmcollectl -s cdnm

<table>
<thead>
<tr>
<th>#</th>
<th>CPU</th>
<th>IOPS</th>
<th>NET</th>
<th>MEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>216</td>
<td>179</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>202</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1678</td>
<td>2073</td>
<td>6876</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2361</td>
<td>216</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2348</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>465</td>
<td>516</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>217</td>
<td>185</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
But...what if we have a problem?

- Automatic not enough...
- Need to eek out the last X percent
- Need Determinism
Tuned Updates for RHEL7

- Installed by default!
- Profiles updated for RHEL7 features and characteristics
- Profiles automatically set based on installation
  - Desktop/Workstation: balanced profile
  - Server/HPC: throughput-performance profile
- Optional hook/callout capability
- Concept of Inheritance (just like httpd.conf)
RHEL “tuned” package

Available profiles:
- balanced
- desktop
- latency-performance
- myprofile << Easy to add your own
- network-latency
- network-throughput
- throughput-performance
- virtual-guest
- virtual-host

Current active profile: myprofile
Tuned Profiles throughout Red Hat's Product Line

- **RHEL7 Desktop/Workstation**: balanced
- **RHEL6/7 KVM Host, Guest**: virtual-host/guest
- **Red Hat Storage**: rhs-high-throughput, virt
- **RHEL Atomic**: atomic-host/guest
- **RHEL7 Server/HPC**: throughput-performance
- **RHEV**: virtual-host
- **RHEL OSP (compute node)**: virtual-host
perf list

List counters/tracepoints available on your system

# perf list

List of pre-defined events (to be used in -e):
  cpu-cycles OR cycles
  instructions
  cache-references
  cache-misses
  branch-instructions OR branches
  branch-misses
  cpu-clock
  task-clock
  page-faults OR faults
  context-switches OR cs
  cpu-migrations OR migrations
  minor-faults
  major-faults
perf top

System-wide 'top' view of busy functions

<table>
<thead>
<tr>
<th>Percentage</th>
<th>Process</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>34.35%</td>
<td>httpd</td>
<td>[kernel.kallsyms] avtab_search_node</td>
</tr>
<tr>
<td>12.70%</td>
<td>httpd</td>
<td>[kernel.kallsyms] _spin_lock</td>
</tr>
<tr>
<td>8.61%</td>
<td>httpd</td>
<td>[kernel.kallsyms] tg_load_down</td>
</tr>
<tr>
<td>7.42%</td>
<td>httpd</td>
<td>[kernel.kallsyms] _spin_lock_irq</td>
</tr>
<tr>
<td>5.79%</td>
<td>init</td>
<td>[kernel.kallsyms] intel_idle</td>
</tr>
<tr>
<td>3.92%</td>
<td>httpd</td>
<td>[kernel.kallsyms] _spin_lock_irqsave</td>
</tr>
<tr>
<td>1.75%</td>
<td>httpd</td>
<td>[kernel.kallsyms] sidtab_search_core</td>
</tr>
<tr>
<td>1.74%</td>
<td>httpd</td>
<td>[kernel.kallsyms] load_balance_fair</td>
</tr>
<tr>
<td>1.18%</td>
<td>httpd</td>
<td>[kernel.kallsyms] tg_nop</td>
</tr>
<tr>
<td>1.13%</td>
<td>init</td>
<td>[kernel.kallsyms] _spin_lock</td>
</tr>
</tbody>
</table>
ificacion

perf record

- Record system-wide (-a)
  - perf record -a sleep 10
  - perf record -a  // Hit ctrl-c when done.

- Or record a single command
  - perf record myapp.exe

- Or record an existing process (-p)
  - perf record -p <pid>

- Or add call-chain recording (-g)
  - perf record -g ls -rl /root

- Or only record specific events (-e)
  - perf record -e branch-misses -p <pid>
perf record

- Record system-wide (-a)

```bash
# perf record -a dd if=/dev/zero of=test bs=1M count=1000 conv=fdatasync oflag=direct

# perf report --stdio |head -20
# To display the perf.data header info, please use --header/--header-only options.
#
# Samples: 2K of event 'cycles'
# Event count (approx.): 438884664
#
# Overhead       Command      Shared Object                                            Symbol
# ........  ............  .................  ................................................
#
<table>
<thead>
<tr>
<th>Overhead</th>
<th>Command</th>
<th>Shared Object</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>38.18%</td>
<td>dd [kernel.kallsyms] [k]</td>
<td>__clear_user</td>
<td></td>
</tr>
<tr>
<td>5.08%</td>
<td>dd [kernel.kallsyms] [k]</td>
<td>do_blockdev_direct_IO</td>
<td></td>
</tr>
<tr>
<td>3.66%</td>
<td>dd [kernel.kallsyms] [k]</td>
<td>gup_pte_range</td>
<td></td>
</tr>
<tr>
<td>3.52%</td>
<td>dd [kernel.kallsyms] [k]</td>
<td>put_page</td>
<td></td>
</tr>
<tr>
<td>1.85%</td>
<td>swapper [kernel.kallsyms] [k]</td>
<td>native_safe_halt</td>
<td></td>
</tr>
<tr>
<td>1.79%</td>
<td>dd [kernel.kallsyms] [k]</td>
<td>__domain_mapping</td>
<td></td>
</tr>
<tr>
<td>1.67%</td>
<td>dd [kernel.kallsyms] [k]</td>
<td>__bio_add_page</td>
<td></td>
</tr>
<tr>
<td>1.64%</td>
<td>dd [kernel.kallsyms] [k]</td>
<td>__blk_segment_map_sg</td>
<td></td>
</tr>
<tr>
<td>1.29%</td>
<td>dd [kernel.kallsyms] [k]</td>
<td>rb_prev</td>
<td></td>
</tr>
<tr>
<td>1.11%</td>
<td>swapper [kernel.kallsyms] [k]</td>
<td>irq_entries_start</td>
<td></td>
</tr>
<tr>
<td>0.99%</td>
<td>dd [kernel.kallsyms] [k]</td>
<td>sg_next</td>
<td></td>
</tr>
<tr>
<td>0.97%</td>
<td>dd [kernel.kallsyms] [k]</td>
<td>dm_table_find_target</td>
<td></td>
</tr>
</tbody>
</table>
```
perf report

```
# Overhead    Command                                Shared Object
#   ............  ..............................................................
#
# 43.53%  dd [kernel.kallsyms] [k] __clear_user
#         | __clear_user
#            |--99.75%-- read_zero.part.5
#                        read_zero
#                        vfs_read
#                        sys_read
#                        system_call_fastpath
#                        __GI___libc_read
#                        --0.25%-- [...]

5.37%  dd [kernel.kallsyms] [k] do_blockdev_direct_IO
#         |-- do_blockdev_direct_IO
#                  __blockdev_direct_IO
#                  xfs_vm_direct_IO
#                  generic_file_direct_write
#                  xfs_file_dio_aio_write
#                  xfs_file_aio_write
#                  do_sync_write
```

/dev/zero

oflag=direct
## perf diff

### Event 'cycles'

<table>
<thead>
<tr>
<th>Baseline</th>
<th>Delta</th>
<th>Shared Object</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.88%</td>
<td>-12.27%</td>
<td>[kernel.kallsyms]</td>
<td></td>
</tr>
<tr>
<td>11.97%</td>
<td>-11.17%</td>
<td>systemd</td>
<td>[k] __lookup_mnt</td>
</tr>
<tr>
<td>4.32%</td>
<td>+6.43%</td>
<td>libdbus-1.so.3.7.4</td>
<td>[.] 0x00000000000064968</td>
</tr>
<tr>
<td>4.06%</td>
<td>+4.72%</td>
<td>dbus-daemon</td>
<td>[.] 0x0000000000029258</td>
</tr>
<tr>
<td>3.79%</td>
<td>-3.79%</td>
<td>libglib-2.0.so.0.3600.3</td>
<td>[.] 0x0000000000014a6e</td>
</tr>
<tr>
<td>3.72%</td>
<td>+0.25%</td>
<td>[kernel.kallsyms]</td>
<td>[k] seq_list_start</td>
</tr>
</tbody>
</table>
perf list

grep for something interesting, maybe to see what numabalance is doing?
New C-2-C RHEL7.3
Cacheline Contention – high level

64 byte chunk of memory
(size of cacheline)

Socket 0
CPU CPU CPU CPU CPU CPU CPU CPU
... Read/write
Offset 0
Offset 8
Offset 16
Offset 24
Offset 32
Offset 40
Offset 48
Offset 56

Socket 1
CPU CPU CPU CPU CPU CPU CPU CPU
... Read/write

Socket 2
CPU CPU CPU CPU CPU CPU CPU CPU
... Read/write

Socket 3
CPU CPU CPU CPU CPU CPU CPU CPU
...

When caches in individual cpus are modified, the cache coherency protocol has to work harder to maintain consistency.

Can really hurt performance.
This shows us:
- The hottest contended cachelines
- The process names, data addr, ip, pids, tids
- The node and CPU numbers they ran on,
- And how the cacheline is being accessed (read or write)
## HSW EX Brickland - recent “perf c2c” activity

### Offset 0x00:

**Heavily read.** Tiny number of writes.

<table>
<thead>
<tr>
<th>Cnt</th>
<th>inst_addr</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>814</td>
<td>0x7f70fd202c99</td>
<td>libhdbbasis.so:ltt::allocated_refcounted::destroyImp</td>
</tr>
<tr>
<td>347</td>
<td>0x7f70fd202cad</td>
<td>libhdbbasis.so:ltt::allocated_refcounted::destroyImp</td>
</tr>
</tbody>
</table>

### Offset 0x10:

**Heavily read.** Tiny number of writes.

<table>
<thead>
<tr>
<th>Cnt</th>
<th>inst_addr</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1427</td>
<td>0x7f70fd12afb4</td>
<td>libhdbbasis.so:MemoryManager::PoolAllocator::deallocateImpl</td>
</tr>
<tr>
<td>148</td>
<td>0x7f70fd12f891</td>
<td>libhdbbasis.so:MemoryManager::PoolAllocator::allocateNoThrowImpl</td>
</tr>
<tr>
<td>662</td>
<td>0x7f7109cb8ab0</td>
<td>libhdbcs.so:ltt::allocated_refcounted::release</td>
</tr>
<tr>
<td>2810</td>
<td>0x7f7109cb8aba</td>
<td>libhdbcs.so:ltt::allocated_refcounted::release</td>
</tr>
<tr>
<td>4934</td>
<td>0x7f7109cb8ae0</td>
<td>libhdbcs.so:ltt::allocated_refcounted::addReference</td>
</tr>
<tr>
<td>15</td>
<td>0x7f7109cb8ae6</td>
<td>libhdbcs.so:ltt::allocated_refcounted::addReference</td>
</tr>
</tbody>
</table>

### Offset 0x20:

**Heavily written.** Tiny number of reads.

<table>
<thead>
<tr>
<th>Cnt</th>
<th>inst_addr</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>8074</td>
<td>0x7f70fd12b230</td>
<td>libhdbbasis.so:MemoryManager::PoolAllocator::addReference</td>
</tr>
<tr>
<td>19</td>
<td>0x7f70fd12b236</td>
<td>libhdbbasis.so:MemoryManager::PoolAllocator::addReference</td>
</tr>
<tr>
<td>474</td>
<td>0x7f70fd130378</td>
<td>libhdbbasis.so:MemoryManager::PoolAllocator::release</td>
</tr>
<tr>
<td>3919</td>
<td>0x7f70fd130383</td>
<td>libhdbbasis.so:MemoryManager::PoolAllocator::release</td>
</tr>
</tbody>
</table>

**Reports include:**

- All simultaneous readers and writers to a cacheline.
- The sockets and cpus the accesses are coming from.
- The data addr, pid, tid, load latencies, where data was sourced from.
Performance Co-Pilot - Overview

- What is PCP?
  - Open source **toolkit**
  - System-level analysis
  - Live and historical
  - Extensible (monitors, collectors)
  - Distributed

Supported in RHEL 7.0 and RHEL 6.6!
Performance Co-Pilot - Architecture

pmlogger
pmchart
pmie
# yum install pcp*
# systemctl enable {pmcd,pmlogger,pmwebd}
# systemctl start {pmcd,pmlogger,pmwebd}

Verify it's working:
# pmclient -t1
Performance Co-Pilot - Firewall Config

RHEL6:

-A INPUT -p tcp -m state --state NEW -m tcp --dport 44321 -j ACCEPT
-A INPUT -p udp -m state --state NEW -m udp --dport 44321 -j ACCEPT

RHEL7:

# firewall-cmd --permanent --zone=public --add-service=pmcd
# firewall-cmd --reload
Client toolkit - *pmchart*

- Arbitrary charts
- Load / Save views
- VCR-style playback
System Tuning Tool - tuna

- Tool for fine grained control
- Display applications / processes
- Displays CPU enumeration
- Socket (useful for NUMA tuning)
- Dynamic control of tuning
  - Process affinity
  - Parent & threads
  - Scheduling policy
  - Device IRQ priorities, etc
Tuna (RHEL6/7)

<table>
<thead>
<tr>
<th>Socket 0</th>
<th>Socket 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Usage</td>
</tr>
<tr>
<td>Filter</td>
<td>0</td>
</tr>
<tr>
<td>Filter</td>
<td>2</td>
</tr>
<tr>
<td>Filter</td>
<td>4</td>
</tr>
<tr>
<td>Filter</td>
<td>6</td>
</tr>
<tr>
<td>Filter</td>
<td>8</td>
</tr>
<tr>
<td>Filter</td>
<td>10</td>
</tr>
<tr>
<td>Filter</td>
<td>12</td>
</tr>
<tr>
<td>Filter</td>
<td>14</td>
</tr>
<tr>
<td>Filter</td>
<td>16</td>
</tr>
<tr>
<td>Filter</td>
<td>18</td>
</tr>
<tr>
<td>Filter</td>
<td>20</td>
</tr>
<tr>
<td>Filter</td>
<td>22</td>
</tr>
</tbody>
</table>

## IRQ Affinity

<table>
<thead>
<tr>
<th>IRQ</th>
<th>Affinity</th>
<th>Events</th>
<th>Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.23</td>
<td>12994</td>
<td>timer</td>
</tr>
<tr>
<td>1</td>
<td>0.2,4,6,8,10</td>
<td>2</td>
<td>i8042</td>
</tr>
<tr>
<td>2</td>
<td>0.2,4,6,8,10</td>
<td>268</td>
<td>serial</td>
</tr>
<tr>
<td>3</td>
<td>0.2,4,6,8,10</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.2,4,6,8,10</td>
<td>1</td>
<td>rtc0</td>
</tr>
<tr>
<td>5</td>
<td>0.2,4,6,8,10</td>
<td>0</td>
<td>acpi</td>
</tr>
<tr>
<td>6</td>
<td>0.2,4,6,8,10</td>
<td>4</td>
<td>i8042</td>
</tr>
<tr>
<td>7</td>
<td>0.2,4,6,8,10</td>
<td>0</td>
<td>pata_ettixp</td>
</tr>
<tr>
<td>8</td>
<td>0.2,4,6,8,10</td>
<td>0</td>
<td>pata_ettixp</td>
</tr>
<tr>
<td>9</td>
<td>0.2,4,6,8,10</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.2,4,6,8,10</td>
<td>0</td>
<td>radem.anhci</td>
</tr>
<tr>
<td>11</td>
<td>0.2,4,6,8,10</td>
<td>0</td>
<td>ehc_hcd.usb2,ohci_hcd.usb3,ohci_hcd.usb4</td>
</tr>
<tr>
<td>12</td>
<td>0.2,4,6,8,10</td>
<td>0</td>
<td>ehc_hcd.usb1,ohci_hcd.usb5,ohci_hcd.usb6</td>
</tr>
<tr>
<td>13</td>
<td>0.2,4,6,8,10</td>
<td>12,14,16,18,20,22</td>
<td>uhci_hcd.usb7.hplio</td>
</tr>
</tbody>
</table>

### MD Policy Priority Affinity VolCtxtSwitch NonVolCtxtSwitch Command Line

<table>
<thead>
<tr>
<th>MD</th>
<th>Policy</th>
<th>Priority</th>
<th>Affinity</th>
<th>VolCtxtSwitch</th>
<th>NonVolCtxtSwitch</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OTHER</td>
<td>0</td>
<td>0.23</td>
<td>1452</td>
<td>55</td>
<td>/sbin/init</td>
</tr>
<tr>
<td>363</td>
<td>OTHER</td>
<td>0</td>
<td>0.23</td>
<td>000000000000</td>
<td>0</td>
<td>/sbin/systemd</td>
</tr>
<tr>
<td>404</td>
<td>OTHER</td>
<td>0</td>
<td>0.2,4,6,8,10</td>
<td>59280707</td>
<td>77026</td>
<td>/usr/libexec/kvm-kvm-name-kvm-broker-S-M:int64:4.6:cpu Option_53, netdev_msr, runc, server, systemd/</td>
</tr>
<tr>
<td>911</td>
<td>OTHER</td>
<td>0</td>
<td>0.23</td>
<td>6688</td>
<td>91</td>
<td>/sbin/systemd</td>
</tr>
<tr>
<td>2248</td>
<td>OTHER</td>
<td>0</td>
<td>0.23</td>
<td>111966</td>
<td>0</td>
<td>auditd</td>
</tr>
<tr>
<td>2446</td>
<td>OTHER</td>
<td>0</td>
<td>0.23</td>
<td>1</td>
<td>0</td>
<td>/sbin/portservice</td>
</tr>
<tr>
<td>2453</td>
<td>OTHER</td>
<td>0</td>
<td>0.23</td>
<td>51</td>
<td>0</td>
<td>/sbin/systemd --systemd-1/var/run/systemd.pid &lt; 5</td>
</tr>
<tr>
<td>2482</td>
<td>OTHER</td>
<td>0</td>
<td>0.23</td>
<td>375632</td>
<td>1387</td>
<td>irqbalance</td>
</tr>
<tr>
<td>2503</td>
<td>OTHER</td>
<td>0</td>
<td>0.23</td>
<td>126446</td>
<td>0</td>
<td>rpcbind</td>
</tr>
<tr>
<td>2510</td>
<td>OTHER</td>
<td>0</td>
<td>0.23</td>
<td>10356</td>
<td>34</td>
<td>sshd: root@pts/2</td>
</tr>
<tr>
<td>2513</td>
<td>OTHER</td>
<td>0</td>
<td>0.23</td>
<td>49</td>
<td>6</td>
<td>bash</td>
</tr>
<tr>
<td>2521</td>
<td>OTHER</td>
<td>0</td>
<td>0.23</td>
<td>12</td>
<td>0</td>
<td>rpc.statd</td>
</tr>
<tr>
<td>2542</td>
<td>OTHER</td>
<td>0</td>
<td>0.23</td>
<td>5567</td>
<td>1302</td>
<td>/usr/bin/python /usr/bin/tuna</td>
</tr>
<tr>
<td>2577</td>
<td>OTHER</td>
<td>0</td>
<td>0.23</td>
<td>1</td>
<td>0</td>
<td>rpc.idmapd</td>
</tr>
<tr>
<td>2677</td>
<td>OTHER</td>
<td>0</td>
<td>0.23</td>
<td>2485</td>
<td>3</td>
<td>dbus-daemon --systemd</td>
</tr>
<tr>
<td>2689</td>
<td>OTHER</td>
<td>0</td>
<td>0.23</td>
<td>7745159</td>
<td>43353</td>
<td>avahi-daemon</td>
</tr>
<tr>
<td>2690</td>
<td>OTHER</td>
<td>0</td>
<td>0.23</td>
<td>3</td>
<td>0</td>
<td>avahi-daemon</td>
</tr>
<tr>
<td>2718</td>
<td>OTHER</td>
<td>0</td>
<td>0.23</td>
<td>2</td>
<td>0</td>
<td>/usr/sbin/acpid</td>
</tr>
<tr>
<td>2737</td>
<td>OTHER</td>
<td>0</td>
<td>0.23</td>
<td>0</td>
<td>0</td>
<td>bash</td>
</tr>
</tbody>
</table>
Tuna GUI Capabilities Updated for RHEL7

Monitoring | Profile management | Profile editing
---|---|---

Current active tuna profile: example.conf

[Save Snapshot] [Save & Apply permanently] [Restore changes] [Apply changes]

**Kernel scheduler**
- `kernel.core_pattern` - core
- `kernel.sched_latency_ns` - 24000000
- `kernel.sched_min_granularity_ns` - 10000000
- `kernel.sched_nr_migrate` - 32
- `kernel.sched_rt_period_us` - 1000000
- `kernel.sched_rt_runtime_us` - 950000
- `kernel.sched_tunable_scaling` - 1
- `kernel.sched_wakeup_granularity_ns` - 4000000

**VM**
- `vm.dirty_expire_centisecs`
- `vm.dirty_ratio`
- `vm.dirty_writeback_centisecs`
- `vm.laptop_mode`
- `vm.memory_failure_early_kill`
- `vm.swappiness`

**Network IPv4**
- `ipv4.conf.all.forwarding`
- `ipv4.conf.all_rp_filter` - 0
- `ipv4.tcp_congestion_control` - cubic

**Network IPv6**
- `ipv6.conf.all.forwarding`
- `ipv6.conf.default.forwarding`
- `ipv6.conf.docker0.forwarding`
- `ipv6.conf.em1.forwarding`
- `ipv6.conf.em2.forwarding`
### Helpful Utilities

**Supportability**
- redhat-support-tool
- sos
- kdump
- perf
- psmisc
- strace
- sysstat
- systemtap
- trace-cmd
- util-linux-ng

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

**Power/Tuning**
- cpupowerutils (R6)
- kernel-tools (R7)
- powertop
- tuna
- tuned

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

**Storage**
- blktrace
- iotop
- iostat
Performance Optimizations RHEL7

**CPU**
- Support for all new CPUs
- AVX2 instruction support
- RHEL-RT released March 5

**Memory**
- Automatic NUMA Balancing
- Tunable workqueues (writeback)

**Networking**
- Full support for PTP1588v2
- Route cache → F.I.B.
- irqbalance handles NUMA
- busy_poll, tcp_fastopen
- nohz_full (tickless while active)
- Byte Queue Limits
- TCP Small Queues

**Power Management**
- intel_pstate
- tuned does most heavy lifting
Disk IO
Tuned-adm profile throughput-performance (R7 default)

- 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
Tuned: Storage Performance Boost

RHEL7 File System In Cache Perf

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

Throughput in MB/Sec

<table>
<thead>
<tr>
<th>File System</th>
<th>not tuned</th>
<th>tuned</th>
</tr>
</thead>
<tbody>
<tr>
<td>ext3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ext4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>xfs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gfs2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Larger is better
I/O Tuning - Configuring I/O Elevators

• **Boot-time**
  - Grub command line –
    elevator=deadline/cfq/noop

• **Dynamically, per device**
  - echo “deadline” >
    /sys/class/block/sda/queue/scheduler

• **tuned (RHEL6 utility)**
  - tuned-adm profile throughput-performance
  - tuned-adm profile enterprise-storage
Impact of I/O Elevators – OLTP Workload

34.26% better than CFQ with higher process count.
Impact of I/O Elevators - DSS Workload

Comparison CFQ vs Deadline

Oracle DSS Workload (with different thread count)
Tuning Memory - Flushing Caches

- Drop unused Cache
  - ✔ Frees unused memory
  - ✔ File cache
  - ✗ If the DB uses cache, may notice slowdown

- Free pagecache
  - echo 1 > /proc/sys/vm/drop_caches

- Free slabcache
  - echo 2 > /proc/sys/vm/drop_caches

- Free pagecache and slabcache
  - echo 3 > /proc/sys/vm/drop_caches
dirty_ratio and dirty_background_ratio

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**
Tuning Memory – swappiness

- Not needed as much in RHEL7
- 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
NonUniform Memory Access NUMA
Understanding NUMA (Non Uniform Memory Access)

- Multi Socket – Multi core architecture
  - NUMA required for scaling
  - RHEL 5 / 6 completely NUMA aware
  - Additional performance gains by enforcing NUMA placement
Understand the Configuration

Sampling of tools we often use. I'm sure you have your own favorites.

• `lscpu`: Display information about the CPU architecture
• `lstopo`: Show the topology of the system
• `numactl --hardware`: Show inventory of available nodes

• `dmidecode`: DMI table decoder
• `lspci -t -vv`: List all PCI devices
• `lsblk`: List block devices
• `blkid`: Locate & print block device attributes
• `cat /proc/cmdline`: See flags kernel booted with.
• `ifconfig -a`: Display available network interfaces

• … <and all your favorites>
How can I visualize my system's NUMA topology in Red Hat Enterprise Linux?

https://access.redhat.com/site/solutions/62879
RHEL NUMA Scheduler

- RHEL6
  - numactl, numastat enhancements
  - numad – usermode tool, dynamically monitor, auto-tune
- RHEL7 – auto numa balancing
  - Moves tasks (threads or processes) closer to the memory they are accessing.
  - Moves application data to memory closer to the tasks that reference it.
  - A win for most apps.
  - Enable / Disable
    - sysctl kernel.numablocking={0,1}
numastat: per-node meminfo (new)

# numastat -mczs

<table>
<thead>
<tr>
<th></th>
<th>Node 0</th>
<th>Node 1</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemTotal</td>
<td>65491</td>
<td>65536</td>
<td>131027</td>
</tr>
<tr>
<td>MemFree</td>
<td>60366</td>
<td>59733</td>
<td>120099</td>
</tr>
<tr>
<td>MemUsed</td>
<td>5124</td>
<td>5803</td>
<td>10927</td>
</tr>
<tr>
<td>Active</td>
<td>2650</td>
<td>2827</td>
<td>5477</td>
</tr>
<tr>
<td>FilePages</td>
<td>2021</td>
<td>3216</td>
<td>5238</td>
</tr>
<tr>
<td>Active(file)</td>
<td>1686</td>
<td>2277</td>
<td>3963</td>
</tr>
<tr>
<td>Active(anon)</td>
<td>964</td>
<td>551</td>
<td>1515</td>
</tr>
<tr>
<td>AnonPages</td>
<td>964</td>
<td>550</td>
<td>1514</td>
</tr>
<tr>
<td>Inactive</td>
<td>341</td>
<td>946</td>
<td>1287</td>
</tr>
<tr>
<td>Inactive(file)</td>
<td>340</td>
<td>946</td>
<td>1286</td>
</tr>
<tr>
<td>Slab</td>
<td>380</td>
<td>438</td>
<td>818</td>
</tr>
<tr>
<td>SReclaimable</td>
<td>208</td>
<td>207</td>
<td>415</td>
</tr>
<tr>
<td>SUNreclaim</td>
<td>173</td>
<td>230</td>
<td>403</td>
</tr>
<tr>
<td>AnonHugePages</td>
<td>134</td>
<td>236</td>
<td>370</td>
</tr>
</tbody>
</table>
numastat – per-PID mode

# numastat -c java (default scheduler – non-optimal)
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>57501 (java)</td>
<td>755</td>
<td>1121</td>
<td>480</td>
<td>698</td>
<td>3054</td>
</tr>
<tr>
<td>57502 (java)</td>
<td>1068</td>
<td>702</td>
<td>573</td>
<td>723</td>
<td>3067</td>
</tr>
<tr>
<td>57503 (java)</td>
<td>649</td>
<td>1129</td>
<td>687</td>
<td>606</td>
<td>3071</td>
</tr>
<tr>
<td>57504 (java)</td>
<td>1202</td>
<td>678</td>
<td>1043</td>
<td>150</td>
<td>3073</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>3674</strong></td>
<td><strong>3630</strong></td>
<td><strong>2783</strong></td>
<td><strong>2177</strong></td>
<td><strong>12265</strong></td>
</tr>
</tbody>
</table>

# numastat -c java (numabalance close to opt)
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>56918 (java)</td>
<td>49</td>
<td><strong>2791</strong></td>
<td>56</td>
<td>37</td>
<td>2933</td>
</tr>
<tr>
<td>56919 (java)</td>
<td><strong>2769</strong></td>
<td>76</td>
<td>55</td>
<td>32</td>
<td>2932</td>
</tr>
<tr>
<td>56920 (java)</td>
<td>19</td>
<td>55</td>
<td>77</td>
<td><strong>2780</strong></td>
<td>2932</td>
</tr>
<tr>
<td>56921 (java)</td>
<td>97</td>
<td>65</td>
<td><strong>2727</strong></td>
<td>47</td>
<td>2936</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>2935</strong></td>
<td><strong>2987</strong></td>
<td><strong>2916</strong></td>
<td><strong>2896</strong></td>
<td><strong>11734</strong></td>
</tr>
</tbody>
</table>
Memory Tuning - Effect of “numa”

4 KVM guest running OLTP workload

Comparison between no-numa / numad / manual numa pin

NUMAD performance is as good as manual NUMA pinning
RHEL 6.6 vs RHEL 7.1 SAP HANA Performance
All due to Auto NUMA balancing (kernel.numa_balancing = 1)
Power Management
Networking
Low Latency
**Tuned network-latency Profile**

### Parents
- throughput-performance
- balanced
- latency-performance

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

### Children/Grandchildren
- Feed Handler
- Gateway
- Matching Eng.

**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

**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
Tuned: Network-Latency Performance Boost

C-state lock improves determinism, reduces jitter

![Graph showing latency improvements across C6, C3, C1, and C0 states](image)
turbostat shows P/C-states on Intel CPUs

turbostat begins shipping in RHEL6.4, cpupowerutils package

### Default

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

### latency-performance

<table>
<thead>
<tr>
<th>pk</th>
<th>cor</th>
<th>CPU</th>
<th>GHz</th>
<th>TSC</th>
<th>%c0</th>
<th>%c1</th>
<th>%c3</th>
<th>%c6</th>
<th>%c7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>3.30</td>
<td>2.90</td>
<td>100</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>3.30</td>
<td>2.90</td>
<td>100</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0.00</td>
<td>3.30</td>
<td>2.90</td>
<td>100</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0.00</td>
<td>3.30</td>
<td>2.90</td>
<td>100</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Impact of CPU Idle Drivers (watts per workload)
Low Latency Performance Tuning Guide for Red Hat Enterprise Linux 7

- Tactical tuning overview for latency-sensitive workloads.
- Emphasizes impactful new features included in RHEL7:
  - CPU/power management
  - NUMA
  - tuned profiles
  - scheduling
  - network tunables
  - kernel timers.
  - "de-jittering" CPU cores
  - tracing techniques

[https://access.redhat.com/articles/1323793](https://access.redhat.com/articles/1323793)
Full DynTicks Patchset - nohz_full=1

Patchset Goal:
● Stop interrupting userspace tasks
● Move timekeeping to non-latency-sensitive cores
● If nr_running=1, then scheduler/tick can avoid that core
● Default disabled...Opt-in via nohz_full cmdline option

Kernel Tick:
● timekeeping (gettimeofday)
● Scheduler load balancing
● Memory statistics (vmstat)
RHEL6 and 7 Tickless

Time (CONFIG_HZ=1000)

Tick

No

No

No

Tick

Userspace Task

Timer Interrupt

Idle

Red Hat Performance Engineering
RHEL7 nohz_full

Time (CONFIG_HZ=1000)

No Ticks

Userspace Task  Timer Interrupt
SO_BUSY_POLL Socket Option

● Socket-layer code polls receive queue of NIC

● Replaces interrupts and NAPI

● Retains full capabilities of kernel network stack
RHEL Atomic / Open Shift
Red Hat Enterprise Linux Atomic Host

**IT IS RED HAT ENTERPRISE LINUX**

Inherits the complete hardware ecosystem, military-grade security, stability, and reliability for which Red Hat Enterprise Linux is known.

**OPTIMIZED FOR CONTAINERS**

**MINIMIZED FOOTPRINT**

Minimized host environment tuned for running Linux containers while maintaining compatibility with Red Hat Enterprise Linux.

**SIMPLIFIED MAINTENANCE**

Atomic updating and rollback means it’s easy to deploy, update, and rollback using image-based technology.

**ORCHESTRATION AT SCALE**

Build composite applications by orchestrating multiple containers as microservices on a single host instance.
RHEL 7 Containers Architecture

Containers

Docker CLI

Systemd

Docker Image

Unit File

Cgroups

Namespaces

SELinux

Drivers

RHEL Kernel

Hardware (Intel, AMD) or Virtual Machine
### Density

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

<table>
<thead>
<tr>
<th>Container Engine (Docker) with SELinux</th>
<th>Red Hat Enterprise Linux Atomic Host</th>
<th>Bare Metal/KVM Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deps</td>
<td>Deps</td>
<td>Deps</td>
</tr>
<tr>
<td>Container Engine (Docker) with SELinux</td>
<td>Red Hat Enterprise Linux Atomic Host</td>
<td>Bare Metal/KVM Host</td>
</tr>
</tbody>
</table>
Container creation speed vs VMs

Creation Speed
Higher is Better

# of running instances

Time (seconds)

# Containers started

# VMs started
Network Performance: netperf, RR tests (latency)

- With cpu pinning and IRQ Affinity in place

![Graph showing performance](image)

- Host to Host Tuned
- Host to Host
- Container to Container Passthrough
- Container to Container tuned
- Host to Container
- Container to Container

<table>
<thead>
<tr>
<th></th>
<th>TCP 1-way-latency (us)</th>
<th>UDP 1-way-latency (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host to Host</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>Container to Container tuned</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>Host to Container</td>
<td>12</td>
<td>14</td>
</tr>
</tbody>
</table>

Bridge adds some overhead
Tuned: RHEL Atomic Profile Inheritance

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 = 10
vm.swappiness=10

vm.dirty_ratio = 30
vm.swappiness = 30

avc_cache_threshold=65536
nf_conntrack_hashsize=131072
kernel.pid_max=131072
net.netfilter.nf_conntrack_max=1048576

atomic-guest

atomic-host
Container performance across multiple workloads

Time to Complete Test Workload
Higher is Better

% difference vs Bare metal

<table>
<thead>
<tr>
<th></th>
<th>Calculate primes</th>
<th>OLTP workload</th>
<th>Analytics App.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHEL7 bare metal</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>RHEL7 container</td>
<td>100</td>
<td>97</td>
<td>94</td>
</tr>
</tbody>
</table>

RHEL7 bare metal  RHEL7 container
RHEL 7.1 + OpenOnload
Bare Metal/Containers/SR-IOV
References

Low Latency Tuning Guide for Red Hat Enterprise Linux 7
https://access.redhat.com/articles/1323793

Accelerating Red Hat Enterprise Linux 7-based Linux Containers with Solarflare OpenOnload
https://access.redhat.com/articles/1407003

How do I create my own tuned profile on RHEL7?
https://access.redhat.com/solutions/731473
KVM / NFV / Realtime
Quick Overview - KVM Architecture

- Guests run as a process in userspace on the host
- A virtual CPU is implemented using a Linux thread
  - The Linux scheduler is responsible for scheduling a virtual CPU, as it is a normal thread
- Guests inherit features from the kernel
  - NUMA
  - Huge Pages
  - Support for new hardware
Virtualization Tuning - Caching

- **Cache = none (Figure 1)**
  - I/O from the guest is not cached on the host

- **Cache = writethrough (Figure 2)**
  - I/O from the guest is cached and written through on the host
    - Works well on large systems (lots of memory and CPU)
    - Potential scaling problems with this option with multiple guests (host CPU used to maintain cache)
    - Can lead to swapping on the host
Virt Tuning - **Effect of I/O Cache Settings**

OLTP testing in 4 VMs

Cache=WT vs Cache=none

![Bar chart showing performance comparison between Cache=WT and Cache=none with user set count and transactions per minute.](chart.png)
Virt Tuning – Using NUMA

4 Virtual Machines running OLTP workload

Trans / min normalized to 100

no NUMA | Manual Pin | NUMAD

7.54 | 7.05
Virt Tuning - Tuning Transparent Huge Pages

4 VM testing

Comparision between THP and huge pages on host

![Graph showing performance comparison between THP and huge pages on host with different user set sizes. The graph shows the number of transactions per minute for 10U, 20U, 40U, 60U, 80U, and 100U user sets. The bars are color-coded to represent THP-scan=10000, THP-scan=100, and Hugepages.]
**Virt Tuning - Kernel Samepage Merging (KSM)**

KSM breaks down THP, so performance advantage of THP is lot. Some of it is recovered by lowering the scan interval.

CPU overhead of KSM results in less CPU time for application.

- THP 10000scan, ksm on, mem opt server (150)
- THP 100scan, ksm on, mem opt server (150)
- THP 100scan, ksm off, mem opt server (150)
- THP 100scan, ksm off, no memory opt
RHEV 3.3 – Migration

Migrating a 108G VM running OLTP ~ 500K Trans/min

Configure – migration_max_bandwidth = <Value> in /etc/vdsm/vdsm.conf
Multi Instances of database with and without Hyperthreads

Each of the 4 instances were aligned to an individual NUMA node. This test shows the best gain in performances as other factors influencing performance like NUMA, I/O are not a factor.
Virtualization Tuning - Network

- VirtIO
  - ✔ VirtIO drivers for network
- vhost_net (low latency – close to line speed)
  - ✔ Bypass the qemu layer
- PCI pass through
  - ✔ Bypass the host and pass the PCI device to the guest
  - ✔ Can be passed only to one guest
- SR-IOV (Single root I/O Virtualization)
  - ✔ Pass through to the guest
  - ✔ Can be shared among multiple guests
  - ✔ Limited hardware support
Virtualization Tuning - Network - Latency Comparison

Network Latency by Guest Interface Method
Guest Receive (Lower is better)

Message Size (Bytes)
- host RX
- virtio RX
- vhost RX
- SR-IOV RX
• Intel Cache Monitoring / Cache Allocation Tech

**Cache Allocation Technology**

- **Guest VM DPDK_QOS**
  - Virtual Machine Monitor
  - Last Level Cache

- **Aggressor VM**
  - Virtual Machine Monitor
  - Last Level Cache

**Graphical Representation**

- DPDK QoS Sched Sample Application: 11 Mpps
- DPDK QoS Sched Sample Application: 4 Mpps
- DPDK QoS Sched Sample Application: 11 Mpps

**Textual Representation**

- **Parallel execution of aggressor workload causes 60% performance degradation**
- **Restore packet performance using Intel® Cache Allocation Technology**

Source: Intel® Corporation, Communications Infrastructure Division

Intel® Data Plane Development Kit (Intel® DPDK)
- **Intel Cache Monitoring / Cache Allocation Tech**

### Intel® Xeon® Processor E5 v4 Product Family Resource Director Technology (RDT)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Benefit</th>
<th>How Does it Work?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache Monitoring Technology (CMT)</td>
<td>Ability to monitor Last Level Cache occupancy for a set of threads</td>
<td>Each thread assigned a RMID (Resource Monitoring ID)</td>
</tr>
<tr>
<td>Cache Allocation Technology (CAT)</td>
<td>Ability to partition Last Level Cache, enforcement on a per thread basis Enables workload prioritization, consolidation, and resource partitioning Enables control over noisy neighbors</td>
<td>Each thread assigned a Class of Service Each Class of Service restricted to portion of LLC</td>
</tr>
<tr>
<td>Code and Data Prioritization (CDP)</td>
<td>A specialized extension of CAT which enables separate masks for code and data. This allows code to be protected at the L3 cache level for instance</td>
<td>Half of the masks are associated with code, the other half of the masks are associated with data</td>
</tr>
<tr>
<td>Memory Bandwidth Monitoring (MBM)</td>
<td>Monitors Memory Bandwidth utilization on an RMID basis. Identify memory bandwidth conflict issues and enable thread migration</td>
<td>RMIDs can be associated with one or a group of threads / applications</td>
</tr>
</tbody>
</table>
**Examples** - Lock target cache lines into the cache via CAT

- Set the default way mask to not include the last two cache ways
  - `wrmsr 0xc90 0xFFFFFC`

- Set a second way mask to only include the last two cache ways
  - `wrmsr 0xc91 0x3`

- Set core 1 to use the second way mask
  - `wrmsr –p 1 0xc8f 0x100000000`

- Use core 1 to touch all of the memory locations one wishes to lock permanently into the cache

- Set core 1 to use the default (first) way mask
  - `wrmsr –p 1 0xc8f 0x000000000`
Know your hardware: `lstopo`

### NUMA Node 0

```
<table>
<thead>
<tr>
<th>Socket #0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>L3 (256KB)</td>
<td></td>
</tr>
<tr>
<td>L2 (256KB)</td>
<td></td>
</tr>
<tr>
<td>L2 (256KB)</td>
<td></td>
</tr>
<tr>
<td>L1d (256KB)</td>
<td></td>
</tr>
<tr>
<td>L1d (256KB)</td>
<td></td>
</tr>
<tr>
<td>L1d (256KB)</td>
<td></td>
</tr>
<tr>
<td>L1d (256KB)</td>
<td></td>
</tr>
<tr>
<td>L1d (256KB)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Core P44</th>
<th>Core P45</th>
<th>Core P46</th>
<th>Core P47</th>
<th>Core P48</th>
<th>Core P49</th>
<th>Core P4A</th>
<th>Core P4B</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0 P44</td>
<td>P1 P44</td>
<td>P2 P44</td>
<td>P3 P44</td>
<td>P4 P44</td>
<td>P5 P44</td>
<td>P6 P44</td>
<td>P7 P44</td>
</tr>
<tr>
<td>P0 P45</td>
<td>P1 P45</td>
<td>P2 P45</td>
<td>P3 P45</td>
<td>P4 P45</td>
<td>P5 P45</td>
<td>P6 P45</td>
<td>P7 P45</td>
</tr>
<tr>
<td>P0 P46</td>
<td>P1 P46</td>
<td>P2 P46</td>
<td>P3 P46</td>
<td>P4 P46</td>
<td>P5 P46</td>
<td>P6 P46</td>
<td>P7 P46</td>
</tr>
<tr>
<td>P0 P47</td>
<td>P1 P47</td>
<td>P2 P47</td>
<td>P3 P47</td>
<td>P4 P47</td>
<td>P5 P47</td>
<td>P6 P47</td>
<td>P7 P47</td>
</tr>
<tr>
<td>P0 P49</td>
<td>P1 P49</td>
<td>P2 P49</td>
<td>P3 P49</td>
<td>P4 P49</td>
<td>P5 P49</td>
<td>P6 P49</td>
<td>P7 P49</td>
</tr>
<tr>
<td>P0 P4A</td>
<td>P1 P4A</td>
<td>P2 P4A</td>
<td>P3 P4A</td>
<td>P4 P4A</td>
<td>P5 P4A</td>
<td>P6 P4A</td>
<td>P7 P4A</td>
</tr>
</tbody>
</table>
```

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

[How can I visualize my system's NUMA topology in Red Hat Enterprise Linux?](https://access.redhat.com/site/solutions/62879)
DPDK Accelerated Virtual Switch

- Enables packet switching from the physical NIC on the host (hypervisor) to the VNF application in the Guest VM and between Guest VMs to be handled almost entirely in user-space
- Support for both DPDK physical and KVM/QEMU vhost-user ports
  - vhost-user is a QEMU feature that provides efficient virtio-net I/O between a guest VM and a user-space vswitch
- DPDK support for OVS is still under development in the upstream
  - OVS 2.4.0 Single Queue DPDK, Shipped in RHEL7.2, OSP8
  - OVS 2.5.0 Multi Queue DPDK support
    - 2.5.0 has been held up because of upstream conntrack work; for RHEL 7.3, 7.2z
- OpNFV – RH Performance Team contributing with Office of Tech, and Kernel / QE
  - VSperf group w/ consortia
  - Develop “MoonGen” (University Munich), control flows, measure loss,
  - Work on other HW packet generators – Xena, XIA, Sprient
RHEL 7.2 vhostuser – multi-queue

Upstream ovs-dpdk (2.5), Intel 40Gb XL710
64-byte frames
33.4 Million packets per second!
OVS: 8 cores (16 threads), 2 bridges, each using 4 i40e PMD threads + 4 vhostuser PMD threads
VM 4: cores (8 threads), 2 vhostuser interfaces, each using 4 virtio PMD thread
Performance with tiny frames 64 Bytes

**Kernel Networking**
- L2 Fwd **0.52** Mpps in one VM out another VM Single Queue
- L2 Fwd **0.90** Mpps Till VMs Multi-Queue
- L2 Fwd **9.05** Mpps Till OVS

**DPDK kernel bypass**
- L2 Fwd **4.04** Mpps in one Vm out another VM Single Queue
- L2 Fwd **14.21** Mpps Till VMs Multi-Queue
- L2 Fwd **14.88** Mpps Till OVS

64 Byte frames, 10G link, Theoretical limit 14.88 Million Packets Per Second (Mpps)
Scalability w/ 6 – 40Gb adapters i40e
Packets/sec Passthrough (RHEL7.1 + DPDK)

Configurations: BM/Atomic/KVM

- Boot options
  - CPU cstate=1
  - Reserve 1GB hugepages
  - isolate CPUs
- DPDK setup
  - Allocate PCI physical functions [using vfio]
  - Run a DPDK application in container (ovs-dpdk, pktgen, l2fwd, or testpmd)
    - container includes dpdk packages, access to PCI dev
    - specify which poll-mode-driver library to use (one for each network interface type)
    - specify which PCI function(s) to use
    - specify which CPUs and memory to use
- Current software versions tested:
  - dpdk-2.0.0-6
  - pktgen-dpdk-2.8.4
  - openvswitch-2.3.90-10031.gitf097013a.3
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
NFV 40G Packets/Sec DPDK (64 byte UDP)

208Mpps+ INTO KVM DPDK

NFV: Millions of Packets Per Second

RHEL 7.x, L2 Forwarding, 12 x 40Gb NICs

<table>
<thead>
<tr>
<th></th>
<th>KVM</th>
<th>Docker</th>
<th>Bare-metal</th>
<th>HW Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packets Per Second (Millions)</td>
<td>208</td>
<td>215</td>
<td>218</td>
<td>225</td>
</tr>
</tbody>
</table>
Realtime, Realtime KVM/NFV Tuned Profiles

Parents
- latency-performance

Children
- network-latency

Children/Grandchildren
- realtime
  - realtime-virtual-host
  - realtime-virtual-guest
Scheduler Latency (cyclictest)

![Graph showing latency comparison between RHEL 7.2, RHEL-RT 7.2, and RHEL-RT 7.2 (KVM). The graph highlights latency in microseconds for different performance metrics including Min, Mean, 99.9%, and Stddev. The comparison is made for cyclic test scenarios on Red Hat Enterprise Linux (RHEL) and RHEL-RT environments.]
Realtime Scheduler Latency Jitter Plot
Valuable Links

- Red Hat Performance Tuning Guide
- Red Hat Low Latency Tuning Guide
- Red Hat Virtualization Tuning Guide
- Resource Management and LXC Guide
- Comprehensive Overview of Storage Scalability in Docker
- RHEL Blog / Developer Blog
- Blog: http://www.breakage.org/ or @jeremyeder
- Reference Architectures on RH Portal
  - Ex: Deploying Oracle RAC Database 12c on RHEL 7 - Best Practices
- Key RH Summit Presentation:
  - Performance analysis & tuning of Red Hat Enterprise Linux: Part I
  - Performance analysis & tuning of Red Hat Enterprise Linux: Part II
Questions
## Performance Utility Summary

<table>
<thead>
<tr>
<th>Supportability</th>
<th>NUMA</th>
<th>Networking</th>
</tr>
</thead>
<tbody>
<tr>
<td>redhat-support-tool</td>
<td>hwloc</td>
<td>dropwatch</td>
</tr>
<tr>
<td>sos</td>
<td>Intel PCM</td>
<td>ethtool</td>
</tr>
<tr>
<td>kdump</td>
<td>numactl</td>
<td>netsniff-ng (EPEL6)</td>
</tr>
<tr>
<td>perf</td>
<td>numad</td>
<td>tcpdump</td>
</tr>
<tr>
<td>psmisc</td>
<td>numatop (01.org)</td>
<td>wireshark/tshark</td>
</tr>
<tr>
<td>strace</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sysstat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>systemd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Util-linux-ng</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pcp</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>cpupowerutils (R6)</td>
<td>blktrace</td>
</tr>
<tr>
<td></td>
<td>kernel-tools (R7)</td>
<td>iotop</td>
</tr>
<tr>
<td></td>
<td>powertop</td>
<td>iostat</td>
</tr>
<tr>
<td></td>
<td>tuna</td>
<td></td>
</tr>
<tr>
<td></td>
<td>tuned</td>
<td></td>
</tr>
</tbody>
</table>