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 – Senior Principal Performance Engineer / Network/Atomic
Joe Mario - Senior Principal Performance Engineer / RHEL / NUMA
Agenda: Performance Analysis Tuning Part I

• Part I
  • RHEL Evolution 5->6->7 – Hybrid Clouds Atomic / OSE / RHOP
  • NonUniform Memory Access (NUMA)
    • What is NUMA, RHEL Architecture, Auto-NUMA-Balance
  • Cgroups cpuset, memory, network and IO
    • Use to prevent IO from consuming 95% of memory
    • Used by RHEV w/ KVM and OSE w/ Atomic
  • System Performance/Tools
    • Tuned, Perf, and Tuna

• “Meet The Experts” - 5:45-7 PM Free - Soda/Beer/Wine
# 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>Static Hugepages</td>
<td>Transparent HugePage</td>
<td>Transparent Hugepages</td>
<td>RHEV – out-of-the-box</td>
</tr>
<tr>
<td>Ktune – on/off</td>
<td>Tuned – choose profile</td>
<td>Tuned – throughput-performance (default)</td>
<td>virt-host/guest</td>
</tr>
<tr>
<td>CPU Affinity</td>
<td>CPU Affinity (ts/numactl)</td>
<td>CPU Affinity (ts/numactl)</td>
<td>RHEL OSP – blueprints</td>
</tr>
<tr>
<td>(taskset)</td>
<td>NUMAD – uerspace tool</td>
<td>NUMAD – uerspace tool</td>
<td>Tuned, Numa pining</td>
</tr>
<tr>
<td>NUMA Pinning</td>
<td>Cgroups - irqbalance – NUMA enhanced</td>
<td>Autonuma-Balance</td>
<td>NIC – jumbo sriov</td>
</tr>
<tr>
<td>(numactl)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Irqbalance</td>
<td></td>
<td></td>
<td>RHEL Atomic Host/Atomic Enterprise</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RH OpenShift v3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cloud Forms</td>
</tr>
</tbody>
</table>

#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 – linpack, Imbench
- Memory – Imbench, McCalpin STREAM
- Disk IO – iozone, fio – SCSI, FC, iSCSI
- Filesystems – iozone, ext3/4, xfs, gfs2, gluster
- Networks – netperf – 10/40Gbit, Infiniband/RoCE, Bypass
- Bare Metal, RHEL6/7 KVM, Atomic Containers
- White box AMD/Intel, with our OEM partners

Application Performance
- Linpack MPI, HPC workloads
- AIM 7 – shared, filesystem, db, compute
- Database: DB2, Oracle 11/12, Sybase 15.x, MySQL, MariaDB, Postgres, MongoDB
- OLTP – TPC-C, TPC-VMS
- DSS – TPC-H/xDS
- Big Data – TPCx-HS, Bigbench
- SPEC cpu, jbb, sfs, virt, cloud
- SAP – SLCS, SD
- STAC = FSI (STAC-N)
- SAS mixed Analytic, SAS grid (gfs2)
Benchmark publications using Red Hat Enterprise Linux over past 24 months

Industry Benchmarks June 2014 - June 2016
(As of June 6, 2016)

- TPCx-BB: 100%
- TPCx-HS: 89%
- SPEC CPU2006: 81%
- SPECjbb2015: 67%
- SPECvirt_sc2013: 63%
- SPEC OMP2012: 39%
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
### Subsystem Analysis: ALL

```bash
# pmcollectl -s cdnm
```

<table>
<thead>
<tr>
<th>CPU</th>
<th>DISK IO</th>
<th>NET</th>
<th>MEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu</td>
<td>sys</td>
<td>inter</td>
<td>ctxsw</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>210</td>
<td>179</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1976</td>
<td>2973</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2340</td>
<td>183</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2361</td>
<td>216</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1760</td>
<td>1629</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2349</td>
<td>183</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>2361</td>
<td>216</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>1875</td>
<td>2850</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>5137</td>
<td>5383</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>2349</td>
<td>183</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1760</td>
<td>1629</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>1691</td>
<td>2526</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>1875</td>
<td>2850</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>1875</td>
<td>2850</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>16997</td>
<td>28627</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
<td>15619</td>
<td>28062</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>4495</td>
<td>7098</td>
</tr>
<tr>
<td>17</td>
<td>5</td>
<td>2380</td>
<td>187</td>
</tr>
<tr>
<td>18</td>
<td>5</td>
<td>2349</td>
<td>188</td>
</tr>
<tr>
<td>19</td>
<td>5</td>
<td>2356</td>
<td>214</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>2348</td>
<td>197</td>
</tr>
<tr>
<td>21</td>
<td>3</td>
<td>1366</td>
<td>225</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>465</td>
<td>516</td>
</tr>
<tr>
<td>23</td>
<td>0</td>
<td>236</td>
<td>205</td>
</tr>
<tr>
<td>24</td>
<td>0</td>
<td>217</td>
<td>185</td>
</tr>
</tbody>
</table>
Performance Tools - Tuned
tuned is a tool to dynamically tune Red Hat Enterprise Linux.

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

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

- New Profiles since last year
  - Realtime
  - NFV
  - RHEL Atomic Host
  - OpenShift
  - Oracle

See `man tuned-profiles` for profile definitions
Tuned: Your Custom Profiles

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
Mapping *tuned* profiles to Red Hat's product portfolio

<table>
<thead>
<tr>
<th>RHEL Desktop/Workstation</th>
<th>RHEL Server/HPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>balanced</td>
<td>throughput-performance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RHEL KVM Host, Guest</th>
<th>RHEV</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual-host/guest</td>
<td>virtual-host</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Red Hat Storage</th>
<th>RHEL OSP (compute node)</th>
</tr>
</thead>
<tbody>
<tr>
<td>rhs-high-throughput, virt</td>
<td>virtual-host</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RHEL Atomic</th>
<th>OpenShift</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic-host, atomic-guest</td>
<td>openshift-master, node</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RHEL for Real Time</th>
<th>RHEL for Real Time KVM/NFV</th>
</tr>
</thead>
<tbody>
<tr>
<td>realtime</td>
<td>realtime-virtual-host/guest</td>
</tr>
</tbody>
</table>
## 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)

64KB Read Untuned vs "Tuned" throughput-performance

Larger is better

Throughput (IOPS)

Untuned

Tuned

64KB Read
RHEL 6/7 Non-Uniform Memory (NUMA)
Typical Four-Node NUMA System

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

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

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

Node 3
- Node 3 RAM
- L3 Cache
- QPI links, IO, etc.
- Core 0
- Core 2
- Core 4
- Core 6
- Core 8
- Core...

Non-optimal numa setup

Process 1 in red, 5 threads
Add in more processes: non-optimal

**Process 1** in red, 5 threads
**Process 2** in green, 4 threads.
**Optimal numa setup**

Process 1 in green, 4 threads
Process 2 in red, 5 threads
Are my processes doing that?

• Variety of commands available to help:
  • lscpu
  • numactl
  • lstopo
  • numastat
  • ps
  • top
Tools to display CPU and Memory (NUMA)

```
# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                40
On-line CPU(s) list:   0-39
Thread(s) per core:    1
Core(s) per socket:    10
CPU socket(s):         4
NUMA node(s):          4

L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              30720K
NUMA node0 CPU(s):     0, 4, 8, 12, 16, 20, 24, 28, 32, 36
NUMA node1 CPU(s):     2, 6, 10, 14, 18, 22, 26, 30, 34, 38
NUMA node2 CPU(s):     1, 5, 9, 13, 17, 21, 25, 29, 33, 37
NUMA node3 CPU(s):     3, 7, 11, 15, 19, 23, 27, 31, 35, 39
```

cpu, core, socket, node info

The cpu numbers for each node
Tools to display CPU and Memory (NUMA)

```bash
# 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
```

### cpus & memory for each node

### Relative “node-to-node” latency costs.
Visualize CPUs via lstopo (hwloc-gui rpm)

# lstopo

NUMA
CACHE
cores
cpus
HT

PCIe
numastat shows need for NUMA management

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

<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><strong>10723</strong></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><strong>10717</strong></td>
<td>10722</td>
</tr>
<tr>
<td>10671 (qemu-kvm)</td>
<td>0</td>
<td>0</td>
<td><strong>10726</strong></td>
<td>0</td>
<td>10726</td>
</tr>
<tr>
<td>10713 (qemu-kvm)</td>
<td><strong>10733</strong></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>
```
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>
What about my processes and threads? Two ways to see “where it last ran”.

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

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

<table>
<thead>
<tr>
<th>PID</th>
<th>TID</th>
<th>PSR</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>3175391</td>
<td>3175391</td>
<td>73</td>
<td>pig</td>
</tr>
<tr>
<td>3175391</td>
<td>3175392</td>
<td>1</td>
<td>pig</td>
</tr>
<tr>
<td>3175391</td>
<td>3175393</td>
<td>25</td>
<td>pig</td>
</tr>
<tr>
<td>3175391</td>
<td>3175394</td>
<td>49</td>
<td>pig</td>
</tr>
<tr>
<td>3175391</td>
<td>3175395</td>
<td>74</td>
<td>pig</td>
</tr>
<tr>
<td>3175391</td>
<td>3175396</td>
<td>2</td>
<td>pig</td>
</tr>
<tr>
<td>3175391</td>
<td>3175397</td>
<td>26</td>
<td>pig</td>
</tr>
<tr>
<td>3175391</td>
<td>3175398</td>
<td>50</td>
<td>pig</td>
</tr>
<tr>
<td>3175391</td>
<td>3175399</td>
<td>75</td>
<td>pig</td>
</tr>
<tr>
<td>3175391</td>
<td>3175400</td>
<td>3</td>
<td>pig</td>
</tr>
</tbody>
</table>
```

"Last Ran CPU" column

2) Run “`top`”, then enter “`f`”, then select “`Last used cpu`” field
Techniques to control placement:

*numactl*:
  • Control NUMA policy for processes or shared memory:

*taskset*:
  • Retrieve or set a process's CPU affinity

*sched_getaffinity(), sched_setaffinity()*,
  • for process affinity from within program

*mbind(), get_mempolicy(), set_mempolicy()*,
  • set default NUMA memory policy for a process children.
Techniques to control placement (cont):

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

Auto-Numa-Balance kernel scheduler:
- Automatically run programs near their memory, and moves memory near the programs using it.
- Default enabled. Available in RHEL 7+
- Great video on how it works:
  - https://www.youtube.com/watch?v=mjVw_oe1hEA
Early SAP HANA benefit from Auto-Numa-Balance 25+% gain. [Recent HANA numa-aware binary closed gap.]

benchBWEMLSim - MultiProvider QueryRuntime (LOWER==BETTER)
### NUMA Nodes and Zones

<table>
<thead>
<tr>
<th>Node 0</th>
<th>Node 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>4GB DMA32 Zone</td>
<td>16MB DMA Zone</td>
</tr>
<tr>
<td>Normal Zone</td>
<td>Normal Zone</td>
</tr>
<tr>
<td>16MB DMA Zone</td>
<td>End of RAM</td>
</tr>
</tbody>
</table>

- **Node 0**: 4GB DMA32 Zone, Normal Zone, 16MB DMA Zone
- **Node 1**: Normal Zone, End of RAM
Per Node / Zone split LRU Paging Dynamics

User Allocations

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

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

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

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

● Readahead parameters
   /sys/block/<bdev>/queue/read_ahead_kb
swappiness

• Controls how aggressively the system reclaims anonymous memory versus pagecache memory:
  • Anonymous memory – swapping and freeing
  • File pages – writing if dirty and freeing
  • System V shared memory – swapping and freeing
• Default is 60
• Decrease: more aggressive reclaiming of pagecache memory
• Increase: more aggressive swapping of anonymous memory
• Can effect Numa nodes differently.
• Tuning not as necessary on RHEL7 than RHEL6 and even less than RHEL5
Memory reclaim Watermarks

Free memory list

- All of RAM
  - Do nothing
- Pages High – kswapd sleeps above High
  - kswapd reclaims memory
- Pages Low – kswapd wakes up at Low
  - Wake up kswapd and it reclaims memory
- Pages Min – all memory allocators reclaim at Min
  - User processes/kswapd reclaim memory

0
### min_free_kbytes

Directly controls the page reclaim watermarks in KB

Distributed between the Numa nodes

Defaults are higher when THP is enabled

```
# cat /proc/sys/vm/min_free_kbytes
90100
```

Node 0 DMA  
min:80 low:100kB high:120kB
Node 0 DMA32 min:15312kB low:19140kB high:22968kB
Node 0 Normal min:29600kB low:37000kB high:44400kB
Node 1 Normal min:45108kB low:56384kB high:67660kB

```
echo 180200 > /proc/sys/vm/min_free_kbytes
```

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

• Controls NUMA specific memory allocation policy
• To see current setting: cat /proc/sys/vm/zone_reclaim_mode
  • Turn ON: echo 1 > /proc/sys/vm/zone_reclaim_mode
    • Reclaim memory from local node rather than allocating from next node
  • Turn OFF: echo 0 > /proc/sys/vm/zone_reclaim_mode
    • Allocate from all nodes before reclaiming memory
• Default is set at boot time based on NUMA factor
• In Red Hat Enterprise Linux 6.6+ and 7+, the default is usually OFF – because this is better for many applications
Low-memory SPEC CPU loses huge performance with wrong zone reclaim mode setting! Several benchmarks off more than 40%.

(BTW, Don't run SPEC CPU with low memory!!)
NUMA tuning for KVM / Atomic is the same!

• Best performance is achieved if the size of the guest/container can fit into a single NUMA node.
  • In RHEL 7, auto-numa kernel scheduler will try to move guest to one node.

• Great doc with numerous examples: See the NUMA chapter in: Red Hat Virtualization Tuning and Optimization Guide
NUMA Performance – SPECjbb2005 on DL980 Westmere EX

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

8 socket, 80 cpu, 1TB mem
Red Hat Enterprise Linux Cgroups
Cgroup default mount points

**RHEL6**

```
# cat /etc/cgconfig.conf

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

**RHEL7**

```
# ls -l /cgroup
```

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

**RHEL7**

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

```
drwxr-xr-x. 2 root root 0 Mar 20 16:40 blkio
drwxr-xr-x. 2 root root 0 Mar 20 16:40 cpu,cpuacct
drwxr-xr-x. 2 root root 0 Mar 20 16:40 cpuset
drwxr-xr-x. 2 root root 0 Mar 20 16:40 devices
drwxr-xr-x. 2 root root 0 Mar 20 16:40 freezer
drwxr-xr-x. 2 root root 0 Mar 20 16:40 hugetlb
drwxr-xr-x. 3 root root 0 Mar 20 16:40 memory
drwxr-xr-x. 2 root root 0 Mar 20 16:40 net_cls
drwxr-xr-x. 2 root root 0 Mar 20 16:40 perf_event
drwxr-xr-x. 4 root root 0 Mar 20 16:40 systemd
```
Create a 2GB/4CPU subset of a 16GB/8CPU system

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

# echo 0-3 > cpuset.cpus
# runmany 20MB 110procs &
# top -d 5

top - 12:24:13 up 1:36, 4 users, load average: 22.70, 5.32, 1.79
Tasks: 315 total, 93 running, 222 sleeping, 0 stopped, 0 zombie

<table>
<thead>
<tr>
<th>Cpu</th>
<th>%us</th>
<th>%sy</th>
<th>%ni</th>
<th>%id</th>
<th>%wa</th>
<th>%hi</th>
<th>%si</th>
<th>%st</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cpu0</td>
<td>100.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Cpu1</td>
<td>100.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Cpu2</td>
<td>100.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Cpu3</td>
<td>100.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Cpu4</td>
<td>0.4%</td>
<td>0.6%</td>
<td>0.0%</td>
<td>98.8%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.2%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Cpu5</td>
<td>0.4%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>99.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.4%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Cpu6</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>100.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Cpu7</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>99.8%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.2%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>
Correct NUMA bindings

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

node0
numa_hit 1648772
numa_miss 23459
local_node 1648648
other_node 23583

node1
numa_hit 438778
numa_miss 2134520
local_node 423162
other_node 2150136

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

# numastat

node0
numa_hit 1623194
numa_miss 23459
local_node 2149354
other_node 23583

node1
numa_hit 418490
numa_miss 1098074
local_node 418490
other_node 1098074

Incorrect NUMA bindings

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

node0
numa_hit 1623318
numa_miss 23459
local_node 1623194
other_node 23583

node1
numa_hit 434106
numa_miss 1082458
local_node 418531
other_node 1098074

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

# numastat

node0
numa_hit 1623341
numa_miss 2133738
local_node 1623217
other_node 23583

node1
numa_hit 434147
numa_miss 2133738
local_node 418531
other_node 2149354
### cpu.shares default

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

---

### cpu.shares throttled

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

---

<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>
<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>20104</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>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>20106</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>4160</td>
<td>360</td>
<td>284</td>
<td>R</td>
<td>99.8</td>
<td>0.0</td>
<td>0:16.31 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>
# cat cpu.cfs_period_us
100000
# cat cpu.cfs_quota_us
-1

top - 10:11:33 up 13 days, 17:31, 11 users,  load average: 6.21, 7.78, 6.80

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

# echo 1000 > cpu.cfs_quota_us

# cat cpu.cfs_period_us
100000
# cat cpu.cfs_quota_us
-1

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

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

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

# ./memory 16G
size = 10485760000
touching 2560000 pages
Killed
# vmstat 1

```
 0  0  52224 1640116 0 3676924 0 0 0 0 202 487 0 0 100 0 0
 1  0  52224 1640116 0 3676924 0 0 0 0 162 316 0 0 100 0 0
 0  1 248532 587268 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
 1  0 885972 585404 0 156844 0 156852 754 1225 0 2 88 10 0
 0  1 1042644 587128 0 156500 0 156508 747 1146 0 2 86 12 0
 0  1 1169708 587396 0 127064 4 127836 702 1429 0 2 88 10 0
0  0  86648 1607092 0 3677020 144 0 148 0 491 1151 0 1 97 1 0
```
Cgroup OOMkils (continued)

```bash
# 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
```
C-group Dynamic resource control

Dynamic CPU Change
Oracle OLTP Workload

Transactions Per Minute

Control Group CPU Count

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

Instance 1
Instance 2
Cgroup – Application Isolation

System Level Memory Swapping

Memory Resource Management
Oracle OLTP Workload

Even though one application does not have resources and starts swapping, other applications are not affected.
Summary - Red Hat Enterprise Linux NUMA

• RHEL6 – NUMAD - With Red Hat Enterprise Linux 6.5
  • NUMAD can significantly improve performance and automate NUMA management on systems with server consolidation or replicated parallel workloads.

• RHEL7, Auto-NUMA-Balance works well for most applications out of the box!

• Use NUMAstat and NUMActl tools to measure and/or fine control your application on RHEL.

• App Developers – use perf to check for false sharing, advise padding

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

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

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

<table>
<thead>
<tr>
<th>Supportability</th>
<th>NUMA</th>
<th>Networking</th>
<th>Networking</th>
</tr>
</thead>
<tbody>
<tr>
<td>• redhat-support-tool</td>
<td>• hwloc</td>
<td>• dropwatch</td>
<td>• pcp</td>
</tr>
<tr>
<td>• sos</td>
<td>• Intel PCM</td>
<td>• ethtool</td>
<td>#redhat #rhsummit</td>
</tr>
<tr>
<td>• kdump</td>
<td>• numactl</td>
<td>• netsniff-ng (EPEL6)</td>
<td></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>

<table>
<thead>
<tr>
<th>Power/Tuning</th>
<th>Storage</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• cpupowerutils (R6)</td>
<td>• blktrace</td>
<td>#redhat #rhsunmit</td>
<td></td>
</tr>
<tr>
<td>• kernel-tools (R7)</td>
<td>• iotop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• powertop</td>
<td>• iostat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• tuna</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• tuned</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Agenda: Performance Analysis Tuning Part II

- Part II
  - Scheduler tunables
  - Transparent Hugepages, Static Hugepages 4K/2MB/1GB
  - Disk and Filesystem IO - Throughput-performance - RHS / Cloud
  - Network Performance and Latency-performance noHZ_full
  - NFV Kernel vs offload DPDK, w/ Virt, Container
  - Demo – low latency profile

- Q+A at “Meet The Experts” - Free as in Soda/Beer/Wine
LEARN. NETWORK. EXPERIENCE OPEN SOURCE.
Performance Tools - Perf
**perf list**

List counters/tracepoints available on your system

<table>
<thead>
<tr>
<th>Event Description</th>
<th>Event Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu-cycles OR cycles</td>
<td>Hardware event</td>
</tr>
<tr>
<td>instructions</td>
<td>Hardware event</td>
</tr>
<tr>
<td>cache-references</td>
<td>Hardware event</td>
</tr>
<tr>
<td>cache-misses</td>
<td>Hardware event</td>
</tr>
<tr>
<td>branch-instructions OR branches</td>
<td>Hardware event</td>
</tr>
<tr>
<td>branch-misses</td>
<td>Hardware event</td>
</tr>
<tr>
<td>cpu-clock</td>
<td>Software event</td>
</tr>
<tr>
<td>task-clock</td>
<td>Software event</td>
</tr>
<tr>
<td>page-faults OR faults</td>
<td>Software event</td>
</tr>
<tr>
<td>context-switches OR cs</td>
<td>Software event</td>
</tr>
<tr>
<td>cpu-migrations OR migrations</td>
<td>Software event</td>
</tr>
<tr>
<td>minor-faults</td>
<td>Software event</td>
</tr>
<tr>
<td>major-faults</td>
<td>Software event</td>
</tr>
</tbody>
</table>
perf top

System-wide 'top' view of busy functions

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

- Record system-wide (-a)
  - `perf record -a sleep 10`
  - `perf record -a // Hit ctrl-c when done.`
- Or record a single command
  - `perf record myapp.exe`
- Or record an existing process (-p)
  - `perf record -p <pid>`
- Or add call-chain recording (-g)
  - `perf record -g ls -rl /root`
- *Or only record specific events* (-e)
  - `perf record -e branch-misses -p <pid>`
perf report

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

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

/dev/zero
offlag=direct
# perf diff / sched

Compare 2 perf recordings

```plaintext
# perf diff
# Event 'cycles'
# Baseline  Delta     Shared Object       Symbol
# 12.88%   -12.27%   [kernel.kallsyms]   [k] _lookup_mnt
11.97%   -11.17%   systemd
4.32%    +6.43%    libdbus-1.so.3.7.4
4.06%    +4.72%    dbus-daemon
3.79%    -3.79%    libglib-2.0.so.0.3600.3
3.72%    +0.25%    [kernel.kallsyms]
```

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

```plaintext
# perf list | grep sched: | grep numa
sched:sched_move numa       [Tracepoint event]
sched:sched_stick numa      [Tracepoint event]
sched:sched_swap numa       [Tracepoint event]
```
False Sharing

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

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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>int a;</td>
<td>0</td>
</tr>
<tr>
<td>pthread_mutex_t mutex1;</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>40</td>
</tr>
<tr>
<td>long b;</td>
<td>48</td>
</tr>
<tr>
<td>long sequence_cnt;</td>
<td>56</td>
</tr>
</tbody>
</table>
Gets you contention like this:
• Can be quite painful

64 byte cache line

Node 0
- CPU
- CPU
- CPU
- CPU
- CPU
- CPU
- CPU
- CPU

Node 1
- CPU
- CPU
- CPU
- CPU
- CPU
- CPU
- CPU

Node 2
- CPU
- CPU
- CPU
- CPU
- CPU
- CPU
- CPU

Node 3
- CPU
- CPU
- CPU
- CPU
- CPU
- CPU
- CPU

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

Gets you contention like this:
• Can be quite painful
Split it up into two lines, with hot items in their own lines:

- **Hot mutex**
  - pthread_mutex_t mutex1;
  - long a;
  - long b;
  - long cold_var;
  - long sequence_cnt;

- **Hot sequence counter**
  - long pad1;
  - long pad2;
  - long pad3;
  - long pad4;
  - long pad5;
  - long pad6;
  - long pad7;

- **Cacheline 1**
- **Cacheline 2**

With padding or cold variables
Future Red Hat update to perf: “c2c data sharing” tool

This shows who is contributing to the false sharing:

- The hottest contended cachelines
- The process names, data addr, ip, pids, tids
- The node and CPU numbers they ran on,
- And how the cacheline is being accessed (read or write)
- Disassemble the binary to find the ip, and track back to the sources.
Performance Tools - Tuna
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 GUI Capabilities Updated for RHEL7