Performance Analysis and Tuning – Part 1

D. John Shakshober (Shak)
Sr Consulting Eng / Director Performance Engineering

Larry Woodman
Senior Consulting Engineer / Kernel VM

Jeremy Eder
Principal Software Engineer/ Performance Engineering
Agenda: Performance Analysis Tuning Part I

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

• Part II
  • Disk and Filesystem IO - Throughput-performance
  • Network Performance and Latency-performance
  • System Performance/Tools – perf, tuna, systemtap, performance-co-pilot

• Q & A
# Red Hat Enterprise Linux Performance Evolution

<table>
<thead>
<tr>
<th>RHEL5</th>
<th>RHEL6</th>
<th>RHEL7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Hugepages</td>
<td>Transparent Hugepages</td>
<td>Transparent Hugepages</td>
</tr>
<tr>
<td>Ktune – on/off</td>
<td>Tuned – choose profile</td>
<td>Tuned – throughput-performance (default)</td>
</tr>
<tr>
<td>CPU Affinity (taskset)</td>
<td>CPU Affinity (ts/numactl)</td>
<td>CPU Affinity (ts/numactl)</td>
</tr>
<tr>
<td>NUMA Pinning (numactl)</td>
<td>NUMAD – userspace tool</td>
<td>Autonuma-Balance</td>
</tr>
<tr>
<td>irqbalance</td>
<td>irqbalance – NUMA enhanced</td>
<td>irqbalance – NUMA enhanced</td>
</tr>
</tbody>
</table>
Industry Standard Benchmark Results 4/2014

- SPEC www.spec.org
- SPECcpu2006
- SPECvirt_sc2010, sc2013
- SPECjbb2013

- TPC www.tpc.org
- TPC-H 3 of top 6 categories
- TPC-C Top virtualization w/ IBM

- SAP SD Users
- STAC benchmark – FSI
RHEL Benchmark platform of choice 2014

Number of Benchmarks Used Intel Xeon Processor E7 v2 Launch

- Red Hat: 12
- Microsoft: 5
- SUSE: 3
- VMware: 2

Percentage of Published Results Using Red Hat Enterprise Linux

Since February 1, 2013
(As of February 20, 2014)

- SPECjbb_2013: 58%
- SPECvirt_sc2013: 71%
- SPECvirt_sc2010: 75%
- SPEC CPU2006: 77%

SPEC® is a registered trademark of the Standard Performance Evaluation Corporation. For more information about SPEC® and its benchmarks see www.spec.org.
RHEL Performance Coverage

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
- White box AMD/Intel, with our OEM partners

Application Performance

- Linpack MPI, SPEC CPU, SPECjbb 05/13
- AIM 7 – shared, filesystem, db, compute
- Database: DB2, Oracle 11/12, Sybase 15.x, MySQL, MariaDB, PostgreSQL, Mongo
- OLTP – Bare Metal, KVM, RHEV-M clusters – TPC-C/Virt
- DSS – Bare Metal, KVM, RHEV-M, IQ, TPC-H/Virt
- SPECsfs NFS
- SAP – SLCS, SD
- STAC = FSI (STAC-N)
Tuned for RHEL7
Overview and What's New
What is “tuned”?

Tuning profile delivery mechanism

Red Hat ships tuned profiles that improve performance for many workloads...hopefully yours!
Yes, but why do I care?
Tuned: Storage Performance Boost

RHEL7 RC 3.10-111 File System In Cache Performance

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

Larger is better

Throughput in MB/Sec

ext3
ext4
xfs
afs2
Tuned: Network Throughput Boost

R7 RC1 Tuned Profile Comparison - 40G Networking

- Balanced
- Throughput-performance
- Network-latency
- Network-throughput

Incorrect Binding  Correct Binding  Correct Binding (Jumbo)

Larger is better
Tuned: Network Latency Performance Boost

C-state lock improves determinism, reduces jitter

Latency (microseconds)

C6 | C3 | C1 | C0
Tuned: Updates for RHEL7

• Installed by default!
Tuned: Updates for RHEL7

• Installed by default!

• Profiles automatically set based on install type:
  • Desktop/Workstation: balanced
  • Server/HPC: throughput-performance
Tuned: Updates for RHEL7

- Re-written for maintainability and extensibility.
Tuned: Updates for RHEL7

• Re-written for maintainability and extensibility.
  • Configuration is now consolidated into a single tuned.conf file
Tuned: Updates for RHEL7

• Re-written for maintainability and extensibility.
• Configuration is now consolidated into a single tuned.conf file
• Optional hook/callout capability
Tuned: Updates for RHEL7

• Re-written for maintainability and extensibility.
  • Configuration is now consolidated into a single tuned.conf file
  • Optional hook/callout capability
  • Adds concept of Inheritance (just like httpd.conf)
Tuned: Updates for RHEL7

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

- Profiles updated for RHEL7 features and characteristics
  - Added bash-completion :-(
Tuned: Profile Inheritance

Parents
- throughput-performance
- balanced
- latency-performance

Children
- network-throughput
- desktop
- network-latency
- virtual-host
- virtual-guest
Tuned: Your Custom Profiles

Parents
- throughput-performance
- balanced
- latency-performance

Children
- network-throughput
- desktop
- network-latency

Children/Grandchildren
- Your Web Profile
- Your Database Profile
- Your Middleware Profile
RHEL “tuned” package

Available profiles:
- balanced
- desktop
- latency-performance
- myprofile
- network-latency
- network-throughput
- throughput-performance
- virtual-guest
- virtual-host

Current active profile: myprofile
RHEL7 Features
RHEL NUMA Scheduler

• RHEL6
  • numactl, numastat enhancements
  • numad – usermode tool, dynamically monitor, auto-tune

• RHEL7 Automatic numa-balancing
  • Enable / Disable
    
    # sysctl kernel.numa_balancing={0,1}
What is NUMA?

• Non Uniform Memory Access
• A result of making bigger systems more scalable by distributing system memory near individual CPUs....
• Practically all multi-socket systems have NUMA
  • Most servers have 1 NUMA node / socket
  • Recent AMD systems have 2 NUMA nodes / socket
• Keep interleave memory in BIOS off (default)
  • Else OS will see only 1-NUMA node!!!
Automatic NUMA Balancing - balanced
Per NUMA-Node Resources

- Memory zones (DMA & Normal zones)
- CPUs
- IO/DMA capacity
- Interrupt processing
- Page reclamation kernel thread (kswapd#)
- Lots of other kernel threads
NUMA Nodes and Zones

Node 0
- 4GB Normal Zone
- DMA32 Zone
- 16MB DMA Zone

Node 1
- End of RAM
- Normal Zone

64-bit
zone_reclaim_mode

• Controls NUMA specific memory allocation policy
• When set and node memory is exhausted:
  • Reclaim memory from local node rather than allocating from next node
  • Slower allocation, higher NUMA hit ratio
• When clear and node memory is exhausted:
  • Allocate from all nodes before reclaiming memory
  • Faster allocation, higher NUMA miss ratio
• Default is set at boot time based on NUMA factor
Per Node/Zone split LRU Paging Dynamics

User Allocations

- Reactivate
- Page aging
- Reclaiming
- swapout
- flush
- User deletions

anonLRU
fileLRU
ACTIVE

anonLRU
fileLRU
INACTIVE

FREE
Visualize CPUs via lstopo (hwloc rpm)

# lstopo
Tools to display CPU and Memory (NUMA)

# lscpu

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

L1d cache: 32K
L1i cache: 32K
L2 cache: 256K
L3 cache: 30720K

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

# numactl --hardware

available: 4 nodes (0-3)
node 0 cpus: 0 4 8 12 16 20 24 28 32 36
node 0 size: 65415 MB
node 0 free: 63482 MB
node 1 cpus: 2 6 10 14 18 22 26 30 34 38
node 1 size: 65536 MB
node 1 free: 63968 MB
node 2 cpus: 1 5 9 13 17 21 25 29 33 37
node 2 size: 65536 MB
node 2 free: 63897 MB
node 3 cpus: 3 7 11 15 19 23 27 31 35 39
node 3 size: 65536 MB
node 3 free: 63971 MB

cnode distances:
node   0   1   2   3
0:  10  21  21  21
1:  21  10  21  21
2:  21  21  10  21
3:  21  21  21  10
RHEL7 Autonuma-Balance Performance – SPECjbb2005

RHEL 3.10-115 RC1 Multi-instance Java peak SPECjbb2005

Multi-instance Java loads fit within 1-node

%gain vs noauto
RHEL7 Database Performance Improvements
Autonuma-Balance

Postgres Sysbench OLTP

2-socket Westmere EP 24p/48 GB

Number of threads vs. transactions per second for different configurations:
- 3.10-54 base
- 3.10-54 numa
- NumaD %
Autonuma-balance and numad aligns process memory and CPU threads within nodes

No NUMA scheduling

With NUMA Scheduling

Node 0  Node 1  Node 2  Node 3


Node 0  Node 1  Node 2  Node 3

Proc 29  Proc 19  Proc 61  Proc 37

No NUMA scheduling

With NUMA Scheduling
Some KVM NUMA Suggestions

• Don't assign extra resources to guests
  • Don't assign more memory than can be used
  • Don't make guest unnecessarily wide
    • Not much point to more VCPUs than application threads
• For best NUMA affinity and performance, the number of guest VCPUs should be $\leq$ number of physical cores per node, and guest memory $<$ available memory per node
• Guests that span nodes should consider SLIT
How to manage NUMA manually

• Research NUMA topology of each system
• Make a resource plan for each system
• Bind both CPUs and Memory
  • Might also consider devices and IRQs
• Use numactl for native jobs:
  • “numactl -N <nodes> -m <nodes> <workload>”
• Use numatune for libvirt started guests
  • Edit xml: <numatune> <memory mode="strict" nodeset="1-2"/> </numatune>
RHEL7 Java Performance Multi-Instance
Autonuma-Balance

RHEL7 Autonuma-Balance SPECjbb2005
multi-instance - bare metal + kvm

8 socket, 80 cpu, 1TB mem

bops

1 instance  2 instance  4 instance  8 instance

RHEL7 – No_NUMA
Autonuma-Balance – BM
Autonuma-Balance - KVM
numastat shows guest memory alignment

```bash
# 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>
</tbody>
</table>

Total               | 11462  | 11045  | 9672   | 10698  | 42877 |
```

unaligned

```bash
# 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>
</tbody>
</table>

Total               | 10733  | 10723  | 10740  | 10717  | 42913 |
```

aligned
RHEL Cgroups / Containers
Resource Management using cgroups

Ability to manage large system resources effectively

- Control Group (Cgroups) for CPU/Memory/Network/Disk
- Benefit: guarantee Quality of Service & dynamic resource allocation
- Ideal for managing any multi-application environment
  - From back-ups to the Cloud
C-group Dynamic resource control

Dynamic CPU Change
Oracle OLTP Workload

Control Group CPU Count

Transactions Per Minute

Instance 1
Instance 2

cgrp 1 (4), cgrp 2 (32)
cgrp 1 (32), cgrp 2 (4)
Cgroup default mount points

**RHEL6**

```bash
# cat /etc/cgconfig.conf

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

**RHEL7**

```bash
RHEL7
# ls -l /sys/fs/cgroup/
```

```bash
drwxr-x-x. 2 root root  0 Mar 20 16:40 blkio
drwxr-x-x. 2 root root  0 Mar 20 16:40 cpu
drwxr-x-x. 2 root root  0 Mar 20 16:40 cpuacct
drwxr-x-x. 2 root root  0 Mar 20 16:40 cpuset
drwxr-x-x. 2 root root  0 Mar 20 16:40 devices
drwxr-x-x. 2 root root  0 Mar 20 16:40 freezer
drwxr-x-x. 2 root root  0 Mar 20 16:40 hugetlb
```
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

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

```

top - 12:24:13 up  1:36,  4 users,  load average: 22.70, 5.32, 1.79

Tasks: 315 total, 93 running, 222 sleeping, 0 stopped, 0 zombie

| Cpu0 : 100.0%us, | 0.0%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st |
| Cpu1 : 100.0%us, | 0.0%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st |
| Cpu2 : 100.0%us, | 0.0%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st |
| Cpu3 : 100.0%us, | 0.0%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st |
| Cpu4 : 0.4%us,   | 0.6%sy, 0.0%ni, 98.8%id, 0.0%wa, 0.0%hi, 0.2%si, 0.0%st |
| Cpu5 : 0.4%us,   | 0.0%sy, 0.0%ni, 99.2%id, 0.0%wa, 0.0%hi, 0.4%si, 0.0%st |
| Cpu6 : 0.0%us,   | 0.0%sy, 0.0%ni,100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st |
| Cpu7 : 0.0%us,   | 0.0%sy, 0.0%ni, 99.8%id, 0.0%wa, 0.0%hi, 0.2%si, 0.0%st |

---

#redhat #rhsummit
Correct bindings

```bash
# echo 0 > cpuset.mems
# echo 0-3 > cpuset.cpus
# numastat
numa_hit       1648772
numa_miss      23459
local_node     1648648
other_node     23583

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

# numastat
numa_hit       2700423
numa_miss      23459
local_node     2700299
other_node     23583
```

Incorrect bindings

```bash
# echo 1 > cpuset.mems
# echo 0-3 > cpuset.cpus
# numastat
numa_hit       1623318
numa_miss      23459
local_node     1623194
other_node     23583

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

# numastat
numa_hit       1623341
numa_miss      23459
local_node     1623217
other_node     23583
```
# cat cpu.shares
1024

cpu.shares default

cpu.shares throttled

# echo 10 > cpu.shares

## Original Data

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

## Modified Data

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

cpu.cfs_quota_us unlimited

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

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

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

# echo 1000 > cpu.cfs_quota_us

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

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

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

# ./memory 16G
size = 10485760000
触及 2560000 页
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 160708 0 160776 719 1161 0 2 86 11 0
0 1 1042644 587128 0 156844 0 156852 754 1225 0 2 87 11 0
0 1 1169708 587396 0 127064 0 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 0 202 487 0 0 100 0 0
1 0  52224 1640116 0 3676924 0 0 0 162 316 0 0 100 0 0
0 1 248532 587268 0 3676948 32 196312 32 196372 912 974 1 4 88 7 0
0 1 406228 586572 0 3677308 0 157696 0 157704 624 696 0 1 87 11 0
0 1 568532 585928 0 3676864 0 162304 0 162312 722 1039 0 2 87 11 0
0 1 729300 584744 0 3676840 0 160768 0 160776 719 1161 0 2 87 11 0
1 0 885972 585404 0 3677008 0 156844 0 156852 754 1225 0 2 88 10 0
0 1 1042644 587128 0 3676784 0 156500 0 156508 747 1146 0 2 86 12 0
0 1 1169708 587396 0 3676748 0 127064 4 127836 702 1429 0 2 88 10 0
0 0  86648 1607092 0 3677020 144 0 148 0 491 1151 0 1 97 1 0
...

# dmesg
...
[506858.413341] Task in /test killed as a result of limit of /test
[506858.413342] memory: usage 1048460kB, limit 1048576kB, failcnt 295377
[506858.413343] memory+swap: usage 2097152kB, limit 2097152kB, failcnt 74
[506858.413344] kmem: usage 0kB, limit 9007199254740991kB, failcnt 0
[506858.413345] Memory cgroup stats for /test: cache:0kB rss:1048460KB rss_huge:10240KB
mapped_file:0KB swap:1048692KB inactive_anon:524372KB active_anon:524084KB inactive_file:0KB
active_file:0KB unevictable:0KB
```
Cgroup node migration

# cd /sys/fs/cgroup/cpuset/
# mkdir test1
# echo 8-15 > test1/cpuset.cpus
# echo 1 > test1/cpuset.mems
# echo 1 > test1/cpuset.memory_migrate
# mkdir test2
# echo 16-23 > test2/cpuset.cpus
# echo 2 > test2/cpuset.mems
# echo 1 > test2/cpuset.memory_migrate
# echo $$ > test2/tasks
# /tmp/memory 31 1 &

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>Node 4</th>
<th>Node 5</th>
<th>Node 6</th>
<th>Node 7</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>3244 (watch)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3477 (memory)</td>
<td>0</td>
<td>0</td>
<td>2048</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2048</td>
</tr>
<tr>
<td>3594 (watch)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Total | 0 | 1 | 2048 | 1 | 0 | 0 | 0 | 0 | 2051 |
Cgroup node migration

# echo 3477 > test1/tasks

<table>
<thead>
<tr>
<th>PID</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>3244 (watch)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3477 (memory)</td>
<td>0</td>
<td>2048</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2048</td>
</tr>
<tr>
<td>3897 (watch)</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>1</td>
</tr>
<tr>
<td>Total</td>
<td>0</td>
<td>2049</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2051</td>
</tr>
</tbody>
</table>
RHEL7 Container Scaling: Subsystems

- cgroups
  - Vertical scaling limits (task scheduler)
- SE Linux supported
- Namespaces
  - Very small impact; documented in upcoming slides.
- Bridge
  - Max ports, latency impact
- iptables
  - Latency impact, large # of rules scale poorly
- devicemapper backend
  - Large number of loopback mounts scaled poorly (fixed)
Cgroup automation with Libvirt and Docker

• Create a VM using i.e. virt-manager, apply some CPU pinning.
• Create a container with docker; apply some CPU pinning.

• Libvirt/Docker constructs cpuset cgroup rules accordingly...
Cgroup automation with Libvirt and Docker

```
# systemd-cgls
├─ machine.slice
│ └─ machine-qemu\x2d20140320\x2drhel7.scope
│   └─ 7592 /usr/libexec/qemu-kvm -name rhel7.vm
vcpu*/cpuset.cpus

<table>
<thead>
<tr>
<th>VCPU: Core</th>
<th>NUMA</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>vcpu0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>vcpu1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>vcpu2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>vcpu3</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>
```
Cgroup automation with Libvirt and Docker

```
# systemd-cgls
cpuset/docker-b3e750a48377c23271c6d636c47605708ee2a02d68ea661c242ccb7bbd8e
│  ├── 5440 /bin/bash
│  │  └── 5500 /usr/sbin/sshd
│  └── 5704 sleep 99999
# cat cpuset.mems
1
```
RHEL Scheduler
RHEL Scheduler Tunables

Implements multilevel run queues for sockets and cores (as opposed to one run queue per processor or per system)

RHEL tunables
- sched_min_granularity_ns
- sched_wakeup_granularity_ns
- sched_migration_cost
- sched_child_runs_first
- sched_latency_ns
Finer Grained Scheduler Tuning

- `/proc/sys/kernel/sched_*`
- RHEL6/7 Tuned-adm will increase quantum on par with RHEL5
  - `echo 10000000 > /proc/sys/kernel/sched_min_granularity_ns`
    - Minimal preemption granularity for CPU bound tasks. See `sched_latency_ns` for details. The default value is 4000000 (ns).
  - `echo 15000000 > /proc/sys/kernel/sched_wakeup_granularity_ns`
    - The wake-up preemption granularity. Increasing this variable reduces wake-up preemption, reducing disturbance of compute bound tasks. Lowering it improves wake-up latency and throughput for latency critical tasks, particularly when a short duty cycle load component must compete with CPU bound components. The default value is 5000000 (ns).
Load Balancing

• Scheduler tries to keep all CPUs busy by moving tasks from overloaded CPUs to idle CPUs

• Detect using “perf stat”, look for excessive “migrations”

• `/proc/sys/kernel/sched_migration_cost`
  • Amount of time after the last execution that a task is considered to be “cache hot” in migration decisions. A “hot” task is less likely to be migrated, so increasing this variable reduces task migrations. The default value is 500000 (ns).
  • If the CPU idle time is higher than expected when there are runnable processes, try reducing this value. If tasks bounce between CPUs or nodes too often, try increasing it.

• Rule of thumb – increase by 2-10x to reduce load balancing (tuned does this)

• Use 10x on large systems when many CGROUPs are actively used (ex: RHEV/KVM/RHOS)
fork() behavior

sched_child_runs_first
• Controls whether parent or child runs first
• Default is 0: parent continues before children run.
• Default is different than RHEL5

RHEL6.3 Effect of sched_migration cost on fork/exit

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

Percent improvement

usec/call default 500us
usec/call tuned 4ms
percent improvement
RHEL Hugepages/ VM Tuning

• Standard HugePages 2MB
  • Reserve/free via
    • /proc/sys/vm/nr_hugepages
    • /sys/devices/node/*/hugepages/*/nrhugepages
  • Used via hugetlbfs

• GB Hugepages 1GB
  • Reserved at boot time/no freeing
  • Used via hugetlbfs

• Transparent HugePages 2MB
  • On by default via boot args or /sys
  • Used for anonymous memory
2MB standard Hugepages

```plaintext
# echo 2000 > /proc/sys/vm/nr_hugepages
# cat /proc/meminfo
MemTotal:       16331124 kB
MemFree:        11788608 kB

HugePages_Total:    2000
HugePages_Free:     2000
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB

# ./hugeshm 1000

# cat /proc/meminfo
MemTotal:       16331124 kB
MemFree:        11788608 kB

HugePages_Total:    2000
HugePages_Free:     1000
HugePages_Rsvd:     1000
HugePages_Surp:        0
Hugepagesize:       2048 kB
```
1GB Hugepages

Boot arguments

- default_hugepagesz=1G, hugepagesz=1G, hugepages=8

# cat /proc/meminfo | more

HugePages_Total:     8
HugePages_Free:      8
HugePages_Rsvd:      0
HugePages_Surp:      0

# mount -t hugetlbfs none /mnt

# ./mmapwrite /mnt/junk 33

writing 2097152 pages of random junk to file /mnt/junk
wrote 8589934592 bytes to file /mnt/junk

# cat /proc/meminfo | more

HugePages_Total:     8
HugePages_Free:      0
HugePages_Rsvd:      0
HugePages_Surp:      0
Transparent Hugepages

echo never > /sys/kernel/mm/transparent_hugepages=never

[root@dhcp-100-19-50 code]# time ./memory 15 0
real   0m12.434s
user   0m0.936s
sys    0m11.416s

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

• Boot argument: transparent_hugepages=always (enabled by default)
• # echo always > /sys/kernel/mm/redhat_transparent_hugepage/enabled

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

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

SPEEDUP 12.4/7.0 = 1.77x, 56%
RHEL7 Performance Tuning Summary

• Use “Tuned”, “NumaD” and “Tuna” in RHEL6 and RHEL7
  • Power savings mode (performance), locked (latency)
  • Transparent Hugepages for anon memory (monitor it)
  • numabalance – Multi-instance, consider “NumaD”
  • Virtualization – virtio drivers, consider SR-IOV
• Manually Tune
  • NUMA – via numactl, monitor numastat -c pid
  • Huge Pages – static hugepages for pinned shared-memory
  • Managing VM, dirty ratio and swappiness tuning
  • Use cgroups for further resource management control
## Helpful Utilities

### Supportability
- redhat-support-tool
- sos
- kdump
- perf
- psmisc
- strace
- sysstat
- systemd
- 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

### Networking
- dropwatch
- ethtool
- netsniff-ng (EPEL6)
- tcpdump
- wireshark/tshark
Q & A