LEARN. NETWORK. EXPERIENCE OPEN SOURCE.

June 11-14, 2013
Boston, MA
Tuning Red Hat Enterprise Linux for Databases

Sanjay Rao
Principal Performance Engineer, Red Hat
June 13, 2013
Objectives of this session

- Share tuning tips
  - RHEL 6 scaling
  - Aspects of tuning
  - Tuning parameters
  - Results of the tuning
    - Bare metal
    - KVM Virtualization
- Tools
RHEL 6 is a lot more scalable but it also offers many opportunities for tuning.
What To Tune

- I/O
- Memory
- CPU
- Network
I/O Tuning – **Hardware**

- **Know Your Storage**
  - SAS or SATA? (Performance comes at a premium)
  - Fibre Channel, Ethernet or SSD?
  - Bandwidth limits (I/O characteristics for desired I/O types)

- **Multiple HBAs**
  - Device-mapper multipath
    - Provides multipathing capabilities and LUN persistence
    - Check for your storage vendors recommendations (upto 20% performance gains with correct settings)

- **How to profile your I/O subsystem**
  - Low level I/O tools – dd, iozone, dt, etc.
  - I/O representative of the database implementation
I/O Tuning – Understanding I/O Elevators

- **Deadline**
  - Two queues per device, one for read and one for writes
  - I/Os dispatched based on time spent in queue
  - Used for multi-process applications and systems running enterprise storage

- **CFQ**
  - Per process queue
  - Each process queue gets fixed time slice (based on process priority)
  - Default setting - Slow storage (SATA)

- **Noop**
  - FIFO
  - Simple I/O Merging
  - Lowest CPU Cost
  - Low latency storage and applications (Solid State Devices)
CFQ vs Deadline

1 thread per multipath device (4 devices)

4 threads per multipath device (4 devices)

% diff (CFQ vs Deadline)
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

Deadline <> Noop
Impact of I/O Elevators – DSS Workload

Comparison CFQ vs Deadline
Oracle DSS Workload (with different thread count)

Parallel degree

<table>
<thead>
<tr>
<th>Parallel degree</th>
<th>CFQ</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>47.69</td>
<td>54.01</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>58.4</td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Impact of I/O Elevators – DSS Workload
I/O Tuning – File Systems

- **Direct I/O**
  - Avoid double caching
  - Predictable performance
  - Reduce CPU overhead

- **Asynchronous I/O**
  - Eliminate synchronous I/O stall
  - Critical for I/O intensive applications
  - **Configure read ahead** (for sequential read operations)
    - Database (parameters to configure read ahead)
    - Block devices (commands – "blockdev -- getra / setra")
    - Configure device read ahead for large data loads

- **Turn off I/O barriers** (RHEL6 and enterprise storage only)
I/O Tuning – Effect of Direct I/O, Asynch I/O

OLTP Workload - 4 Socket 2 cores - 16G mem
Mid-level Fibre channel storage

Best performance with AIO and DIO enabled
I/O Tuning – Effect of read ahead during data load

Completion time for loading 30G data

DB2 v9.7 (fp4)

Completion time 42% better with device read ahead set to 1024 from 256
I/O Tuning – Database Layout

• Separate files by I/O (data, logs, undo, temp)
• **OLTP** – data files / undo / logs
  • All transactions generate logs and undo information
• **DSS** – data files / temp files
  • Merge / joins / indexes generally use temp segments
• Use low latency / high bandwidth devices for hot spots
  • Use database statistics

Linux Tool used for Identifying I/O hotspots
• `iostat -dmxz <interval>`
  • This shows I/O for all the disks that are in use
I/O Tuning – OLTP - Logs

OLTP workload - Logs on FC vs Fusion-io

Single Instance

- Transactions / Min

- 10U
- 40U
- 80U

- Logs – Fusion-io
- Logs – FC
- % diff

- 23.77
- 22.3
- 21.01
I/O Tuning – Storage (OLTP database)

OLTP workload - Fibre channel vs Fusion-io

4 database instances

Transactions / Min

FC

Fusion-io
I/O Tuning – DSS - Temp

DSS Workload - Sort-Merge table create - Time Metric - Smaller is better
Memory Tuning

- NUMA
- Huge Pages
- Manage Virtual Memory pages
  - Flushing of dirty pages
  - Swapping behavior
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
Memory Tuning – Finding NUMA layout

[root@perf30 ~]# numactl --hardware
available: 4 nodes (0-3)
node 0 cpus: 0 4 8 12 16 20 24 28 32 36 40 44 48 52 56 60
node 0 size: 32649 MB
node 0 free: 30868 MB
node 1 cpus: 1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61
node 1 size: 32768 MB
node 1 free: 29483 MB
node 2 cpus: 2 6 10 14 18 22 26 30 34 38 42 46 50 54 58 62
node 2 size: 32768 MB
node 2 free: 31082 MB
node 3 cpus: 3 7 11 15 19 23 27 31 35 39 43 47 51 55 59 63
node 3 size: 32768 MB
node 3 free: 31255 MB
node distances:
node 0 1 2 3
  0: 10 21 21 21
  1: 21 10 21 21
  2: 21 21 10 21
  3: 21 21 21 10
Memory Tuning – NUMA

- Enforce NUMA placement
  - numactl
    - CPU and memory pinning
- Taskset
  - CPU pinning
- cgroups (only in RHEL6)
  - cpusets
  - cpu and memory cgroup
- Libvirt
  - for KVM guests – CPU pinning
Memory Tuning – Effect of NUMA Tuning

OLTP workload - Multi Instance

NUMA pinning improves performance by 8.23%
Memory Tuning – NUMA - “numad”

- **What is numad?**
  - User-level daemon to automatically improve out of the box NUMA system performance
  - Added to Fedora 17
  - Added to RHEL 6.3 as tech preview
  - Not enabled by default

- **What does numad do?**
  - Monitors available system resources on a per-node basis and assigns significant consumer processes to aligned resources for optimum NUMA performance.
  - Rebalances when necessary
  - Provides pre-placement advice for the best initial process placement and resource affinity.
Memory Tuning – **Effect of “numad”**

4 KVM guest running OLTP workload

Comparison between no numa / numad / manual numa pin

NUMAD performance is as good as manual NUMA pinning
Memory Tuning – Huge Pages

- 2M pages vs 4K standard Linux page
- Virtual to physical page map is 512 times smaller
- TLB can map more physical pages, resulting in fewer misses
- Traditional Huge Pages always pinned
- Most databases support Huge pages
- 1G pages supported on newer hardware
- Transparent Huge Pages in RHEL6 (cannot be used for Database shared memory – only for process private memory)

- How to configure Huge Pages (16G)
  - echo 8192 > /proc/sys/vm/nr_hugepages
  - vi /etc/sysctl.conf (vm.nr_hugepages=8192)
Memory Tuning – Effect of huge pages

- 4 Inst – no pin
- 4 inst – no pin Hugepages
- 4 inst – numa pin Hugepages

Total Transactions / Minute

<table>
<thead>
<tr>
<th>Number of CPUs</th>
<th>10U</th>
<th>40U</th>
<th>80U</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Inst – no pin</td>
<td>13.23</td>
<td>19.32</td>
<td>23.96</td>
</tr>
<tr>
<td>4 Inst – no pin Hugepages</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 Inst – numa pin Hugepages</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

% diff – huge pgs pin Vs no huge pgs no pin
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
Tuning Memory – swappiness

- Not needed as much in RHEL6
- Controls how aggressively the system reclaims “mapped” memory:
  - Default - 60%
  - Decreasing: more aggressive reclaiming of unmapped pagecache memory
  - Increasing: more aggressive swapping of mapped memory
CPU Tuning – Power Savings / cpuspeed

- Power savings mode
  - cpuspeed off
  - performance
  - ondemand
  - powersave

How To

- `echo "performance" > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor`
- best of both worlds – cron jobs to configure the governor mode
- tuned-adm profile server-powersave (RHEL6)
Scaling governors testing with OLTP workload using Violin Memory Storage

- Powersave has lowest performance
- Performance difference between ondemand and performance goes down as system gets busier
CPU Tuning – **Effect of Power Savings - OLTP**

Scaling governors testing with OLTP workload using Violin Memory Storage

Effect of turbo is seen when ondemand and performance mode is set

### Transactions / Minute

<table>
<thead>
<tr>
<th>User Set / CPU Consumption</th>
<th>Transactions / Minute</th>
</tr>
</thead>
<tbody>
<tr>
<td>10U / 15%</td>
<td></td>
</tr>
<tr>
<td>40U / 40%</td>
<td></td>
</tr>
<tr>
<td>80U / 63%</td>
<td></td>
</tr>
<tr>
<td>10U / 76%</td>
<td></td>
</tr>
</tbody>
</table>

- **ondemand**
- **performance**
- **Bios-no power savings**
CPU Tuning – Effect of Power Savings - OLTP

DSS workload (I/O intensive)

Time Metric (Lower is better)

<table>
<thead>
<tr>
<th>Time Metric</th>
<th>performance</th>
<th>ondemand</th>
<th>powersave</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:00</td>
<td>7 12</td>
<td>7 12</td>
<td>1 15</td>
</tr>
<tr>
<td>01:26</td>
<td>5884 122884416 485900 734376</td>
<td>5884 122885024 485900 734376</td>
<td>5884 122885176 485920 734376</td>
</tr>
<tr>
<td>02:53</td>
<td>0 184848 39721 9175 37669</td>
<td>0 217766 27468 9904 42807</td>
<td>0 248283 19807 7710 37788</td>
</tr>
<tr>
<td>04:19</td>
<td>42807</td>
<td>42807</td>
<td>4190</td>
</tr>
<tr>
<td>05:46</td>
<td>42807</td>
<td>42807</td>
<td>4190</td>
</tr>
<tr>
<td>07:12</td>
<td>42807</td>
<td>42807</td>
<td>4190</td>
</tr>
<tr>
<td>08:38</td>
<td>42807</td>
<td>42807</td>
<td>4190</td>
</tr>
</tbody>
</table>

vmstat output during test:

```
7 12 5884 122884416 485900 734376 0 0 184848 39721 9175 37669 4 1 89 6 0
7 12 5884 122885024 485900 734376 0 0 217766 27468 9904 42807 4 2 87 6 0
2 0 5884 122884928 485908 734376 0 0 168496 45375 6294 27759 4 1 90 5 0
7 11 5884 122885056 485912 734372 0 0 178790 40969 9433 38140 4 1 90 5 0
1 15 5884 122885176 485920 734376 0 0 248283 19807 7710 37788 5 2 86 7 0
```
CPU Tuning – Effect of Power Savings - OLTP

Full table scans using Violin Storage

Changes in power savings on the chip. Turbo mode helps performance.
CPU Tuning – C-states

- Various states of the CPUs for power savings
- C0 through C6
- C0 – full frequency (no power savings)
- C6 (deep power down mode – maximum power savings)
- OS can tell the processors to transition between these states

How To

- Turn off power savings mode in BIOS (No OS control or Turbo mode)
  - ls /sys/devices/system/cpu/cpu0/cpuidle/state*

Linux Tool used for monitoring c-states (only for Intel)

- turbostat -i <interval>
RHEL6 Technology Innovation – Networking

- Multi-queue
- Tools to monitor dropped packets – tc, dropwatch.
- RCU adoption in stack
- Multi-CPU receive to pull in from the wire faster.
- 10GbE driver improvements.
- Data center bridging in ixbge driver.
- FcoE performance improvements throughout the stack.
Network Tuning – Databases

- Network Performance
  - Separate network for different functions (Private network for database traffic)
  - If on same network, use arp_filter to prevent ARP flux
  - `echo 1 > /proc/sys/net/ipv4/conf/all/arp_filter`

- Hardware
  - 10GigE
    - Supports RDMA w/ RHEL6 high performance networking package (ROCE)
  - Infiniband (Consider the cost factor)
  - Packet size (Jumbo frames)

- Linux Tool used for monitoring network
  - `sar -n DEV <interval>`
Network tuning – Jumbo Frames with iSCSI storage

OLTP Workload

Transactions / min

10U
40U
100U

1500MTU
9000MTU

DSS Workloads

Derived metric based on Query completion time

9000MTU
1500MTU

1500
9000

OLTP Workload

DSS Workloads
Performance Setting Framework – tuned

tuned for RHEL6

• Configure system for different performance profiles
  • laptop-ac-powersave
  • spindown-disk
  • latency-performance
  • laptop-battery-powersave
  • server-powersave
  • throughput-performance
  • desktop-powersave
  • enterprise-storage
  • Default

• Create your own custom profiles

• Can be rolled back
## Tuned Profile Summary

<table>
<thead>
<tr>
<th>Tunable</th>
<th>default</th>
<th>latency-performance</th>
<th>throughput-performance</th>
<th>enterprise-storage</th>
<th>virtual-host</th>
<th>virtual-guest</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>kernel.sched_min_granularity_ns</code></td>
<td>4ms</td>
<td>10ms</td>
<td>10ms</td>
<td>10ms</td>
<td>10ms</td>
<td>10ms</td>
</tr>
<tr>
<td><code>kernel.sched_wakeup_granularity_ns</code></td>
<td>4ms</td>
<td>15ms</td>
<td>15ms</td>
<td>15ms</td>
<td>15ms</td>
<td>15ms</td>
</tr>
<tr>
<td><code>vm.dirty_ratio</code></td>
<td>20% RAM</td>
<td>40%</td>
<td>40%</td>
<td>10%</td>
<td>40%</td>
<td></td>
</tr>
<tr>
<td><code>vm.dirty_background_ratio</code></td>
<td>10% RAM</td>
<td></td>
<td>5%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>vm.swappiness</code></td>
<td>60</td>
<td></td>
<td></td>
<td>10</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>I/O Scheduler (Elevator)</td>
<td>CFQ</td>
<td>deadline</td>
<td>deadline</td>
<td>deadline</td>
<td>deadline</td>
<td>deadline</td>
</tr>
<tr>
<td>Filesystem Barriers</td>
<td>On</td>
<td></td>
<td></td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>CPU Governor</td>
<td>ondemand</td>
<td>performance</td>
<td>performance</td>
<td>performance</td>
<td>performance</td>
<td>performance</td>
</tr>
<tr>
<td>Disk Read-ahead</td>
<td></td>
<td></td>
<td></td>
<td>4x</td>
<td>4x</td>
<td>4x</td>
</tr>
</tbody>
</table>
Tuned – **Important files / customization**

- Turning tuned on
- tuned-adm profile enterprise-storage
- Important tuned files
  - `/etc/tune-profiles` # Directory with config files
  - `/etc/tune-profiles/enterprise-storage/ktune.sh`
  - `/etc/tune-profiles/enterprise-storage/sysctl.ktune`
  - `/etc/tune-profiles/enterprise-storage/ktune.sysconfig`
Tuned – OLTP workload

I/O based workloads benefit from profiles built for storage / throughput
Tuned – BenchmarkSQL 2.3.3 on PostgreSQL 9.2

I/O based workloads benefit from profiles built for storage / throughput
Tuned – Sybase IQ – DSS Workload – 300GB database

Metric calculated using completion times of series of queries

Default | Latency-performance | Enterprise-storage | Throughput-performance

Power with RF | Power | Throughput with RF
Tuned – MongoDB – YCSB workload - Using Journals

The diagram shows the performance comparison of two MongoDB configurations:

- **TuneD-off**
- **TuneD-ES**

Operations compared are:
- Insert
- Update
- Reads

The x-axis represents the operation, and the y-axis represents the time in seconds. The diagram illustrates how each configuration performs under different workloads.
Database Performance

- Application tuning
  - Design
  - Reduce locking / waiting
  - Database tools (optimize regularly)
C-group – Resource Management and Performance

- Resource Management
  - Memory, cpus, IO, Network
  - For performance
  - For application consolidation
  - Dynamic resource allocation

- Application Isolation

- I/O Cgroups
  - At device level control the % of I/O for each Cgroup if the device is shared
  - At device level put a cap on the throughput
Cgroups – Resource management

Performance of applications can be managed by using resource control.
Cgroups – NUMA pinning

Cgroup NUMA Pinning
OLTP Workload

Transactions Per Minute

4 Instance
4 Instance Cgroup

- Instance 4
- Instance 3
- Instance 2
- Instance 1
Cgroups – Dynamic Resource Control

Dynamic CPU Change in the C-Groups
OLTP Workload

Transactions Per Minute

Control Group CPU Count

cgrp 1 (4), cgrp 2 (32)
cgrp 1 (32), cgrp 2 (4)
Cgroups – Application Isolation

Instance 1 was throttled to show that swapping within a C-group does not affect the performance of applications running in other C-groups.

Swapping activity during application isolation

Data was captured during the run.
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
• **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

• **How To**
  • Configure I/O - Cache per disk in qemu command line or libvirt
Virt Tuning – Effect of I/O Cache Settings

OLTP workload

The overhead of cache is more pronounced with multiple guests

Configurable per device:
- Virt-Manager - drop-down option under “Advanced Options”
- Libvirt xml file - driver name='qemu' type='raw' cache='writethrough' io='native'
Virt Tuning – Using NUMA

4 Virtual Machines running OLTP workload

- no NUMA: 7.54
- Manual Pin: 7.05
- NUMAD: 7.05

Trans / min normalized to 100
RHEV – Migration

Migration tuning – configure migration bandwidth to facilitate migration

Configure – migration_max_bandwidth = <Value> in /etc/vdsm/vdsm.conf
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)

Lowest network latency from virtual machine using SR-IOV
Performance Monitoring Tools

- Monitoring tools
  - top, vmstat, ps, iostat, netstat, sar, perf, turbostat
- Kernel tools
  - /proc, sysctl, AltSysRq
- Networking
  - ethtool, ifconfig
- Profiling
  - oprofile, strace, ltrace, systemtap, perf
Performance Monitoring Tool – perf

- Performance analysis tool
  - perf top (dynamic)
  - perf record / report (save and replay)
  - perf stat <command> (analyze a particular workload)
Performance Monitoring Tool – `perf top`

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>oracle</td>
<td>5.25%</td>
</tr>
<tr>
<td>oracle</td>
<td>2.06%</td>
</tr>
<tr>
<td>oracle</td>
<td>1.95%</td>
</tr>
<tr>
<td>oracle</td>
<td>1.77%</td>
</tr>
<tr>
<td>oracle</td>
<td>1.65%</td>
</tr>
<tr>
<td>oracle</td>
<td>1.47%</td>
</tr>
<tr>
<td>oracle</td>
<td>1.16%</td>
</tr>
<tr>
<td>oracle</td>
<td>1.13%</td>
</tr>
<tr>
<td>oracle</td>
<td>1.11%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.79%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.78%</td>
</tr>
<tr>
<td>[Kernel]</td>
<td>0.75%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.72%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.65%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.58%</td>
</tr>
<tr>
<td>[Kernel]</td>
<td>0.58%</td>
</tr>
<tr>
<td>libclntsh.so.11.1</td>
<td>0.56%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.55%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.52%</td>
</tr>
<tr>
<td>[Kernel]</td>
<td>0.50%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.50%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.48%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.48%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.46%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.46%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.44%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.44%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.43%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.43%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.43%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.42%</td>
</tr>
<tr>
<td>oracle</td>
<td>0.42%</td>
</tr>
</tbody>
</table>

#redhat #rhsummit
## Performance Monitoring Tool – `perf record / report`

<table>
<thead>
<tr>
<th>Event</th>
<th>Percentage</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5.74%</td>
<td>oracle oracle</td>
</tr>
<tr>
<td></td>
<td>2.11%</td>
<td>oracle oracle</td>
</tr>
<tr>
<td></td>
<td>2.10%</td>
<td>oracle oracle</td>
</tr>
<tr>
<td></td>
<td>1.74%</td>
<td>oracle oracle</td>
</tr>
<tr>
<td></td>
<td>1.73%</td>
<td>oracle oracle</td>
</tr>
<tr>
<td></td>
<td>1.52%</td>
<td>oracle oracle</td>
</tr>
<tr>
<td></td>
<td>1.22%</td>
<td>oracle oracle</td>
</tr>
<tr>
<td></td>
<td>1.20%</td>
<td>oracle oracle</td>
</tr>
<tr>
<td></td>
<td>1.15%</td>
<td>oracle oracle</td>
</tr>
<tr>
<td></td>
<td>0.79%</td>
<td>oracle oracle</td>
</tr>
<tr>
<td></td>
<td>0.75%</td>
<td>oracle oracle</td>
</tr>
<tr>
<td></td>
<td>0.74%</td>
<td>oracle oracle</td>
</tr>
<tr>
<td></td>
<td>0.65%</td>
<td>oracle oracle</td>
</tr>
<tr>
<td></td>
<td>0.59%</td>
<td>oracle oracle</td>
</tr>
</tbody>
</table>

### Runas Example
- `runas Toltpb.ex libclntsh.so.11.1`

Other events include:
- `kcbgtcr`
- `ktrexc`
- `kcbgc"ur`
- `_intel_new_memset`
- `_intel_new_mymcpy`
- `kdxlr2s`
- `kcbgtcrf`
- `kdxb"rs`
- `opiexe`
- `opipls`
- `kslfre`
- `ktbgfi`
- `ksslgetl`
- `kcbbsacc`
- `ttcacr`
- `kduovw`
- `kernel.kallsyms`
- `kcrfw_redo_gen`
- `ktuchg2`
- `qerixStart`
- `ksl_get_shared_latch`
- `kcbget`
- `kssadf numa_intl`
- `kcb_commit_main`
- `ktichg`
- `kszgtlctx`
- `dkcmp1`
- `kpobii`
- `kdiins0`
- `kdudcp`
- `kdimodnu0`
- `kcbget`
- `kernel.kallsyms`
- `radix_tree_lookup_slot`
Performance Monitoring Tool – `perf stat`

- `perf stat <command>`
  - monitors any workload and collects variety of statistics
  - can monitor specific events for any workload with `-e` flag ("perf list" give list of events)

"perf stat" - with regular 4k pages

```
oracle@perf30 ~/oast/home> perf stat ./database_workload_command

Performance counter stats for './database_workload_command'

  1198816.385221 task-clock                        #    2.690 CPUs utilized
     24,468,186 context-switches                  #    0.020 M/sec
     3,603,875 CPU-migrations                    #    0.003 M/sec
          282,197 page-faults                     #    0.000 M/sec
      2,589,984,107,267 cycles                              #    2.160 GHz                     [83.36%]
     2,052,981,463,592 stalled-cycles-frontend     #   79.27% frontend cycles idle    [83.41%]
     1,447,156,041,144 stalled-cycles-backend    #   55.88% backend  cycles idle    [66.62%]
        988,260,844,982 instructions                       #    0.38  insns per cycle
                195,178,277,195 branches                          #  162.809 M/sec                   [83.33%]
       14,063,695,242 branch-misses                   #    7.21% of all branches     [83.29%]

445.726643364 seconds time elapsed
```
```
1223064.068933 task-clock  # 2.726 CPUs utilized
25,521,110 context-switches # 0.021 M/sec
4,242,520 CPU-migrations  # 0.003 M/sec
151,366 page-faults       # 0.000 M/sec

2,640,419,666,995 cycles  # 2.159 GHz  [83.35%]
2,085,237,230,532 stalled-cycles-backend # 78.97% frontend cycles idle [83.33%]
1,459,622,166,670 stalled-cycles-backend # 55.28% backend cycles idle [66.68%]
1,020,193,451,957 instructions      # 0.39 insns per cycle
201,608,008,922 branches        # 2.04 stalled cycles per insn [83.32%]
14,310,983,194 branch-misses    # 164.838 M/sec  [83.36%]
14,310,983,194 branch-misses    # 7.10% of all branches  [83.29%]

448.643139666 seconds time elapsed
```
Performance Monitoring Tool – *sar*

Output of “*sar -N DEV 3*”
For a DSS workload running on iSCSI storage using different MTUs

### 1500 MTU

<table>
<thead>
<tr>
<th>Time</th>
<th>Interface</th>
<th>rxpck/s</th>
<th>txpck/s</th>
<th>rxkB/s</th>
<th>txkB/s</th>
<th>rxcmp/s</th>
<th>txcmp/s</th>
<th>rxmcst/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>01:40:08 PM</td>
<td>eth0</td>
<td>0.34</td>
<td>0.34</td>
<td>0.02</td>
<td>0.02</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>01:40:11 PM</td>
<td>eth5</td>
<td>135016.78</td>
<td>19107.72</td>
<td>199178.19</td>
<td>1338.53</td>
<td>0.00</td>
<td>0.00</td>
<td>0.34</td>
</tr>
<tr>
<td>01:40:14 PM</td>
<td>eth0</td>
<td>0.66</td>
<td>0.00</td>
<td>0.05</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.66</td>
</tr>
<tr>
<td>01:40:17 PM</td>
<td>eth5</td>
<td>133676.74</td>
<td>18911.30</td>
<td>197199.84</td>
<td>1310.25</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

### 9000 MTU

<table>
<thead>
<tr>
<th>Time</th>
<th>Interface</th>
<th>rxpck/s</th>
<th>txpck/s</th>
<th>rxkB/s</th>
<th>txkB/s</th>
<th>rxcmp/s</th>
<th>txcmp/s</th>
<th>rxmcst/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>06:58:43 PM</td>
<td>eth0</td>
<td>0.91</td>
<td>0.00</td>
<td>0.07</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.91</td>
</tr>
<tr>
<td>06:58:46 PM</td>
<td>eth5</td>
<td>104816.36</td>
<td>48617.27</td>
<td>900444.38</td>
<td>3431.15</td>
<td>0.00</td>
<td>0.00</td>
<td>0.91</td>
</tr>
<tr>
<td>06:58:49 PM</td>
<td>eth0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
</tr>
<tr>
<td>06:58:52 PM</td>
<td>eth5</td>
<td>118269.80</td>
<td>54965.84</td>
<td>101615.64</td>
<td>3867.91</td>
<td>0.00</td>
<td>0.00</td>
<td>0.98</td>
</tr>
<tr>
<td>06:58:55 PM</td>
<td>eth0</td>
<td>0.94</td>
<td>0.00</td>
<td>0.06</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.47</td>
</tr>
<tr>
<td>06:58:55 PM</td>
<td>eth5</td>
<td>115853.05</td>
<td>53515.49</td>
<td>995087.67</td>
<td>3766.28</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
### Performance Monitoring Tool – `vmstat`

Output of “vmstat -n 3”

<table>
<thead>
<tr>
<th>Procs</th>
<th>-------memory--------</th>
<th>---swap---</th>
<th>-----io----</th>
<th>--system--</th>
<th>-----cpu-----</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>swpd  free  buff  cache</td>
<td>si  so  bi  bo  in  cs  us  sy  id  wa  st</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25 1</td>
<td>39748 1102508 170976 23568380</td>
<td>0 0 7952 163312 40235 97711 81 11 7 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39 4</td>
<td>39748 1081552 171036 23568948</td>
<td>0 0 8150 162642 40035 97585 82 11 6 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>54 0</td>
<td>39748 1071600 171064 23569452</td>
<td>0 0 7498 166835 40494 97413 82 12 6 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 0</td>
<td>39748 1077892 171104 23569980</td>
<td>0 0 6841 159781 40150 95170 83 12 5 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49 2</td>
<td>39748 1139520 171128 23570568</td>
<td>0 0 5950 138597 40117 94040 85 12 4 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46 1</td>
<td>39748 1192436 171144 23571136</td>
<td>0 0 5895 139294 40487 94423 84 12 4 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40 3</td>
<td>39748 1213212 171168 23571660</td>
<td>0 0 5906 136871 40401 94313 84 11 4 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50 2</td>
<td>39748 1210840 171176 23572212</td>
<td>0 0 5890 135288 40744 95360 84 12 4 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>52 1</td>
<td>39748 1090252 171200 23572732</td>
<td>0 0 7866 174045 39702 97930 80 11 8 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46 3</td>
<td>39748 1082532 171244 23573300</td>
<td>0 0 7217 174223 40516 94969 82 11 5 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>59 3</td>
<td>39748 1129396 171268 23573892</td>
<td>0 0 5682 218917 41571 94576 84 12 4 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46 7</td>
<td>39748 1159372 171284 23574428</td>
<td>0 0 5488 357287 45871 96181 83 12 4 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46 3</td>
<td>39748 1196788 171328 23574912</td>
<td>0 0 5456 257984 45617 97021 84 12 3 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>51 0</td>
<td>39748 1199880 171336 23575584</td>
<td>0 0 5518 161104 41572 96639 84 12 4 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42 3</td>
<td>39748 1198720 171352 23576148</td>
<td>0 0 5440 159580 41191 95308 85 11 3 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Performance Monitoring Tool – iostat

#### Output of “iostat -dmxz 3”

<table>
<thead>
<tr>
<th>Device</th>
<th>rrqm/s</th>
<th>wrqm/s</th>
<th>r/s</th>
<th>w/s</th>
<th>rMB/s</th>
<th>wMB/s</th>
<th>avgrq-sz</th>
<th>avgqu-sz</th>
<th>await</th>
<th>svctm</th>
<th>%util</th>
</tr>
</thead>
<tbody>
<tr>
<td>sda</td>
<td>0.00</td>
<td>25.20</td>
<td>0.00</td>
<td>2.40</td>
<td>0.00</td>
<td>0.11</td>
<td>90.67</td>
<td>0.04</td>
<td>17.50</td>
<td>11.50</td>
<td>2.76</td>
</tr>
<tr>
<td>dm-0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>8.00</td>
<td>0.05</td>
<td>47.00</td>
<td>15.20</td>
<td>1.52</td>
</tr>
<tr>
<td>dm-2</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>26.20</td>
<td>0.00</td>
<td>0.10</td>
<td>8.00</td>
<td>0.43</td>
<td>16.43</td>
<td>0.47</td>
<td>1.24</td>
</tr>
<tr>
<td>fioa</td>
<td>0.00</td>
<td>41.80</td>
<td>1057.60</td>
<td>3747.60</td>
<td>28.74</td>
<td>114.75</td>
<td>61.16</td>
<td>1.72</td>
<td>0.36</td>
<td>0.16</td>
<td>76.88</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Device</th>
<th>rrqm/s</th>
<th>wrqm/s</th>
<th>r/s</th>
<th>w/s</th>
<th>rMB/s</th>
<th>wMB/s</th>
<th>avgrq-sz</th>
<th>avgqu-sz</th>
<th>await</th>
<th>svctm</th>
<th>%util</th>
</tr>
</thead>
<tbody>
<tr>
<td>sda</td>
<td>0.00</td>
<td>2.99</td>
<td>0.00</td>
<td>3.19</td>
<td>0.00</td>
<td>0.02</td>
<td>15.00</td>
<td>0.01</td>
<td>4.50</td>
<td>4.44</td>
<td>1.42</td>
</tr>
<tr>
<td>dm-2</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>5.99</td>
<td>0.00</td>
<td>0.02</td>
<td>8.00</td>
<td>0.01</td>
<td>2.43</td>
<td>2.37</td>
<td>1.42</td>
</tr>
<tr>
<td>fioa</td>
<td>0.00</td>
<td>32.93</td>
<td>950.70</td>
<td>3771.46</td>
<td>25.33</td>
<td>127.18</td>
<td>66.14</td>
<td>1.77</td>
<td>0.38</td>
<td>0.16</td>
<td>76.57</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Device</th>
<th>rrqm/s</th>
<th>wrqm/s</th>
<th>r/s</th>
<th>w/s</th>
<th>rMB/s</th>
<th>wMB/s</th>
<th>avgrq-sz</th>
<th>avgqu-sz</th>
<th>await</th>
<th>svctm</th>
<th>%util</th>
</tr>
</thead>
<tbody>
<tr>
<td>sda</td>
<td>0.00</td>
<td>22.80</td>
<td>0.00</td>
<td>1.60</td>
<td>0.00</td>
<td>0.09</td>
<td>121.00</td>
<td>0.01</td>
<td>6.25</td>
<td>6.12</td>
<td>0.98</td>
</tr>
<tr>
<td>dm-2</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>24.20</td>
<td>0.00</td>
<td>0.09</td>
<td>8.00</td>
<td>0.11</td>
<td>4.69</td>
<td>0.40</td>
<td>0.98</td>
</tr>
<tr>
<td>fioa</td>
<td>0.00</td>
<td>40.00</td>
<td>915.00</td>
<td>3868.60</td>
<td>24.10</td>
<td>118.31</td>
<td>60.97</td>
<td>1.63</td>
<td>0.34</td>
<td>0.16</td>
<td>75.34</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Device</th>
<th>rrqm/s</th>
<th>wrqm/s</th>
<th>r/s</th>
<th>w/s</th>
<th>rMB/s</th>
<th>wMB/s</th>
<th>avgrq-sz</th>
<th>avgqu-sz</th>
<th>await</th>
<th>svctm</th>
<th>%util</th>
</tr>
</thead>
<tbody>
<tr>
<td>sda</td>
<td>0.00</td>
<td>54.20</td>
<td>0.00</td>
<td>1.60</td>
<td>0.00</td>
<td>0.22</td>
<td>278.00</td>
<td>0.01</td>
<td>6.00</td>
<td>5.00</td>
<td>0.80</td>
</tr>
<tr>
<td>dm-2</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>55.60</td>
<td>0.00</td>
<td>0.22</td>
<td>8.00</td>
<td>0.24</td>
<td>4.26</td>
<td>0.14</td>
<td>0.80</td>
</tr>
<tr>
<td>fioa</td>
<td>0.00</td>
<td>39.80</td>
<td>862.00</td>
<td>3800.60</td>
<td>21.93</td>
<td>131.67</td>
<td>67.47</td>
<td>1.72</td>
<td>0.37</td>
<td>0.16</td>
<td>75.96</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Device</th>
<th>rrqm/s</th>
<th>wrqm/s</th>
<th>r/s</th>
<th>w/s</th>
<th>rMB/s</th>
<th>wMB/s</th>
<th>avgrq-sz</th>
<th>avgqu-sz</th>
<th>await</th>
<th>svctm</th>
<th>%util</th>
</tr>
</thead>
<tbody>
<tr>
<td>sda</td>
<td>0.00</td>
<td>2.40</td>
<td>0.00</td>
<td>0.80</td>
<td>0.00</td>
<td>0.01</td>
<td>30.00</td>
<td>0.01</td>
<td>6.75</td>
<td>6.75</td>
<td>0.54</td>
</tr>
<tr>
<td>dm-2</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>8.00</td>
<td>0.00</td>
<td>0.01</td>
<td>8.00</td>
<td>0.01</td>
<td>1.80</td>
<td>1.80</td>
<td>0.54</td>
</tr>
<tr>
<td>fioa</td>
<td>0.00</td>
<td>36.00</td>
<td>811.20</td>
<td>3720.80</td>
<td>20.74</td>
<td>116.78</td>
<td>62.15</td>
<td>1.56</td>
<td>0.34</td>
<td>0.16</td>
<td>72.72</td>
</tr>
</tbody>
</table>
Performance Monitoring Tool – turbostats (Intel only)

The tool is found in cpupowerutils.x86_64 in RHEL6.4
Wrap up – **Bare Metal**

- I/O
  - Choose the right elevator
  - Eliminated hot spots
  - Direct I/O or Asynchronous I/O
  - Virtualization – Caching

- Memory
  - NUMA
  - Huge Pages
  - Swapping
  - Managing Caches

- RHEL has many tools to help with debugging / tuning
Wrap Up – **Bare Metal**

- **CPU**
  - Check cpuspeed settings
- **Network**
  - Separate networks
  - arp_filter
  - Packet size

**New Tools**

- tuned
- perf options - top, stats, record, report
- turbostats (Intel only)
Wrap Up – Virtualization

- VirtIO drivers
- aio (native)
- NUMA
- Cache options (none, writethrough)
- Network (vhost-net)
Stay connected through the Red Hat Customer Portal

Maximizing Oracle Database Efficiency for RHEL

Improving OpenJDK and Oracle JDK Garbage Collection Performance

Watch video

Review Tech brief

access.redhat.com