High Performance I/O with NUMA Systems in Linux

Lance Shelton
How prevalent is NUMA?

- All major vendors
  - HP, Dell, IBM, Cisco, SuperMicro, Hitachi, etc.
- As small as 1U
- 2, 4, and 8 socket systems
- 2 to 10 cores per socket
- Number of cores doubles with HyperThreading
  \[ 8 \times 10 \times 2 = 160 \text{ CPU cores} \]

**NUMA is mainstream.**
Why use large NUMA servers?

▸ NUMA is key to the current evolution of performance
  • Storage rates are drastically increasing
  • Processor speeds are stagnant
  • Multi-core cannot scale infinitely without NUMA

▸ NUMA meets the needs of the application
  • No partitioning is required
  • Faster than clustering multiple servers
  • Works well in combination with scale out
Nodes, Sockets, Cores, Threads

Memory + Socket = NUMA Node

Cores + Threads
Non-Uniform I/O Access

I/O = PCIe
- Internal flash
- Host bus adapter to storage network
2, 4, and 8 Socket Servers

# numactl --hardware

<table>
<thead>
<tr>
<th>node</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>10</td>
<td>12</td>
<td>17</td>
<td>17</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>1:</td>
<td>12</td>
<td>10</td>
<td>17</td>
<td>17</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>2:</td>
<td>17</td>
<td>17</td>
<td>10</td>
<td>12</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>3:</td>
<td>17</td>
<td>17</td>
<td>12</td>
<td>10</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>4:</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>10</td>
<td>12</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>5:</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>12</td>
<td>10</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>6:</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>17</td>
<td>17</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>7:</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>17</td>
<td>17</td>
<td>12</td>
<td>10</td>
</tr>
</tbody>
</table>
Enterprise Storage

Storage  Database Servers  Clients
PCIe Local Nodes

Intel Nehalem

Intel Sandy Bridge
# cat /sys/devices/pci0000:50/0000:50:09.0/numa_node
0

Only one local node is presented to the OS
PCle Local Node?

# cat /sys/devices/pci0000:50/0000:50:09.0/numa_node
-1

Not detected? Update the BIOS.
Localizing I/O from Storage

- 50-100% performance improvement vs. non-NUMA aware volume placement
Localizing I/O – High Availability

- Node locality of volumes must still be maintained
Localizing I/O – HA Failover

- Node locality maintained
- All available ports are still used
Localizing I/O – Components

- HBA placement
- Interrupt affinity
- Kernel thread affinity
- Application affinity

Analyze everything in the data path.
Localize all components to the HBA local nodes.

Reduces latency and improves caching
Discovering Device Locality

- Devices associated with each volume
  - `multipath -ll`
  - `ls -d /sys/bus/pci/devices/*/host*`

- Device location
  - `dmidecode -t slot`
  - `lspci -tv`

- NUMA node of devices
  - `/sys/devices/pci/*/numa_node`
Pinning Interrupts

- Pin IRQs to the local node
- Distribute IRQs between cores in the local node

```
# grep [driver] /proc/interrupts
[ num]: ...
# cat /proc/irq/[num]/node
[node]
# echo [CPU mask] > /proc/irq/[num]/smp_affinity
```

~100% improvement

Reduces latency & CPU contention, improves caching
Persistently Pinning Interrupts

- irqbalance: run-time load balancing
  - Mixed results
  - Better with RHEL 6.4 + kernel update + Sandy Bridge
- Customized init service to set affinity on boot
  - Best results for a known application
Driver Kernel Thread Affinity

- Sometimes handled by Linux drivers
- Verify and adjust

# taskset -p -c [cpumask] [pid]

pid [pid]'s current affinity list: 0-159
pid [pid]'s new affinity list: 20-29
Block Device Tuning

```bash
# echo noop > /sys/block/[device]/queue/scheduler
  • 10% improvement
# echo 0 > /sys/block/[device]/queue/add_random
  • 10% improvement
# echo 2 > /sys/block/[device]/queue/rq_affinity
  • 5% improvement
# echo [0 or 1] > /sys/block/[device]/queue/rotational
  • Mixed results
```

Reduces latency and CPU utilization
Persistent Block Device Tuning

- **udev rules (/etc/udev/rules.d/)**
  - I/O devices
    
    ```
    ACTION="add|change", SUBSYSTEM="block", ATTR{device/vendor}="FUSIONIO", ATTR{queue/scheduler} ="noop", ATTR{queue/rq_affinity}="2", ATTR{queue/add_random}="0"
    ```

  - devices with I/O slave devices (DM multipath)
    
    ```
    ACTION="add|change", KERNEL="dm-*", PROGRAM="/bin/bash -c 'cat /sys/block/$name/slaves/*/device/vendor | grep FUSIONIO'", ATTR{queue/scheduler}="noop", ATTR{queue/rq_affinity}="2", ATTR{queue/add_random}="0"
    ```

Note: udev rules may only rely on sysfs parameters that are available at the time the device is created.
Power/Performance Tuning

- Disable c-states in the BIOS
- Disable c-states in the boot loader (grub)
  - intel_idle.max_cstate=0
  - processor.max_cstate=0

- 10% improvement

Keeping processors in active states reduces latency
Application Tuning

- OS-provided tools
  - taskset
  - cgroups
  - numad

- Application-specific settings
  - Oracle
    - _enable_NUMA_support
Benchmarking Performance

- Test thread affinity
  - FIO
    - cpus_allowed
    - numa_cpu_nodes
    - numa_mem_policy
  - Oracle Orion and others
    - taskset -c [cpulist] [test command]

- Measuring performance
  - iostat
  - Application-specific
8 Socket Total Performance Gains

May 10, 2013
High Performance I/O with NUMA Servers
## Tools for NUMA Tuning

<table>
<thead>
<tr>
<th>numactl</th>
<th>cgroups</th>
</tr>
</thead>
<tbody>
<tr>
<td>taskset</td>
<td>lstopo</td>
</tr>
<tr>
<td>dmidecode</td>
<td>sysfs</td>
</tr>
<tr>
<td>irqbalance</td>
<td>numad</td>
</tr>
<tr>
<td>top</td>
<td>numatop</td>
</tr>
<tr>
<td>htop</td>
<td>tuna</td>
</tr>
<tr>
<td>irqstat</td>
<td>tuned-adm</td>
</tr>
</tbody>
</table>
top: terminal is not big enough

```
root@RHEL980:~ root@RHEL980:~ ssh...
top - 13:31:23 up 1 day, 5:17, 2 users, load average: 3.66, 0.96, 0.42
Tasks: 3897 total, 1 running, 3896 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.2%us, 1.4%sy, 0.0%ni, 95.7%id, 0.5%wa, 0.0%hi, 2.2%si, 0.0%st
Mem: 264521876k total, 118689448k used, 145832428k free, 243928k buffers
Swap: 20971504k total, 0k used, 20971504k free, 2139856k cached

Sorry, terminal is not big enough

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>124807</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>824m</td>
<td>59m</td>
<td>48m</td>
<td>S</td>
<td>5.1</td>
<td>0.0</td>
<td>1:47:06</td>
<td>fio</td>
</tr>
<tr>
<td>89</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>5.1</td>
<td>0.0</td>
<td>0:46:11</td>
<td>ksoftirqd/21</td>
</tr>
<tr>
<td>85</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>4.5</td>
<td>0.0</td>
<td>0:36:00</td>
<td>ksoftirqd/20</td>
</tr>
<tr>
<td>405</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>4.5</td>
<td>0.0</td>
<td>0:37:30</td>
<td>ksoftirqd/100</td>
</tr>
<tr>
<td>421</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>4.2</td>
<td>0.0</td>
<td>0:12:97</td>
<td>ksoftirqd/104</td>
</tr>
<tr>
<td>124743</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>17964</td>
<td>4232</td>
<td>956</td>
<td>R</td>
<td>3.9</td>
<td>0.0</td>
<td>0:02:39</td>
<td>top</td>
</tr>
<tr>
<td>121</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>2.6</td>
<td>0.0</td>
<td>0:10:04</td>
<td>ksoftirqd/29</td>
</tr>
<tr>
<td>441</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>2.6</td>
<td>0.0</td>
<td>0:10:26</td>
<td>ksoftirqd/109</td>
</tr>
<tr>
<td>101</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>2.2</td>
<td>0.0</td>
<td>0:13:09</td>
<td>ksoftirqd/24</td>
</tr>
<tr>
<td>113</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>1.3</td>
<td>0.0</td>
<td>0:16:79</td>
<td>ksoftirqd/27</td>
</tr>
<tr>
<td>10953</td>
<td>oracle</td>
<td>20</td>
<td>0</td>
<td>1388m</td>
<td>135m</td>
<td>114m</td>
<td>S</td>
<td>1.3</td>
<td>0.1</td>
<td>24:00:31</td>
<td>oracle</td>
</tr>
<tr>
<td>400</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>1.0</td>
<td>0.0</td>
<td>0:12:80</td>
<td>ksoftirqd/101</td>
</tr>
<tr>
<td>413</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>1.0</td>
<td>0.0</td>
<td>0:24:10</td>
<td>ksoftirqd/102</td>
</tr>
<tr>
<td>425</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>1.0</td>
<td>0.0</td>
<td>0:05:92</td>
<td>ksoftirqd/105</td>
</tr>
<tr>
<td>437</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>1.0</td>
<td>0.0</td>
<td>0:08:67</td>
<td>ksoftirqd/108</td>
</tr>
<tr>
<td>10945</td>
<td>oracle</td>
<td>-2</td>
<td>0</td>
<td>1365m</td>
<td>14m</td>
<td>12m</td>
<td>S</td>
<td>1.0</td>
<td>0.0</td>
<td>6:36:87</td>
<td>oracle</td>
</tr>
<tr>
<td>97</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.6</td>
<td>0.0</td>
<td>0:31:75</td>
<td>ksoftirqd/23</td>
</tr>
<tr>
<td>117</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.6</td>
<td>0.0</td>
<td>0:08:93</td>
<td>ksoftirqd/28</td>
</tr>
</tbody>
</table>
```

(top in RHEL 6.4)
mpstat with 160 cores

```plaintext
[root@RHEL980 ~]# mpstat -I ALL
Linux 2.6.32-358.el6.x86_64 (RHEL980.int.fusionio.com) 04/13/2013 _x86_64_ (16
01:57:00 PM CPU intr/s
01:57:00 PM all 14284.04
01:57:00 PM CPU 0/s 1/s 3/s 4/s 8/s 9/s 12
    103/s  104/s  105/s  106/s  107/s  108/s  109/s  110/s
    120/s  121/s  122/s  123/s  124/s  125/s  126/s  127/s  12
    s    138/s  139/s  140/s  141/s  NMI/s  LOC/s  SPU/s  PMI/s
01:57:00 PM 0  0.08  0.01  0.00  0.01  0.00  0.01  0.00  0.00  0.00  0.01  19.33  0.00  0.01
01:57:00 PM 1  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  131.68  0.00  0.00  0.00  0.0
01:57:00 PM 2  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  4.17  0.00  0.01
```
/proc/interrupts with 160 cores
top: added NUMA support

May 10, 2013
High Performance I/O with NUMA Servers

https://gitorious.org/procps/procps
### irqstat: IRQ viewer for NUMA

<table>
<thead>
<tr>
<th>IRQ#</th>
<th>TOTAL</th>
<th>NODE0</th>
<th>NODE1</th>
<th>NODE2</th>
<th>NODE3</th>
<th>NODE4</th>
<th>NODE5</th>
<th>NODE6</th>
<th>NODE7</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td>361145</td>
<td>0</td>
<td>0</td>
<td>361145</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>PCI-MSI-edge qla2xxx</td>
</tr>
<tr>
<td>123</td>
<td>356357</td>
<td>0</td>
<td>0</td>
<td>356357</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>PCI-MSI-edge qla2xxx</td>
</tr>
<tr>
<td>109</td>
<td>46118</td>
<td>46118</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>PCI-MSI-edge lpfc:fp</td>
</tr>
<tr>
<td>111</td>
<td>46085</td>
<td>46085</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>PCI-MSI-edge lpfc:fp</td>
</tr>
<tr>
<td>115</td>
<td>44369</td>
<td>0</td>
<td>0</td>
<td>44369</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>PCI-MSI-edge lpfc:fp</td>
</tr>
<tr>
<td>117</td>
<td>43951</td>
<td>0</td>
<td>0</td>
<td>43951</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>PCI-MSI-edge lpfc:fp</td>
</tr>
<tr>
<td>107</td>
<td>37149</td>
<td>37149</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>PCI-MSI-edge lpfc:fp</td>
</tr>
<tr>
<td>113</td>
<td>36305</td>
<td>36305</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>PCI-MSI-edge lpfc:fp</td>
</tr>
<tr>
<td>126</td>
<td>11</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>PCI-MSI-edge eth0[i0]</td>
</tr>
<tr>
<td>129</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>PCI-MSI-edge eth0[i3]</td>
</tr>
</tbody>
</table>

[https://github.com/lanceshelton/irqstat](https://github.com/lanceshelton/irqstat)
Must end users be NUMA-aware?

- Unfortunately, yes.
  
  - Users must be aware of PCIe device slot placement
  
  - Optimal NUMA tuning is not yet performed by the OS
  
  - Persistent tuning is a non-trivial task
  
  - Performance challenges are changing faster than tools
How can this be improved?

• NUMA architectures must be detected properly and tuned by default
  ▶ Phase out Nehalem or add SLIT support for multiple local nodes

• Linux distributions need to provide optimal tuning across applications and devices at the OS level

• Improve existing tools
What’s next?

▸ More cores per socket
  • 15 core CPUs by the end of next year?

▸ Removal of existing bottlenecks
  • Multi-queue block layer: http://kernel.dk/blk-mq.pdf

▸ Improved tools
  • numatop: https://01.org/numatop
  • top: https://gitorious.org/procps/procps
  • irqstat: https://github.com/lanceshelton/irqstat
References

- HP DL980 Architecture

- HP NUMA Support

- Performance profiling methods
THANK YOU

LShelton@fusionio.com