Performance Tuning and Debugging

Don Porter
Why is my application slow?

• No silver bullet
• Part science, part art
  – Science: Measure performance, test hypotheses
  – Art: Finding practical balances of concerns
Most common culprits

• Insufficient resources
  • Configuration error
  • Hardware problems
Digression: Throughput and Latency

• What are they?

• Throughput: Operations over time
  – Requests per second
  – Transactions per minute
  – Higher is better

• Latency: Time to complete one operation
  – My server can complete an HTTP GET in .01 seconds
  – Lower is better
What happens when you are overloaded?

![Graph showing % Maximum Throughput vs Load (%)]

The graph illustrates the relationship between load and % maximum throughput. The blue line represents the ideal scenario, while the red line depicts a more realistic scenario. As the load increases, the throughput initially increases to a peak and then decreases sharply beyond a certain point.
What happens when you are overloaded?

![Graph showing latency vs. load percentage]

- **Latency**
- **Load (%)**

The graph illustrates the ideal latency under different load percentages. As the load increases, the latency also increases linearly.
What happens when you are overloaded?

![Diagram showing latency vs load with note: Note Change in Y Axis Scale---approaches infinity]
Graceful Degradation

• Ideally, when a system is overloaded, by n%, operation latency would increase by n% and throughput would stay constant

• In practice, systems rarely degrade gracefully when they are overloaded

• Thus, finding the “limiting factor” is essential
## atop

<table>
<thead>
<tr>
<th>ATOP - aria</th>
<th>2010/02/28 12:16:22</th>
<th>----</th>
<th>10s elapsed</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRC</td>
<td>sys 5.20s</td>
<td>user 6.20s</td>
<td>#proc 157</td>
</tr>
<tr>
<td>CPU</td>
<td>sys 27%</td>
<td>user 61%</td>
<td>irq 25%</td>
</tr>
<tr>
<td>cpu</td>
<td>sys 8%</td>
<td>user 59%</td>
<td>irq 25%</td>
</tr>
<tr>
<td>cpu</td>
<td>sys 13%</td>
<td>user 1%</td>
<td>irq 0%</td>
</tr>
<tr>
<td>cpu</td>
<td>sys 5%</td>
<td>user 1%</td>
<td>irq 0%</td>
</tr>
<tr>
<td>cpu</td>
<td>sys 1%</td>
<td>user 0%</td>
<td>irq 0%</td>
</tr>
<tr>
<td>CPL</td>
<td>avg1 2.08</td>
<td>avg5 1.91</td>
<td>avg15 1.31</td>
</tr>
<tr>
<td>MEM</td>
<td>tot 7.6G</td>
<td>free 5.0G</td>
<td>cache 1.9G</td>
</tr>
<tr>
<td>SWP</td>
<td>tot 2.6G</td>
<td>free 2.0G</td>
<td>vmcom 624.2M</td>
</tr>
<tr>
<td>DSK</td>
<td>sda busy 66%</td>
<td>read 0</td>
<td>write 1040</td>
</tr>
<tr>
<td>DSK</td>
<td>sdb busy 56%</td>
<td>read 0</td>
<td>write 848</td>
</tr>
<tr>
<td>NET</td>
<td>transport tcpi 188125</td>
<td>tcpo 99797</td>
<td>udp0 0</td>
</tr>
<tr>
<td>NET</td>
<td>network ipi 188125</td>
<td>ipo 99796</td>
<td>ipfrw 0</td>
</tr>
<tr>
<td>NET</td>
<td>eth0 21%</td>
<td>pcki 188120</td>
<td>pcko 99793</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PID</th>
<th>SYSCPU</th>
<th>USRCPU</th>
<th>VGROW</th>
<th>RGROW</th>
<th>RRDSK</th>
<th>WRDSK</th>
<th>ST</th>
<th>EXC</th>
<th>S</th>
<th>CPUNR</th>
<th>CPU</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>17063</td>
<td>1.12s</td>
<td>4.98s</td>
<td>0K</td>
<td>0K</td>
<td>-</td>
<td>-</td>
<td>E</td>
<td>0</td>
<td>E</td>
<td>-</td>
<td>61%</td>
<td>&lt;bzip2&gt;</td>
</tr>
<tr>
<td>17059</td>
<td>2.25s</td>
<td>1.08s</td>
<td>0K</td>
<td>0K</td>
<td>0K 246.6M</td>
<td>--</td>
<td>-</td>
<td>R</td>
<td>3</td>
<td>33%</td>
<td>rsync</td>
<td></td>
</tr>
<tr>
<td>17064</td>
<td>1.23s</td>
<td>0.13s</td>
<td>0K</td>
<td>0K</td>
<td>-</td>
<td>-</td>
<td>E</td>
<td>0</td>
<td>E</td>
<td>-</td>
<td>14%</td>
<td>&lt;tar&gt;</td>
</tr>
<tr>
<td>17011</td>
<td>0.27s</td>
<td>0.00s</td>
<td>0K</td>
<td>0K</td>
<td>0K 27012K</td>
<td>--</td>
<td>-</td>
<td>S</td>
<td>2</td>
<td>3%</td>
<td>pdflush</td>
<td></td>
</tr>
<tr>
<td>16991</td>
<td>0.14s</td>
<td>0.00s</td>
<td>0K</td>
<td>0K</td>
<td>0K 10808K</td>
<td>--</td>
<td>-</td>
<td>S</td>
<td>2</td>
<td>1%</td>
<td>pdflush</td>
<td></td>
</tr>
<tr>
<td>662</td>
<td>0.11s</td>
<td>0.00s</td>
<td>0K</td>
<td>0K</td>
<td>0K 4940K</td>
<td>--</td>
<td>-</td>
<td>S</td>
<td>2</td>
<td>1%</td>
<td>kjournal</td>
<td></td>
</tr>
<tr>
<td>1957</td>
<td>0.04s</td>
<td>0.00s</td>
<td>0K</td>
<td>0K</td>
<td>0K 114.2M</td>
<td>--</td>
<td>-</td>
<td>S</td>
<td>1</td>
<td>0%</td>
<td>kjournal</td>
<td></td>
</tr>
<tr>
<td>17047</td>
<td>0.02s</td>
<td>0.01s</td>
<td>0K</td>
<td>0K</td>
<td>0K</td>
<td>--</td>
<td>-</td>
<td>R</td>
<td>1</td>
<td>0%</td>
<td>atop</td>
<td></td>
</tr>
<tr>
<td>2669</td>
<td>0.01s</td>
<td>0.00s</td>
<td>0K</td>
<td>0K</td>
<td>0K</td>
<td>--</td>
<td>-</td>
<td>S</td>
<td>0</td>
<td>0%</td>
<td>httpd</td>
<td></td>
</tr>
<tr>
<td>2045</td>
<td>0.01s</td>
<td>0.00s</td>
<td>0K</td>
<td>0K</td>
<td>0K</td>
<td>--</td>
<td>-</td>
<td>S</td>
<td>0</td>
<td>0%</td>
<td>kondemand/0</td>
<td></td>
</tr>
<tr>
<td>2687</td>
<td>0.00s</td>
<td>0.00s</td>
<td>0K</td>
<td>0K</td>
<td>0K</td>
<td>--</td>
<td>-</td>
<td>S</td>
<td>3</td>
<td>0%</td>
<td>mythbackend</td>
<td></td>
</tr>
</tbody>
</table>

atop

• Super-useful tool that shows usage of
  – CPU
  – Memory
  – Disk
  – Network

• On a color terminal, highlights over-used resources
CPU

• Very rarely the bottleneck
  – Actually degrades gracefully in most cases

• Nonetheless, overloaded CPUs will seem less responsive

• Note that when another resource is scarce, CPU time is used trying to compensate
**Load Average**

- The average number of processes waiting for the CPU
  - Less than 1, the CPU is idle
  - Higher than 1 is ok, just means CPU is fully utilized
  - Very high values (>8) can indicate a problem

- Read from the uptime command:

  ```
  $ uptime
  20:10:13 up 20 days, 11:08, 5 users, load average: 0.00, 0.03, 0.05
  ```
Memory

• Often the biggest troublemaker
• Why?
  – OSes over-commit memory to applications
  – In other words, if I have 1GB RAM, I can have 5 applications that all think they have 300 MB
  – How is this possible?
  – Swapping
Swapping

• If the OS is running low on memory, it can take RAM away from applications
  – Save the contents to disk
  – Reuse the RAM

• If the application tries to read or write to this memory, the application is interrupted, OS notified
  – OS has to then find free RAM, replace contents for app
The problem with swapping

• Disk reads and writes are slow (relative to CPU)
  – You very rarely wait for them before making progress
  – Except when swapping

• Mitigation: OS makes educated guesses about unlikely-to-be-used data to swap out
  – In the best case, things slow down a bit, and then return to normal

• In the worst case, data ping-pongs between disk and RAM
  – Called thrashing
Recommendation

• If you see substantial swap usage in atop, buy more RAM
  – It is cheap, and more RAM is cheaper now than when you bought the computer

• Note: OS often uses substantial amount of RAM to cache the file system contents, so don’t be mislead if total RAM usage is near 100%
  – Look at swap to detect insufficient RAM
In a crisis...

• Linux has an out-of-memory killer
• As advertised, it just kills programs until there is enough memory
Swappiness

- Linux tries to swap some data out before there is a crisis
- Linux has a parameter that sets how aggressively to swap data. This can get out of whack
  - /proc/sys/vm/swappiness
- I’ve personally had to dial this back on an Ubuntu release that set the default too high, in order for a nearly idle system to be usable
Network

• When the network is overloaded, packets are dropped
  – But the other end usually retries

• Two biggest culprit for network overload:
  – Attack (denial of service, brute-force password guessing, spam, etc)
  – Legitimate overload (slashdotted website, peak usage time)

• Need to figure out which
Network advice

• If the overload is not legitimate, good security practice can help to reduce wasteful traffic
  – Firewall, denyhosts, spam filter, etc.
  – For DoS, there are also quality-of-service tools on many network devices to limit the share of packets delivered from any one source

• If the overload is legitimate, you may need more servers and a load-balancer
  – Like round-robin DNS
Disks

- Very rarely the bottleneck, except:
  - (Implicitly when thrashing swap)
  - Actual disk-intensive workloads (e.g., database)
  - And when disk is nearing end-of-life

- Why rarely a problem?
  - Most disk requests are asynchronous
  - Most disk-intensive applications inherently rate-limited

- Why a problem at end-of-life?
  - Heavy remapping yields poor scheduling
  - For SSDs, internal bookkeeping can take longer as the device ages
Disks

• In general, if the disk is getting old, the best advice is replace it
  – You also don’t want to lose data

• Some file systems perform worse as they age, but these are increasingly uncommon
  – Running a “defragmenter” can help
General advice

• Measure a performance baseline for your system
  – Application performance
  – Microbenchmarks (e.g., lmbench)

• If things seem slower, re-measure the component
  – Has my disk bandwidth degraded?

• This is the science of tuning
Other tools

• /proc/cpuinfo, /proc/meminfo, /proc/diskstats – useful system statistics
  – Lots of goodies in /proc
• vmstat – more details on memory usage
• nice/renice – adjust scheduling priority, giving more CPU time to important applications
• swapinfo – more details on swapping
• netstat – more details about network usage
• hdparm/sdparm – measure raw disk performance
• iostat – more details about disk I/O