Analyzing OS X Systems Performance with the USE Method

Brendan Gregg, Senior Performance Architect, Netflix
March, 2014
Find the Bottleneck
This Talk

• Summarizes casual to serious performance analysis of OS X
• From the *systems perspective*, not the application
• Many application issues can be found easily this way
• Covering not just current tools, but suggestions for future work
• May change how you think about performance!
whoami

- Senior Performance Architect at Netflix
- Primary author of the DTrace book
- Wrote many DTrace scripts included with OS X. Eg: dtruss, iosnoop, iotop, opensnoop, execsnoop, procsystime, bitesize.d, seeksize.d, setuids.d, etc...
  
- These were ported and enhanced by Apple engineering (thanks!)
- Created the USE method and USE method checklist for OS X
Agenda

- The Tools Method
- The USE Method
- Future work
The Tools Method
The Tools Method

• A tool-based performance analysis approach, commonly followed today. For reference, I've called it the "Tools Method".

  • 1. List available performance tools
  • 2. For each tool, list its useful metrics
  • 3. For each metric, list possible interpretation

• Simple, useful, but analysis is limited to what the tools provide easily
Tool Examples

- Activity Monitor
- atMonitor, Temperature Monitor Lite
- Command Line
- DTrace
- Instruments
Activity Monitor

• High level process and system summaries. A GUI version of top(1)
• Table shows processes by %CPU, memory
• CPU load over time
• Quit, info, and system diagnosis buttons
Activity Monitor

Network

- Quick way to see current and recent network throughput
- Like the CPU summary, shows aggregate device stats, and not per-device
Activity Monitor

CPU Usage

- Per-CPU utilization from previous 0.5 - 5 seconds (tunable)
- Handy to leave running. Look for single hot CPUs/threads
Earlier OS X also had a compact version (gone in Mavericks)

Was nice, but what I really want is a compact visualization for both per-CPU and historical data
Activity Monitor
CPU/Disk Suggestion

• Could show both per-device and history using a *utilization heat map*:

Activity Monitor
Sample Process

• The cog button ("System diagnostics options") has a "Sample process" option for profiling CPU code paths

• Explains %CPU usage

• Although output usually very long and time consuming to read (see scroll bar):
Activity Monitor
Flame Graphs?

• Suggestion: include a Flame Graph view

• Visualizes entire profile output in one screen

• http://github.com/brendangregg/FlameGraph
atMonitor

• 3rd party app. Version 2.7b crashes for me if "Top Window" is visible.

• Shows many useful metrics: per-CPU, RAM, GPU, per-disk, and per-network interface utilization percentages with histories.

• Currently the easiest way to see GPU, disk, and network utilization.

• Utilization is easy to interpret. I/O per second is not.
Temperature Monitor Lite

• Another 3rd party application

• Easy way to infer GPU utilization

• Normal: GPU1: 57°C

• Video: GPU1: 81°C
Command Line

• Accessed via the Terminal application

• Numerous performance tools available, from UNIX/BSD/OSX

• Eg, the `uptime(1)` command shows recent and historic CPU load:

```
$ uptime
14:36  up 43 days,  2:39, 30 users, load averages: 0.72 1.02 1.29
```

• There numbers are the 1, 5, and 15 minute load averages. Values are really constants in an exponential decay moving sum.

• Interpret: if average > number of CPUs, then CPUs are overloaded
Command Line: top

- top(1): high level process and system summary:

```
$ top -o cpu
Processes: 272 total, 4 running, 268 sleeping, 1546 threads            14:47:36
Load Avg: 1.14, 0.75, 0.95  CPU usage: 13.95% user, 2.78% sys, 83.26% idle
SharedLibs: 12M resident, 5112K data, 0B linkedit.
MemRegions: 339218 total, 6689M resident, 184M private, 2153M shared.
PhysMem: 3429M wired, 6502M active, 5910M inactive, 15G used, 537M free.
VM: 552G vsize, 1052M framework vsize, 111312590(1) pageins, 1437348(0) pageouts
Networks: packets: 120030109/127G in, 70582570/38G out.
Disks: 22089197/1050G read, 26756359/1163G written.

<table>
<thead>
<tr>
<th>PID</th>
<th>COMMAND</th>
<th>%CPU</th>
<th>TIME</th>
<th>#TH</th>
<th>#WQ</th>
<th>#PORT</th>
<th>#MREGS</th>
<th>RPRVT</th>
<th>RSHRD</th>
<th>RSIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>602</td>
<td>bash</td>
<td>100.0</td>
<td>47:42:28</td>
<td>1/1</td>
<td>0</td>
<td>21</td>
<td>27</td>
<td>236K</td>
<td>816K</td>
<td>760K</td>
</tr>
<tr>
<td>94370</td>
<td>top</td>
<td>17.2</td>
<td>00:03:77</td>
<td>1/1</td>
<td>0</td>
<td>24</td>
<td>39</td>
<td>4368K</td>
<td>216K</td>
<td>5116K</td>
</tr>
<tr>
<td>52617</td>
<td>firefox</td>
<td>6.3</td>
<td>47:30:58</td>
<td>45/1</td>
<td>2</td>
<td>576-</td>
<td>177307+</td>
<td>1984M+</td>
<td>200M</td>
<td>2530M+</td>
</tr>
<tr>
<td>92489</td>
<td>GoogleChrom</td>
<td>2.2</td>
<td>13:31:85</td>
<td>34</td>
<td>2</td>
<td>530</td>
<td>2454</td>
<td>273M</td>
<td>271M</td>
<td>734M</td>
</tr>
</tbody>
</table>
```

hey...
Command Line: `vm_stat`

- `vm_stat(1)`: virtual memory statistics, including free memory, paging
Command Line: iostat

- `iostat(1)`: block device I/O statistics. Disks, USB drives.

```bash
$ iostat 1
```

<table>
<thead>
<tr>
<th>KB/t</th>
<th>tps</th>
<th>MB/s</th>
<th>KB/t</th>
<th>tps</th>
<th>MB/s</th>
<th>us</th>
<th>sy</th>
<th>id</th>
<th>1m</th>
<th>5m</th>
<th>15m</th>
</tr>
</thead>
<tbody>
<tr>
<td>47.03</td>
<td>13</td>
<td>0.60</td>
<td>96.67</td>
<td>0</td>
<td>0.00</td>
<td>5</td>
<td>2</td>
<td>92</td>
<td>0.94</td>
<td>1.01</td>
<td>0.99</td>
</tr>
<tr>
<td>972.42</td>
<td>19</td>
<td>18.02</td>
<td>128.00</td>
<td>141</td>
<td>17.60</td>
<td>2</td>
<td>3</td>
<td>95</td>
<td>0.94</td>
<td>1.01</td>
<td>0.99</td>
</tr>
<tr>
<td>315.60</td>
<td>10</td>
<td>3.08</td>
<td>128.00</td>
<td>24</td>
<td>3.00</td>
<td>6</td>
<td>2</td>
<td>92</td>
<td>0.94</td>
<td>1.01</td>
<td>0.99</td>
</tr>
<tr>
<td>4.00</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0.00</td>
<td>6</td>
<td>2</td>
<td>92</td>
<td>0.94</td>
<td>1.01</td>
<td>0.99</td>
</tr>
<tr>
<td>1024.00</td>
<td>8</td>
<td>7.99</td>
<td>128.00</td>
<td>69</td>
<td>8.61</td>
<td>6</td>
<td>2</td>
<td>92</td>
<td>0.94</td>
<td>1.01</td>
<td>0.99</td>
</tr>
<tr>
<td>1024.00</td>
<td>18</td>
<td>17.97</td>
<td>128.00</td>
<td>143</td>
<td>17.85</td>
<td>2</td>
<td>2</td>
<td>95</td>
<td>0.86</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>1024.00</td>
<td>17</td>
<td>16.98</td>
<td>128.00</td>
<td>142</td>
<td>17.72</td>
<td>2</td>
<td>2</td>
<td>96</td>
<td>0.86</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>165.27</td>
<td>272</td>
<td>43.84</td>
<td>127.13</td>
<td>146</td>
<td>18.10</td>
<td>6</td>
<td>5</td>
<td>89</td>
<td>0.95</td>
<td>1.01</td>
<td>0.99</td>
</tr>
<tr>
<td>1024.00</td>
<td>18</td>
<td>17.98</td>
<td>128.00</td>
<td>143</td>
<td>17.85</td>
<td>2</td>
<td>2</td>
<td>96</td>
<td>0.95</td>
<td>1.01</td>
<td>0.99</td>
</tr>
</tbody>
</table>

- No percent utilization/busy, like other OSes? Makes it hard to interpret.
Command Line: netstat

- `netstat(1)`: various network statistics. `-i` for interface stats:

`$ netstat -iI en0 1`

<table>
<thead>
<tr>
<th>packets</th>
<th>errs</th>
<th>bytes</th>
<th>packets</th>
<th>errs</th>
<th>bytes</th>
<th>colls</th>
</tr>
</thead>
<tbody>
<tr>
<td>237</td>
<td>0</td>
<td>296232</td>
<td>167</td>
<td>0</td>
<td>18555</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>0</td>
<td>19374</td>
<td>16</td>
<td>0</td>
<td>4617</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>661</td>
<td>1</td>
<td>0</td>
<td>2020</td>
<td>0</td>
</tr>
<tr>
<td>1601</td>
<td>0</td>
<td>2231882</td>
<td>535</td>
<td>0</td>
<td>50072</td>
<td>0</td>
</tr>
<tr>
<td>3519</td>
<td>0</td>
<td>5027348</td>
<td>1005</td>
<td>0</td>
<td>62086</td>
<td>0</td>
</tr>
<tr>
<td>1362</td>
<td>0</td>
<td>1923223</td>
<td>627</td>
<td>0</td>
<td>39699</td>
<td>0</td>
</tr>
<tr>
<td>1338</td>
<td>0</td>
<td>1866404</td>
<td>296</td>
<td>0</td>
<td>17166</td>
<td>0</td>
</tr>
<tr>
<td>878</td>
<td>0</td>
<td>1203230</td>
<td>182</td>
<td>0</td>
<td>14803</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1302</td>
<td>11</td>
<td>0</td>
<td>2900</td>
<td>0</td>
</tr>
</tbody>
</table>

[...]

- No percent utilization, but can figure it out: throughput / known max
Command Line: tcpdump

- tcpdump(1): sniff and examine network packets:

  $ tcpdump -n
  tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
  listening on en0, link-type EN10MB (Ethernet), capture size 65535 bytes
  18:00:55.228744 IP 10.0.1.92.53 > 10.0.1.148.49228: 26359 1/0/0 A 69.192.253.15 (81)
  18:00:55.311056 ARP, Reply 10.0.1.162 is-at 2c:54:2d:a4:25:4c, length 28
  18:00:55.342793 IP 74.125.28.189.443 > 10.0.1.148.62998: Flags [P.], seq 3544891232:3544891287, ack 3832081572, win 661, options [nop,nop,TS val 2936982235 ecr 2331923799], length 55
  18:00:55.342933 IP 10.0.1.148.50359 > 67.195.141.201.443: Flags [P.], seq 696365506:696365533, ack 1903095540, win 16384, length 27

- Also dump to a file and examine later. Does incur overhead.
Observability So Far...

- We can see all the things!
- Not really...
Observability So Far...

- **Device Interconnect (PCIe/USB)**
- **Interface**
- **Transports**
- **I/O Controller**
- **Network Controller**
- **FSB**
- **Memory**
- **CPU**
- **Northbridge**
- **Southbridge**
- **DRAM**
- **GPU**
- **Other Devices**

### Applications
- **Darwin Operating System**
  - **XNU Kernel**
  - **Applications**
  - **System Libraries**
    - **BSD**
    - **VFS**
    - **HFS+...**
    - **Block Devices**
    - **Ethernet**
    - **Device Drivers**

### System Libraries
- **OSFMK**
- **Scheduler**
- **Virtual Memory**
- **I/O Kit**

### Device Drivers
- **I/O Kit**
- **Disk**
- **Disk Transports**
- **Port**

### Monitoring Tools
- **Netstat**
- **ActivityMonitor**
- **tcpdump**
- **iostat**
- **vm_stat**
- **top**
- **atMonitor**
- **Temp.Monitor**

DTrace

- Programmable, real-time, dynamic and static tracing
- Write your own one-liners and scripts, or use other people's; including those in /usr/bin
- There is a great book about it...
DTrace: Scripts

- Over 40 DTrace scripts are shipped with OS X (which I mostly wrote originally). Listing them:

```bash
$ man -k dtrace
bitesize.d(1m) - analyse disk I/O size by process. Uses DTrace
cpuwalk.d(1m) - Measure which CPUs a process runs on. Uses DTrace
creatbyproc.d(1m) - snoop creat()s by process name. Uses DTrace
dappprof(1m) - profile user and lib function usage. Uses DTrace
dapptrace(1m) - trace user and library function usage. Uses DTrace
diskhits(1m) - disk access by file offset. Uses DTrace
dispqlen.d(1m) - dispatcher queue length by CPU. Uses DTrace
dtrace(1) - generic front-end to the DTrace facility
dtruss(1m) - process syscall details. Uses DTrace
erroinfo(1m) - print errno for syscall fails. Uses DTrace
execsnoop(1m) - snoop new process execution. Uses DTrace
[...]```
DTrace: iosnoop

- iosnoop(1m): trace block device I/O

```
$ iosnoop

<table>
<thead>
<tr>
<th>UID</th>
<th>PID</th>
<th>D</th>
<th>BLOCK</th>
<th>SIZE</th>
<th>COMM</th>
<th>PATHNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>503</td>
<td>176</td>
<td>R</td>
<td>148471184</td>
<td>8192</td>
<td>SystemUIServer</td>
<td>??/vm/swapfile10</td>
</tr>
<tr>
<td>503</td>
<td>176</td>
<td>R</td>
<td>835310312</td>
<td>4096</td>
<td>SystemUIServer</td>
<td>??/vm/swapfile4</td>
</tr>
<tr>
<td>503</td>
<td>92489</td>
<td>W</td>
<td>746204600</td>
<td>61440</td>
<td>Google Chrome</td>
<td>??/Chrome/.com.google.Chrome.hw1Inp</td>
</tr>
<tr>
<td>503</td>
<td>92489</td>
<td>W</td>
<td>746204720</td>
<td>23472</td>
<td>Google Chrome</td>
<td>??/Default/.com.google.Chrome.76k4tG</td>
</tr>
<tr>
<td>0</td>
<td>19</td>
<td>W</td>
<td>425711304</td>
<td>4096</td>
<td>syslogd</td>
<td>??/DiagnosticMessages/2014.02.14.asl</td>
</tr>
<tr>
<td>0</td>
<td>19</td>
<td>W</td>
<td>57246896</td>
<td>4096</td>
<td>syslogd</td>
<td>??/DiagnosticMessages/StoreData</td>
</tr>
<tr>
<td>0</td>
<td>19</td>
<td>W</td>
<td>425710304</td>
<td>4096</td>
<td>syslogd</td>
<td>??/DiagnosticMessages/2014.02.14.asl</td>
</tr>
<tr>
<td>503</td>
<td>52617</td>
<td>W</td>
<td>214894232</td>
<td>4096</td>
<td>firefox</td>
<td>??/iw4rbe19.default/<em>CACHE_CLEAN</em></td>
</tr>
<tr>
<td>0</td>
<td>19</td>
<td>W</td>
<td>57246896</td>
<td>4096</td>
<td>syslogd</td>
<td>??/DiagnosticMessages/StoreData</td>
</tr>
<tr>
<td>0</td>
<td>19</td>
<td>W</td>
<td>425710304</td>
<td>4096</td>
<td>syslogd</td>
<td>??/DiagnosticMessages/2014.02.14.asl</td>
</tr>
</tbody>
</table>
```

- Identify processes and files causing disk I/O
DTrace: hfsslower.d

- hfsslower.d: trace HFS calls slower than a threshold. Eg, 10 ms:

```
$ ~/dtbook_scripts/Chap5/hfsslower.d 10

<table>
<thead>
<tr>
<th>TIME</th>
<th>PROCESS</th>
<th>D</th>
<th>KB</th>
<th>ms</th>
<th>FILE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014 Feb 14 17:35:59</td>
<td>Terminal</td>
<td>R</td>
<td>5751</td>
<td></td>
<td>data.data</td>
</tr>
<tr>
<td>2014 Feb 14 17:35:59</td>
<td>Terminal</td>
<td>R</td>
<td>6166</td>
<td></td>
<td>data.data</td>
</tr>
<tr>
<td>2014 Feb 14 17:35:59</td>
<td>Terminal</td>
<td>W</td>
<td>11921</td>
<td></td>
<td>data.data</td>
</tr>
</tbody>
</table>

[...]```

- Traces *all* application I/O to the file system, not just disk I/O

- Script is on http://www.dtracebook.com
DTrace: execsnoop

- execsnoop(1m): trace process execution

$ execsnoop -v
STRTIME                UID    PID   PPID ARGS
2014 Feb 14 19:40:55   503  94835    551 man
2014 Feb 14 19:40:55   503  94835    551 man
2014 Feb 14 19:40:55   503  94841  94837 groff
2014 Feb 14 19:40:55   503  94839  94837 tbl
2014 Feb 14 19:40:55   503  94840  94838 cat
2014 Feb 14 19:40:55   503  94841  94845 grotty
2014 Feb 14 19:40:55   503  94844  94841 troff
2014 Feb 14 19:40:55   503  94843  94842 less
2014 Feb 14 19:40:58   503  94846  92489 Google Chrome He
2014 Feb 14 19:41:03   503  94847  92489 Google Chrome He

- Shows what programs are launched
DTrace: dtruss

- dtruss(1m): trace system calls, from one or many processes

```
$ dtruss -en bash

<table>
<thead>
<tr>
<th>PID/THRD</th>
<th>ELAPSD</th>
<th>SYSCALL(args)</th>
<th>= return</th>
</tr>
</thead>
<tbody>
<tr>
<td>475/0x1199:</td>
<td>87917</td>
<td>read(0x0, &quot;a\0&quot;, 0x1)</td>
<td>= 1 0</td>
</tr>
<tr>
<td>475/0x1199:</td>
<td>12</td>
<td>write_nocancel(0x2, &quot;a\0&quot;, 0x1)</td>
<td>= 1 0</td>
</tr>
<tr>
<td>475/0x1199:</td>
<td>3</td>
<td>sigprocmask(0x1, 0x0, 0x7FF55F898E0)</td>
<td>= 0x0 0</td>
</tr>
<tr>
<td>475/0x1199:</td>
<td>2</td>
<td>sigaltstack(0x0, 0x7FF55F898D0, 0x0)</td>
<td>= 0 0</td>
</tr>
<tr>
<td>475/0x1199:</td>
<td>48163</td>
<td>read(0x0, &quot;t\0&quot;, 0x1)</td>
<td>= 1 0</td>
</tr>
<tr>
<td>475/0x1199:</td>
<td>10</td>
<td>write_nocancel(0x2, &quot;t\0&quot;, 0x1)</td>
<td>= 1 0</td>
</tr>
<tr>
<td>475/0x1199:</td>
<td>3</td>
<td>sigprocmask(0x1, 0x0, 0x7FF55F898E0)</td>
<td>= 0x0 0</td>
</tr>
<tr>
<td>475/0x1199:</td>
<td>2</td>
<td>sigaltstack(0x0, 0x7FF55F898D0, 0x0)</td>
<td>= 0 0</td>
</tr>
<tr>
<td>475/0x1199:</td>
<td>12</td>
<td>write_nocancel(0x2, &quot;m\0&quot;, 0x1)</td>
<td>= 1 0</td>
</tr>
<tr>
<td>475/0x1199:</td>
<td>2</td>
<td>sigprocmask(0x1, 0x0, 0x7FF55F898E0)</td>
<td>= 0x0 0</td>
</tr>
</tbody>
</table>
```

- dtruss is a script - edit it to add/modify it as desired
DTrace: sotop

- **sotop**: summarize socket I/O by-process, top-style:

```
$ sotop

<table>
<thead>
<tr>
<th>PROCESS</th>
<th>PID</th>
<th>READS</th>
<th>WRITES</th>
<th>READ_KB</th>
<th>WRITE_KB</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel_task</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>475</td>
</tr>
<tr>
<td>firefox</td>
<td>52617</td>
<td>205</td>
<td>14</td>
<td>84</td>
<td>22</td>
<td>118</td>
</tr>
<tr>
<td>Terminal</td>
<td>165</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>35</td>
</tr>
<tr>
<td>WindowServer</td>
<td>89</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>34</td>
</tr>
<tr>
<td>SIDPLAY</td>
<td>51232</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>31</td>
</tr>
<tr>
<td>Google Chrome H</td>
<td>92513</td>
<td>6</td>
<td>12</td>
<td>0</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>Google Chrome H</td>
<td>94477</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>clear</td>
<td>94909</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>Google Chrome</td>
<td>92489</td>
<td>16</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>sh</td>
<td>94909</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
</tbody>
</table>
```

- Also from the DTrace book.
Instruments

- Advanced analysis GUI
- Includes many "Instruments", which profile applications in different ways:
- Data sources include DTrace, CPU counters
Instruments
Thread States
Instruments
Low Level CPU Counters

- Performance monitor counter (PMC) and performance monitor interrupts can be instrumented
- Hard work, but can be used to understand bus and interconnect activity
Tools Method provides reasonable coverage

- Some observability gaps, some uneven coverage

- Can improve coverage by adding more tools: ps, ping, traceroute, latency, df, sysctl, plockstat, opensnoop, dispqlen.d, runocc.d, nfsstat, iopending, soconnect_mac.d, httpdstat.d, sc_usage, fs_usage, ...

- I could keep covering tools for the rest of this talk...
Most DTrace scripts are in /usr/bin. Some are from my DTrace book and are available online.

Custom Instruments using CPU counters/interrupts can be added for bus observability.
The Focus on Tools

• Useful, however, learning tools & metrics becomes laborious.

• Still limited by what the tools provide, or provide easily.

• You can try to approach this in a different way...
Instead of starting with the tools, start with the questions
The USE Method
The USE Method

- For every resource, check:
  1. Utilization
  2. Saturation
  3. Errors
The USE Method

- For every resource, check:
  - 1. Utilization: time resource was busy, or degree used
  - 2. Saturation: degree of queued extra work
  - 3. Errors: any errors
• If it helps, consider all resources as a queueing system:

• Also check errors
Hardware Resources

- CPUs
- Main Memory
- Network Interfaces
- Storage Devices
- Controllers, Interconnects

Find the *functional diagram* and examine every item in the data path...
Hardware Functional Diagram

- For each check:
  - 1. Utilization
  - 2. Saturation
  - 3. Errors
USE Method Checklists

- Build a checklist for all combinations, identifying tools/metrics to use
## OS X Checklist

<table>
<thead>
<tr>
<th>Resource</th>
<th>Type</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Utilization</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>Saturation</td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>Errors</td>
<td></td>
</tr>
</tbody>
</table>
## OS X Checklist

<table>
<thead>
<tr>
<th>Resource</th>
<th>Type</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Utilization</td>
<td>system-wide: iostat 1, &quot;us&quot; + &quot;sy&quot;; per-cpu: DTrace [1]; Activity Monitor → CPU Usage or Floating CPU Window; per-process: top -o cpu, &quot;%CPU&quot;; Activity Monitor → Activity Monitor, &quot;%CPU&quot;; ...</td>
</tr>
<tr>
<td>CPU</td>
<td>Errors</td>
<td>dmesg; /var/log/system.log; Instruments → Counters, for PMC and whatever error counters are supported (eg, thermal throttling)</td>
</tr>
</tbody>
</table>
OS X Checklist

<table>
<thead>
<tr>
<th>Resource</th>
<th>Type</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Utilization</td>
<td>system-wide: iostat 1, &quot;us&quot; + &quot;sy&quot;; per-cpu: DTrace [1]; Activity Monitor → CPU Usage or Floating CPU Window; per-process: top -o cpu, &quot;%CPU&quot;; Activity Monitor → Activity Monitor, &quot;%CPU&quot;; ...</td>
</tr>
<tr>
<td>CPU</td>
<td>Errors</td>
<td>dmesg; /var/log/system.log; Instruments → Counters, for PMC and whatever error counters are supported (eg, thermal throttling)</td>
</tr>
<tr>
<td>Resource</td>
<td>Type</td>
<td>Metric</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>Memory Capacity</td>
<td>Utilization</td>
<td></td>
</tr>
<tr>
<td>Memory Capacity</td>
<td>Saturation</td>
<td></td>
</tr>
<tr>
<td>&quot;</td>
<td>Errors</td>
<td></td>
</tr>
</tbody>
</table>
## OS X Checklist, cont.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Type</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Capacity</td>
<td>Utilization</td>
<td>system-wide: vm_stat 1, main memory free = &quot;free&quot; + &quot;inactive&quot;, in units of pages; Activity Monitor → Activity Monitor → System Memory, &quot;Free&quot; for main memory; per-process: top -o rsize, &quot;RSIZE&quot; is resident main memory size, &quot;FSIZE&quot; is virtual memory size; ps -alx, &quot;RSS&quot; is resident set size, &quot;SZ&quot; is virtual memory size;</td>
</tr>
<tr>
<td>&quot;</td>
<td>Errors</td>
<td>System Information → Hardware → Memory, &quot;Status&quot; for physical failures; DTrace failed malloc()s</td>
</tr>
</tbody>
</table>
## OS X Checklist, cont.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Type</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Capacity</td>
<td>Utilization</td>
<td>system-wide: <code>vm_stat 1</code>, main memory free = &quot;free&quot; + &quot;inactive&quot;, in units of pages; Activity Monitor → Activity Monitor → System Memory, &quot;Free&quot; for main memory; per-process: <code>top -o rsize</code>, &quot;RSIZE&quot; is resident main memory size, &quot;VSIZE&quot; is virtual memory size; <code>ps -alx</code>, &quot;RSS&quot; is resident set size, &quot;SZ&quot; is virtual memory size;</td>
</tr>
<tr>
<td></td>
<td>Errors</td>
<td>System Information → Hardware → Memory, &quot;Status&quot; for physical failures; DTrace failed malloc()s</td>
</tr>
</tbody>
</table>
OS X Checklist, cont.

- Full list: http://www.brendangregg.com/USEmethod/use-macosx.html

- Includes references from earlier tables
Software Resources

- Can be studied using USE metrics as well, if possible
- OS X Checklist includes some example software resources:
  - Processes, file descriptors, kernel mutexes, user-level mutexes
Mutex Lock

- Can you think of what these could mean for a mutex lock?:
  - Utilization
  - Saturation
  - Errors
Mutex Lock

- Can you think of what these could mean for a mutex lock?:
  - Utilization: held time per second
  - Saturation: measure of contention time or waiters
  - Errors: EDEADLK, EINVAL
Future Work
Future Work

- Tools/Metrics for USE Method
- More methodologies, and then tools
USE Method Tools

- Tools can be developed to fetch USE metrics more easily
- Especially for busses and interconnects
- Would love to see USE metrics in Activity Monitor
USE Method New Uses

• Can be applied new areas, developing new metrics
• May not always work, but worth trying
• Find a functional diagram of your system, application, or environment, and look for U.S.E. metrics for each component
# USE Metrics for all of:

Darwin Operating System

<table>
<thead>
<tr>
<th>Applications</th>
<th>System Libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSD</td>
<td>System Call Interface</td>
</tr>
<tr>
<td>VFS</td>
<td>Sockets</td>
</tr>
<tr>
<td>HFS+/...</td>
<td>TCP/UDP</td>
</tr>
<tr>
<td>Block Devices</td>
<td>Ethernet</td>
</tr>
<tr>
<td></td>
<td>Device Drivers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BSD</th>
<th>OSFMK</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Call Interface</td>
<td>Scheduler</td>
</tr>
<tr>
<td>Sockets</td>
<td>Virtual Memory</td>
</tr>
<tr>
<td>TCP/UDP</td>
<td>I/O Kit</td>
</tr>
<tr>
<td>IP</td>
<td>Device Drivers</td>
</tr>
</tbody>
</table>

Hardware

- CPU
- GPU
- FSB
- DMI
- DRAM
- Northbridge
- Southbridge
- Other Devices

Device Interconnect (PCIe/USB)

- I/O Controller
- Network Controller
- Other Devices

XNU Kernel

- USE Metrics for all of:
  - Applications
  - System Libraries
  - BSD
  - VFS
  - HFS+/...
  - Block Devices
  - OSFMK
  - System Call Interface
  - Sockets
  - TCP/UDP
  - IP
  - Ethernet
  - Device Drivers
  - Scheduler
  - Virtual Memory
  - I/O Kit
### Stranger Example: TCP

- **"netstat -s" output has over 50 metrics for TCP**
- **Do you understand them all?**
- **Could USE metrics provide a high level summary, treating TCP as a software resource? (might be a stretch)**

```plaintext
$ netstat -s
tcp:
  8 0 4 4 4 4 9 9  packets sent
  2 8 7 0 6 7 1 9  data packets (36136450 bytes)
  7 6 5 9 9  data packets (6571215 bytes) retransmitted
  60 session initiated by WDH discovery
  618746 ack-only packets (648964 delayed)
  0 UNO only packets
  0 window probe packets
  5026129 window update packets
  707630 control packets
  0 packets sent after flow control

  17714976 packets received
  16294694 acks (for 3602941580 bytes)
  556279 duplicate acks
  0 acks for unsent data
  12477568 packets (124692675 bytes) received in-sequence
  1230891 duplicate packets (115103377 bytes)
  1084 old duplicate packets
  70 packets with same dup. data (17270 bytes duped)
  615283 out-of-order packets (439507901 bytes)
  67 packets (0 bytes) of data after window
  6 window probes
  14858 window update packets
  70285 packets received after close
  65 bad resets
  6 discarded for bad checksum
  6 discarded for bad header offset fields
  6 discarded because packet too short
  374941 connection requests
  01 connection requests
  01 bad connection attempts
  00 on-going connections
  32048 connections established (including accepts)
  38180 connections closed (including 37038 drops)
  14327 connection established cached RTT on close
  14327 connections updated cached RTT variance on close
  545 connections updated cached arstablish on close
  1791 embryonic connections dropped
  1420452 segments updated rtt (of 8478926 attempts)
  274144 retransmit timeouts
  4445 connections dropped by retransmit timeout
  6 connections dropped after retransmitting FIM
  91 persist timeouts
  0 connections dropped by persist timeout
  12794 keepalive timeouts
  225 keepalive probes sent
  225 keepalive probes dropped
  152 keepalive probes dropped by keepalive
  1312411 correct ACK header predictions
  12989016 correct data packet header predictions
  177546 ACK recovery episodes
  512584 segments retransmit ACK recovery episodes
  2650726 bytes retransmit in ACK recovery episodes
  656366 ACK options (ACK blocks) received
  5642516 ACK options (ACK blocks) sent
  0 ACK scoreboard overflow
```

#macIT2014
USE Method: TCP

- TCP as a software resource metrics:
  - Utilization
  - Saturation
  - Errors
USE Method: TCP

- TCP as a software resource metrics:
  - Utilization: time data was buffered per second
  - Saturation: listen queue overflows
  - Errors: bad connection attempts, bad resets, bad checksums, ...
  - I think I'd classify retransmits and duplicates as errors.
Other Methodologies

- Other methodologies include:
  - Drill Down Analysis Method
  - Workload Characterization
  - Thread State Analysis (TSA) Method
- These too can pose questions that tools then answer
References

- http://dtracebook.com - has DTrace book scripts online
- http://www.brendangregg.com/FlameGraphs/cpuflamegraphs.html - flame graphs
Thanks

- http://www.brendangregg.com
- bgregg@netflix.com
- @brendangregg