Need for Performance Tuning

- Reduce IT spend - find and eliminate waste, find areas to tune, and do more with less
- Build scalable architectures - understand system limits and develop around them
- Solve issues - locate bottlenecks and latency outliers
Find and fix kernel-based performance issues
- 2-20% wins: I/O or buffer size tuning, NUMA config, etc
- 2-200x wins: bugs, disabled features, perturbations causing latency outliers
- Kernels change, new devices are added, workloads scale, and new performance issues are encountered

Analyze application performance from kernel/system context
- 2-2000x wins: identifying and eliminating unnecessary work

This is why we study OS and system software
Perspectives

- System analysis can be top-down, or bottom-up:

  - Workload
  - Application
  - System Libraries
  - System Calls
  - Kernel
  - Devices

Developers

Operating System Software Stack

Sysadmins

Resource Analysis

Workload Analysis
Anti-Methodologies

- The lack of a deliberate methodology
- Street light anti-method
  1. Pick observability tools that are
     - Familiar
     - Found on the Internet, or at random
  2. Run tools
  3. Look for obvious issues
- Drunk man anti-method
  - Tune things at random until problem goes away
Methodologies

- For example, USE method
  - For every resource, check:
    - Utilization
    - Saturation
    - Errors
- 5 Whys
  - Ask “why” 5 times
- Other methods include
  - Workload characterization, drill-down analysis, event-tracing, baseline stats, static performance tuning
- Start with questions, then find tools
Command Line Tools

- Useful to study even if you never use them
  - GUIs and commercial products often use the same interfaces

```
$ vmstat 1
procs -----------memory---------- --swap-- ...
  r  b  swpd free  buff  cache  si  so ...
 9  0   0 29549320 29252 9299060  0  ... 
 2  0   0 29547876 29252 9299332  0  ... 
 4  0   0 29548124 29252 9299460  0  ... 
 5  0   0 29548840 29252 9299592  0  ...
```

Kernel
/proc, /sys, ...
Observability Tools

- Tools that watch diverse activities during execution of given workloads
- Useful for troubleshooting or performance optimization of specific workloads
How Do You Measure These?
Basic Tools

- uptime
- top (or htop)
- ps
- vmstat
- iostat
- mpstat
- free
uptime

- One way to print load averages

```
MacPro:~ euiseong$ uptime
19:30  up 5:26, 2 users, load averages: 1.05 0.90 0.75
MacPro:~ euiseong$
```

- A measure of resource demand
- Exponentially-damped moving averages
  - Historic trend without line graphs
- “Load > number of CPUs” may mean CPU saturation
top (or htop)

- System and per-process interval summary
- Can miss short-lived processes
- Can consume noticeable CPU to read /proc
Process status listing

```
$ ps -ef f

<table>
<thead>
<tr>
<th>UID</th>
<th>PID</th>
<th>PPID</th>
<th>C</th>
<th>STIME</th>
<th>TTY</th>
<th>STAT</th>
<th>TIME</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>4546</td>
<td>1</td>
<td>0</td>
<td>11:08</td>
<td>?</td>
<td>Ss</td>
<td>0:00</td>
<td>/usr/sbin/sshd -D</td>
</tr>
<tr>
<td>root</td>
<td>28261</td>
<td>4546</td>
<td>0</td>
<td>17:24</td>
<td>?</td>
<td>Ss</td>
<td>0:00</td>
<td>_ sshd: prod [priv]</td>
</tr>
<tr>
<td>prod</td>
<td>28287</td>
<td>28261</td>
<td>0</td>
<td>17:24</td>
<td>?</td>
<td>S</td>
<td>0:00</td>
<td>_ sshd: prod@pts/0</td>
</tr>
<tr>
<td>prod</td>
<td>28288</td>
<td>28287</td>
<td>0</td>
<td>17:24</td>
<td>pts/0</td>
<td>Ss</td>
<td>0:00</td>
<td>_ -bash</td>
</tr>
<tr>
<td>prod</td>
<td>3156</td>
<td>28288</td>
<td>0</td>
<td>19:15</td>
<td>pts/0</td>
<td>R+</td>
<td>0:00</td>
<td>_ ps -ef f</td>
</tr>
<tr>
<td>root</td>
<td>4965</td>
<td>1</td>
<td>0</td>
<td>11:08</td>
<td>?</td>
<td>Ss</td>
<td>0:00</td>
<td>/bin/sh /usr/bin/svscanboot</td>
</tr>
<tr>
<td>root</td>
<td>4969</td>
<td>4965</td>
<td>0</td>
<td>11:08</td>
<td>?</td>
<td>S</td>
<td>0:00</td>
<td>_ svscan /etc/service</td>
</tr>
</tbody>
</table>
```

[...]

```
vmstat

- Virtual memory statistics and more

- First output line has some summary since boot values
iostat

- Block I/O (disk I/O) stats

```bash
eulseong@accept:~$ iostat -xmdz 2
Linux 3.13.0-37-generic (accept) 12/09/2014 _x86_64_ (8 CPU)
Device: rrqm/s wrqm/s r/s w/s rMB/s wMB/s avgrq-sz avgq-sz await r-await w-await svctm %util
sda 0.00 0.31 0.02 0.35 0.00 0.01 38.33 0.01 13.97 4.02 14.64 10.08 0.38
```
mpstat

- Multi-processor statistics
- Look for unbalanced workloads, hot CPUs

```
euiseong@accept:~$ mpstat -P ALL 1
Linux 3.13.0-37-generic (accept) 12/09/2014  x86_64  (8 CPU)

<table>
<thead>
<tr>
<th>Time</th>
<th>CPU</th>
<th>%usr</th>
<th>%nice</th>
<th>%sys</th>
<th>%iowait</th>
<th>%irq</th>
<th>%soft</th>
<th>%steal</th>
<th>%guest</th>
<th>%gnice</th>
<th>%idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>07:42:16 PM</td>
<td>all</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>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>100.00</td>
</tr>
<tr>
<td>07:42:17 PM</td>
<td>0</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>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>100.00</td>
</tr>
<tr>
<td>07:42:17 PM</td>
<td>1</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>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>100.00</td>
</tr>
<tr>
<td>07:42:17 PM</td>
<td>2</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>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>100.00</td>
</tr>
<tr>
<td>07:42:17 PM</td>
<td>3</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>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>100.00</td>
</tr>
<tr>
<td>07:42:17 PM</td>
<td>4</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>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>100.00</td>
</tr>
<tr>
<td>07:42:17 PM</td>
<td>5</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>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>100.00</td>
</tr>
<tr>
<td>07:42:17 PM</td>
<td>6</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>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>100.00</td>
</tr>
<tr>
<td>07:42:17 PM</td>
<td>7</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>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>100.00</td>
</tr>
</tbody>
</table>
```
free

- **Main memory usage**

```plaintext
euiseong@accept:~$ free

            total       used       free     shared    buffers   cached
Mem:     8153632  6372528     1781104      15864   250300    5601128
-/+ buffers/cache:   521100   7632532
Swap:   19533820       0  19533820
```

- **buffers:** block device I/O cache
- **cached:** virtual page cache
Basic Observability Tools
Where to Observe

- **gprof**
  - Shows total amount of time your program spent executing each function
  - Also shows call graph of your program
- **Use gprof to first figure out where to attack**
- **Steps to use gprof**
  1. Compile and link your program with profiling enabled
  2. Execute your program to generate a profile data file
  3. Run gprof to analyze profile data
Using gprof

- Compiling
  - `cc -g -c myprog.c -pg`
- Execution of compile program will produce `gmon.out`
- Invoking gprof will produce analysis results

### Flat profile:

Each sample counts as 0.01 seconds.

<table>
<thead>
<tr>
<th>time</th>
<th>cumulative</th>
<th>self</th>
<th>calls</th>
<th>ms/call</th>
<th>total</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>33.34</td>
<td>0.02</td>
<td>0.02</td>
<td>7208</td>
<td>0.00</td>
<td>0.00</td>
<td>open</td>
</tr>
<tr>
<td>16.67</td>
<td>0.03</td>
<td>0.01</td>
<td>244</td>
<td>0.04</td>
<td>0.12</td>
<td>offline</td>
</tr>
<tr>
<td>16.67</td>
<td>0.04</td>
<td>0.01</td>
<td>8</td>
<td>1.25</td>
<td>1.25</td>
<td>memcpy</td>
</tr>
<tr>
<td>16.67</td>
<td>0.05</td>
<td>0.01</td>
<td>7</td>
<td>1.43</td>
<td>1.43</td>
<td>write</td>
</tr>
<tr>
<td>0.00</td>
<td>0.06</td>
<td>0.01</td>
<td>236</td>
<td>0.00</td>
<td>0.00</td>
<td>tsort</td>
</tr>
<tr>
<td>0.00</td>
<td>0.06</td>
<td>0.00</td>
<td>192</td>
<td>0.00</td>
<td>0.00</td>
<td>tlower</td>
</tr>
<tr>
<td>0.00</td>
<td>0.06</td>
<td>0.00</td>
<td>47</td>
<td>0.00</td>
<td>0.00</td>
<td>strlen</td>
</tr>
<tr>
<td>0.00</td>
<td>0.06</td>
<td>0.00</td>
<td>45</td>
<td>0.00</td>
<td>0.00</td>
<td>strncat</td>
</tr>
<tr>
<td>0.00</td>
<td>0.06</td>
<td>0.00</td>
<td>1</td>
<td>0.00</td>
<td>50.00</td>
<td>main</td>
</tr>
<tr>
<td>0.00</td>
<td>0.06</td>
<td>0.00</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td>memcmp</td>
</tr>
<tr>
<td>0.00</td>
<td>0.06</td>
<td>0.00</td>
<td>1</td>
<td>0.00</td>
<td>10.11</td>
<td>print</td>
</tr>
<tr>
<td>0.00</td>
<td>0.06</td>
<td>0.00</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td>profil</td>
</tr>
<tr>
<td>0.00</td>
<td>0.06</td>
<td>0.00</td>
<td>1</td>
<td>0.00</td>
<td>50.00</td>
<td>report</td>
</tr>
</tbody>
</table>

Granularity: each sample hit covers 2 bytes for 20.00% of 0.06 seconds

<table>
<thead>
<tr>
<th>index</th>
<th>time</th>
<th>self</th>
<th>children</th>
<th>called</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100.0</td>
<td>0.00</td>
<td>0.05</td>
<td>1/1</td>
<td>&lt;anonymous&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>start [1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>main [2]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>on_exit [58]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>exit [59]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>100.0</td>
<td>0.00</td>
<td>0.05</td>
<td>1/1</td>
<td>&lt;anonymous&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>start [1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>report [3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>100.0</td>
<td>0.00</td>
<td>0.05</td>
<td>1/1</td>
<td>&lt;anonymous&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>main [2]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>report [3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>time_sleep [6]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>print [9]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>fmtstr [12]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>strftime [18]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>timeval [24]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>sleep [44]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>59.8</td>
<td>0.01</td>
<td>0.02</td>
<td>244+260</td>
<td>&lt;anonymous&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;anonymous&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>report [1]</td>
</tr>
</tbody>
</table>

Compiled with `-O3 -g -c myprog.c -pg`