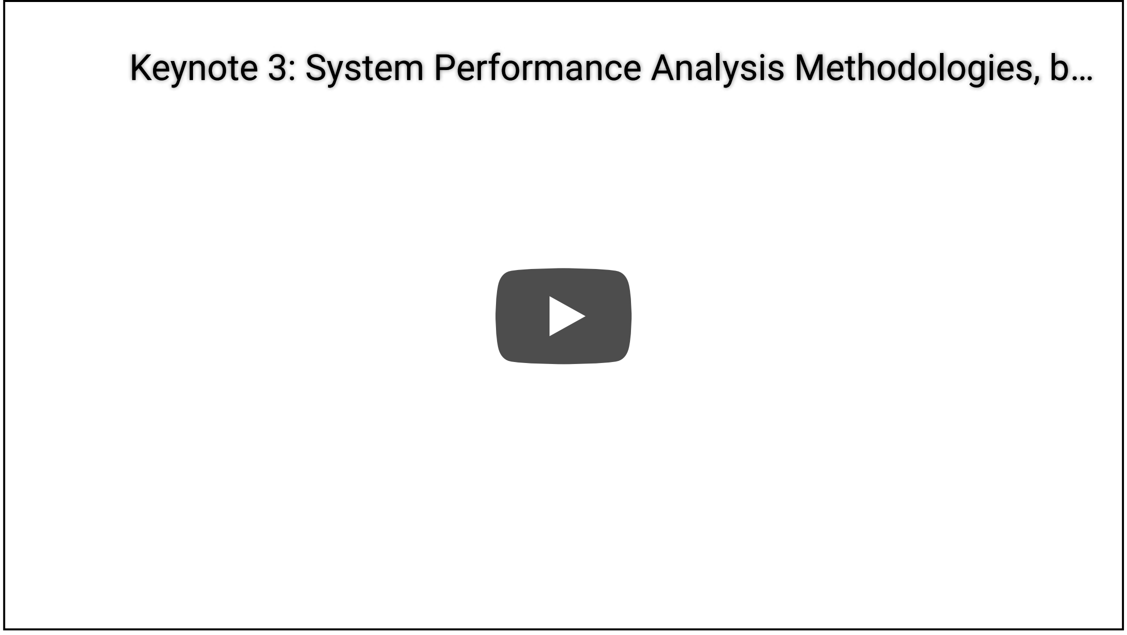
EuroBSDcon: System Performance Analysis Methodologies

28 Oct 2017

For my first trip to Paris I gave the closing keynote at EuroBSDcon 2017 on performance methodologies, using FreeBSD 11.1 as an analysis target. In the past I've shared similar methodologies applied to other operating systems, and finished porting them to BSD for this talk. It was a few days of work, which is really not bad. That's a virtue of these methodologies: once you learn them, you can apply them to anything throughout your career, and it doesn't take too much time to re-apply them.

The video is on youtube:



Keynote 3: System Performance Analysis Methodologies, b…

And the slides are on slideshare:

EuroBSDcon 2017

System Performance Analysis Methodologies

Brendan Gregg

Senior Performance Architect

NETFL

1 of 65

FreeBSD has an excellent range of analysis tools, and this was an opportunity to show them off. Among the new content I developed for the talk was a **FreeBSD performance checklist**:

1. `uptime` → load averages
2. `dmesg -a | tail` → kernel errors
3. `vmstat 1` → overall stats by time
4. `vmstat -P` → CPU balance
5. `ps -auxw` → process usage
6. `iostat -xz 1` → disk I/O
7. `systat -ifstat` → network I/O
8. `systat -netstat` → TCP stats
9. `top` → process overview
10. `systat -vmstat` → system overview

I also developed a new tool to support my thread state analysis methodology on FreeBSD, tstates.d:

```
# ./tstates.d
Tracing scheduler events... Ctrl-C to end.
^C
Time (ms) per state (read script for info):
COMM             PID        CPU  RUNQ     SLP     USL    SUS    SWP    LCK    IWT    YLD
irq15: ata1      12           0     0       0       0      0      0      0  15024      0
[...]
sleep            877          0     0     505       0      0      0      0      0      0
bufdaemon        19           0    11       0   15057      0      0      0      0      0
sleep            879          0     0    2614       0      0      0      0      0      0
devd             523          0     0   15024       0      0      0      0      0      0
syncer           21           1     9       0   15055      0      0      0      0      0
fsck_ufs         878          1     0       0      10      0      0      0      0      0
fsck             836          1     0      12       0      0      0      0      0      0
dd               883          2     0       0       0      0      0      0      0      0
bufspacedaemon   20           3     5       0   15019      0      0      0      0      0
dtrace           873          3    23   15980       0      0      0      0      0      0
sh               881          4     0       3       1      0      0      0      0      0
csh              865          5     7   13882       0      0      0      0      0      0
rand_harvestq    6            8    20       0   15846      0      0      0      0      0
kernel           0           29    15       0       0      0      0      0      0      0
cam              4           45    14       0       0      0      0      0      0      0
sshd             863         52    85   13757       0      0      0      0      0      0
intr             12          79   192       0       0      0      0      0      0      0
cksum            876       1591   177       0     234      0      0      0      0      0
idle             11       14114  1902       0       0      0      0      0      0      0
```

This tool breaks down thread time into different states by tracing scheduler events (which can have noticeable overhead: measure in a lab environment before use). The states are:

- **CPU**: on-CPU
- **RUNQ**: Waiting on a CPU run queue
- **SLP**: Interruptible sleep
- **USL**: Uninterruptible sleep (eg, disk I/O)
- **SUS**: Suspended
- **SWP**: Swapped
- **LCK**: Waiting for a lock
- **IWT**: Waiting for an interrupt
- **YLD**: Yield

I added USL since the talk to split out disk I/O from the sleep state. The output above includes a `sleep 0.5` command, and a `cksum`.

EuroBSDcon was a great conference, and I had a lot of fun catching up with the BSD folk and meeting new people. If you missed my talk, you can see it online above, and I hope you find it useful.

You can comment here, but I can't guarantee your comment will remain here forever: I might switch comment systems at some point (eg, if disqus add advertisements).

About this blog