DTrace and Samba

By: Ira Cooper
Senior Systems Software Engineer – The MathWorks Inc.
Team Member – Samba Team

Normal System Introspection:

- iostat – tells about disk latency
- netstat – networking connections
- mpstat – cpu usage
- lockstat – locking stats and information

What is DTrace?

Philosophy of DTrace

- DTrace is production safe.
- DTrace has no impact when not in use.
- DTrace has very minimal impact when in use.
  - If you ask for a ton of data, clearly there will be some impact.

- DTrace is more about asking questions:
  - How often is a fcntl being called?
  - What locks are being contended on?
  - What system calls are being called? By what applications?

Providers – What you can ask

- Providers:
  - fbt
  - usdt
  - pid
  - syscall
  - profile
  - tick
  - And that's just a few of them…

How do you ask questions?

- "D" – The language
  - Providers to probes
    - My development VM shows over 62,000 probes
  - Specially constructed to have no loops
  - Global variables
  - Associative arrays

Example “one liners”

- syscall:::entry /execname==“smbd”/ { @[probefunc] = count() }
- profile-1001 { @[ustack()] = count() }
- profile-1001 { @[stack()] = count() }
- syscall::fcntl:entry /execname==“smbd”/ { ustack(); }

Flame Graphs

- Helps you visualize any stack counting type result.
- X axis: Probe count
- Y axis: The stack
- It makes more sense if you just see it…

Note for the following examples:

- Almost all of the data gathering was done on the real production boxes.
  - Using the real production software + load.
- Minimized impact through the following techniques:
  - Probing less used functions
  - Probing for short durations

Example Case: “Too many system calls.”

- mpstat –a 1 shows ~4-5 million system calls a second.
- What are they?
- One-liner to find out:
  - syscall:::entry {@[probefunc] = count()}

```
[ other system calls, that have less calls ]
readv      294245
lseek      901852
kill       1187757
fcntl      2796337
```

Now what?

- fcntl is getting called an absurd amount.
  - Note, it is getting called about twice as much as “kill”.
  - Do we care?
- What is calling it?
- Can we fix it…
- syscall::fcntl:entry /execname==“smbd”/ { @[ustack()] = count() }
- Look at the flame graph!

How do we fix it?

- Look at the source of the calls.
  - Many were asking “does the current process exist.”
  - Samba is not allowed to be existential.
- But that wasn’t enough.
  - fill_share_mode_lock.
- Robust Mutexes!
  - Volker fixed this in part.
- serverid.index
  - Removing all TDB access from this hot path.

How do we fix it, part 2.

- Remove the code via refactoring
  - Volker + metze did this, in 4.0 and master
  - We have not tested how effective this change is, yet.
  - The serverid.index may help in addition to this refactoring.
- There is a secondary lock on locking.tdb
  - Fixed via an existing smb.conf parameter.

Results After the mmap + smb.conf Fix:

- Before:

```
[other less called system calls]
readv      294245
lseek      901852
kill       1187757
fcntl      2796337
```

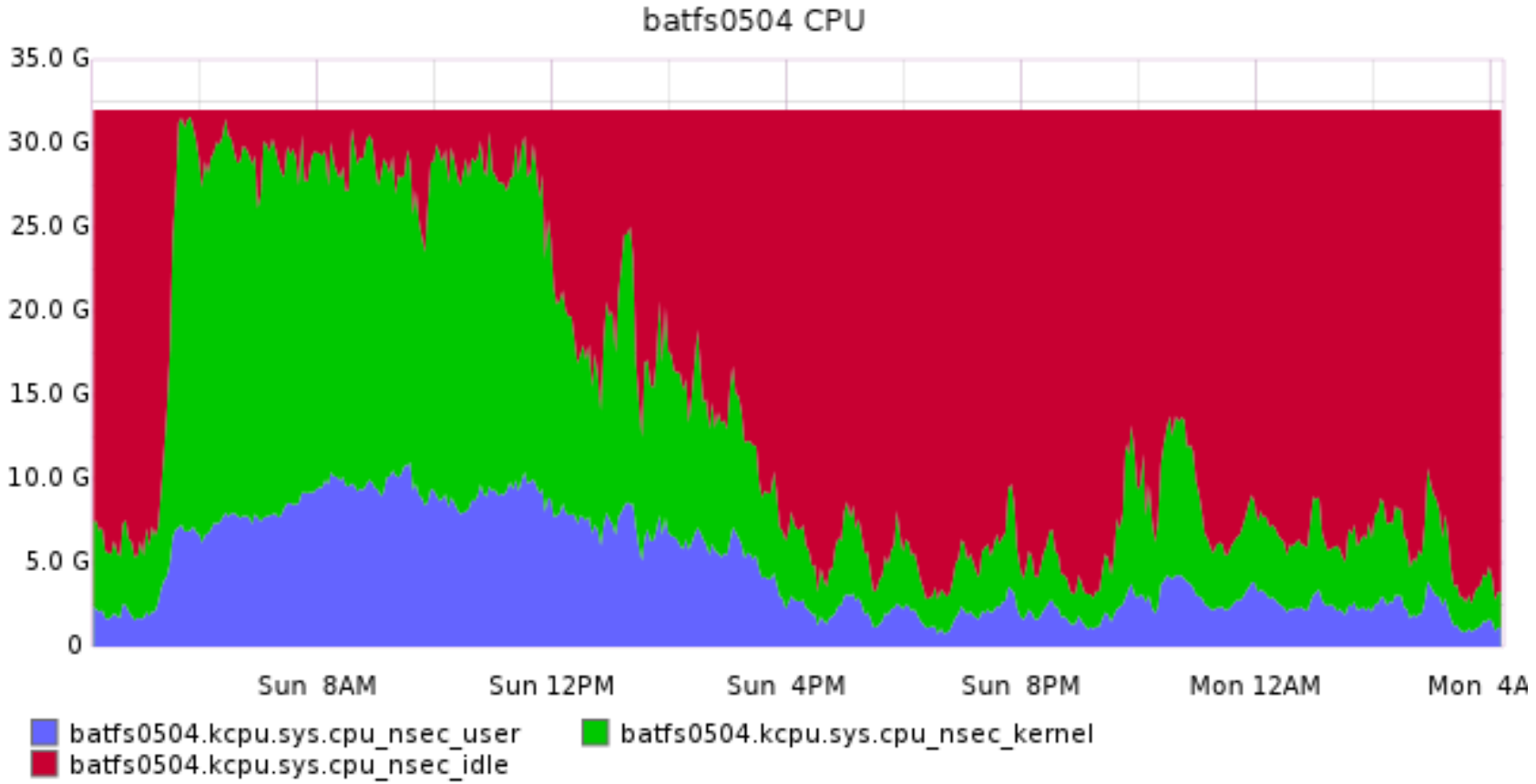
- After:

```
[other less called system calls]
 fcntl     762954
 stat     1236821
 kill     2752993
 lseek    3645825
```

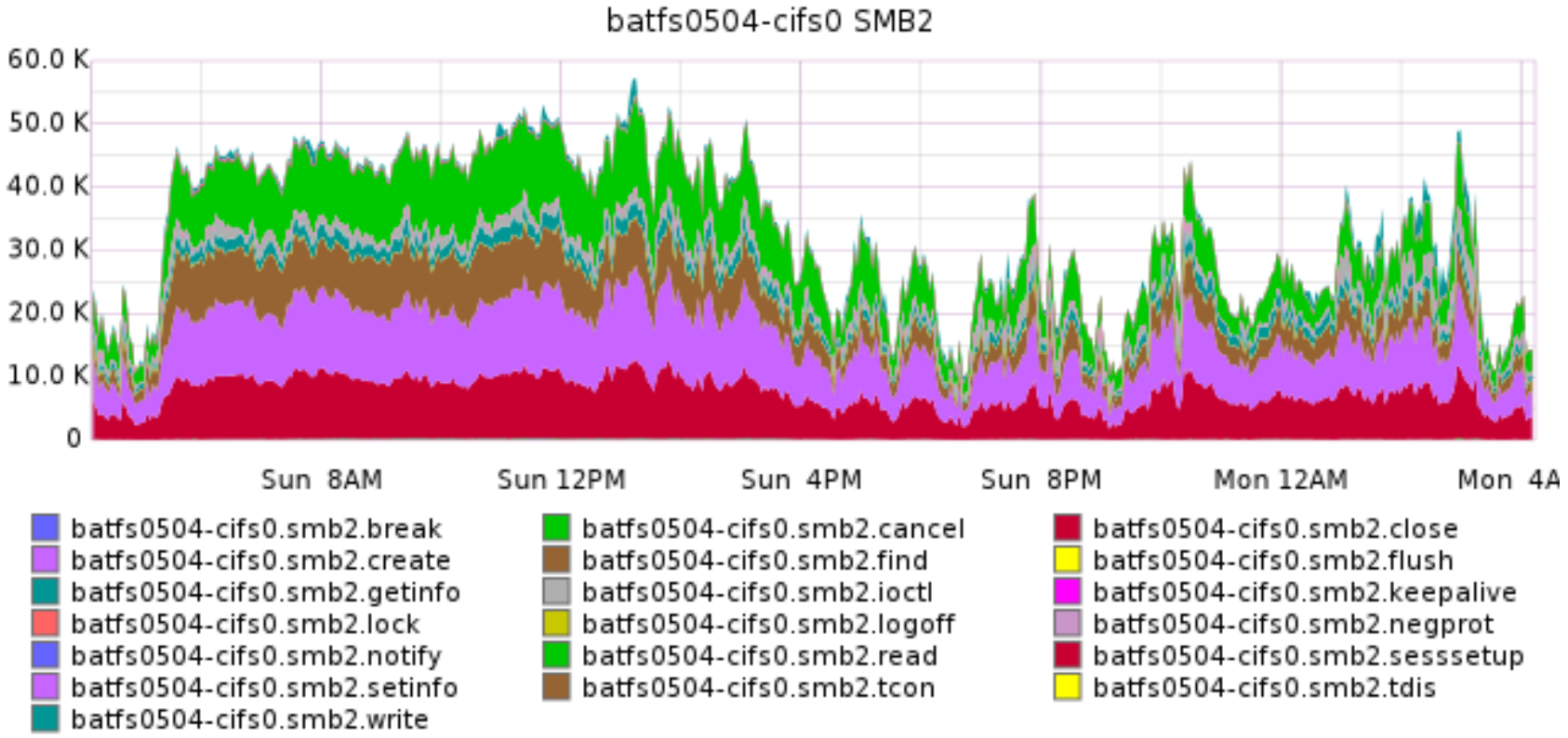
The CPU is Maxed?

- The graphs are from all different sources
- But, they are just to help us target in on the real issue with DTrace

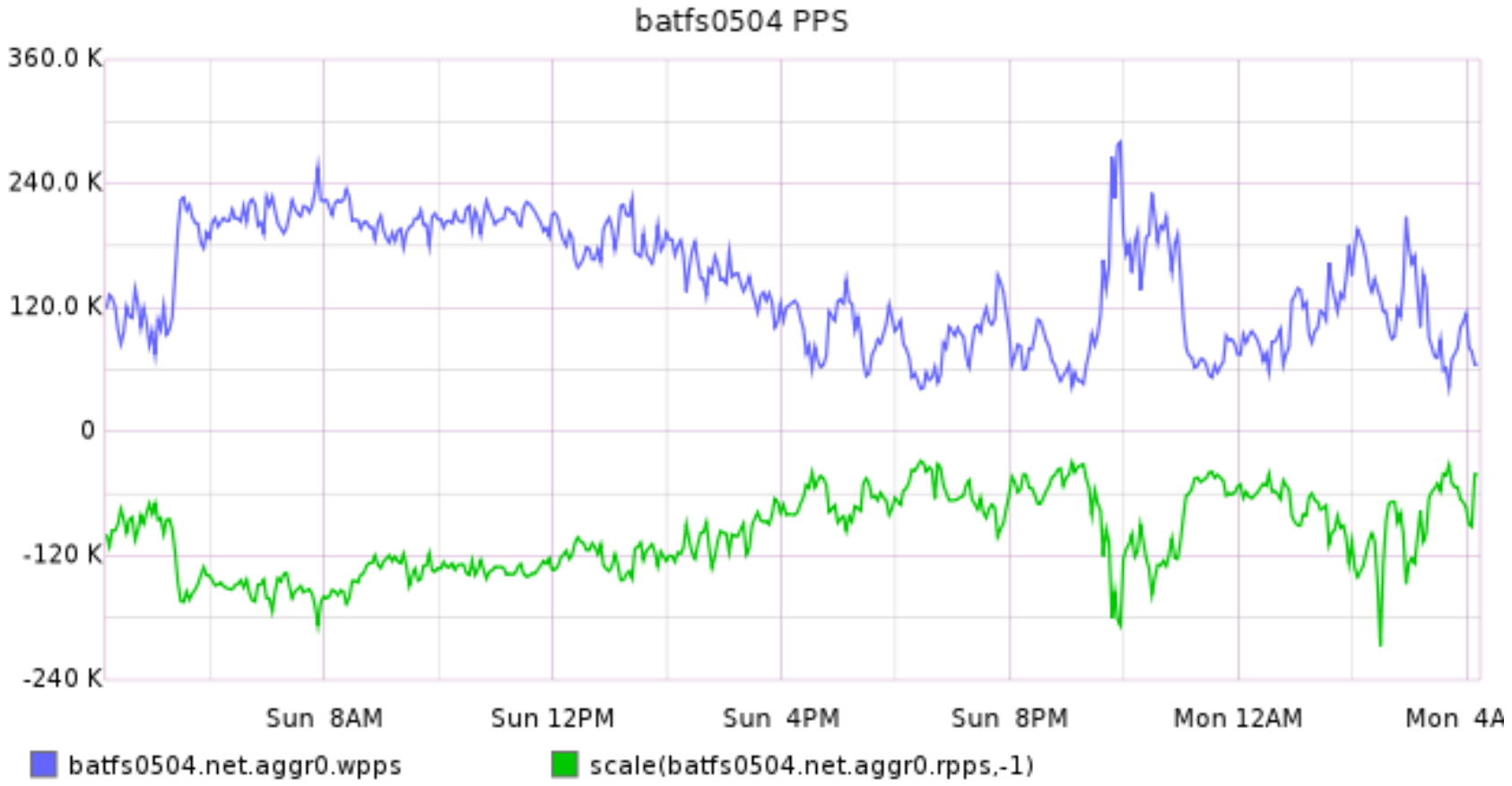
Why is the CPU Maxed?



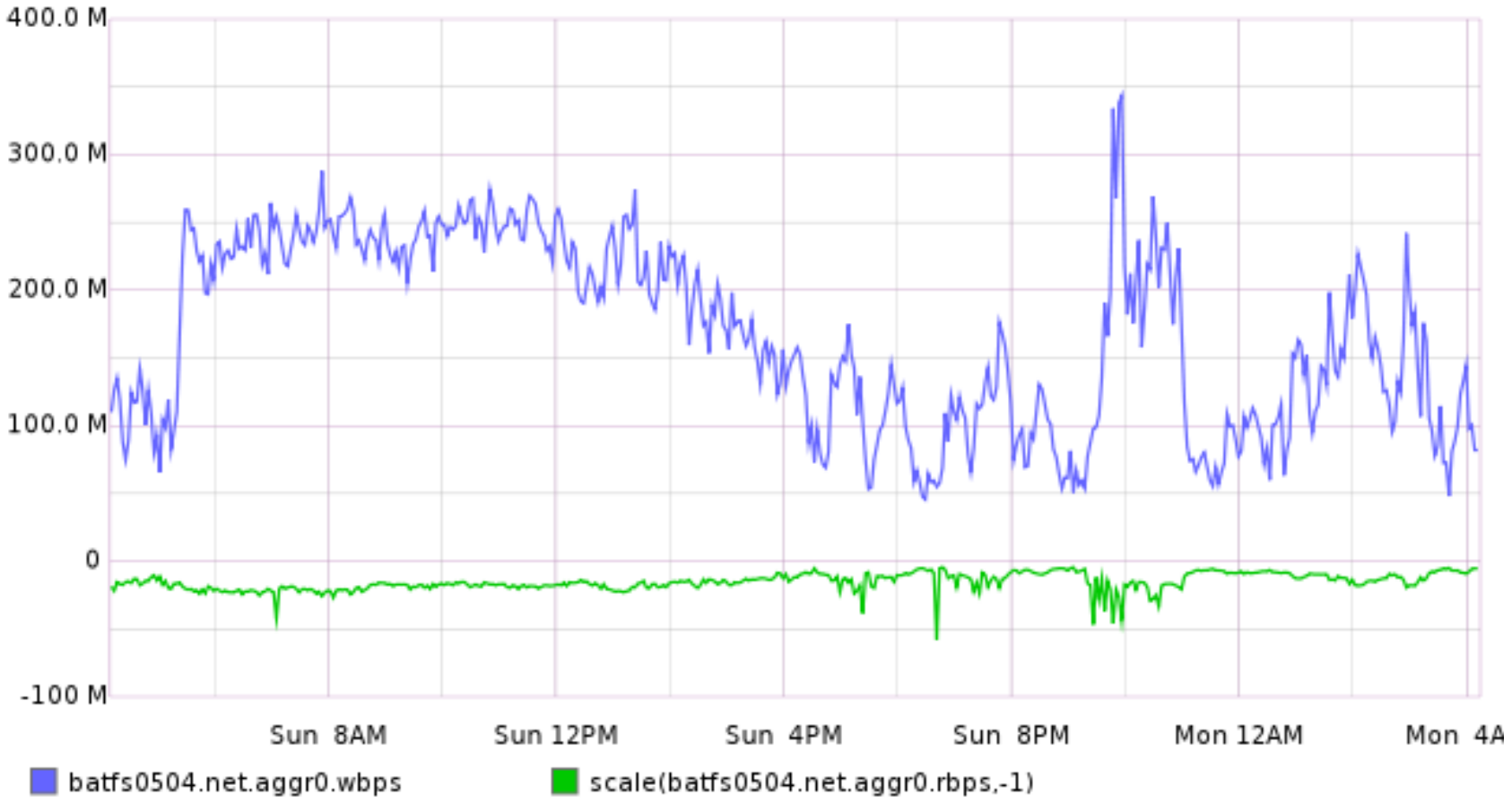
Look at ops.


batfs0504-cifs0 SMB2
60.0 K
50.0 K
40.0 K
30.0 K
20.0 K
10.0 K
0
Sun 8AM
Sun 12PM
Sun 4PM
Sun 8PM
Mon 12AM
Mon 4A
batfs0504-cifs0.smb2.break
batfs0504-cifs0.smb2.cancel
batfs0504-cifs0.smb2.close
batfs0504-cifs0.smb2.create
batfs0504-cifs0.smb2.find
batfs0504-cifs0.smb2.flush
batfs0504-cifs0.smb2.getinfo
batfs0504-cifs0.smb2.ioctl
batfs0504-cifs0.smb2.keepalive
batfs0504-cifs0.smb2.lock
batfs0504-cifs0.smb2.logoff
batfs0504-cifs0.smb2.negprot
batfs0504-cifs0.smb2.notify
batfs0504-cifs0.smb2.read
batfs0504-cifs0.smb2.sesssetup
batfs0504-cifs0.smb2.setinfo
batfs0504-cifs0.smb2.tcon
batfs0504-cifs0.smb2.tdis
batfs0504-cifs0.smb2.write

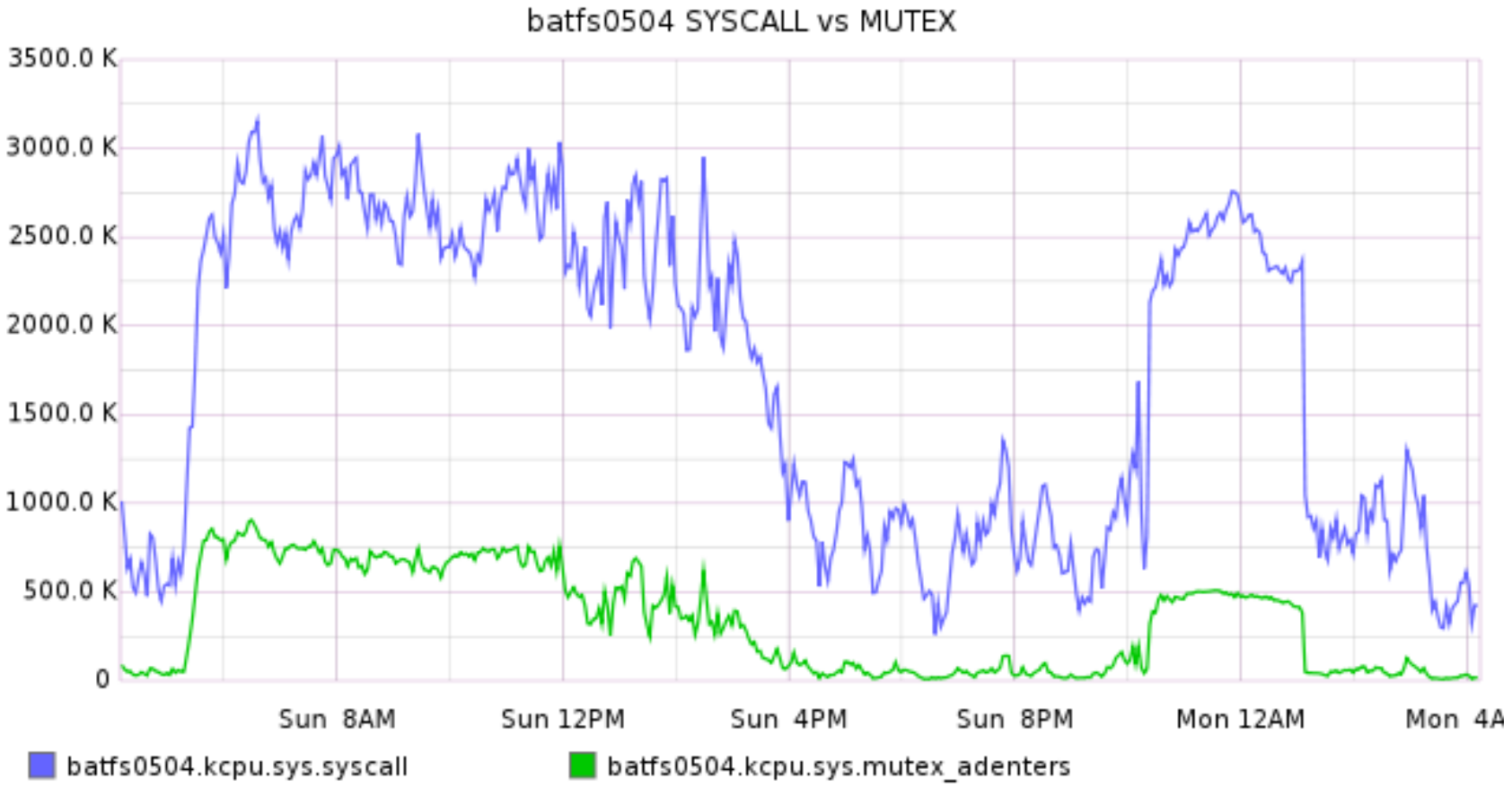
Look at Network Traffic



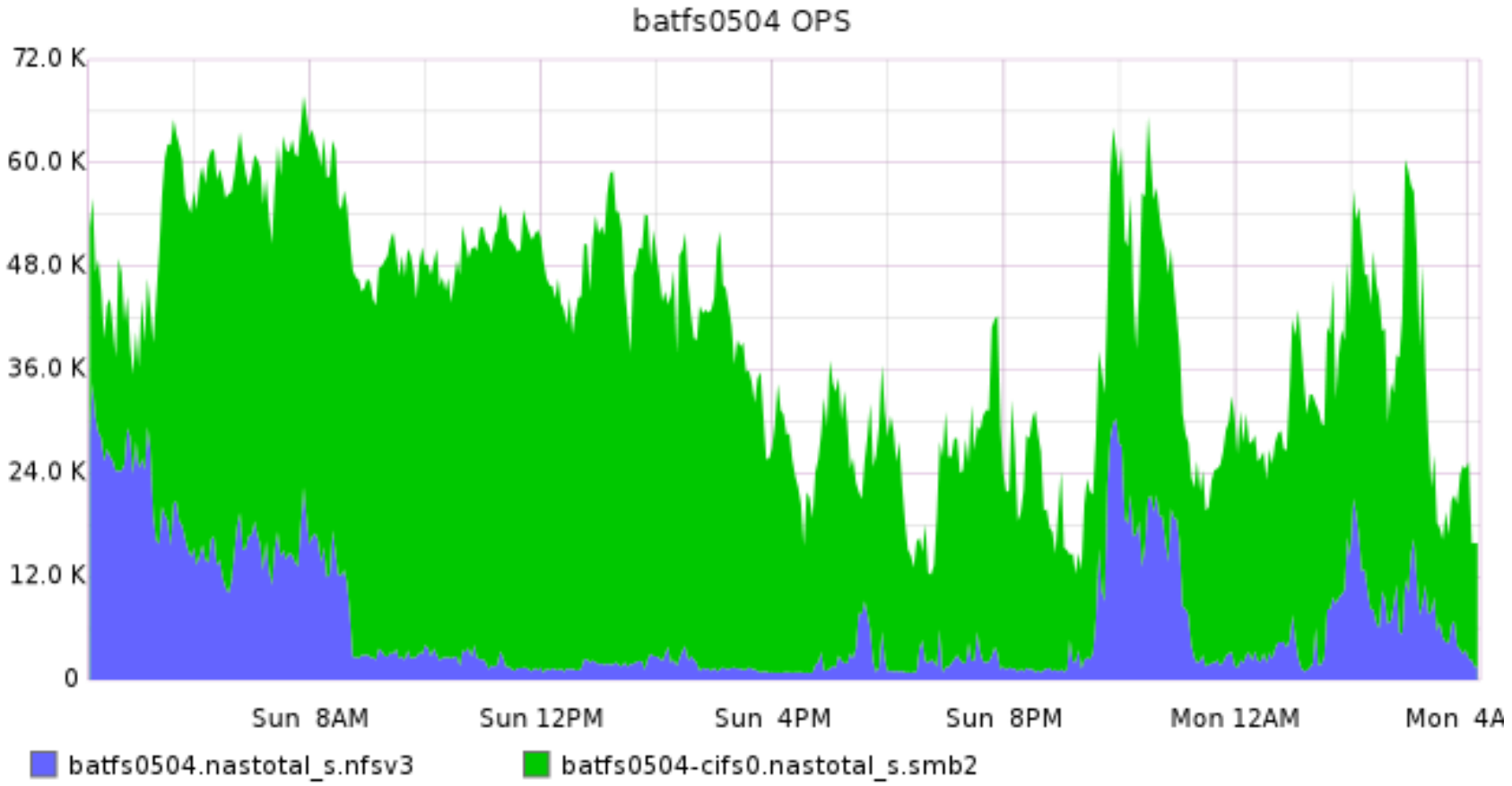
Look at Bandwidth



Are we locking too much?



Not due to NFS:



lockstat + other tools.

- lockstat –s 100 sleep 3
- (show data here)

Conclusion

- Stat is causing a problem
  - Deep directory hierarchy
  - Approximate value, at least 10%.

SMB2, Random Disconnect

- Server appears to be randomly disconnecting users
- Why?
- Added instrumentation to the server exit path
  - Normal stack logging function function
  - Also had samba output the “error status”
  - NT_STATUS_NO_MEMORY?!

Why, NT_STATUS_NO_MEMORY?

- What returned ENOMEM/NT_STATUS_NO_MEMORY
  - All signs point to writev
  - man writev – it is an undocumented return
- Prove it is doing it
  - syscall::writev:return /errno == ENOMEM && arg1== -1/ { ustack(); }
    - arg1 is always the return value of a “return” probe
    - Note, this is “pseudo code” for the real code
  - Wow, it is happening
- The fix was easy, once we knew what it was

MathWorks

Future directions with DTrace + Samba

- UDST (User-Level Statically Defined Tracing)
  - Create a samba provider
- Initial areas of interest:
  - smbd exit
  - oplocks/leases
  - Whatever else might interest us
- Overhead should be low/none
  - Setting up the data for a probe can have cost
  - There are ways around the issue

Now we should be able to:

Questions?

Thank you for attending!

Resources:

- DTrace resources: http://www.brendangregg.com/dtrace.html
  - DTrace toolkit
  - One-liners (great way to learn)

- Flame Graphs: http://dtrace.org/blogs/brendan/2011/12/16/flame-graphs/