NETFLIX

The Magical Mystery Merge

Or

Why we run FreeBSD-current at Netflix

Drew Gallatin

Openfest, November 2023

NETFLIX

Open Connect is Netflix's CDN. It is global, efficient, and purpose-built for distributing Netflix's content.

NETFLIX

Open Connect Appliances are Netflix's CDN servers.

NETFLIX

Netflix OCAs:

- Run FreeBSD-current
  - UFS for content / ZFS for root
- Serve content using ngnix
  - Pre-encoded for all codecs/bitrates
- Storage fails in place
  - No RAID for content, etc

NETFLIX

Netflix Workload:

- Serve only static media files
- Pre-encoded for all codecs/bitrates
  - Video quality is of the utmost importance, so we don’t transcode on the server
- Greatly simplifies server workload

NETFLIX

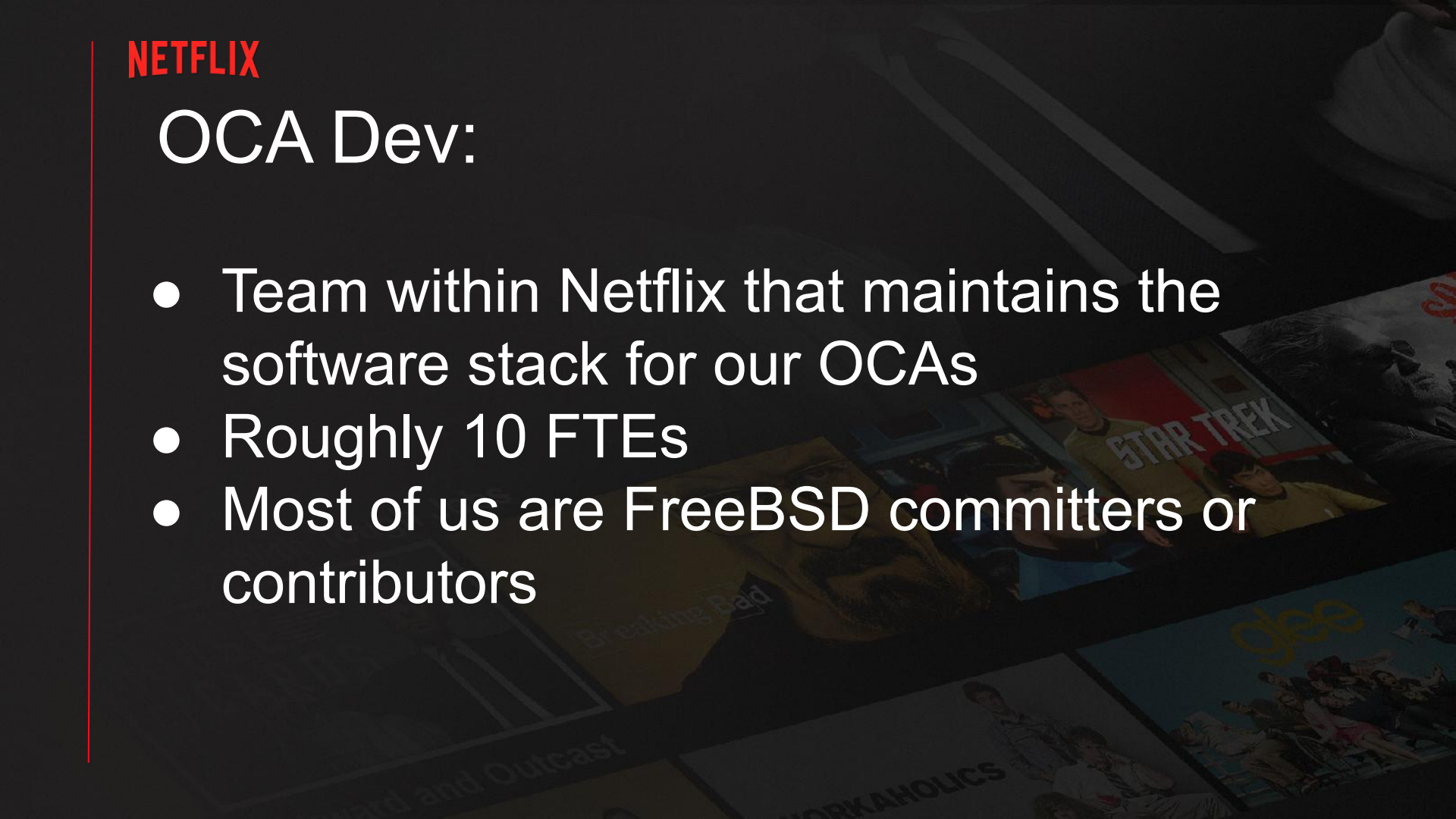
FreeBSD:

- Free, open source OS with BSD license
- Forked from 386BSD in 1993
- Focus on performance
- Main distro includes kernel, base utils, compilers, all source, and packaged 3rd party software
- Netflix has an internal “distro”

NETFLIX

OCA Dev:

- Team within Netflix that maintains the software stack for our OCAs
- Roughly 10 FTEs
- Most of us are FreeBSD committers or contributors



NETFLIX

OCA OS development, then & now

- Merge FreeBSD from -stable every few weeks.
- Moved to a new -stable branch every few years
  - This sometimes took months.
- Upstreaming patches required porting them to -current

NETFLIX

OCA OS development, then & now

- Merge upstream from FreeBSD-current every 3 weeks
  - We notice & resolve new upstream bugs that impact us immediately
  - Much easier to upstream code and collaborate with upstream developers

NETFLIX

Upstreaming code to FreeBSD:

- Small changes & bugfixes are done upstream, and brought back via the 3-week upstream merge process (or via cherrypicks for critical issues)
- Larger changes are done locally
  - kTLS took ~5 years to upstream

NETFLIX

Testing:

- Each change is built and regression tested automatically using Jenkins on amd64 & arm64
- Nightly smoke tests on dozens of OCAs running production traffic
- Release testing on a limited number of OCAs running production traffic

NETFLIX

Our contributions to FreeBSD:

- Asynchronous sendfile
- Unmapped mbufs
- Kernel TLS
- CAM IO Scheduler
- RACK and BBR TCP
- TCP HPTS (TCP pacing)
- Performance enhancements for NUMA

NETFLIX

More contributions to FreeBSD:

- Pfil memory pointer hooks for efficient firewall packet handling
- Many scalability fixes
- Kboot (kexec of FreeBSD from a Linux kernel) for arm64 and amd64

NETFLIX

Contributions to FreeBSD:

- Improved support for FreeBSD from various hardware vendors
- Financial support of the FreeBSD Foundation

NETFLIX

Performance Goals:

- Improve efficiency by reducing CPU use while improving our maintaining member QoE
  - Improves bandwidth at the high end
  - Reduces power consumption at the low end

NETFLIX

Performance Milestones:

2017: First 100Gb/s CDN server
    Intel Xeon E5-2697A, software kTLS
2020: First 200Gb/s CDN server
    AMD 7502P, software kTLS
2021: First 400Gb/s CDN server
    AMD 7502P, NIC kTLS offload

NETFLIX

Important Performance Milestones:

2022: First 800Gb/s CDN server
2x AMD 7713, NIC kTLS offload

2023: First 100Gb/s CDN server consuming only 100W of power
Nvidia Bluefield-3, NIC kTLS offload

NETFLIX

Magical Mystery Merge: *A case study in why we track FreeBSD-current*

- Upstream merge from 3 Aug -> 30 Aug
- Testing merge branch, my 8 year old 100GbE Xeons showed an 8% increase in CPU usage
- Neither profilers nor performance metrics showed any new bottlenecks

NETFLIX

Axis 0: Bits/sec

■ Total OUT bps (hw 1.27)
Max : 93.243G Min : 90.003G
Avg : 91.530G Last : 91.267G
Tot : 5.492T Cnt : 60.000

■ Total IN bps (negative)
Max : -480.257M Min : -507.484M
Avg : -492.367M Last : -507.484M
Tot : -29.542G Cnt : 60.000

□ Approx capacity bps (grey shaded)
Max : 90.822G Min : 90.813G
Avg : 90.817G Last : 90.822G
Tot : 181.635G Cnt : 2.000

□ Potential cap ignoring interfaces (grey line)
Max : 172.318G Min : 172.286G
Avg : 172.302G Last : 172.286G
Tot : 344.604G Cnt : 2.000

□ LIVE (green area)
Max : 1.000 Min : 1.000
Avg : 1.000 Last : 1.000
Tot : 60.000 Cnt : 60.000

□ Fill window (darkened area)
Max : 0.000 Min : 0.000
Avg : 0.000 Last : 0.000
Tot : 0.000 Cnt : 60.000

■ mce0 OUT bps
Max : 93.243G Min : 90.003G
Avg : 91.530G Last : 91.267G
Tot : 5.492T Cnt : 60.000

■ mce0 IN bps
Max : -480.257M Min : -507.484M
Avg : -492.367M Last : -507.484M
Tot : -29.542G Cnt : 60.000

■ Zero line
Max : 0.000 Min : 0.000
Avg : 0.000 Last : 0.000
Tot : 0.000 Cnt : 60.000

... 0 of 1 lines matched filter ...

Axis 1: IPv4 routes

□ IPv4 Routes
Max : NaN Min : NaN
Avg : NaN Last : NaN
Tot : NaN Cnt : 0.000

Frame: 1h, End: 2023-09-11T02:25Z[UTC], Step: 1m
Fetch: 46ms (L: 64.0, 32.0, 11.0; D: 3.8k, 2.0k, 660.0k)

NETFLIX


HealthCpu:c112.ord001.dev.ix.nflxvideo.net
Cpu Busy Percent
100.0
90.0
80.0
70.0
60.0
50.0
40.0
30.0
20.0
10.0
0.0
UTC 01:24 01:27 01:30 01:33 01:36 01:39 01:42 01:45 01:48 01:51 01:54 01:57 02:00 02:03 02:06 02:09 02:12 02:15 02:18 02:21 02:24
c112.ord001.dev.ix.nflxvideo.net CPU limit serve
Max : 80.000 Min : 80.000
Avg : 80.000 Last : 80.000
Tot : 4.800k Cnt : 60.000
c112.ord001.dev.ix.nflxvideo.net CPU limit fill
Max : 55.000 Min : 55.000
Avg : 55.000 Last : 55.000
Tot : 3.300k Cnt : 60.000
c112.ord001.dev.ix.nflxvideo.net CPU limit SOD
Max : NaN Min : NaN
Avg : NaN Last : NaN
Tot : NaN Cnt : 0.000
c112.ord001.dev.ix.nflxvideo.net CPU busy pct
Max : 43.733 Min : 41.520
Avg : 42.512 Last : 42.234
Tot : 2.551k Cnt : 60.000
FillWindow
Max : 0.000 Min : 0.000
Avg : 0.000 Last : 0.000
Tot : 0.000 Cnt : 60.000
Frame: 1h, End: 2023-09-11T02:25Z[UTC], Step: 1m
Fetch: 48ms (L: 16.0, 8.0, 5.0; D: 960.0, 488.0, 300.0k)

NETFLIX

Drew Gallatin
NAB Show, April 2022

NETFLIX

Bisect, and Bisect some more

- We bisected in the upstream FreeBSD tree, re-doing the merge into our tree for every bisection step.
- Then built, installed, and tested an image with production traffic
- Each bisection step took ~4 hours (1hr build & install, 2 hours to ramp OCA up and down, 1 hour to collect CPU use)

NETFLIX

NETFLIX

Found it!

9a7add6d01f3 init_main: Switch from sysinit array to SLIST

But this makes no sense. This just changes the sorting algorithm for kernel initialization functions..
This is one of the most famous commits in recent years

NETFLIX


Colin Percival @cperciva · Aug 20
FreeBSD (HEAD) no longer spends time running a bubblesort on its SYSINITs. We're now running a mergesort which is ~100x faster:
cgit.freebsd.org/src/commit/?id...
Colin Percival @cperciva · May 19
When the FreeBSD kernel boots in Firecracker (1 CPU, 128 MB RAM), it now spends 7% of its time running a bubblesort on its SYSINITs.
O(N^2) can bite hard when you're sorting over a thousand items. Time to replace the bubblesort with something faster.
8
123
482
102K


Y Hacker News new | threads | past | comments | ask | show | jobs | submit
▲ FreeBSD replaces bubblesort with mergesort on SYSINTs (twitter.com/cperciva)
339 points by gslin 63 days ago | hide | past | favorite | 297 comments

NETFLIX

SYSINIT

- kernel subsystem initializers, sorted by linker into alphabetical order
- There are 79 subsystems and 10 “order” hints within subsystems
- Sorted at boot by subsystem, then order.
- SYSINITS from the same subsystem with the same order *should* be able to run in any order

NETFLIX

The (easy) bug

- Original sort was not a bubble sort, but a selection sort
- This means ties are handled differently, and the order for all ties are different
- Colin and I both realized this when we both verified SYSINITS were called in a different order now using TSLOG

NETFLIX

The (easy) fix

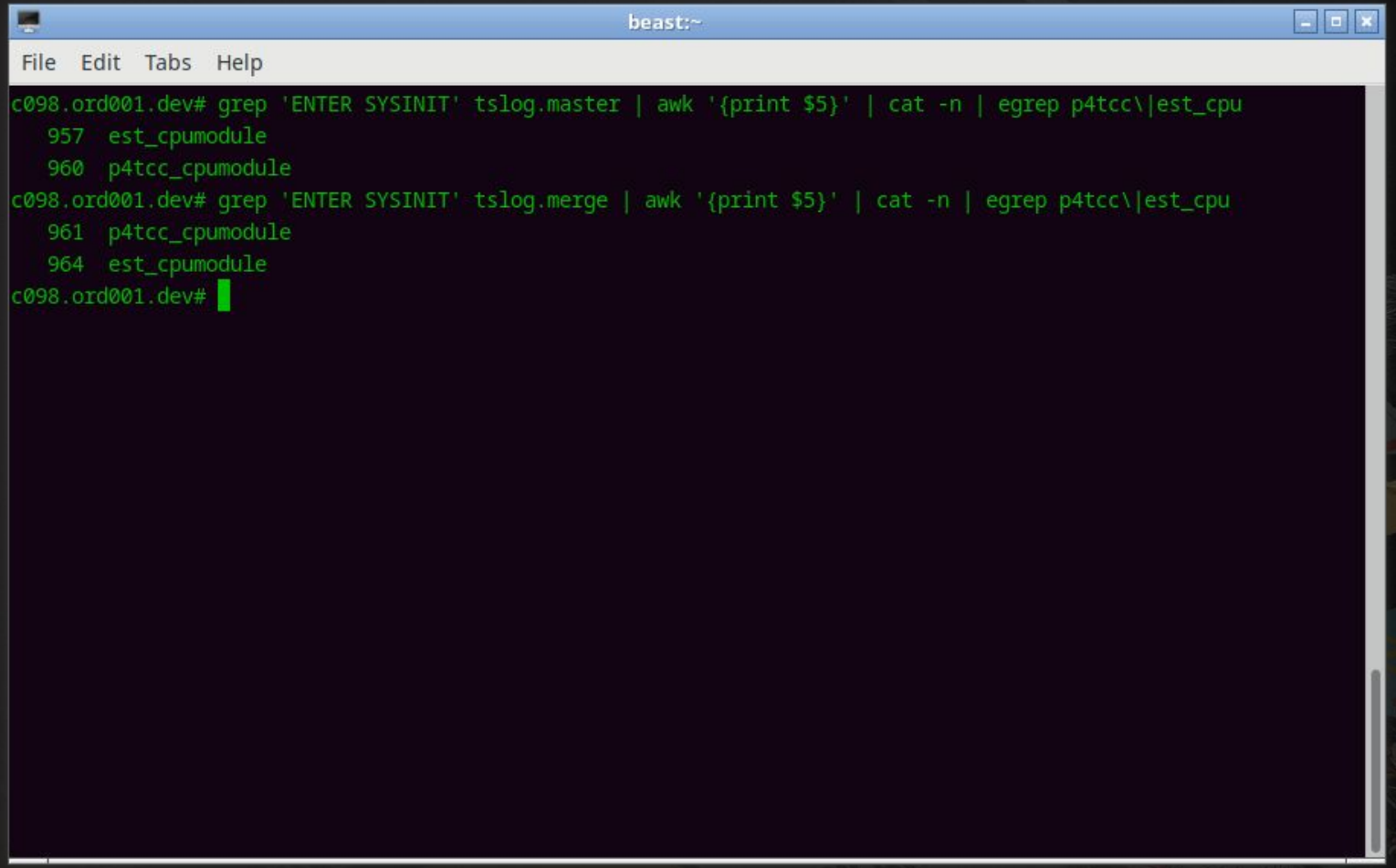
- 71679cf468ba init_main: Switch from SLIST to STAILQ, fix order
- This reverts the ordering to what we had before.
- But why does it work?

NETFLIX

Bisecting to find the real bug

- Reverse selected SYSINITS in multiple subsystems (by using a simple patch controlled by kernel args)
    - SI_SUB_DRIVERS
- Reverse selected SYSINITS in SI_SUB_DRIVERS

NETFLIX


beast:~
File Edit Tabs Help
c098.ord001.dev# grep 'ENTER SYSINIT' tslog.master | awk '{print $5}' | cat -n | egrep p4tcc\|est_cpu
957 est_cpumodule
960 p4tcc_cpumodule
c098.ord001.dev# grep 'ENTER SYSINIT' tslog.merge | awk '{print $5}' | cat -n | egrep p4tcc\|est_cpu
961 p4tcc_cpumodule
964 est_cpumodule
c098.ord001.dev#

NETFLIX

NETFLIX

The real bug

- Old p4tcc cpu frequency driver was put into control when it was initialized first
- The correct driver (est) never got a chance to attach
- Things had worked accidentally for years, due to linkerset alphabetical ordering
- A colleague (Warner Losh) is working on a real fix for CPU frequency driver selection

NETFLIX

p4tcc:

```
dev.cpu.0.freq_levels: 2599/-1 2274/-1 1949/-1
1624/-1 1299/-1 974/-1 649/-1 324/-1
```

est:

```
dev.cpu.0.freq_levels: 2601/145000 2600/145000
2500/137619 2400/130381 2300/123284 2200/116324
2100/109501 2000/102810 1900/97595 1800/91158
1700/84855 1600/78682 1500/72640 1400/66724
1300/62183 1200/56509
```

NETFLIX

Community interaction

- I reached out to Colin in an internal FreeBSD chatroom
- He remembered the change & was happy to help.
- He posted a fix for review within hours
- It landed the next day, and I cherry-picked it into our codebase

NETFLIX

Community benefits

- Netflix noticed this bug almost immediately after it hit the tree. We were the first to notice a regression and be able to attribute it to this change.
- At least one other driver bug was “fixed” by reverting to the old order (amdtemp)

NETFLIX

Netflix benefits

- This was a bug that made no sense, and required bisection.
- Bisecting 3 weeks of changes took days. Had we moved between -stable branches, bisecting 3+ years of changes could have taken weeks

NETFLIX

Netflix benefits

- Since we found the bug within a week or so of it hitting the tree, the developer responsible was incredibly responsive. All the details were fresh in his memory.
- Contrast this to somebody reporting a bug in something you did 3-4 years ago.

Thank you

Slides at: https://people.freebsd.org/~gallatin/talks/

NETFLIX