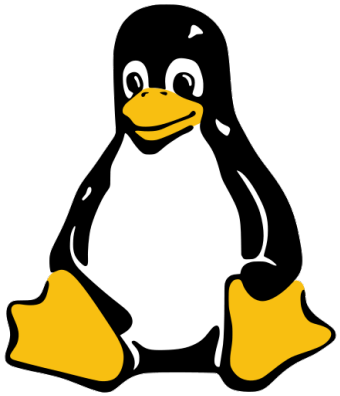
TRACING SUMMIT
EUROPE

Oct, 2014

From DTrace To Linux:



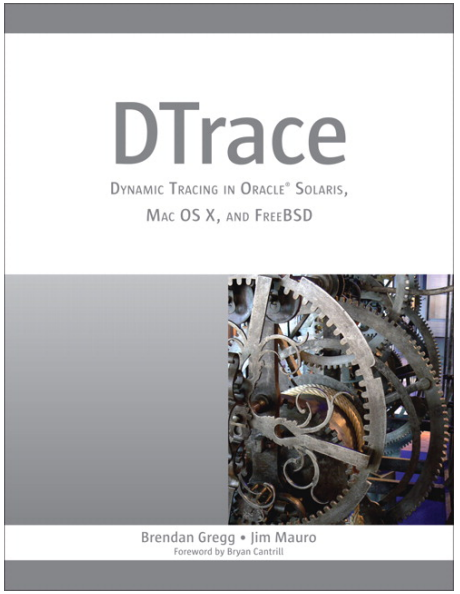
What can Linux learn from DTrace?

Brendan Gregg
Senior Performance Architect

bgregg@netflix.com
@brendangregg

Brendan Gregg

- DTrace contributions include:
  - Primary author of the DTrace book
  - DTraceToolkit
  - dtrace-cloud-tools
  - DTrace network providers
- I now work on Linux at Netflix
  - using: ftrace, perf_events, SystemTap, ktap, eBPF, …
  - created: perf-tools, msr-cloud-tools
- Opinions in this talk are my own



Agenda

1. DTrace
   - What is DTrace, really?
   - Who is DTrace for, really?
   - Why doesn't Linux have DTrace?
   - What worked well?
   - What didn't work well?
2. Linux Tracers
   - ftrace, perf_events, eBPF, …

Topics include adoption, marketing, technical challenges, and our usage at Netflix.

What is DTrace, really?

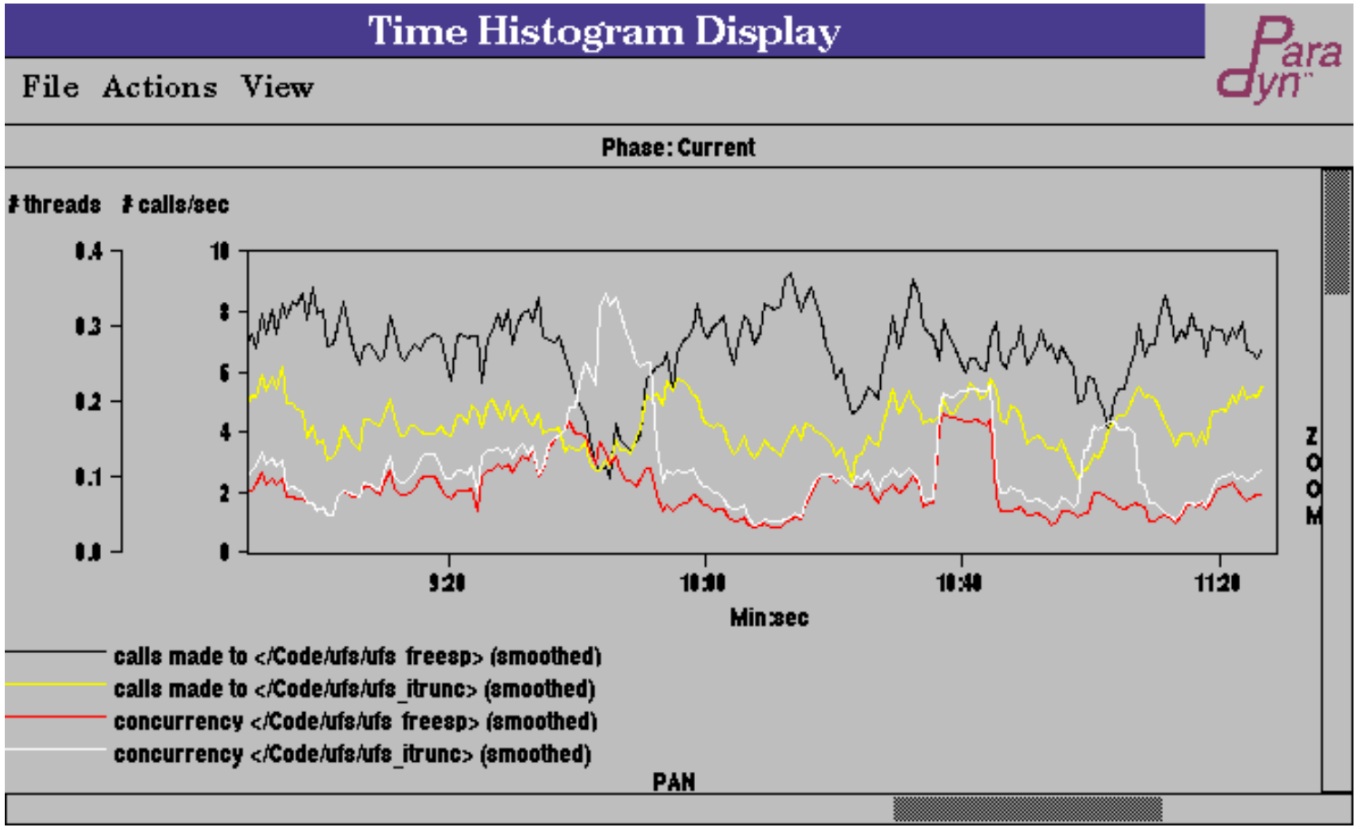
Technology

+

Marketing

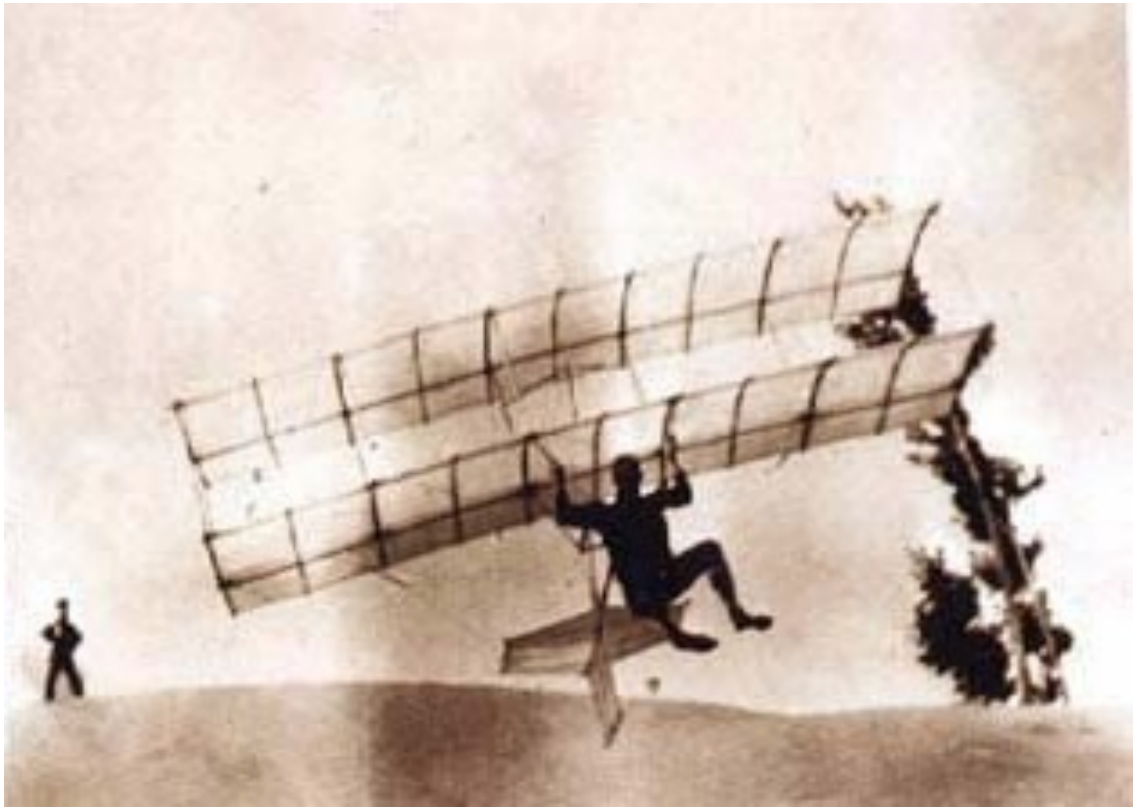
(Like many other company products)

Prior Technology



Kerninst: kernel dynamic tracing, Solaris 2.5.1, 1999

Prior Technology



Early dynamic tracers weren’t safe

Prior Technology

- Also:
  - Sun's TNF
  - DProbes: user + kernel dynamic tracing
  - Linux Trace Toolkit (LTT)
  - Others, including offline binary instrumentation
- DProbes and LTT were combined in Nov 2000, but not integrated into the Linux kernel[1]
- Sun set forth to produce a production-safe tool

[1] http://lkml.iu.edu/hypermail/linux/kernel/0011.3/0183.html

Sun delivers Unix shocker with DTrace

It slices, it dices, it spins, it whirls

By Ashlee Vance, 8 Jul 2004

Analysis Try to imagine a geeky version of famed salesman Ron Popeil. Keep Popeil's exuberance, keep his pitchman savvy and keep his verbal overflow. Then erase his age, sturdy frame and Ronco Food Dehydrator and replace all this with a young, lanky kernel engineer hawking something called DTrace, and you have Bryan Cantrill.

Cantrill is one of three Sun Microsystems Solaris engineers who developed DTrace

Technology

- DTrace:
  - Safe for production use
    - You might step on your foot (overhead), but you won't shoot it off
  - Dynamic tracing, static tracing, and profiling
  - User- and kernel-level, unified
  - Programmatic: filters and summaries
  - Solved countless issues in dev and prod
- That's what DTrace is for *me*
  - An awesome technology, often needed to root cause kernel & app issues
- But for most people….

A Typical Conversation…

“Does Linux have DTrace yet?”

“No.”

“That’s a pity”

“Why?”

“DTrace is awesome!”

“Why, specifically?”

“I’m not sure”

“Have you used it?”

“No.”

Marketing

Early Marketing

Early Marketing

- DTrace had *awesome* marketing
  - People still want it but don't really know why
- Early marketing: traditional, $$$
  - Great marketing product managers
    - 10 Moves Ahead campaign: airports, stations, etc.
  - Sun sales staff pitched DTrace directly
  - Sun technology evangelists
- Benefits
  - Not another Sun tech no one knew about
  - Compelled people to learn more, try it out

Marketing Evolved

- Sun marketing become innovative
  - Engineering blogs, BigAdmin
  - Marketing staff who used and understood DTrace
    - Who could better articulate its value
  - Marketing more directly from the engineers

DTrace for System Administrators, with Brendan Gregg

Rick Ramsey, Solaris Community Leader, interviews Brendan Gregg (OTN Live 2010)

Later Marketing

Later Marketing

- Many initiatives by Deirdré Straughan:
  - Social media, blogs, events, the ponycorn mascot, ...
  - Video and share everything: all meetups, talks
- Blogs:
  - including http://dtrace.org; my own > 1M views
- Books:
  - my own > 30k sold
- Videos:
  - me shouting while DTracing disks, ~1M views
- Language support exposed new communities to DTrace

dtrace.conf

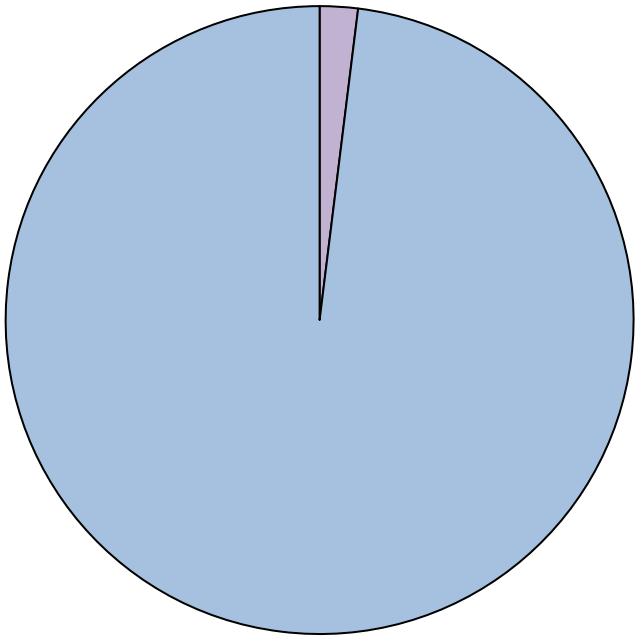

3
PACK
dtrace

???

Who is DTrace for, really?

DTrace end-users: Current

Estimated

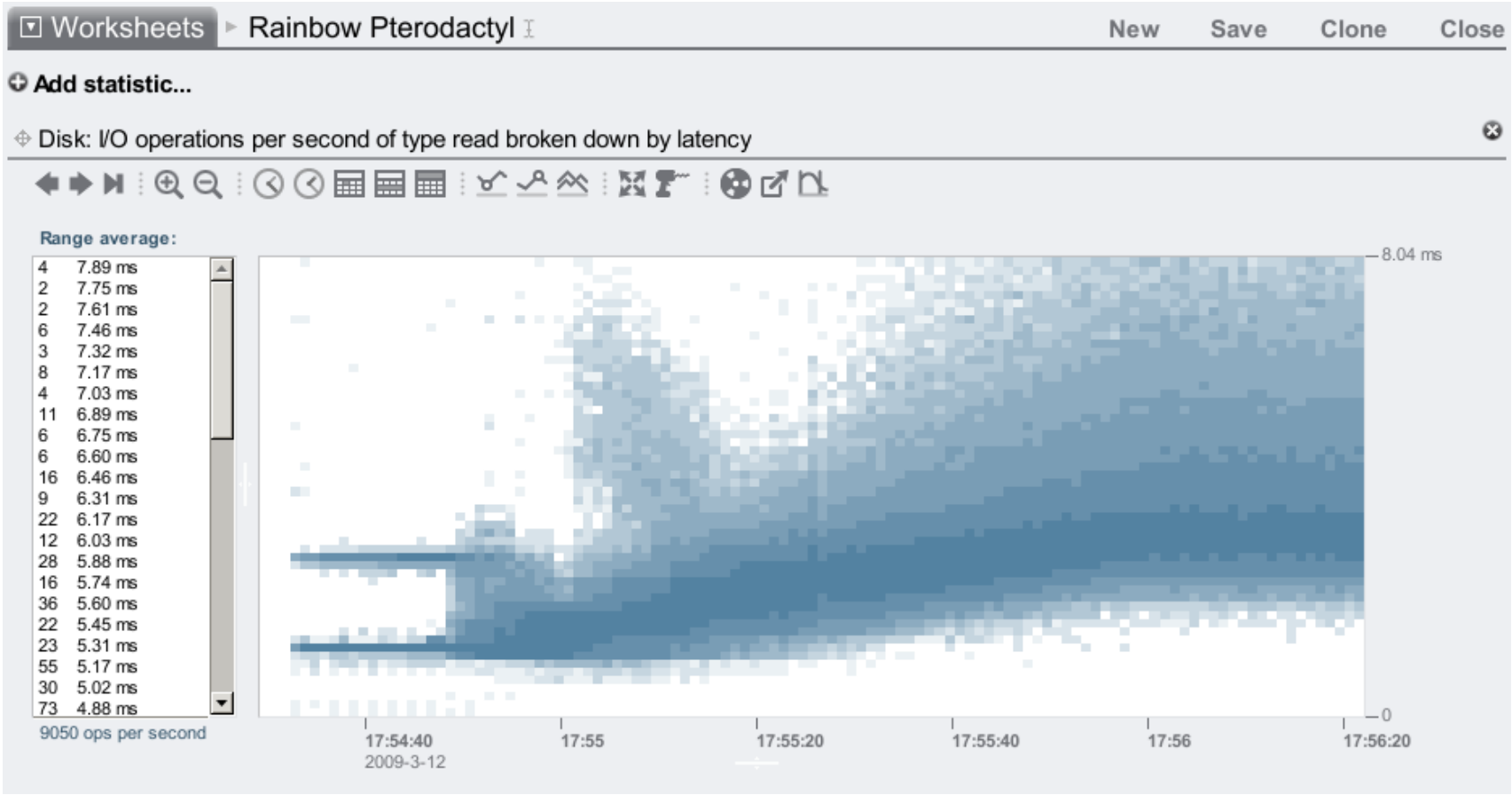


Note: 91.247% of statistics are made up

DTrace end-users: Current

- DTrace guide users: ~100
  - Understand the ~400 page Dynamic Tracing Guide
  - Develop their own scripts from scratch
  - Understand overhead intuitively
- Script end-users: ~5000
  - DTraceToolkit, Google
  - Run scripts. Some tweaks/customizations.

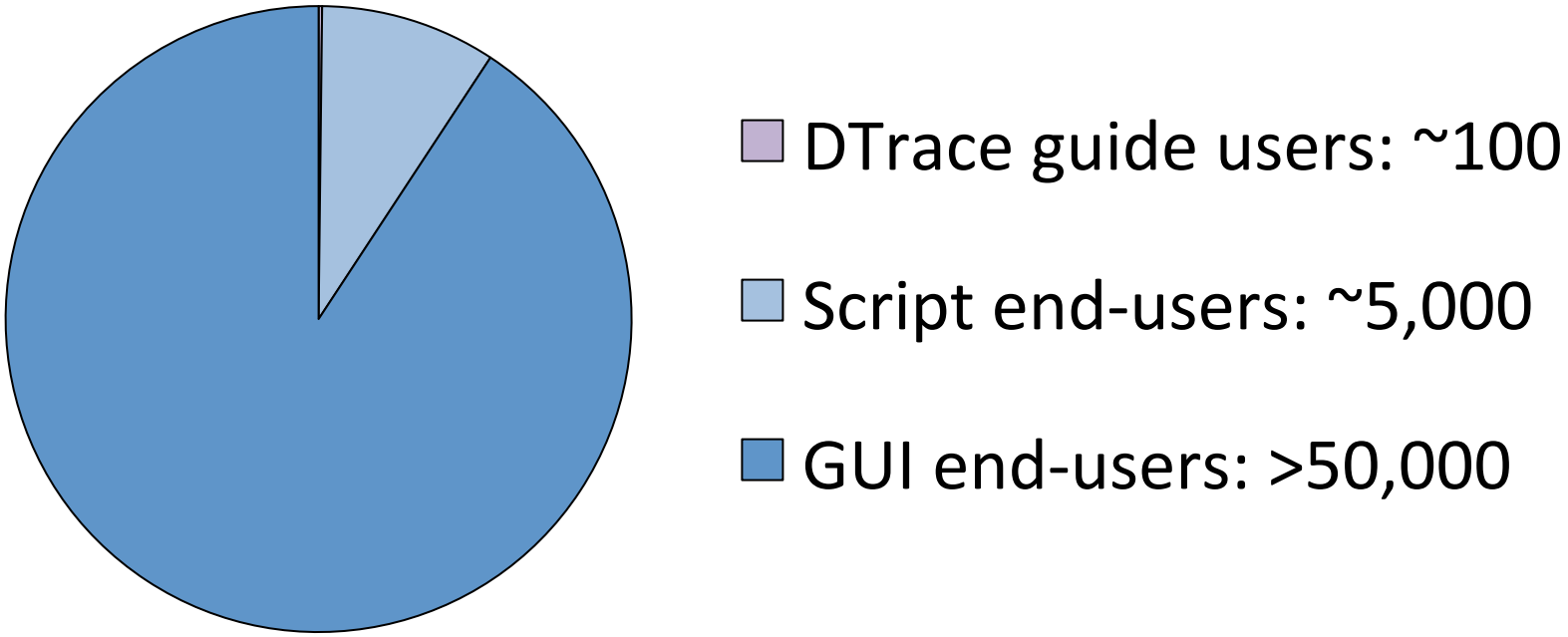
DTrace end-users: Future



eg, Oracle ZFS Storage Appliance Analytics

DTrace end-users: Future

Possible Future



Company Usage

Company Usage

- Practical usage for most companies:
  - A) A performance team (or person)
    - Acquires useful scripts
    - Develops custom scripts
  - B) The rest of the company asks (A) for script/help
    - They need to know what's possible, to know to ask
  - Or, you buy/develop a GUI that everyone can use
- There are some exceptions

Why doesn’t Linux have DTrace?

Why doesn’t Linux have a DTrace equivalent?

4 Answers…

1. It does (sort of)

ftrace

perf_events

1. It does (sort of)

- Linux has changed
  - In 2005, numerous Linux issues were difficult or impossible to solve. Linux *needed* a DTrace equivalent.
  - By 2014, many of these are now solvable, especially using ftrace, perf_events, kprobes, uprobes: all part of the Linux kernel

2. Technical


```
semantic error: missing x86_64
kernel/module debuginfo
```

2. Technical

- Linux is a more difficult environment
  - Solaris always has symbols, via CTF, which DTrace uses for dynamic tracing
  - Linux doesn't always have symbols/debuginfo

3. Linux isn't a Company

"All the wood behind one arrow"

– Scott McNealy, CEO, Sun Microsystems

3. Linux isn't a Company

- Linus can refuse patches, but can't stop projects
  - The tracing wood is split between many arrows
    - ftrace, perf_events, eBPF, SystemTap, ktap, LTTng, …
  - And we are a small community: there's not much wood to go around!

4. No Trace Race

4. No Trace Race

- Post 2001, Solaris was losing ground to Linux. Sun desperately needed differentiators to survive
  - Three top Sun engineers spent years on DTrace
  - Sun marketing gave it their best shot…
- This circumstance will never exist again
  - For Linux today, it would be like having Linus, Ingo, and Steven do tracing full-time for three years, followed by a major marketing campaign
- There may never be another trace race. Unless…

Why doesn’t Linux have DTrace itself?

2 Answers…

1. The CDDL

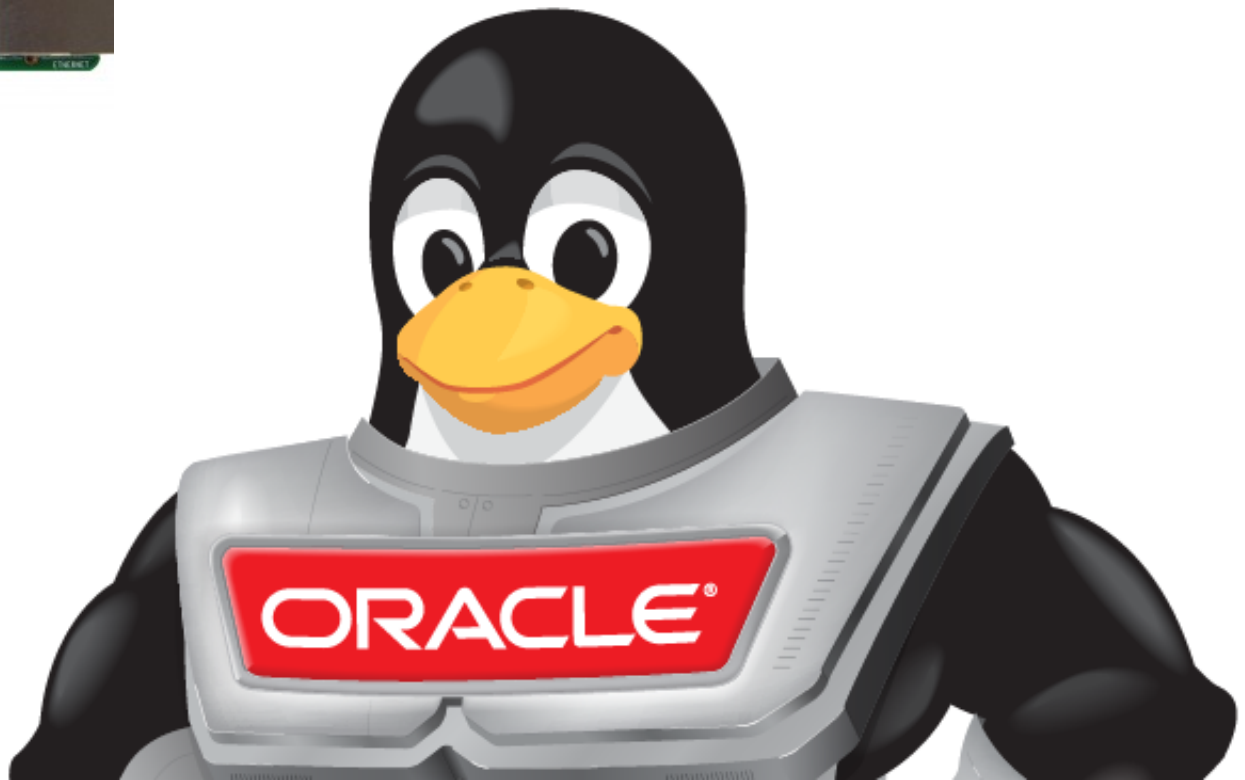
From: Claire Giordano <claire.giordano@sun.com>
To: license-discuss@opensource.org [Open Source Initiative]
Subject: For Approval: Common Development and Distribution License (CDDL)
Date: Wed, 01 Dec 2004 19:47:39 -0800
[…]
Like the MPL, the CDDL is not expected to be compatible with the GPL, since it contains requirements that are not in the GPL (for example, the "patent peace" provision in section 6). Thus, it is likely that files released under the CDDL will not be able to be combined with files released under the GPL to create a larger program.
[…]
CDDL Team, Sun Microsystems

Source: http://lwn.net/Articles/114840/

1. The CDDL

- Linux traditionally includes the tracer/profiler in the (GPL) kernel, but the DTrace license is CDDL
  - Out-of-tree projects have maintenance difficulties
  - Oracle (who own the DTrace copyrights) could relicense it as GPL, but haven't, and may never do this
- Note that ZFS on Linux is doing well, despite being CDDL, and out of tree

2. DTrace ports



2. DTrace ports

- There are two ports, but both currently incomplete
- A) https://github.com/dtrace4linux/linux:
  - Mostly one UK developer, Paul Fox, as a hobby since 2008 (when he isn't developing on the Rasberry Pi)
- B) Oracle Linux DTrace:
  - Open source kernel, closed source user-level ($)
    - We pay for monitoring tools; why not this too?
  - Experienced engineers, test suite focused
  - Had been good progress, but no updates for months

What with DTrace worked well?

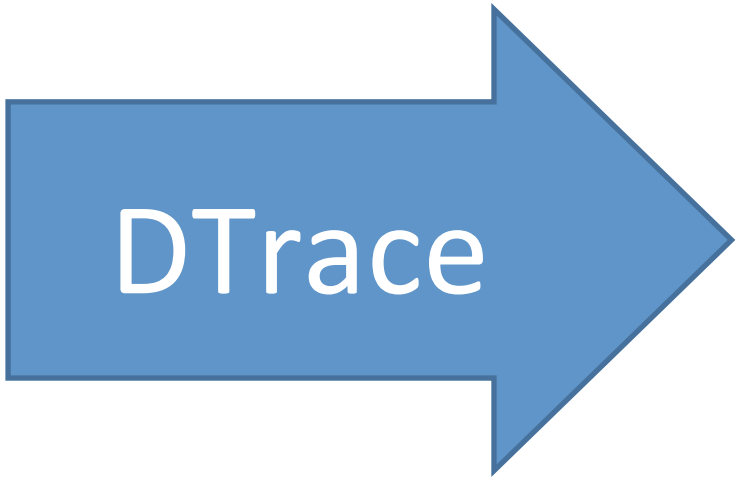
5 Key items…

1. Production Safety

1. Production Safety

- DTrace architecture
  - Restricted probe context: no kernel facility calls, restricted instructions, no backwards branches, restricted loads/stores
  - Heartbeat: aborted due to systemic unresponsiveness
- DTrace Test Suite
  - Hundreds of tests
- Linux is learning this:
  - Oracle Linux DTrace is taking the test suite seriously
  - ftracetest

2. All the wood behind one arrow



2. All the wood behind one arrow

- Can Linux learn this?
  - Can we vote some off the Linux tracing island?
- At least, no new tracers in 2015, please!

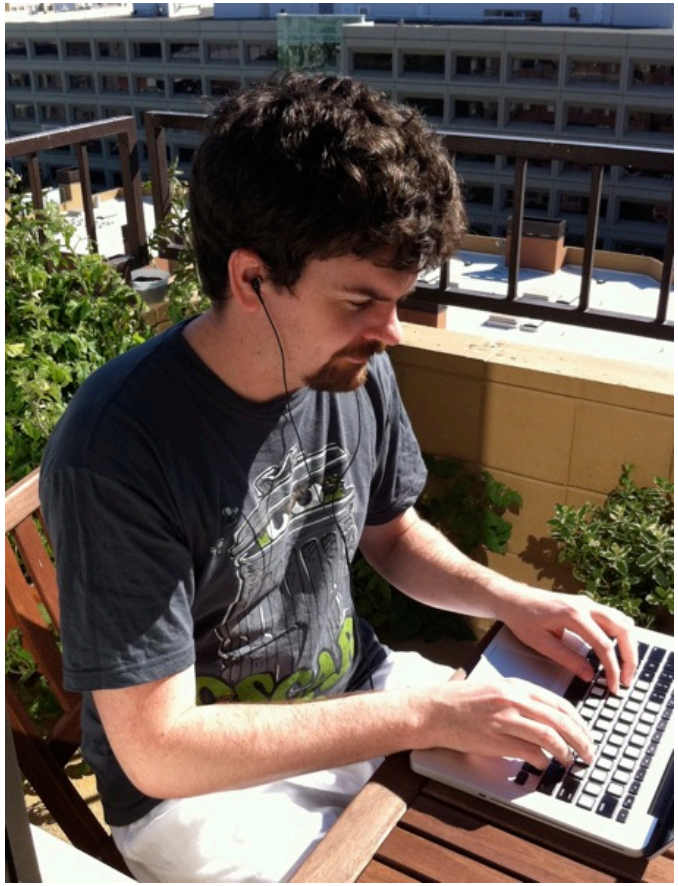
3. In-Kernel Aggregations

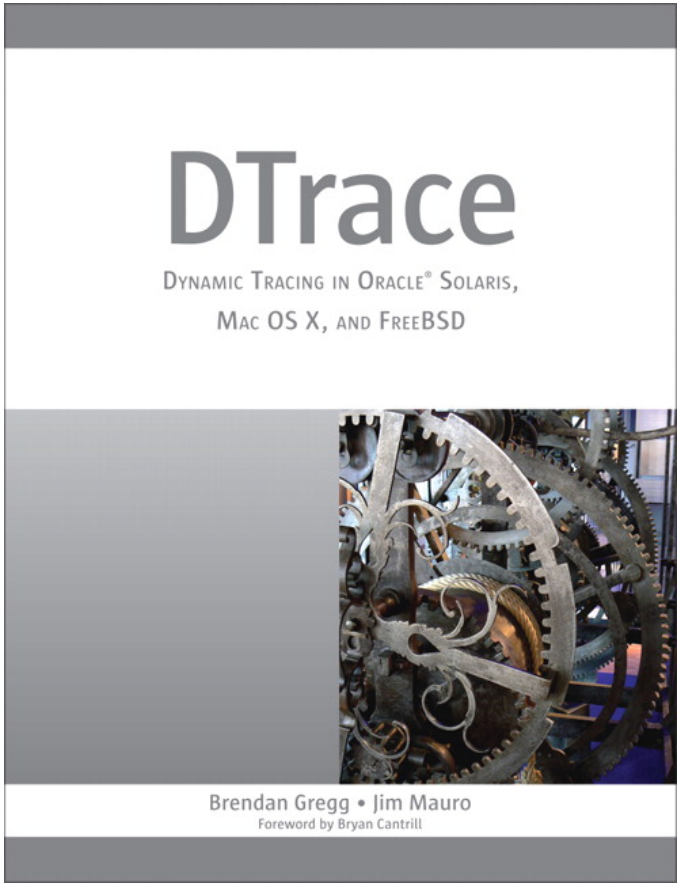
```
     value  ------------- Distribution ------------- count
      4096 |                                         0
      8192 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 1085
     16384 |@@@@@@@@@@@@@@@@                         443
     32768 |@@                                       98
     65536 |                                         5
    131072 |                                         1
    262144 |                                         1
    524288 |                                         0
   1048576 |                                         11
   2097152 |                                         0
```

3. In-Kernel Aggregations

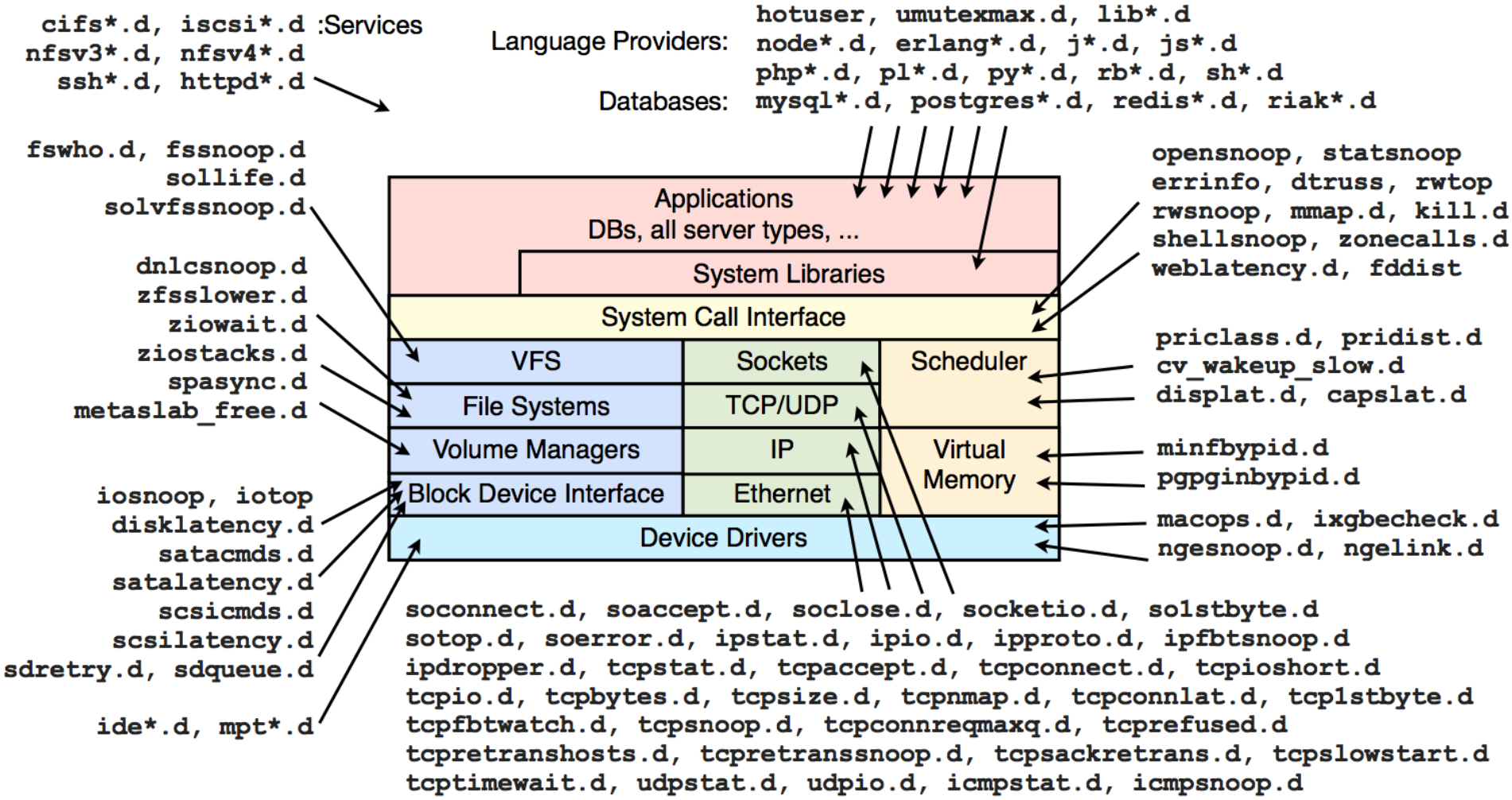
- Changed how investigations are conducted
  - rapid, live analysis
- Low overhead:
  - per-CPU storage, asynchronous kernel->user transfers
- Key uses:
  - summary statistics: count, avg, min, max
  - histograms: quantize, lquantize
  - by-key: execname, kernel stacks, user stacks
- Linux can learn:
  - Need aggregations (eBPF maps, SystemTap, ktap?)

4. Many Example Scripts



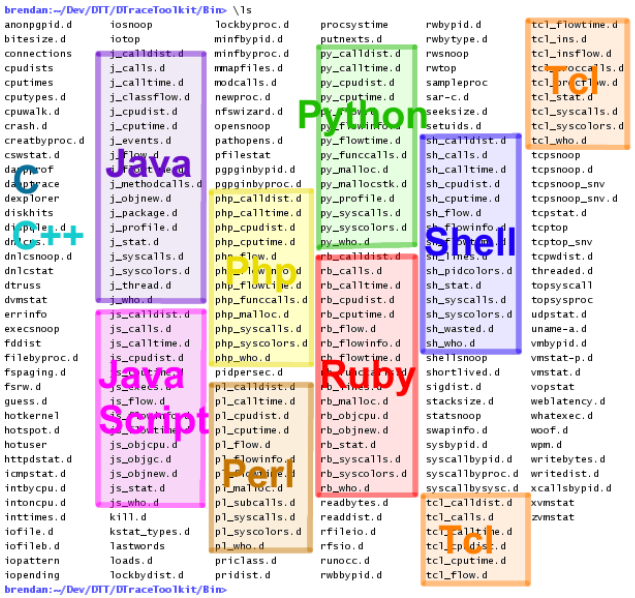


4. Many Example Scripts



4. Many Example Scripts

- Scripts serve many needs:
  - tools: ready to run
  - examples: learn by-example
  - marketing: each is a use case
- DTrace book scripts
  - 150+ short examples
- DTraceToolkit
  - 230 more scripts
  - all have man pages, example files, and are tested
  - An essential factor in DTrace's adoption



DTraceToolkit scripts

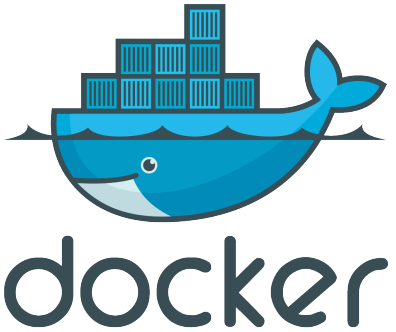
4. Many Example Scripts

- Linux can learn:
  - Many users will just run scripts, not write them
  - People want good short examples
  - If they aren't tested, they don't work
    - It's easy to generate metrics that kind-of work; it's hard to make them reliable for different workloads.
  - Maintenance of dynamic tracing scripts is painful
    - The instrumented code can change
    - Need more static tracepoints

5. Marketing

5. Marketing

- DTrace was effectively marketed in many ways
  - Traditional, social, blogs, scripts, ponycorn, …
- Linux has virtually no marketing for its tracers
  - ftrace is great, *if* you ever discover it; etc.
  - Marketing spend is on commercial products instead
- Linux can learn to market what it has
  - Tracers may also benefit from "a great name and a cute logo"[1]
  - "eBPF" is not catchy, and doesn't convey meaning



[1] http://thenewstack.io/why-did-docker-catch-on-quickly-and-why-is-it-so-interesting/

Cute Tracing Logos

ftrace

perf_events

SystemTap

ktap

LTTng

dtrace4linux

Ponies by Deirdré Straughan, using: http://generalzoi.deviantart.com pony creator

Other Things

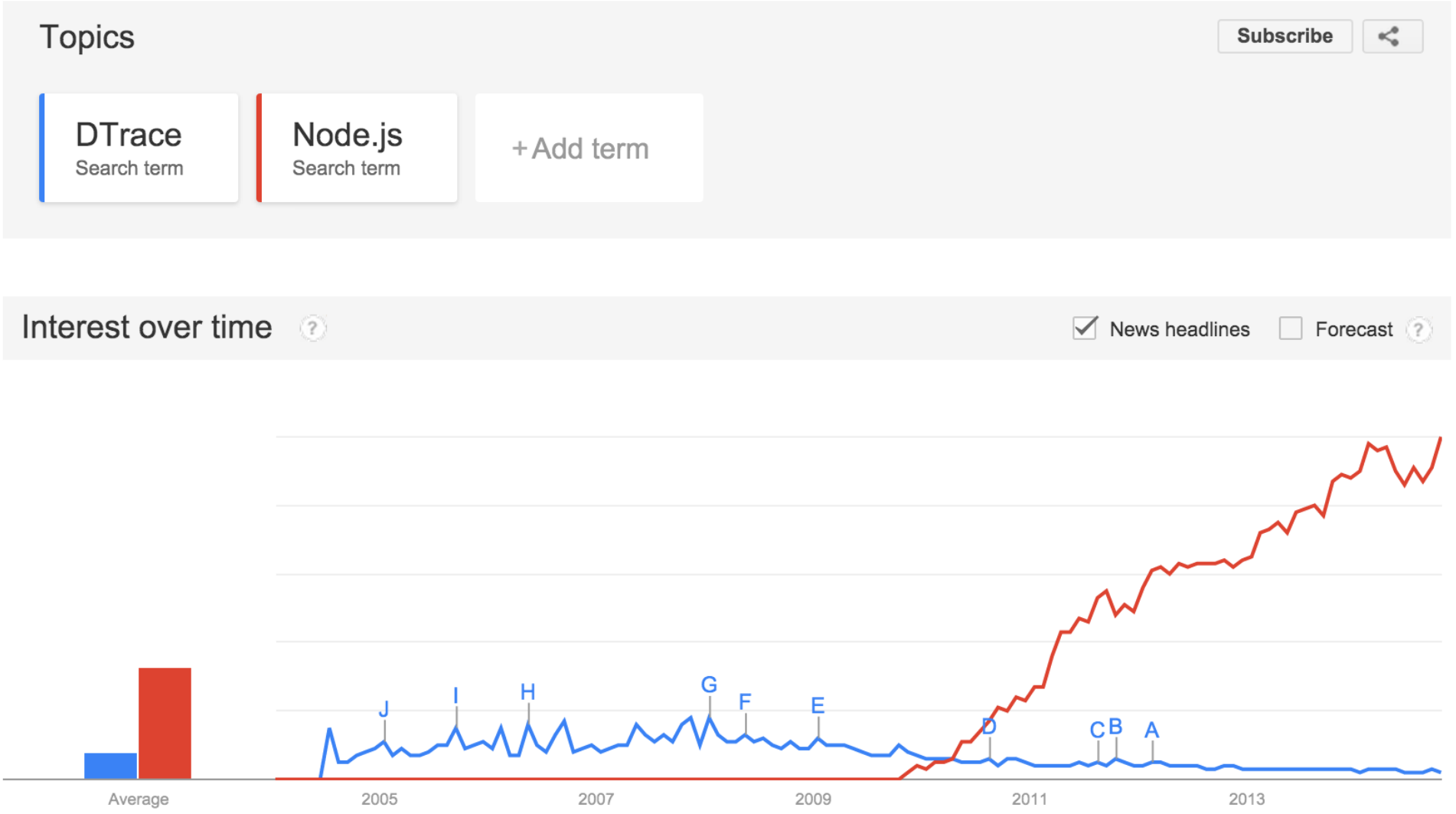
- Programmable/scriptable
- Built-in stability semantics

What with DTrace didn’t work well?

What with DTrace didn't work well?

5 Key Issues…

1. Adoption


Topics
Subscribe
DTrace
Search term
Node.js
Search term
+ Add term
Interest over time
News headlines
Forecast
J
I
H
G
F
E
D
C
B
A
Average
2005
2007
2009
2011
2013

1. Adoption

- Few customers ever wrote DTrace scripts
  - DTrace should have been used more than it was
  - Sun's "killer" tool just wasn't
  - Better pickup rate with developers, not sysadmins
- Many customers just ran my scripts
  - Not ideal, but better than nothing
  - This wasn't what many at Sun dreamed
- Internal adoption was slow, limited
  - Sun could have done much more, but didn't
- The problem was knowing what to do with it
  - The syntax was the easy part

1. Adoption

- Linux can learn:
  - Adoption is about more than just the technology
    - Documentation, marketing, training, community
  - Teaching what it does is more important than how
    - Everyone needs to know when to ask for it, not necessarily how to use it
  - Needs an adoption curve (not a step function)
    - Tools, one-liners, short scripts, …

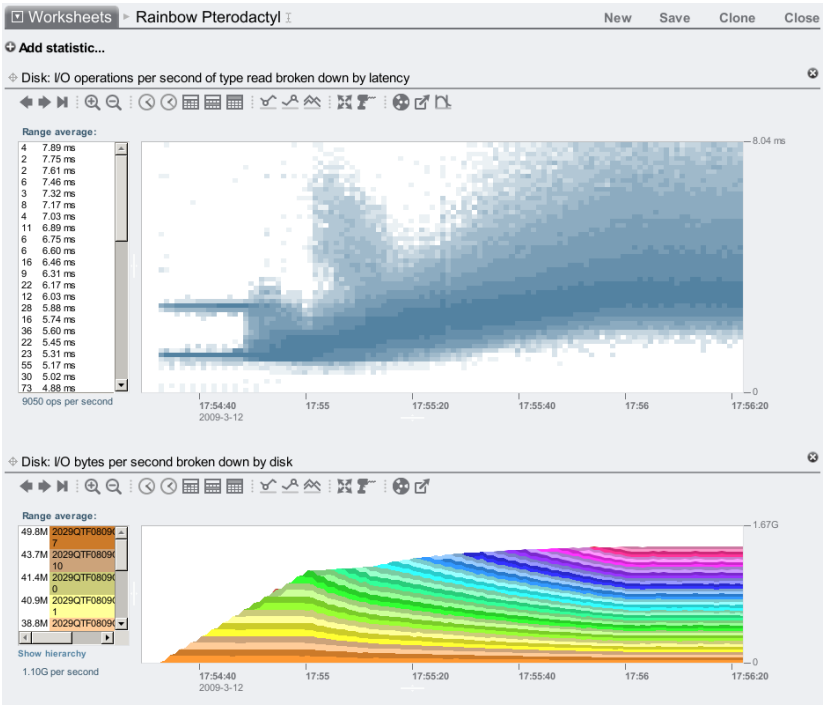
2. Training

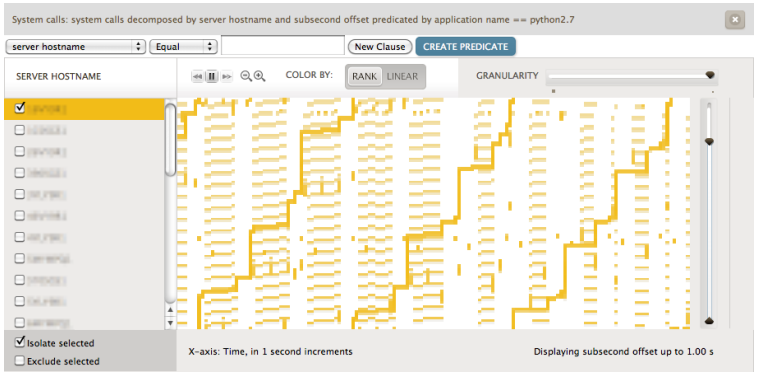

Sun
microsystems
We're the dot in .com
COURSE COMPLETION CERTIFICATE
This is to certify that
Brendan Gregg
Has Completed the Sun Educational course
DTrace is a Solaris differentiator

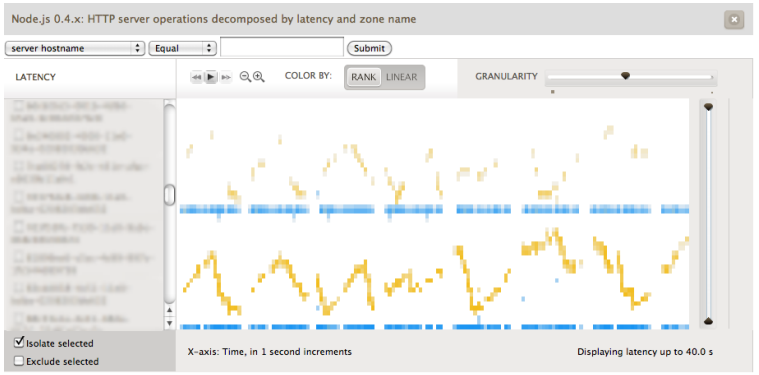
2. Training

- Early training was not very effective
  - Sun began including the DTraceToolkit in courses, with better success
- It gradually improved
  - The last courses I developed and taught (after Sun) used simulated problems for the students to solve *on their own* with DTrace
- Linux can learn:
  - Lab-based training is most effective. Online tutorials?

3. GUIs


Worksheets
Rainbow Pterodactyl
New
Save
Clone
Close
Add statistic...
Disk: I/O operations per second of type read broken down by latency
Range average:
4 7.89 ms
2 7.75 ms
2 7.61 ms
6 7.46 ms
3 7.32 ms
8 7.17 ms
4 7.03 ms
11 6.89 ms
6 6.75 ms
6 6.60 ms
16 6.46 ms
9 6.31 ms
22 6.17 ms
12 6.03 ms
28 5.88 ms
16 5.74 ms
36 5.60 ms
22 5.45 ms
23 5.31 ms
55 5.17 ms
30 5.02 ms
73 4.88 ms
8.04 ms
0
9050 ops per second
17:54:40
2009-3-12
17:55
17:55:20
17:55:40
17:56
17:56:20
Disk: I/O bytes per second broken down by disk
Range average:
49.8M
43.7M
41.4M
40.9M
38.8M
Show hierarchy
1.10G per second
1.67G
0
17:54:40
2009-3-12
17:55
17:55:20
17:55:40
17:56
17:56:20


System calls: system calls decomposed by server hostname and subsecond offset predicated by application name == python2.7
server hostname
Equal
New Clause
CREATE PREDICATE
SERVER HOSTNAME
COLOR BY:
RANK
LINEAR
GRANULARITY
Isolate selected
Exclude selected
X-axis: Time, in 1 second increments
Displaying subsecond offset up to 1.00 s


Node.js 0.4.x: HTTP server operations decomposed by latency and zone name
server hostname
Equal
Submit
LATENCY
COLOR BY:
RANK
LINEAR
GRANULARITY
Isolate selected
Exclude selected
X-axis: Time, in 1 second increments
Displaying latency up to 40.0 s

3. GUIs

- Dozens of performance monitoring products, but almost no meaningful DTrace support
- A couple of exceptions:
  - Oracle ZFS Storage Appliance Analytics
    - Formally the Sun Storage 7000 Analytics
    - Should be generalized. Oracle Solaris 11.3?
  - Joyent Cloud Analytics

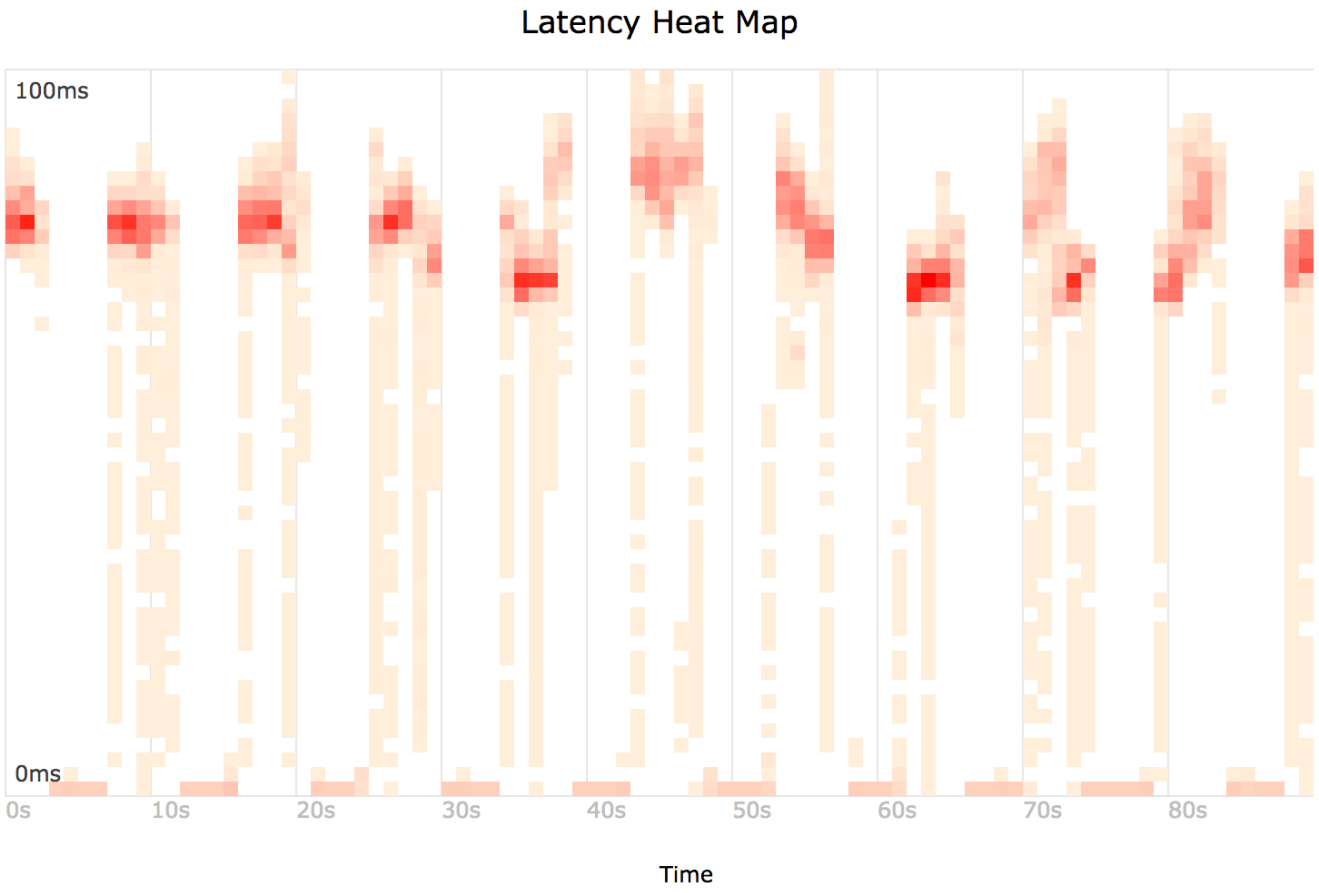
3. GUIs

- Linux can learn:
  - Real adoption possible through scripts & GUIs
  - Use the GUI to add value to the data
    - Heat maps: latency, utilization, offset
    - Flame graphs
    - Time series thread visualizations (Trace Compass)
    - ie, not just line graphs!
  - Commercial GUI products have marketing budget
    - Application perf monitoring was $2.4B in 2013[1]

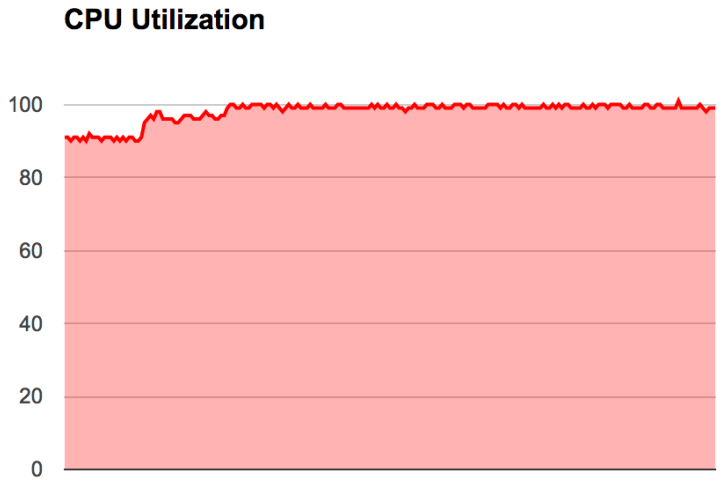
[1] https://www.gartner.com/doc/2752217/market-share-analysis-application-performance

3. GUIs

- Heat maps are an example must-have use case for trace data



4. Overheads

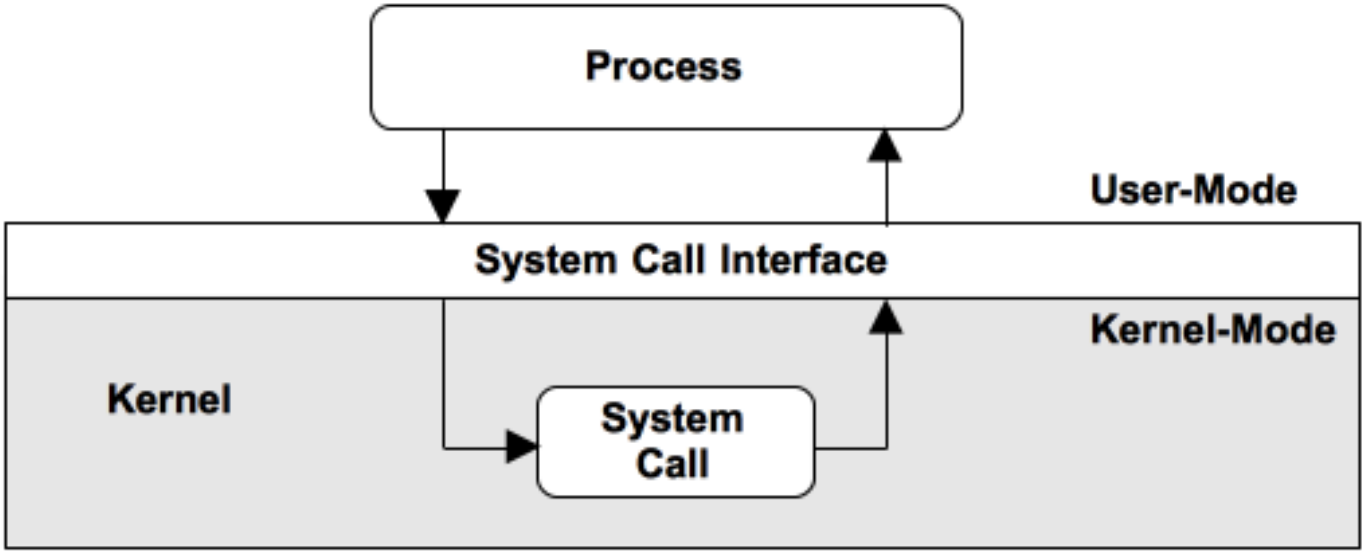


While the DTrace technology is awesome,
it does have some minor technical challenges as well

4. Overheads

- While optimized, for many targets the DTrace CPU overheads can still be too high
  - Scheduler tracing, memory allocation tracing
  - User-level dynamic tracing (fast trap)
  - VM probes (eg, Java disables some probes by default)
  - 10 GbE Network I/O, etc…
- In some cases it doesn't matter
  - Desperation: system already melting down
  - Troubleshooting in dev: speed not a concern
- Linux can learn:
  - Speed can matter, faster makes more possible

5. Syscall Provider



5. Syscall Provider

- Solaris DTrace instrumented the trap table, and called it the syscall provider
  - Which is actually an unstable interface
  - Breaks between Solaris versions
    - And really broke in Oracle Solaris 11
  - Other weird caveats
- Linux can learn:
  - syscalls are the #1 target for users learning system tracers. The API should be easy and stable.

Other Issues

- The lack of:
  - Bounded loops (like SystemTap)
  - Kernel instruction tracing (like perf_events)
  - Easy PMC interface (like perf stat)
  - Aggregation key/value access (stap, ktap, eBPF)
  - Kernel source (issue for Oracle Solaris only)
- 4+ second startup times
  - Several Linux tracers start instantly

From DTrace to Linux Tracers (2014)

NETFLIX

- Massive AWS EC2 Linux cloud, with FreeBSD appliances for content delivery
- Performance is critical: >50M subscribers
- Just launched in Europe!

System Tracing at Netflix

- Present:
  - ftrace can serve many needs
  - perf_events some more, esp. with debuginfo
  - SystemTap as needed, esp. for Java
  - ad hoc other tools
- Future:
  - ftrace/perf_events/ktap with eBPF, for a fully featured and mainline tracer?
  - One of the other tracers going mainline?
- Summarizing 4 tracers…

1. ftrace

1. ftrace

- Tracing and profiling: /sys/kernel/debug/tracing
  - added by Steven Rostedt and others since 2.6.27, and already enabled on our servers (3.2+)
- Experiences:
  - very useful capabilities: tracing, counting
  - surprising features: graphing (latencies), filters
- Front-end tools to ease use
  - https://github.com/brendangregg/perf-tools
  - WARNING: these are unsupported hacks
  - There's also the trace-cmd front-end by Steven
- 4 examples…

perf-tools: iosnoop

- Block I/O (disk) events with latency:

```
# ./iosnoop –ts
Tracing block I/O. Ctrl-C to end.
STARTs          ENDs            COMM       PID   TYPE DEV    BLOCK     BYTES LATms
5982800.302061 5982800.302679 supervise  1809  W    202,1  17039600  4096   0.62
5982800.302423 5982800.302842 supervise  1809  W    202,1  17039608  4096   0.42
5982800.304962 5982800.305446 supervise  1801  W    202,1  17039616  4096   0.48
5982800.305250 5982800.305676 supervise  1801  W    202,1  17039624  4096   0.43
[…]
```

```
# ./iosnoop –h
USAGE: iosnoop [-hQst] [-d device] [-i iotype] [-p PID] [-n name] [duration]
                 -d device       # device string (eg, "202,1)
                 -i iotype       # match type (eg, '*R*' for all reads)
                 -n name         # process name to match on I/O issue
                 -p PID          # PID to match on I/O issue
                 -Q              # include queueing time in LATms
                 -s              # include start time of I/O (s)
                 -t              # include completion time of I/O (s)
                 -h              # this usage message
                 duration        # duration seconds, and use buffers
[…]
```

perf-tools: iolatency

- Block I/O (disk) latency distributions:

```
# ./iolatency
Tracing block I/O. Output every 1 seconds. Ctrl-C to end.

  >=(ms) .. <(ms)   : I/O      |Distribution                          |
      0 -> 1        : 1144     |######################################|
      1 -> 2        : 267      |#########                             |
      2 -> 4        : 10       |#                                     |
      4 -> 8        : 5        |#                                     |
      8 -> 16       : 248      |#########                             |
     16 -> 32       : 601      |####################                  |
     32 -> 64       : 117      |####                                  |
[…]
```

- User-level processing sometimes can't keep up
  - Over 50k IOPS. Could buffer more workaround, but would prefer in-kernel aggregations

perf-tools: opensnoop

- Trace open() syscalls showing filenames:

```
# ./opensnoop -t
Tracing open()s. Ctrl-C to end.
TIMEs             COMM          PID      FD FILE
4345768.332626    postgres      23886   0x8 /proc/self/oom_adj
4345768.333923    postgres      23886   0x5 global/pg_filenode.map
4345768.333971    postgres      23886   0x5 global/pg_internal.init
4345768.334813    postgres      23886   0x5 base/16384/PG_VERSION
4345768.334877    postgres      23886   0x5 base/16384/pg_filenode.map
4345768.334891    postgres      23886   0x5 base/16384/pg_internal.init
4345768.335821    postgres      23886   0x5 base/16384/11725
4345768.347911    svstat        24649   0x4 supervise/ok
4345768.347921    svstat        24649   0x4 supervise/status
4345768.350340    stat          24651   0x3 /etc/ld.so.cache
4345768.350372    stat          24651   0x3 /lib/x86_64-linux-gnu/libselinux…
4345768.350460    stat          24651   0x3 /lib/x86_64-linux-gnu/libc.so.6
4345768.350526    stat          24651   0x3 /lib/x86_64-linux-gnu/libdl.so.2
4345768.350981    stat          24651   0x3 /proc/filesystems
4345768.351182    stat          24651   0x3 /etc/nsswitch.conf
[…]
```

perf-tools: kprobe

- Just wrapping capabilities eases use. Eg, kprobes:

```
# ./kprobe 'p:open do_sys_open filename=+0(%si):string' 'filename ~ "*stat"'
Tracing kprobe myopen. Ctrl-C to end.
        postgres-1172  [000] d... 6594028.787166: open: (do_sys_open
+0x0/0x220) filename="pg_stat_tmp/pgstat.stat"
        postgres-1172  [001] d... 6594028.797410: open: (do_sys_open
+0x0/0x220) filename="pg_stat_tmp/pgstat.stat"
        postgres-1172  [001] d... 6594028.797467: open: (do_sys_open
+0x0/0x220) filename="pg_stat_tmp/pgstat.stat”
^C
Ending tracing...
```

- By some definition of “ease”. Would like easier symbol usage, instead of +0(%si).

1. ftrace

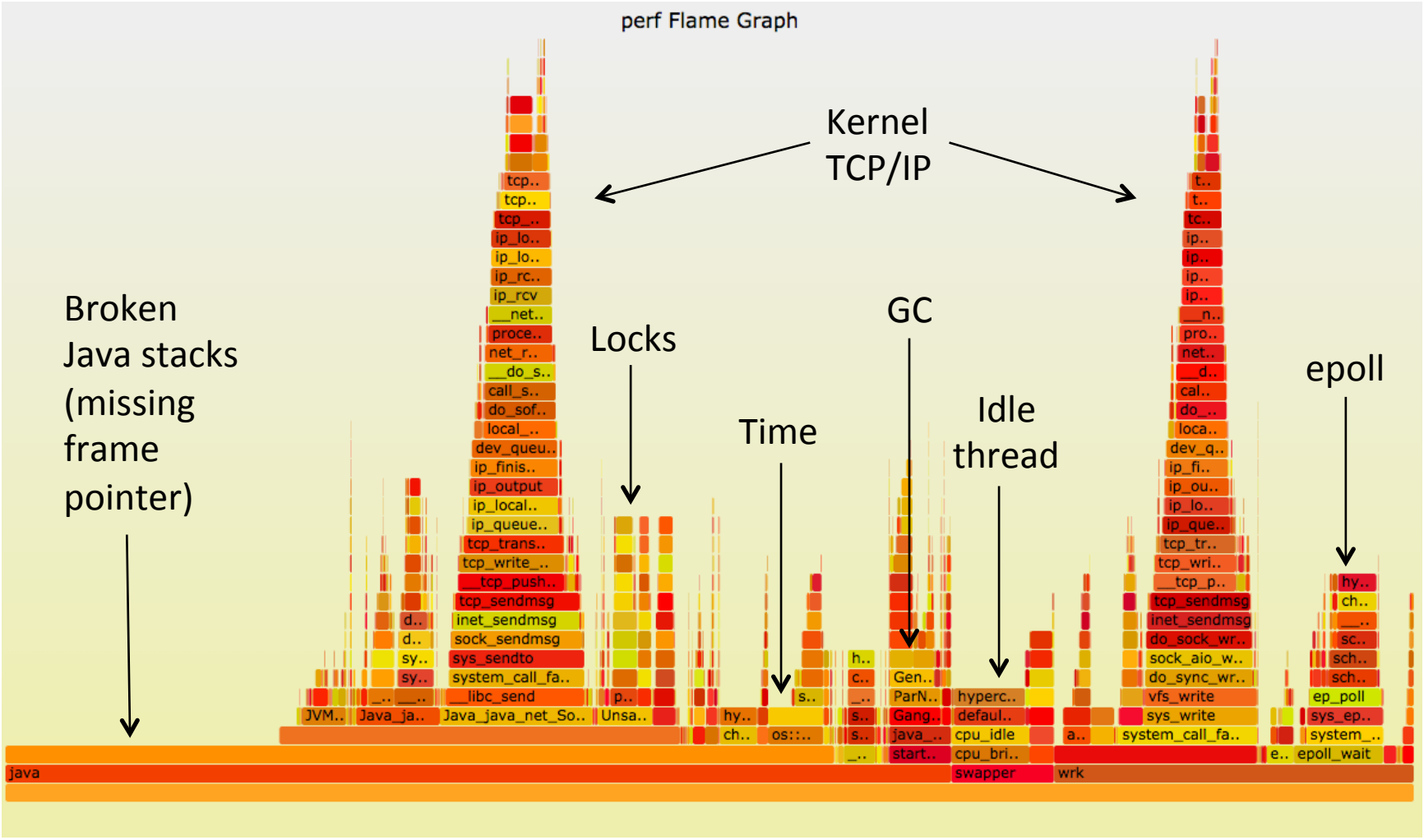
- Suggestions:
  - I'm blogging and so can you!
  - Function profiler:
    - Can these in-kernel counts be used for other vars? Eg, associative array or histogram of %dx
  - Function grapher:
    - Can the timing be exposed by some vars? Picture histogram of latency
  - Multi-user access possible?

2. perf_events

2. perf_events

- In-kernel, tools/perf, multi-tool, “perf” command
- Experiences:
  - Stable, powerful, reliable
  - The sub options can feel inconsistent (perf bench?)
  - Amazing with kernel debuginfo, when we have it
  - We use it for CPU stack profiles all the time
    - And turn them into flame graphs, which have solved numerous issues so far…

perf CPU Flame Graph



2. perf_events

- Suggestions:
  - Support for function argument symbols without a full debuginfo
  - Rework scripting framework (eg, try porting iosnoop)
    - eg, “perf record” may need a tunable timeout to trigger data writes, for efficient interactive scripts
  - Break up the multi-tool a bit (separate perf bench)
  - eBPF integration for custom aggregations?

3. SystemTap

3. SystemTap

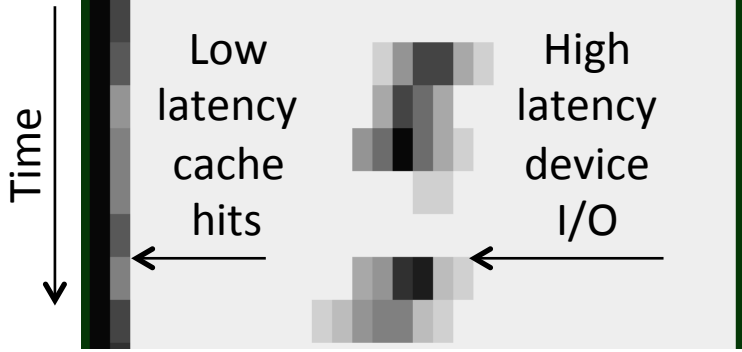
- The most powerful of the tracers
- Used for the deepest custom tracing
  - Especially Java hotspot probes
- Experiences:
  - Undergoing a reset. Switching to the latest SystemTap version, and a newer kernel. So far, so good.
  - Trying out nd_syscall for debuginfo-less tracing
- Suggestions:
  - More non-debuginfo tapset functionality

4. eBPF

4. eBPF

- Extended BPF: programs on tracepoints
  - High performance filtering: JIT
  - In-kernel summaries: maps
- eg, in-kernel latency heat map (showing bimodal):

```
root@bgregg-test-i-b7874e9d:/mnt/src/linux-3.16bpf2/samples/bpf# ./ex3
writing bpf-7 -> /sys/kernel/debug/tracing/events/block/block_rq_issue/filter
writing bpf-9 -> /sys/kernel/debug/tracing/events/block/block_rq_complete/filter
waiting for events to determine average latency...
  IO latency in usec
   - many events with this latency
   - few events
0 usec     ...          17634 usec
```




```
captured=270 missed=0 max_lat=0 usec
captured=3694 missed=0 max_lat=0 usec
captured=3485 missed=12 max_lat=18902 usec
captured=3541 missed=19 max_lat=82377 usec
captured=1945 missed=33 max_lat=24441 usec
captured=1636 missed=0 max_lat=0 usec
captured=3441 missed=18 max_lat=51263 usec
captured=2864 missed=71 max_lat=60497 usec
```

4. eBPF

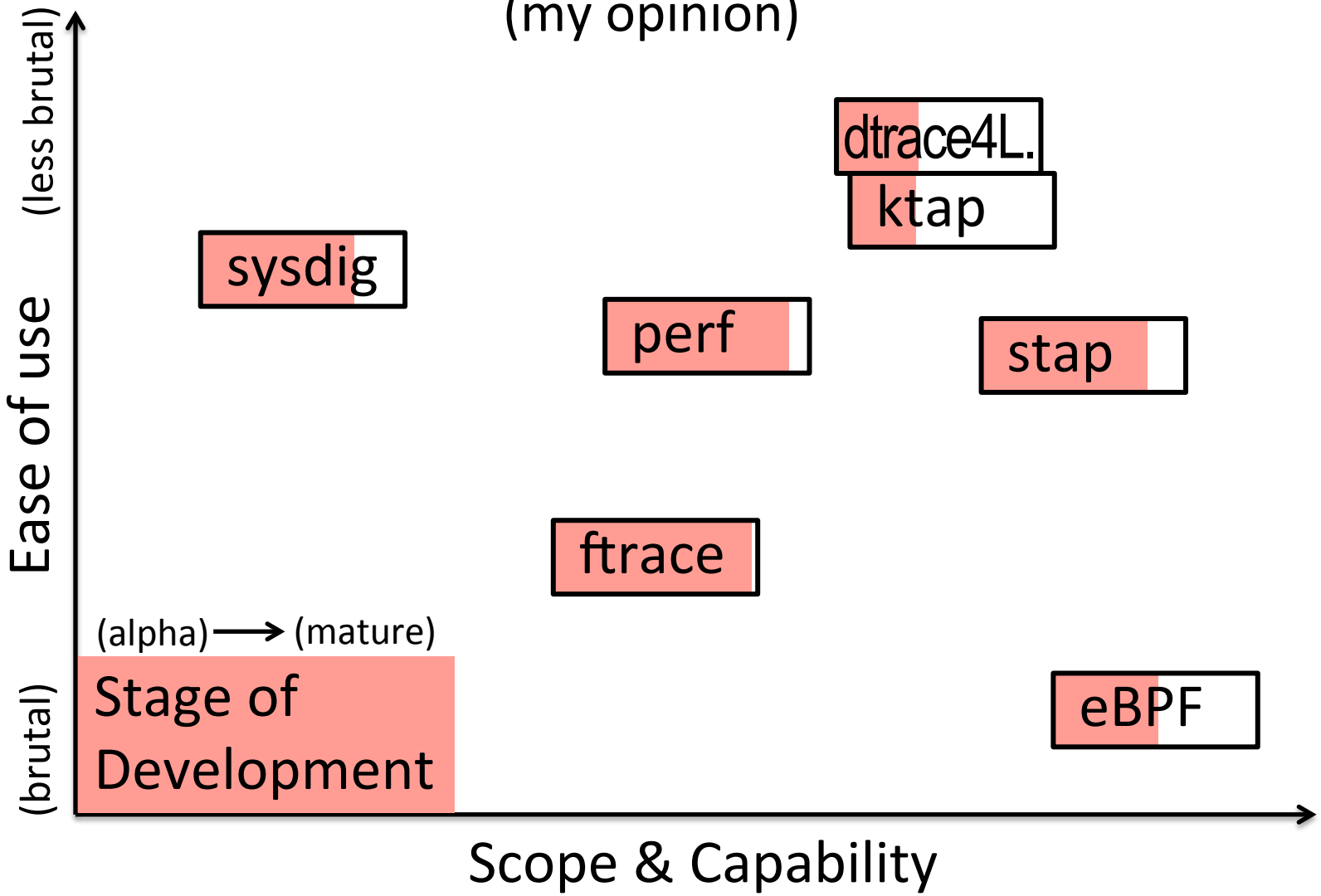
- Experiences:
  - Can have lower CPU overhead than DTrace
  - Very powerful: really custom maps
  - Assembly version very hard to use; C is better, but still not easy
- Suggestions:
  - Integrate: custom in-kernel aggregations is the missing piece

Other Tracers

- Experiences and suggestions:
  - ktap
  - LTTng
  - Oracle Linux DTrace
  - dtrace4linux
  - sysdig

The Tracing Landscape, Oct 2014

(my opinion)



Summary

- DTrace is an awesome technology
  - Which has also had awesome marketing
    - Traditional, social, sales, blogs, …
  - Most people won't use it directly, and that's ok
    - Drive usage via GUIs and scripts
- Linux Tracers are catching up, and may surpass
  - It's not 2005 anymore
    - Now we have ftrace, perf_events, kprobes, uprobes, …
  - Speed and aggregations matter
    - If DTrace is Kitty Hawk, eBPF is a jet engine

Acks

- dtrace.conf X-ray pony art by substack
- http://www.raspberrypi.org/ rasberry PI image
- http://en.wikipedia.org/wiki/Crash_test_dummy photo by Brady Holt
- https://findery.com/johnfox/notes/all-the-wood-behind-one-arrow
- http://en.wikipedia.org/wiki/Early_flying_machines hang glider image
- http://www.beginningwithi.com/2010/09/12/how-the-dtrace-book-got-done/
- http://www.cafepress.com/joyentsmartos.724465338
- http://generalzoi.deviantart.com/art/Pony-Creator-v3-397808116
- Tux by Larry Ewing; Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.
- Thanks Dominic Kay and Deirdré Straughan for feedback

Links

- https://www.usenix.org/legacy/event/usenix04/tech/general/full_papers/cantrill/cantrill.pdf
- ftrace & perf-tools
  - https://github.com/brendangregg/perf-tools
  - http://lwn.net/Articles/608497/
- eBPF: http://lwn.net/Articles/603983/
- ktap: http://www.ktap.org/
- SystemTap: https://sourceware.org/systemtap/
- sysdig: http://www.sysdig.org/
- http://lwn.net/Articles/114840/ CDDL
- http://dtrace.org/blogs/ahl/2011/10/05/dtrace-for-linux-2/
- ftp://ftp.cs.wisc.edu/paradyn/papers/Tamches99Using.pdf
- http://www.brendangregg.com/heatmaps.html
- http://lkml.iu.edu/hypermail/linux/kernel/0011.3/0183.html LTT + DProbes

Thanks

- Questions?
- http://slideshare.net/brendangregg
- http://www.brendangregg.com
- bgregg@netflix.com
- @brendangregg