Unified Tracing Platform

Bringing tracing together

Steven Rostedt
Open Source Engineer

srostedt@vmware.com / rostedt@goodmis.org

Github / twitter

The many faces of tracing

- Ftrace
- Perf
- LTTng
- Dtrace
- systemtap
- BPF tracing
- ktap
- strace
- gdb

A brief history (ftrace)

- logdev - 1998
  - One of the “parents” of ftrace
  - Created during my masters thesis (to see what was happening)
  - Mostly trace_printk() like functionality (with a flexible ring buffer)
  - Very portable (even ported it to Xen hypervisor!)
- latency tracer (preempt-realtime patch) - 2004
  - From Nadia Yvette Chambers and Ingo Molnar
  - Included tracing of functions via mcount
  - Function tracing had to be compiled in
    - either traced functions on boot, or did not
  - Included the preemption and interrupt disabled latency tracing
  - Very basic ring buffer (all events were the same size)
- Merged into ftrace in 2008

A brief history (perf)

- Perf - 2009
  - Started as just a profiler
  - Added to the Linux kernel git tree (under tools/perf)
  - Created its own ring buffer
    - ftrace ring buffer was not lockless yet
    - Needed large event sizes (greater than a page)
  - Hooked into the ftrace trace event infrastructure
    - Was able to duplicate several ftrace features (event tracing)
  - Optimized for profiling (but has gotten better)

A brief history (LTT - before ng)

- Linux Trace Toolkit (LTT) - 2000?
  - Written by Karim Yaghmour
  - First real attempt to do Linux tracing in mainline
  - Ingo Molnar and Linus Torvalds did not like how it was implemented

A brief history (LTT - before ng)

- Linux Trace Toolkit (LTT) - 2000?
  - Written by Karim Yaghmour
  - First real attempt to do Linux tracing in mainline
  - Ingo Molnar and Linus Torvalds did not like how it was implemented

https://lore.kernel.org/lkml/3D8E1AF8.91C1915D@opersys.com/T/#m2cea7cab3f7e7b6720a00645981582e000130e77

```
my problem with this stuff is conceptual: it introduces a constant drag on
the kernel sourcecode, while 99% of development will not want to trace,
ever. When i do need tracing occasionally, then i take those 30 minutes to
write up a tracer from pre-existing tracing patches, tailored to specific
problems. Eg. for the scheduler i wrote a simple tracer, but the rate of
trace points that started to make sense for me from a development and
debugging POV also made kernel/sched.c butt-ugly and unmaintainable, so i
always kept the tracer separate and did the hacking in the untained code.
```

- Ingo Molnar (9/22/2002)

A brief history (LTTng)

- Linux Trace Toolkit next generation (LTTng) - 2006
  - Complete rewrite and implementation by Mathieu Desnoyers
  - Implemented the original tracepoints
    - What ftrace trace events are built on top of
  - Mostly maintained as an out of tree kernel module (and user tools)
  - Hooks into the tracepoint infrastructure that is upstream

A brief history (Dtrace)

- Dtrace - Solaris 10 (Available 2003 / officially released 2005)
  - First to allow scripting in the kernel (custom traces)
  - Now owned by Oracle
  - Oracle ported Dtrace to Linux
    - Released under the GPL in 2017
    - Never made it into mainline
    - Only available with Oracle Linux (maybe some others too)

A brief history (Dtrace)

- Dtrace - Solaris 10 (Available 2003 / officially released 2005)
  - First to allow scripting in the kernel (custom traces)
  - Now owned by Oracle
  - Oracle ported Dtrace to Linux
    - Released under the GPL in 2017
    - Never made it into mainline
    - Only available with Oracle Linux (maybe some others too)
- Note - Some say that Dtrace came from IBM's dprobe work that was shipped by SLES (SUSE's Linux) But latter dropped.

A brief history (SystemTap)

- SystemTap (Released in 2009)
  - Red Hat's Linux's answer to Dtrace
  - Never made it to mainline

A brief history (SystemTap)

- SystemTap (Released in 2009)
  - Red Hat's Linux's answer to Dtrace
  - Never made it to mainline

2008 Kernel Summit

https://lwn.net/Articles/298685/

Linus came in to proclaim that he hates every tracing tool he has seen. SystemTap is far too complicated; these tools need to be simpler. [...]

We should be making better use of the simple tools which are currently in the kernel before trying to put more complicated stuff in.

A brief history (eBPF)

- BPF (Berkeley Packet Filter)
  - Just in Time (JIT) compiler added to x86 in 2011
  - Customize network packet filtering extremely fast
- eBPF (Extended BPF)
  - Introduced by Alexei Starovoitov in 2014
  - Extends the BPF JIT to other areas of the kernel besides just network filtering
  - Can be used for tracing with custom scripts (Dtrace / SystemTap like)
  - bpftrace (user tool incorporating eBPF)

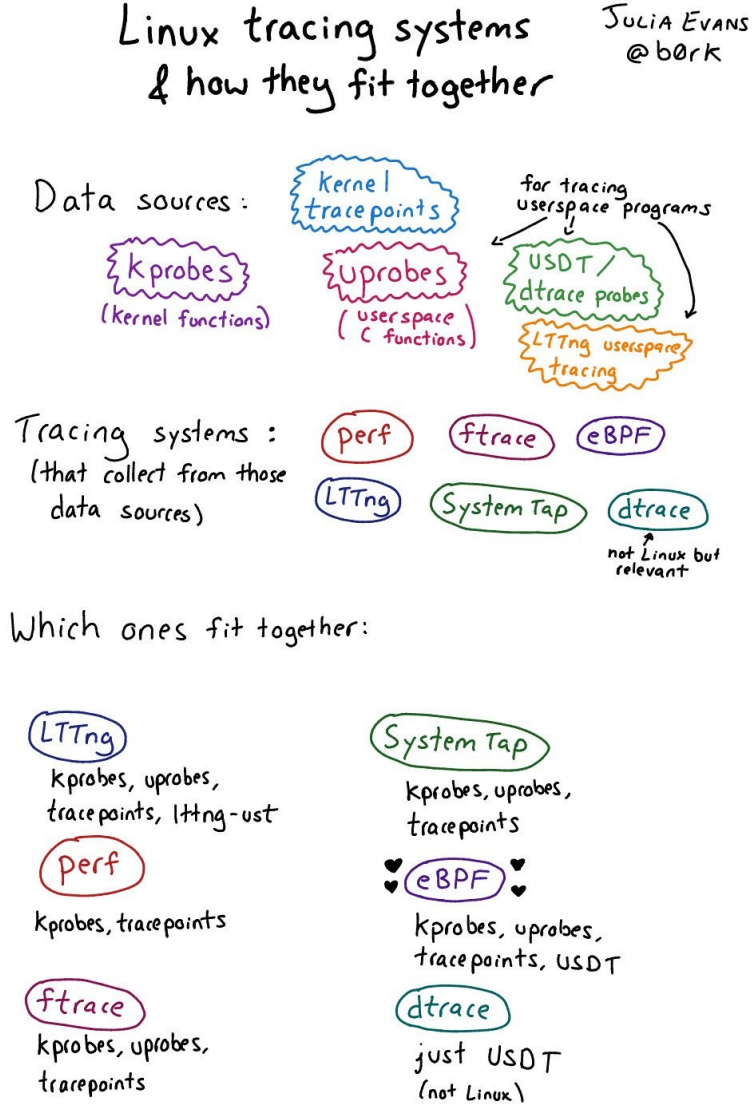
A brief history (eBPF)

- BPF (Berkeley Packet Filter)
  - Just in Time (JIT) compiler added to x86 in 2011
  - Customize network packet filtering extremely fast
- eBPF (Extended BPF)
  - Introduced by Alexei Starovoitov in 2014
  - Extends the BPF JIT to other areas of the kernel besides just network filtering
  - Can be used for tracing with custom scripts (Dtrace / SystemTap like)
  - bpftrace (user tool incorporating eBPF)
- Both SystemTap and Dtrace are working to build on top of eBPF

Where are we today?

Where are we today?

Julia Evans
@b0rk
30 Jun 2017



Can't we just have one tracer?

Can't we just have one tracer?

- Isn't too much choice a problem?

Can't we just have one tracer?

- Isn't too much choice a problem?

Can't we just have one tracer?

- Isn't too much choice a problem?
- Wasting effort on multiple approaches
  - Splitting the skill set?

Can't we just have one tracer?

- Isn't too much choice a problem?
- Wasting effort on multiple approaches
  - Splitting the skill set?
  - But diversity brings innovation!

Can't we just have one tracer?

Creative Commons 2006 - Holger.Ellgaard

Can't we just have one tracer?

- Nothing is one size fits all

Can't we just have one tracer?

- Nothing is one size fits all

Tabs vs Spaces

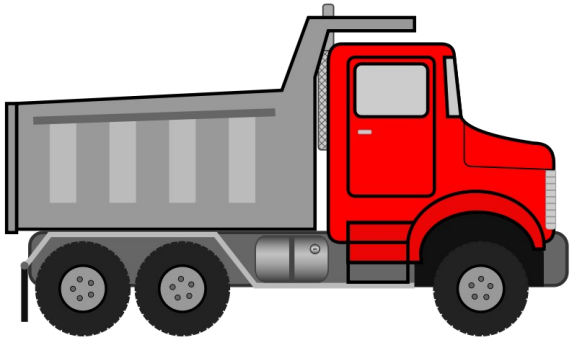
Can't we just have one tracer?

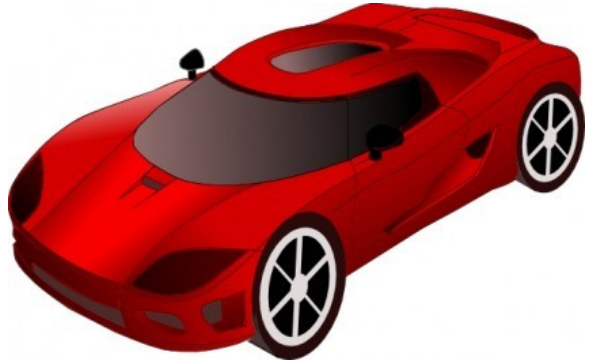
- Nothing is one size fits all

Tabs vs Spaces

Vim vs Emacs

Can't we just have one tracer?





http://clipart-library.com

Can't we just have one tracer?

- But diversity killed Unix!
  - Too many flavors

Can't we just have one tracer?

- But diversity killed Unix!
  - Too many flavors
- They were proprietary!
  - Could not share features

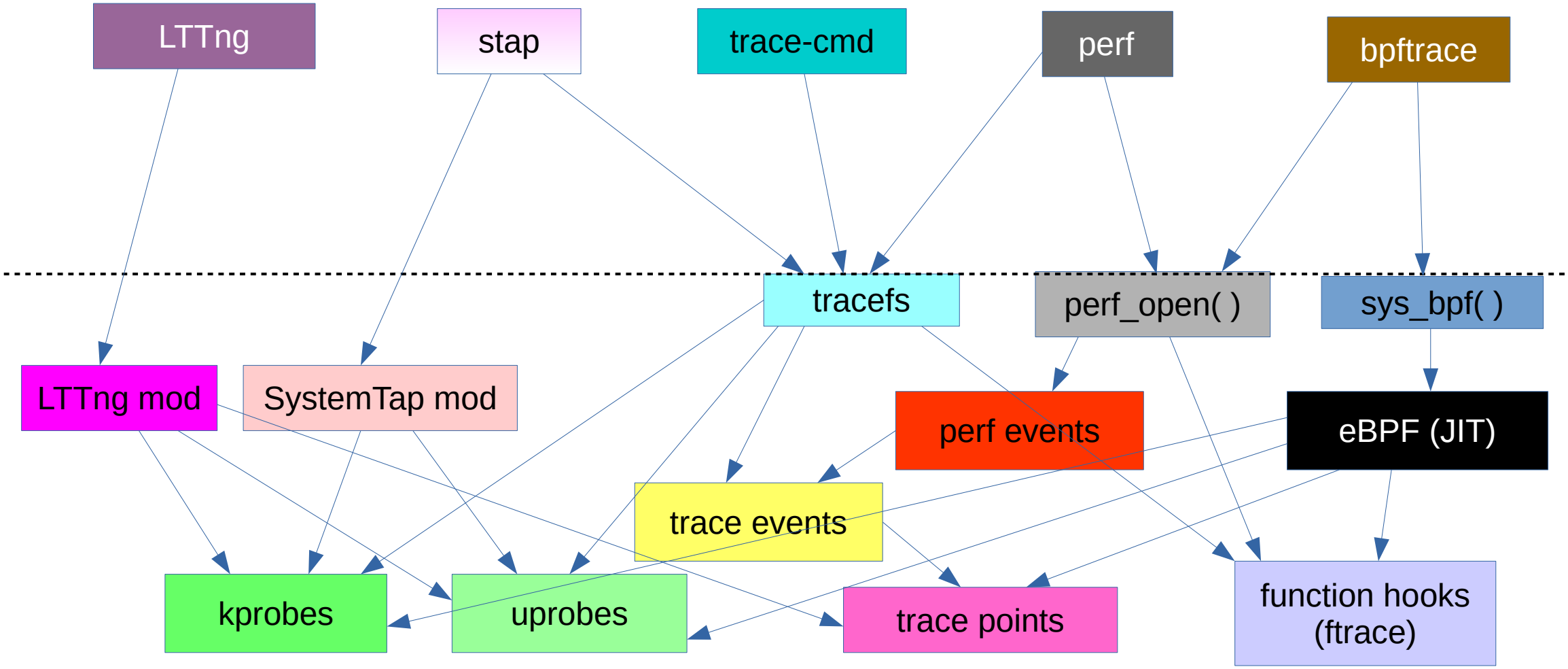
Can't we just have one tracer?

- But diversity killed Unix!
  - Too many flavors
- They were proprietary!
  - Could not share features
- Diversity is the strength of Open Source!
  - We can share!

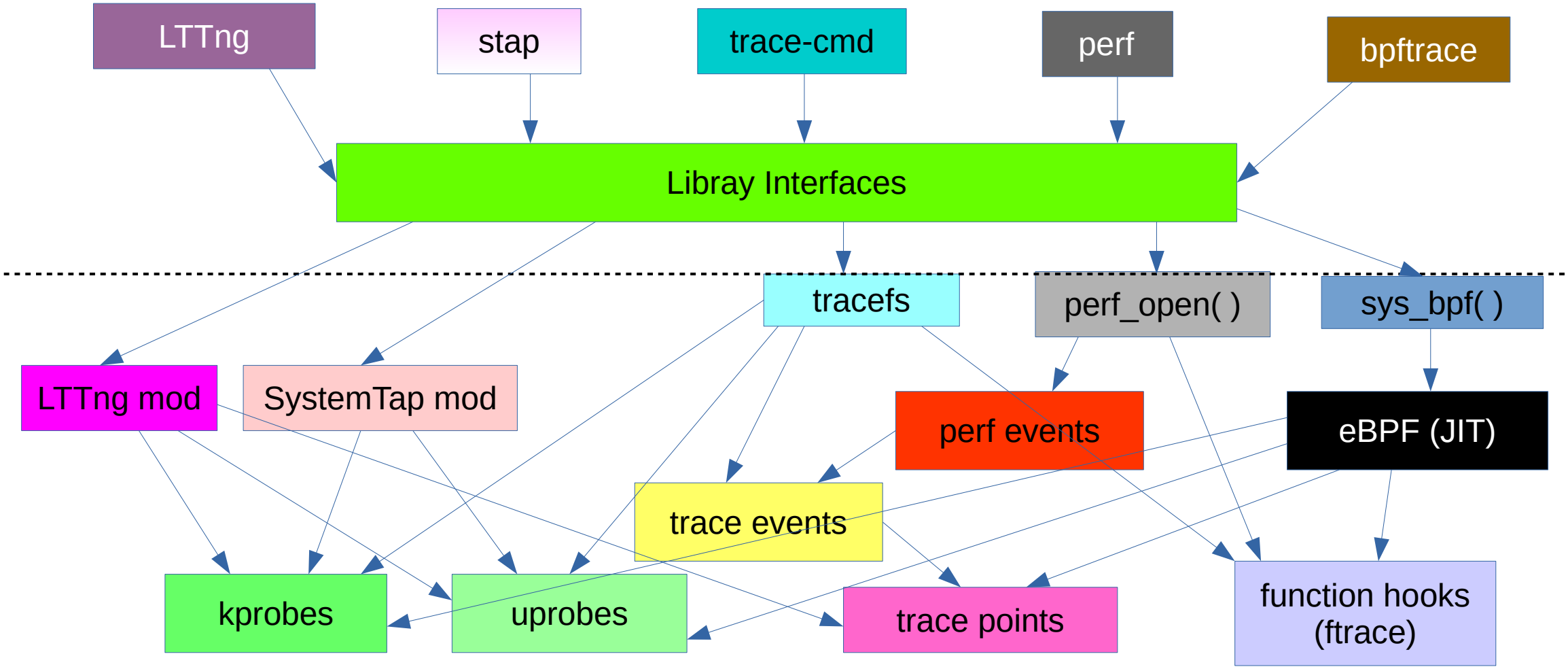
Commonality

- Trace point
  - static location in the kernel, that passes specific data via a “hook”
- Kprobes
  - Dynamic “hook” (like placing a static trace point almost anywhere)
- Uprobes
  - Dynamic “hook” into an application (takes a breakpoint TBD)
- ftrace - Function hooks
  - Allows callbacks from most kernel functions

Commonality



Commonality



Progress!

- Babel trace
- libtraceevent
- libperf

Babel Trace

- Developed from LTTng
- Parses the Common Trace Format (CFT)
- Goal is to allow any utility to read any tracing data format

libtraceevent

- Developed from ftrace utilities
- Parses the trace event format files
- Tells user applications how the trace events are written by the kernel in binary format
- Works with both perf and ftrace event data
- Works with kprobe and uprobe events

libperf

- Wraps the perf_event_open( ) system call
  - Which acts like a large ioctl to the perf interface

```
 1 #include <perf/cpumap.h>
 2
 3 int main(int argc, char **Argv)
 4 {
 5         struct perf_cpu_map *cpus;
 6         int cpu, tmp;
 7
 8         cpus = perf_cpu_map__new(NULL);
 9
10         perf_cpu_map__for_each_cpu(cpu, tmp, cpus)
11                 fprintf(stdout, "%d ", cpu);
12
13         fprintf(stdout, "\n");
14
15         perf_cpu_map__put(cpus);
16         return 0;
17 }
```

Coming Soon!

- libftrace
- libtracecmd
- libkshark

libftrace

- Interface to the tracefs directory
  - Start and stop tracers (function, latency, etc)
  - Enable or disable events
  - Create kprobes and uprobes
  - Read the tracing data
    - Only handles the raw data
    - Does not create the file

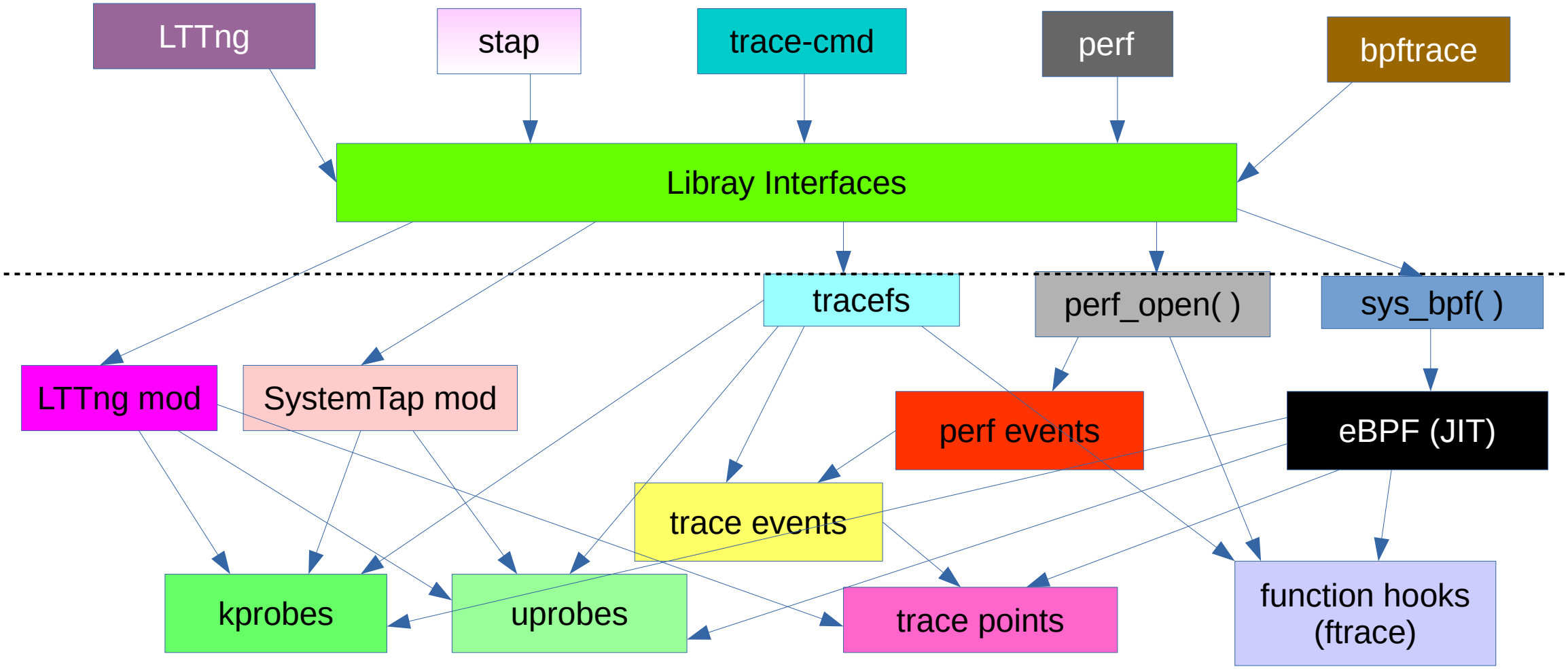
libtrace-cmd

- Allow you to do everything trace-cmd does!
  - Enable traces (will use libftrace)
  - Write to a file (saves a trace.dat)
  - Connect to guests (virt-server tracing)

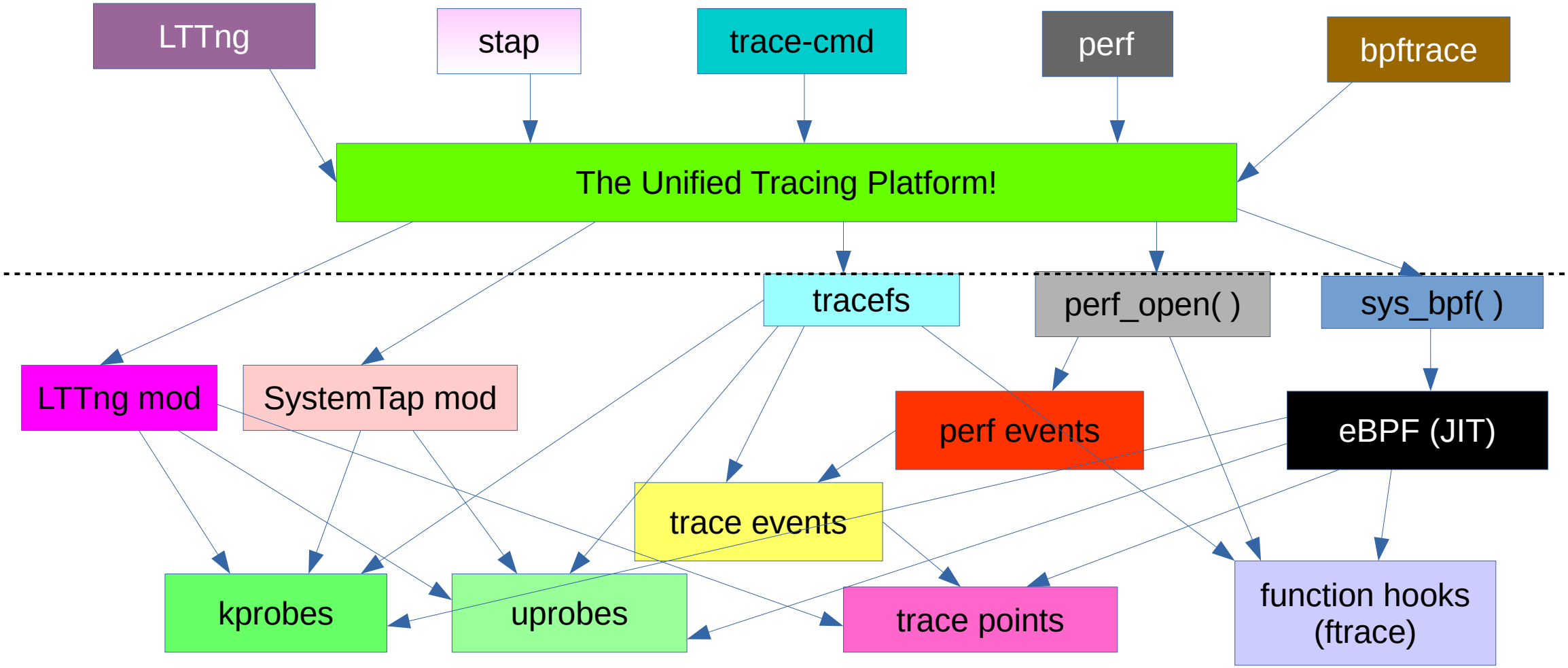
libkshark

- A GUI library
- Will allow you to do whatever KernelShark does
- Plot CPU data
- Plot Task data
- Make flame graphs
- Does not have to be trace-cmd format
  - Will work with CTF (in the near future)

Commonality



Commonality



The Unix way DOTADIW

Do One Thing And Do It Well

The Linux way DOLADIW

Do One Library And Do It Well

Conclusion

- The tools are not competing with each other
  - Does vim really compete with emacs?
- Each has its strengths and weaknesses
- All are open source
- All can utilize each others work
- This is what makes Linux the best OS in the world!

Thank You

vmware®