Linux BPF/bcc Road Ahead, March 2016

28 Mar 2016

This summarizes the current state of Linux BPF for system tracing, and the bcc front-end. BPF is a Linux mainline technology for event tracing and manipulation (formally eBPF for extended BPF), which has had many features added in the Linux 4.x series. bcc is an open source Python front-end.

Linux Now (4.6-rc1)

The latest Linux kernel (4.6-rc1) supports the following:

- dynamic tracing, kernel-level (BPF support for kprobes)
- dynamic tracing, user-level (BPF support for uprobes)
- filtering (via BPF programs)
- debug output (bpf_trace_printk())
- per-event output (bpf_perf_event_output())
- basic variables (global & per-thread variables, via BPF maps)
- associative arrays (via BPF maps)
- frequency counting (via BPF maps)
- histograms (power-of-2, linear, and custom, via BPF maps)
- timestamps and time deltas (bpf_ktime_get_(), and BPF programs)
- stack traces, kernel (BPF stackmap)*
- stack traces, user (BPF stackmap)*

And in user-level, the bcc package currently adds:

- debug output (Python with BPF.trace_pipe() and BPF.trace_fields())
- per-event output (BPF_PERF_OUTPUT macro and BPF.open_perf_buffer())
- interval output (BPF.get_table() and table.clear())
- histogram printing (table.print_log2_hist())
- C struct navigation, kernel-level (maps to bpf_probe_read())
- symbol resolution, kernel-level (ksym(), ksymaddr())
- symbol resolution, user-level (usymaddr())*
- BPF tracepoint support (via bcc Tracepoint.enable_tracepoint())*
- BPF stack trace support (incl. walk method for stack flames)*
- various other helper macros and functions
- many tools (under /tools)

* these were added in the last few weeks (thanks Alexei Starovoitov, Sasha Goldshtein, and Vicent Marti).

The Road Ahead

The following are the major items left to add to the Linux kernel, for BPF-based tracing:

- static tracing, kernel-level (BPF support for tracepoints #232; in progress by Alexei Starovoitov)
- timed sampling events (BPF support #330)
- overwrite ring buffers (in progress by Wang Nan)

And to add to bcc:

- static tracing, user-level (USDT probes via uprobes #327; in progress by Sasha Goldshtein)
- more helper routines for ease of use, but not too many (in progress by Brenden Blanco and others)
- more tools (in progress by me and others)
- more tests

Wow. I first drafted this post a few months ago, when it felt like we were at the half way mark. Many things have been completed since then, and I've updated this post. It's getting really exciting – the finish is in sight.

Other Misc Work

There's also some other related and optional work worth mentioning:

- llvm BPF debug enhancements (incl. better error messages)
- llvm 32-bit subregisters
- BPF accounting (/proc for what is running)
- GUI for bcc (heat maps, flame graphs, etc; Netflix Vector will be one)
- TUI for bcc (like atop)
- bcc built-in compiler instead of llvm/clang
- Higher-level language support (see #425)
- BPF/bcc optimizations, minor features, and bug fixes as needed (in progress by Alexei, Brenden, ...).

Plus new features we haven't thought of yet.

How You Can Help

1. Promotion

Most people have yet to hear of BPF or bcc. You can help by promoting them, especially to anyone doing performance work. Here are some links so far to share:

- https://github.com/iovisor/bcc#tools browse the "example" links
- http://www.brendangregg.com/blog/2016-03-05/linux-bpf-superpowers.html has BPF links
- https://www.kernel.org/doc/Documentation/networking/filter.txt

The 2nd URL has my recent BPF talk (Facebook, 30 mins):

2. Kernel BPF work

If you've done Linux kernel work before and can contribute in this area, keep an eye on lkml posts containing BPF, and become familiar with the samples under samples/bpf. You can run these standalone. To get started, I get llvm setup in /opt/local then do this with a pre-built kernel source:

```
cd linux-4....
vi samples/bpf/Makefile   # change LLC to LLC=/opt/local/llvm/bin/llc or whatever
make samples/bpf/
cd samples/bpf/
./tracex1
```

You can also contribute with code reviews and testing, which is often the bottleneck.

3. bcc Python/C/assembly work

Browse the bcc issues and help out where you can. If you are very good at Python, or C, or assembly, you might notice areas that can be improved just by browsing the code. Our focus so far has been getting BPF and

bcc capable on just x86_64, but there's going to be a lot of additional work to get it working on all processor types. Eg, aliases for registers (#225).

4. Distro Packaging

There need to be bcc packages for different Linux distros and repositories. bcc has a build-time dependency on clang/llvm, but when packaged right it shouldn't have any install-time dependencies – it should be an easy package add.

5. bcc Testing

Test the tools in bcc, and file detailed bugs as you find them.

6. bcc Documentation

There isn't much documentation yet on bcc/BPF, as it's an evolving interface. In the coming months the interface will be settling down, and we can write a reference guide, a tutorial, specific distro install instructions, etc. If you're totally new to bcc/BPF, you'll have a valuable perspective as you'll encounter every detail that must be learned in order to use it. Make notes, and please help with the docs.

In particular, the Ubuntu 16.04LTS release will be the first Ubuntu release with substantial BPF support, and we need docs to show how people can get started. Not everything is available in its 4.4 kernel, but there's enough for a significant preview of BPF capabilities. A couple of dozen tools, at least, should work.

7. Use cases

I wouldn't encourage everyone to start coding their own bcc tools just yet, as the interface still has rough edges. But if you have some tolerance for really gritty engineering work, then publishing early use cases will help the project in many ways.

If you do try bcc and find it too frustrating or difficult, then check again in a few months. It keeps getting better.

Thanks to everyone who has contributed so far!

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).