Contribution ID: **27** Type: **not specified**

Tracing / Perf events MC

The Linux kernel has grown in complexity over the years. Complete understanding of how it works via code inspection has become virtually impossible. Today, tracing is used to follow the kernel as it performs its complex tasks. Tracing is used today for much more than simply debugging. Its framework has become the way for other parts of the Linux kernel to enhance and even make possible new features. Live kernel patching is based on the infrastructure of function tracing, as well as BPF. It is now even possible to model the behavior and correctness of the system via runtime verification which attaches to trace points. There is still much more that is happening in this space, and this microconference will be the forum to explore current and new ideas.

This year, focus will also be on perf events:

Perf events are a mechanism for presenting performance counters and software events that occur running Linux to users. There are kernel and userland components to perf events, with the kernel presenting or extending APIs and the perf tool presenting this to users

Results and accomplishments from the last time (2023):

- Masami's work on accessing function entry data from function *return* probes (kprobe and fprobe) was merged for v6.9.
- eventfs is now dynamically created and fully working following *robust* discussions with Linus.
- Work on sframes was paused due to other priorities but is still a topic of interest.
- Discussions on integrating User events with libside are ongoing.
- User events added multi-format events.

Topics for this year:

- Feedback about the tracing/perf subsystems overall (e.g. how can people help the maintainers).
- Reboot persistent in-memory tracing buffers, this would make ftrace a very powerful debugging and performance analysis tool for kexec and could also be used for post crash debugging.
- Dynamic change of ftrace events to improve symbolic printing.
- Userspace instrumentation (libside), including discussion of its impacts on the User events ABI.
- Collect state dump events from kernel drivers (e.g. dump wifi interfaces configuration at a given point in time through trace buffers).
- Current work implementing performance monitoring in the kernel,
- User land profiling and analysis tools using the perf event API,
- Improving the kernel perf event and PMU APIs,
- Interaction between perf events and subsystems like cgroups, kvm, drm, bpf, etc.,
- Improving the perf tool and its interfaces in particular w.r.t. to scalability of the tool,
- Implementation of new perf features and tools using eBPF, like the ones in tools/perf/util/bpf_skel/.
- Further use of type information to augment the perf tools,
- Novel uses of perf events for debugging and correctness,
- New challenges in performance monitoring for the Linux kernel,
- Regression testing/CI integration for the perf kernel infrastructure and tools,
- Improving documentation,
- Security aspects of using tracing/perf tools,

Key attendees:

- Steven Rostedt
- Masami Hiramatsu
- Mathieu Desnoyers
- Alexei Starovoitov
- Peter Zijlstra
- Mark Rutland
- Beau Belgrave
- Daniel Bristot de Oliveira
- Florent Revest
- Jiri Olsa
- Tom Zanussi
- Alexander Graf
- Johannes Berg
- Arnaldo Carvalho de Melo
- Ian Rogers
- Namhyung Kim
- Stephane Eranian

Primary authors: CARVALHO DE MELO, Arnaldo (Red Hat Inc.); ROGERS, Ian (Google); DESNOYERS, Mathieu (EfficiOS Inc.); JEANSON, Michael (EfficiOS); KIM, Namhyung (Google); ROSTEDT, Steven

Session Classification: Tracing / Perf events MC

Track Classification: Tracing / Perf events MC