LTTng as a fast system call tracer

Mathieu Desnoyers
EfficiOS Inc.

Problem and Goals

- Fast tracing of system calls with their input/output argument payload,
- Current upstream ring buffers cannot handle page faults while copying user-space data into the buffers,
  - Prevents zero-copy of user-space data into buffers.

Current Status

- Upstream tracer ring buffers are tuned for their specific use-cases:
  - Perf is specialized for sampling,
  - Ftrace is specialized for tracing at high speed with preemption disabled.
- None of the upstream tracers allow reading user-space data reliably:
  - Perf and Ftrace only trace the register contents on system call entry/exit,
  - eBPF uses a zero-padding fallback when a fault would occur.
- System call tracers based on ptrace such as *strace* are slow due to scheduling and ptrace peek overhead.

Proposed Solution

- Upstream a new tracer based on components of the LTTng kernel tracer,
- Relevant components:
  - **LTTng's ring buffer** is designed to be used both in the kernel and from user-space, thus allowing preemption and page faults,
  - An **ABI derived from the LTTng kernel tracer**, allowing interaction with an existing ecosystem of user-space trace tooling:
    - Expose concepts compatible with the LTTng tracer user-space tooling,
    - Common Trace Format consumed by trace viewers.

Open Questions

- How do we implement the code which copies system call arguments into buffers ?
  - Macros,
  - Open coded with static inline/macro helpers,
  - … ?
- Is the scope of this project reasonable:
  - Brings enough value to be upstreamed ?
  - Reviewable within a reasonable effort ?