Performance profiling with eBPF

Alessandro Decina

https://github.com/aya-rs/aya

https://discord.gg/xHW2cb2N6G

Alessandro Decina

eBPF? 🤔

- Bytecode format understood by the kernel
- Set of APIs to integrate with the kernel
- The way to hook into the kernel and extend it with new functionality

profiler? 

- Tool to observe program execution
- Instrumenting profiler: program is instrumented with special code to trace and report metrics
- Sampling profiler: snapshots of the program are taken at some frequency

how it started 🐢

$ time tar -C /home/a/tmp -xf snapshot.tar.zst

real 4m15.094s
user 1m57.897s
sys 3m33.882s

```
Total DISK READ:        125.98 M/s | Total DISK WRITE:        407.74 M/s
Current DISK READ:      125.31 M/s | Current DISK WRITE:      567.71 M/s
    TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN     IO>    COMMAND
2590816 be/4 root        7.12 K/s    0.00 B/s  ?unavailable?  [kworker/u9
```

how it’s going 🏃

$ time target/release/tartar snapshot.tar.zst

real 1m37.991s
user 1m26.679s
sys 4m24.559s

```
Total DISK READ:         515.51 M/s | Total DISK WRITE:      1933.86 M/s
Current DISK READ:       514.29 M/s | Current DISK WRITE:       26.46 M/s
    TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN     IO>    COMMAND
```

perf: the OG linux profiler

```
Samples: 95K of event 'cycles', Event count (approx.): 73675690566
  Children      Self  Command          Shared Object      Symbol
+   42.96%     0.00%  iou-wrk-2585714  [unknown]          [.] 0000000000000000
+   42.96%     0.00%  iou-wrk-2585714  [kernel.kallsyms]  [k] ret_from_fork
+   42.96%     0.00%  iou-wrk-2585714  [kernel.kallsyms]  [k] io_wqe_worker
+   42.94%     0.02%  iou-wrk-2585714  [kernel.kallsyms]  [k] io_worker_handle_work
+   42.88%     0.00%  iou-wrk-2585714  [kernel.kallsyms]  [k] io_wq_submit_work
+   42.87%     0.00%  iou-wrk-2585714  [kernel.kallsyms]  [k] io_issue_sqe
+   42.87%     0.01%  iou-wrk-2585714  [kernel.kallsyms]  [k] io_write
+   42.76%     0.00%  iou-wrk-2585714  [kernel.kallsyms]  [k] ext4_file_write_iter
+   42.76%     0.01%  iou-wrk-2585714  [kernel.kallsyms]  [k] ext4_buffered_write_iter
+   42.66%     0.00%  iou-wrk-2585714  libc.so.6          [.] start_thread
+   42.66%     0.00%  iou-wrk-2585714  tartar             [.] std::sys::pal::unix::threa
+   42.66%     0.00%  iou-wrk-2585714  tartar             [.] core::ops::function::FnOnc
+   42.66%     0.00%  iou-wrk-2585714  tartar             [.] std::sys_common::backtrace
+   42.19%     0.17%  iou-wrk-2585714  [kernel.kallsyms]  [k] generic_perform_write
+   28.22%     0.00%  iou-sqp-2585712  [unknown]          [k] 0000000000000000
+   28.22%     0.00%  iou-sqp-2585712  [kernel.kallsyms]  [k] ret_from_fork
+   27.90%    11.99%  iou-sqp-2585712  [kernel.kallsyms]  [k] io_sq_thread
+   25.68%     0.00%  tartar           libc.so.6          [.] __libc_start_call_main
+   25.68%     0.00%  tartar           tartar             [.] main
+   25.68%     0.00%  tartar           tartar             [.] std::rt::lang_start_intern
+   25.68%     0.00%  tartar           tartar             [.] std::rt::lang_start::_$u7b
+   25.68%     0.00%  tartar           tartar             [.] std::sys_common::backtrace
-   25.68%     0.00%  tartar           tartar             [.] tartar::main
   - 25.67% tartar::main
      - 25.48% tar::entry::Entry<R>::unpack_in_with_io
         - 24.83% core::ops::function::impls::<impl core::ops::function::FnMut<A> for &m
            + 24.70% std::io::Read::read_exact
```

eBPF profiler core logic

- Samples the cpu-clock counter at some frequency
- Executes our eBPF program to take a snapshot of the profiled program

eBPF sampling program

```rust
// Define the `perf_samples` eBPF program. This is going to be loaded
// and executed by the kernel.
#[perf_event]
pub fn perf_samples(ctx: PerfEventContext) -> u32 {
    if let Err(e) = do_sample(&ctx) {
        error!(&ctx, "perf_samples hook failed: {}", e);
    }

    0
}
```

user space loader

```rust
// Load the file that contains all our eBPF programs
let mut bpf = aya::BpfLoader::new()
    .set_global("PID", &pid, true)
    .load(include_bytes_aligned!(
        "../../target/bpfel-unknown-none/release/ngmi"
    ))?;

// Attach the `perf_samples` program to the cpu clock cycles counter, with a frequency of 500
// samples per second
let program: &mut PerfEvent = bpf.program_mut("perf_samples").unwrap().try_into()?;
program.load()?;
program.attach(
    PerfTypeId::Software,
    // perf_hw_id::PERF_COUNT_HW_CPU_CYCLES as u64,
    perf_sw_ids::PERF_COUNT_SW_CPU_CLOCK as u64,
    PerfEventScope::OneProcessAnyCpu { pid },
    SamplePolicy::Frequency(500),
    true,
)?;
```

user space loader

```rust
// Load the file that contains all our eBPF programs
let mut bpf = aya::BpfLoader::new()
    .set_global("PID", &pid, true)
    .load(include_bytes_aligned!(
        "../../target/bpfel-unknown-none/release/ngmi"
    ))?;

// Attach the `perf_samples` program to the cpu clock cycles counter, with a frequency of 500
// samples per second
let program: &mut PerfEvent = bpf.program_mut("perf_samples").unwrap().try_into()?;
program.load()?;
program.attach(
    PerfTypeId::Software,
    // perf_hw_id::PERF_COUNT_HW_CPU_CYCLES as u64,
    perf_sw_ids::PERF_COUNT_SW_CPU_CLOCK as u64,
    PerfEventScope::OneProcessAnyCpu { pid },
    SamplePolicy::Frequency(500),
    true,
)?;
```

eBPF sampling logic

```rust
#[map]
static EVENTS: RingBuf = RingBuf::with_byte_size(4096 * 10000, 0);

fn do_sample<C: BpfContext>(ctx: &C) -> Result<(), Error> {
    let mut event = EVENTS
        .reserve::<Event<SamplesEvent>>(0)
        .ok_or(error::RINGBUF)?;

    unsafe {
        let user_stack_len = bpf_get_stack(
            ctx.as_ptr(),
            event.as_mut_ptr().byte_add(SamplesEvent::USER_STACK_OFFSET) as *mut _,
            SamplesEvent::STACK_SIZE,
            BPF_F_USER_STACK as u64,
        );

        ptr::write_unaligned(
            event.as_mut_ptr().byte_add(EVENT_DATA_OFFSET) as *mut _,
            SamplesEventHeader {
                pid: ctx.tgid(),
                tid: ctx.pid(),
                ts: unsafe { bpf_ktime_get_ns() },
                user_stack_len: user_stack_len as u64,
            },
        );
        ptr::write_unaligned(event.as_mut_ptr() as *mut EventType, EventType::Samples);
    }

    event.submit(0);

    Ok(())
}
```

user space sample processing

```
// Get a handle to the `EVENTS` map and process events as they come
let events: RingBuf<_> = bpf.take_map("EVENTS").unwrap().try_into()?;
 ...
while let Some(ring_event) = events.next() {
    let (event_type, data) = unsafe { ring_buf_read(&*ring_event) };
    match event_type {
        EventType::Samples => {
            let (event, _) = unsafe { ring_buf_read::<SamplesEvent>(data) };
            event_processor.send(Event::Samples(event))
        }
         ...
    }
}
```

call stack

0xffff95d454dc
0xffff95bd3cac
0xaaaaad662698
0xffff95b47858
0xaaaaad646c30
...

symbolicated call stack

```
...
tartar::untar_zstd_file::{{closure}} [/home/a/src/tartar/src/main.rs]
core::ops::function::impls::<impl core::ops::function::FnMut<A> for &mut F>::call_mut [/home/a/.r
<core::result::Result<T,E> as core::ops::try_trait::Try>::branch [/home/a/.rustup/toolchains/nigh
tar::entry::EntryFields::unpack_with_io::{{closure}} [/home/a/src/tar-rs/src/entry.rs]
tar::entry::EntryFields::unpack_with_io [/home/a/src/tar-rs/src/entry.rs]
tar::entry::EntryFields::unpack_in_with_io [/home/a/src/tar-rs/src/entry.rs]
tar::entry::Entry<R>::unpack_in_with_io [/home/a/src/tar-rs/src/entry.rs]
<core::result::Result<T,E> as core::ops::try_trait::Try>::branch [/home/a/.rustup/toolchains/nigh
tartar::untar_zstd_file [/home/a/src/tartar/src/main.rs]
tartar::main [/home/a/src/tartar/src/main.rs]
std::sys_common::backtrace::__rust_begin_short_backtrace [/home/a/.rustup/toolchains/nightly-x86_
<core::result::Result<T,E> as std::process::Termination>::report [/home/a/.rustup/toolchains/nigh
std::rt::lang_start::{{closure}} [/home/a/.rustup/toolchains/nightly-x86_64-unknown-linux-gnu/lib
core::result::Result<T,E>::unwrap_or [/home/a/.rustup/toolchains/nightly-x86_64-unknown-linux-gnu
std::rt::lang_start_internal::{{closure}} [/home/a/.rustup/toolchains/nightly-x86_64-unknown-linu
std::panicking::try::do_call [/home/a/.rustup/toolchains/nightly-x86_64-unknown-linux-gnu/lib/rus
std::panicking::try [/home/a/.rustup/toolchains/nightly-x86_64-unknown-linux-gnu/lib/rustlib/src/
std::panic::catch_unwind [/home/a/.rustup/toolchains/nightly-x86_64-unknown-linux-gnu/lib/rustlib
std::rt::lang_start_internal [/home/a/.rustup/toolchains/nightly-x86_64-unknown-linux-gnu/lib/rus
main [tartar]
_libc_start_call_main [./csu/../sysdeps/nptl/libc_start_call_main.h]
```

off-cpu samples

```
let program: &mut TracePoint = bpf.program_mut("sched_switch")
    .unwrap()
    .try_into()?;
program.load()?;
program.attach("sched", "sched_switch")?;
```

disk IO

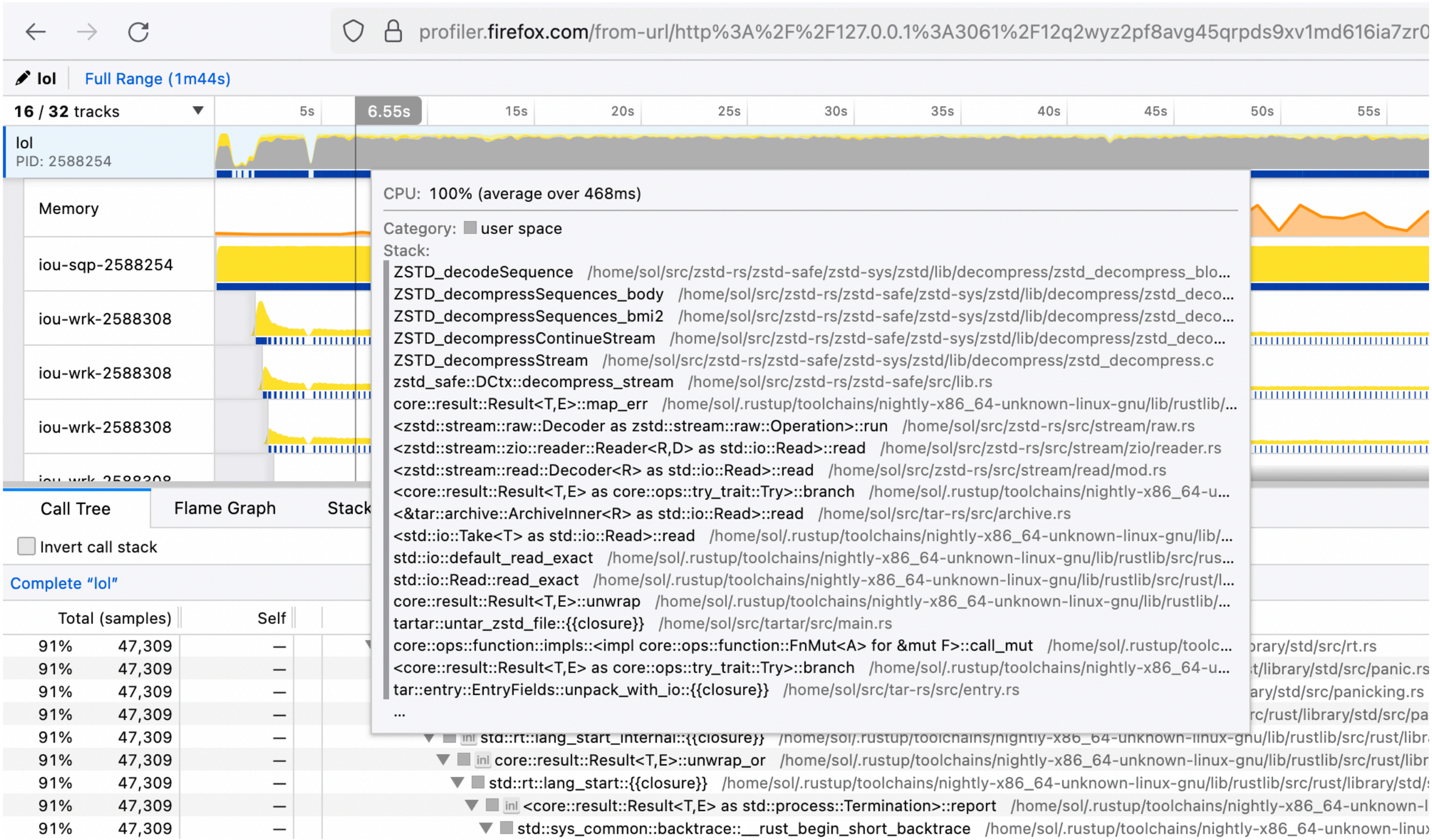
```rust
let program: &mut TracePoint = bpf.program_mut("block_rq_insert")
    .unwrap()
    .try_into()?;
program.load()?;
program.attach("block", "block_rq_insert")?;
```

```rust
let program: &mut TracePoint = bpf.program_mut("block_rq_complete")
    .unwrap()
    .try_into()?;
program.load()?;
program.attach("block", "block_rq_complete")?;
```

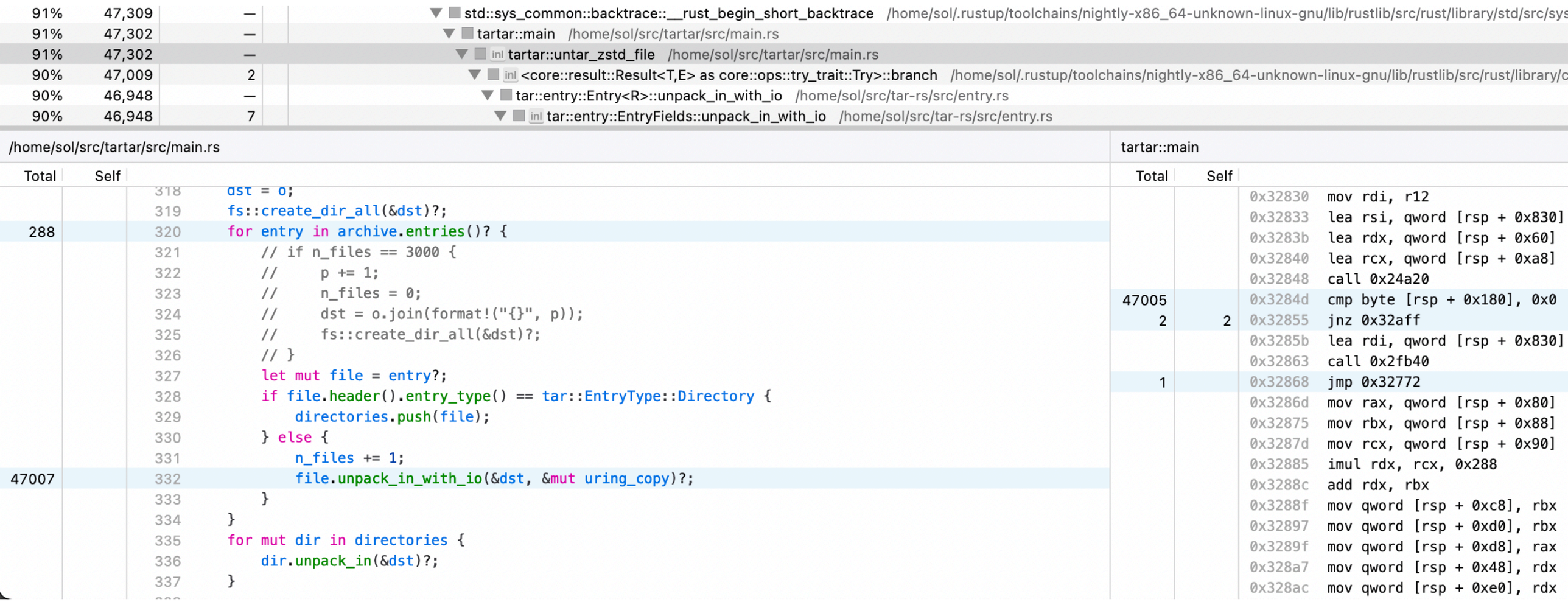
page cache faults

```
let program: &mut FEntry = bpf.program_mut("fe_handle_mm_fault")
    .unwrap()
    .try_into()?;
program.load("handle_mm_fault", &btf)?;
program.attach()?;

let program: &mut FExit = bpf.program_mut("fex_handle_mm_fault")
    .unwrap()
    .try_into()?;
program.load("handle_mm_fault", &btf)?;
program.attach()?;

let program: &mut FExit = bpf
    .program_mut("fex_filemap_fault_exit")
    .unwrap()
    .try_into()?;
program.load("filemap_fault", &btf)?;
program.attach()?;
```

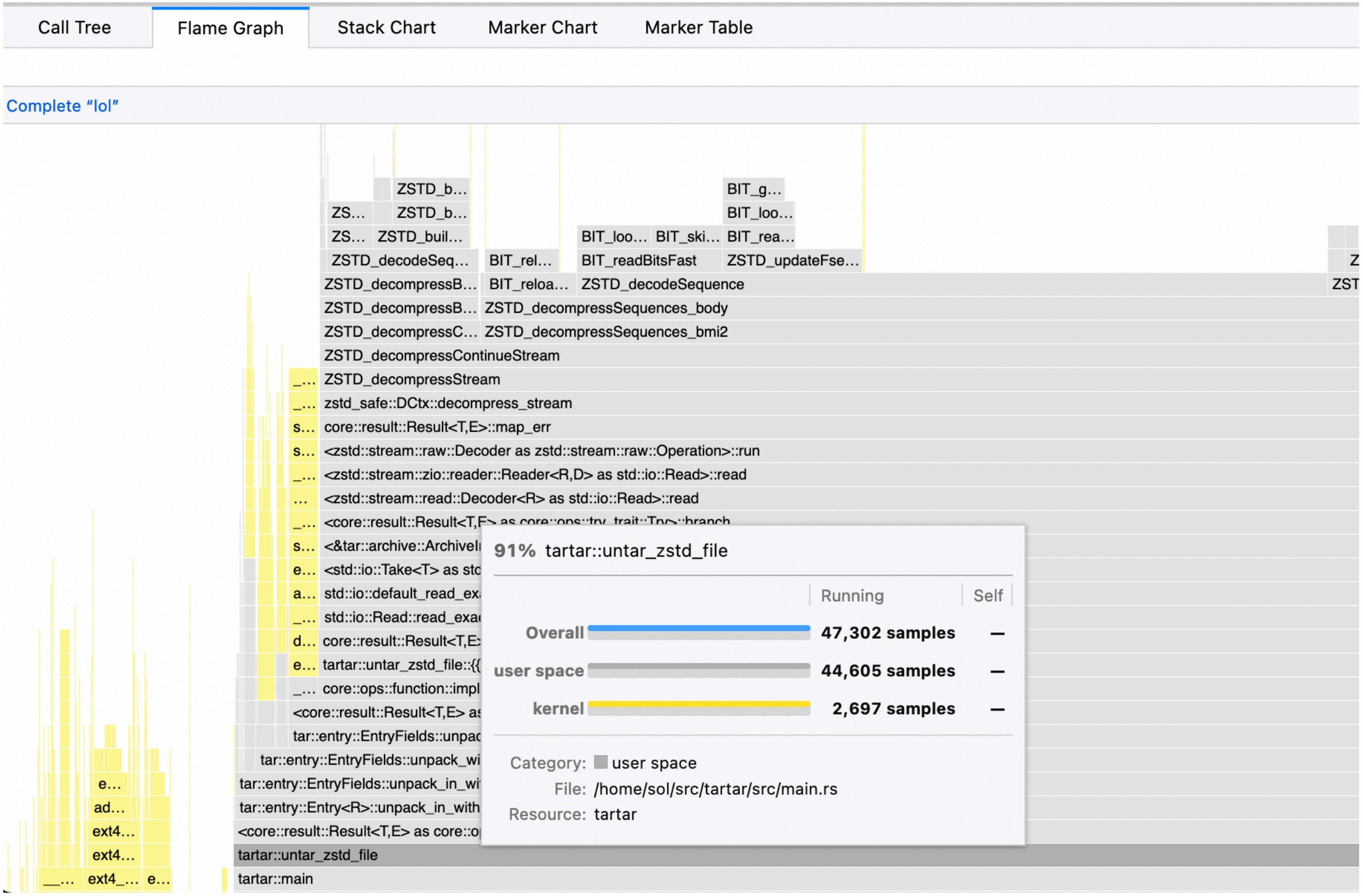
https://profiler.firefox.com



source code, assembly and much more



can’t believe it’s free



https://github.com/aya-rs/aya

https://discord.gg/xHW2cb2N6G