Batteries-included symbolization with

blazesym

Daniel Müller

May 2023

Agenda

- Motivation
- Library Overview
- Current Status
- Look ahead
- Discussion

Motivation

- Address symbolization is intricate:
  - Need to understand individual formats (ELF, DWARF, Gsym)
  - System-specific details and corner cases
  - Symbolization trade-offs (fidelity, performance, memory usage)
- Frequently tools use fairly simple means of symbolization
- Idea: Have a default go-to library
  - Take burden off of tooling developer’s shoulders

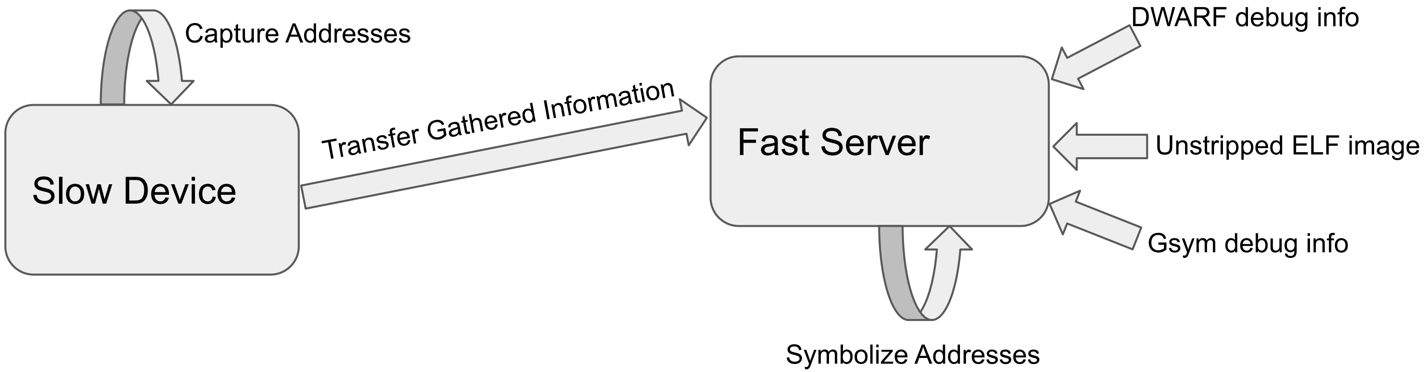
Library Overview

- Identified two main functionalities:
    - Symbolization (address -> name)
    - Reverse symbolization (name -> address + info)

- Ability to handle user space and kernel addresses
    - Process addresses: based on PID (using /proc/<pid>/maps)
    - Kernel addresses: /proc/kallsyms
    - All: based on ELF file, DWARF debug symbols, or Gsym

Library Overview (2) – Remote Symbolization

- Capture addresses on “local” system (potentially resource constrained)
  - Addresses are “normalized” as a result
  - Data about binary an address belongs to is captured (e.g., path, build ID, …)
- Symbolize on different system (e.g., beefier)
  - Find symbolization source based on captured information (e.g., build ID)



Current Status

- Basic symbolization logic support
    - Can use ELF, DWARF (*), GSYM as source
- Can lookup symbols in ELF & DWARF
- Converging on API surface

Current Status (2)

- We provide C bindings
  - Header is auto generated
  - Just link archive/shared object
- Started integration with Meta-internal Profiler
  - Will provide validation at large scale
- Used for VR headset tracing/analysis
- Shout out to Andrii Nakryiko & Kui-Feng Lee

- Repository: https://github.com/libbpf/blazesym
- Rendered Documentation: https://docs.rs/blazesym

Looking ahead…

- Support more DWARF versions
- Support split DWARF
- Symbolization of addresses in APKs
- On-demand symbol demangling
- Advanced use cases such as debuginfod

Discussion

- What may be missing? What are your use cases?
- Integration opportunities?
- Anybody working on something similar?
- Usage of Rust a problem in your context?

- Repository: https://github.com/libbpf/blazesym
- Documentation: https://docs.rs/blazesym