Linux Plumbers Conference | Richmond, VA | Nov. 13-15, 2023

Developing a Continuous eBPF Profiler

Looking Beneath the Kernel to Beyond the Clouds

by Sumera | @sylfrena | @SumoOfShinovar

Polar Signals

Agenda

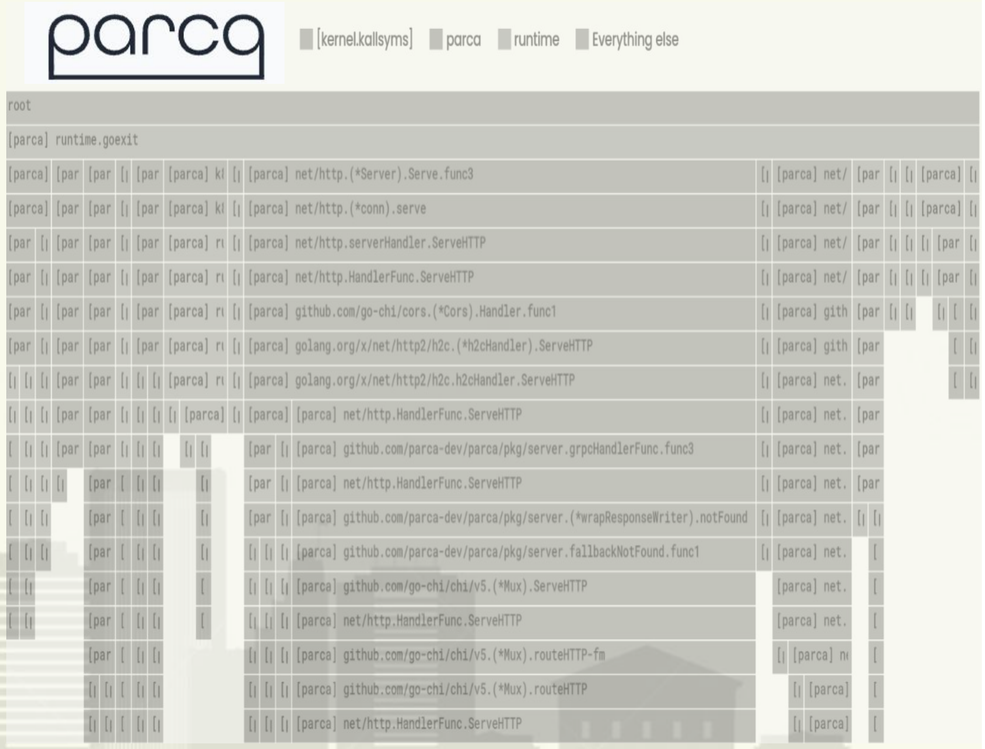
- High Resolution Continuous Profiling
- Userspace and Kernelspace
- Compilers and Runtimes
- Clouds and Kernels
- Low effort Debugging
- Future work

STORY TIME

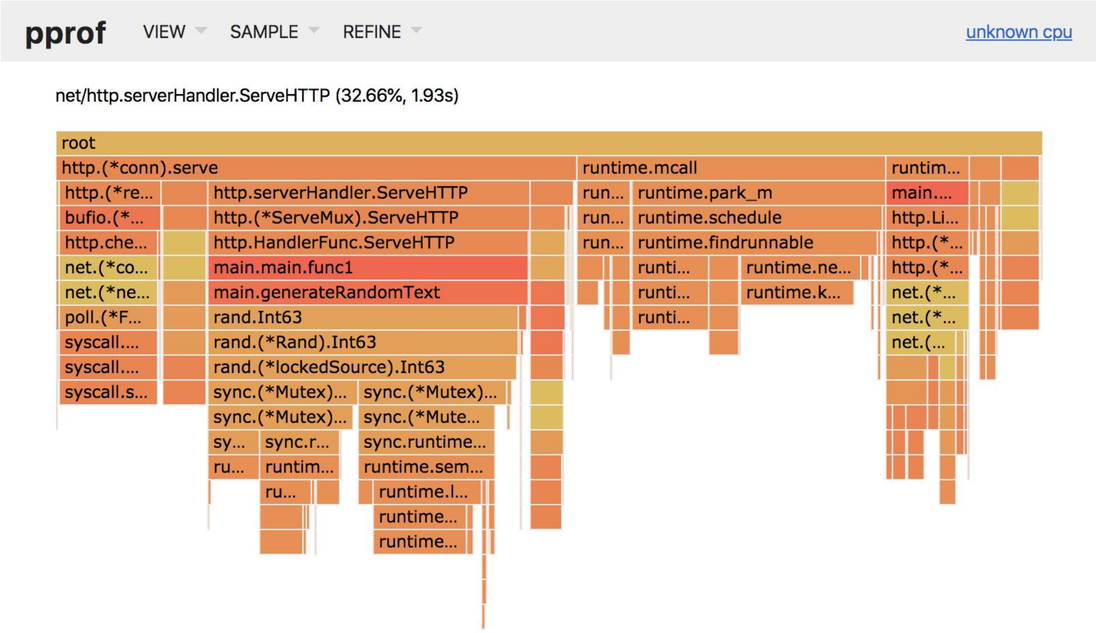
newbie kernel contributor

TO

newbie profiler maintainer



- **ftrace**: dynamic tracing
- **dmesg**: dynamic logging
- **flamegraphs**: static, infrawide profile snapshots



Debugging tools

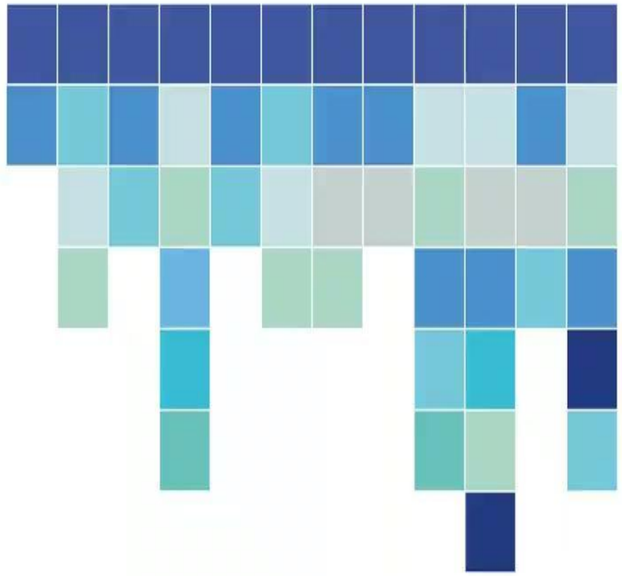
Polar Signals

About Me

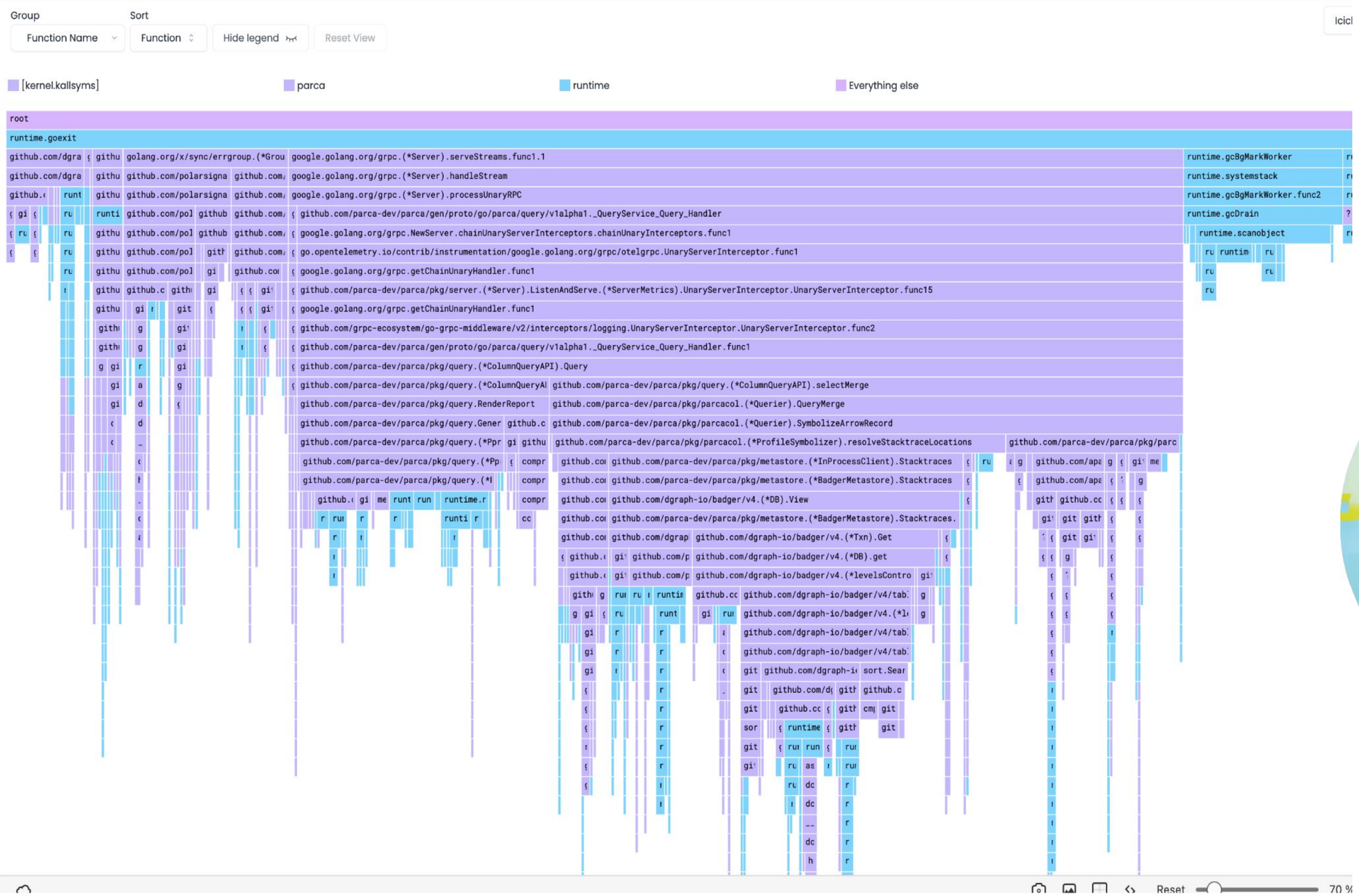
- Junior Software Engineer at Polar Signals
- Open Source Maintainer
  - Parca and Parca-Agent

Sumera Priyadarsini

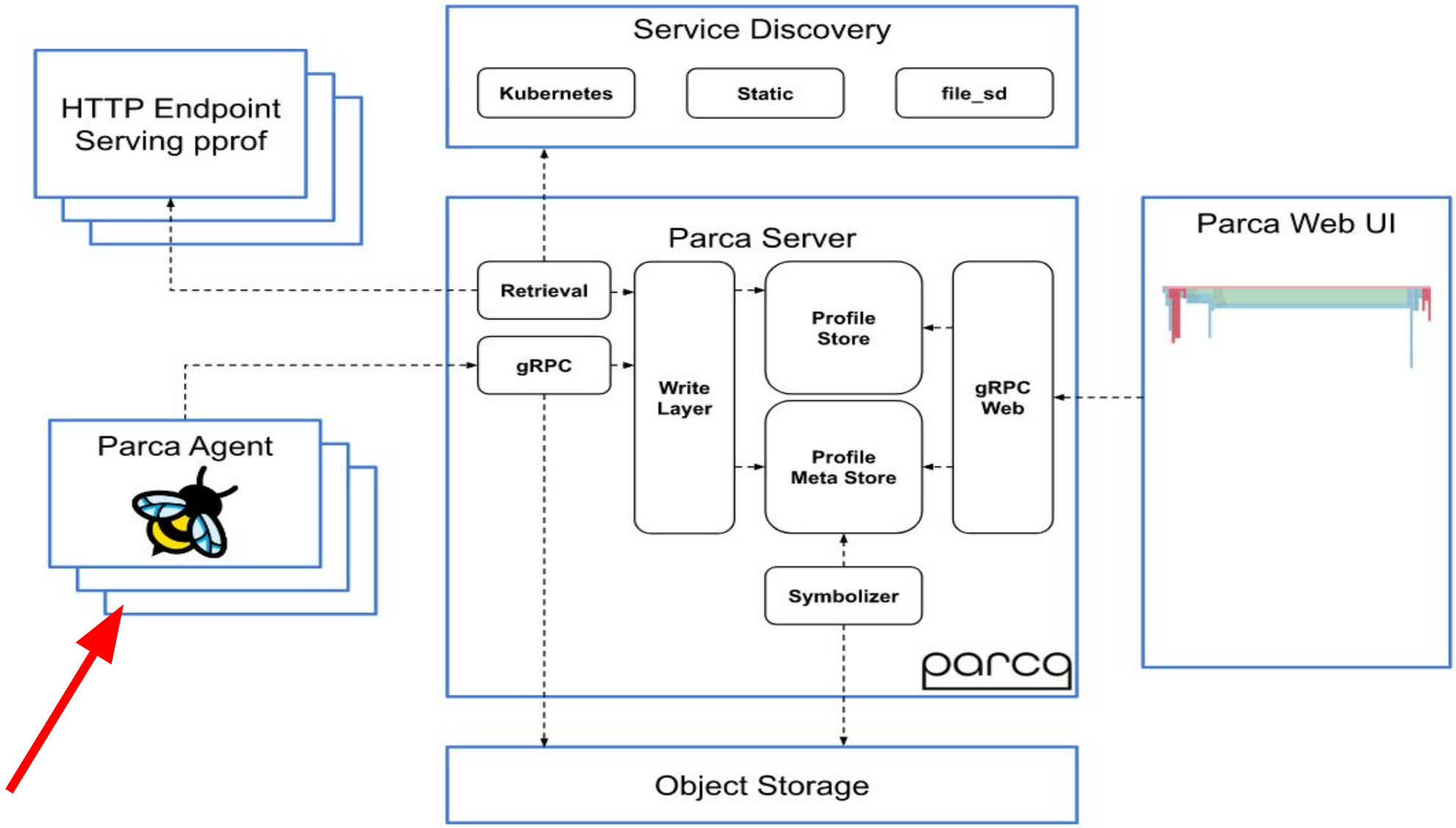
@sylfrena
@SumoOfShinovar



Polar Signals



Polar Signals



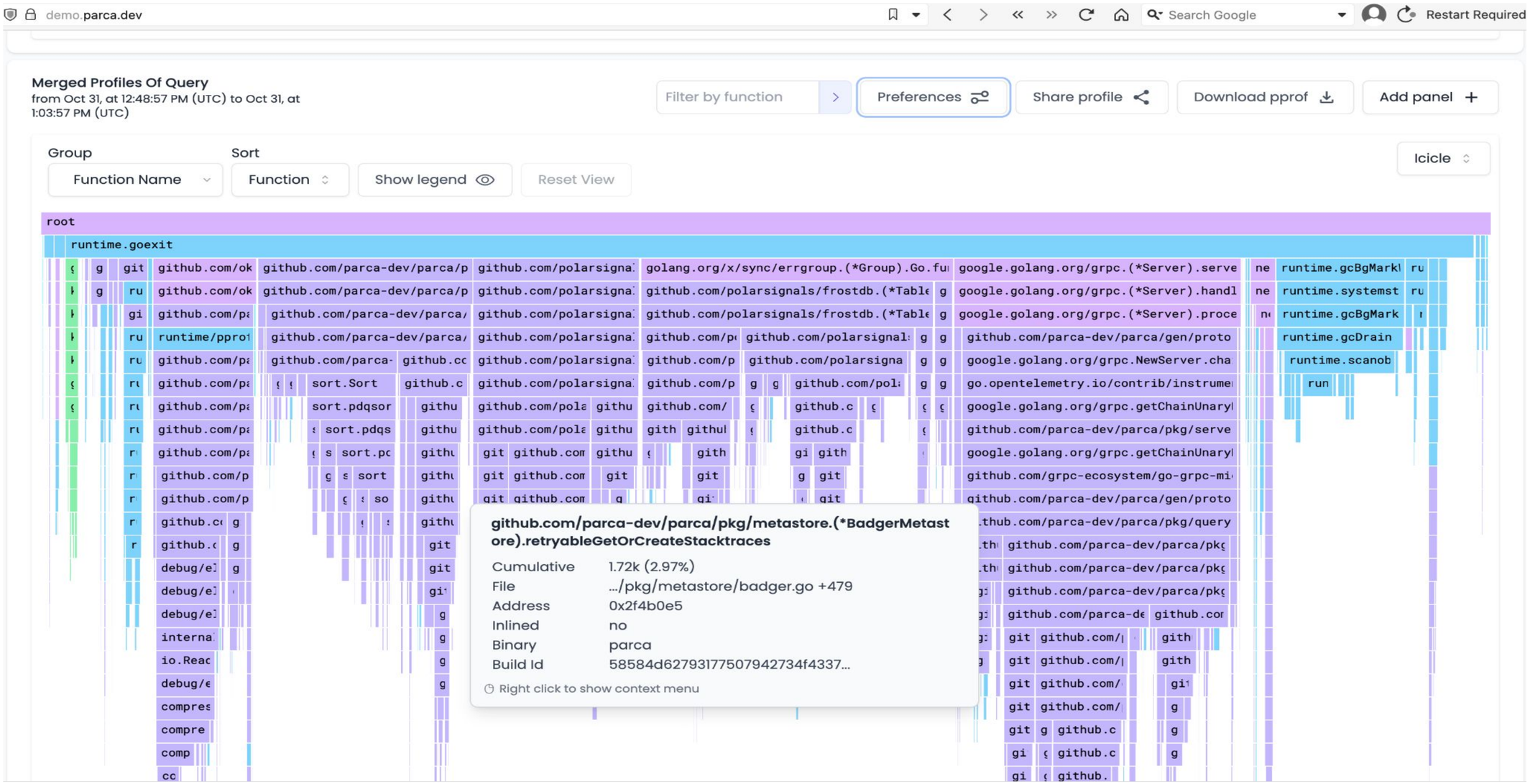
Polar Signals

High Resolution Dynamic Continuous Profiling

High Resolution Profiling

ignals

High Resolution Profiling



Polar Signals

High Resolution Profiling

Merged Profiles Of Query
from Oct 11, at 11:11:32 PM (UTC) to Oct 11, at 11:26:32 PM (UTC)

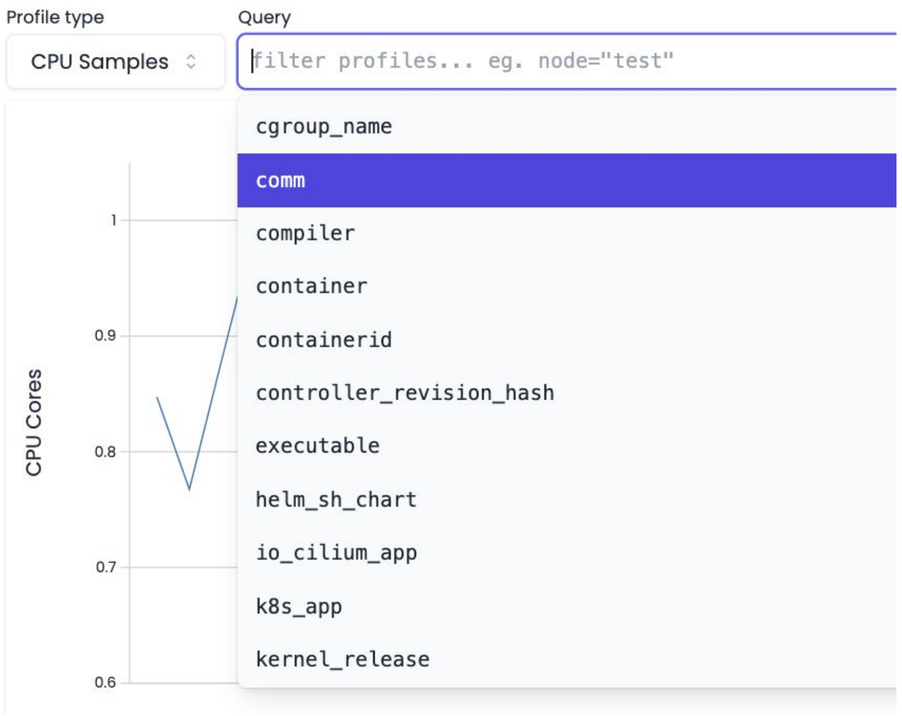
Filter by function | Preferences | Share profile | Download pprof | Add panel +

Columns | Table

Flat	Cumulative	Cumulative (%)	Name
0	13.52k	106.27%	google.golang.org/grpc.getChainUnaryHandler.func1
0	12.7k	99.83%	runtime.goexit
0	6.77k	53.18%	google.golang.org/grpc.(*Server).processUnaryRPC
0	6.77k	53.18%	google.golang.org/grpc.(*Server).handleStream
0	6.77k	53.18%	google.golang.org/grpc.(*Server).serveStreams.func1.1
0	6.76k	53.16%	go.opentelemetry.io/contrib/instrumentation/google.golang.org/grpc/otelgrpc.UnaryServerInterceptor.func1
0	6.76k	53.16%	google.golang.org/grpc.NewServer.chainUnaryServerInterceptors.chainUnaryInterceptors.func1
0	6.76k	53.14%	github.com/parca-dev/parca/pkg/server.(*Server).ListenAndServe.(*ServerMetrics).UnaryServerInterceptor.UnaryServerInterceptor.func15
0	6.76k	53.13%	github.com/grpc-ecosystem/go-grpc-middleware/v2/interceptors/logging.UnaryServerInterceptor.UnaryServerInterceptor.func2
0	6.52k	51.23%	github.com/parca-dev/parca/pkg/query.(*ColumnQueryAPI).Query
0	6.52k	51.23%	github.com/parca-dev/parca/gen/proto/go/parca/query/v1alpha1._QueryService_Query_Handler.func1
0	6.52k	51.23%	github.com/parca-dev/parca/gen/proto/go/parca/query/v1alpha1._QueryService_Query_Handler
0	5.42k	42.59%	github.com/parca-dev/parca/pkg/parcacol.(*Querier).QueryMerge
0	5.42k	42.59%	github.com/parca-dev/parca/pkg/query.(*ColumnQueryAPI).selectMerge
1	5.41k	42.52%	github.com/parca-dev/parca/pkg/parcacol.(*Querier).SymbolizeArrowRecord
44	3.28k	25.79%	github.com/parca-dev/parca/pkg/parcacol.(*ProfileSymbolizer).resolveStacktraceLocations
5	2.99k	23.51%	runtime.systemstack
0	2.95k	23.19%	github.com/dgraph-io/badger/v4.(*DB).View
0	2.52k	19.82%	github.com/parca-dev/parca/pkg/metastore.(*BadgerMetastore).Stacktraces
0	2.52k	19.82%	github.com/parca-dev/parca/pkg/metastore.(*InProcessClient).Stacktraces
4	2.49k	19.59%	github.com/parca-dev/parca/pkg/metastore.(*BadgerMetastore).Stacktraces.func1
10	2.33k	18.33%	github.com/dgraph-io/badger/v4.(*Txn).Get
0	2.22k	17.44%	runtime.gcBgMarkWorker
0	2.21k	17.36%	runtime.gcBgMarkWorker.func2
94	2.21k	17.34%	runtime.gcDrain
13	2.15k	16.86%	github.com/dgraph-io/badger/v4.(*DB).get
808	2.13k	16.74%	runtime.scanobject
148	2.11k	16.58%	github.com/parca-dev/parca/pkg/parcacol.BuildArrowLocations
0	1.98k	15.55%	golang.org/x/sync/errgroup.(*Group).Go.func1

Polar Signals

parca



Polar Signals

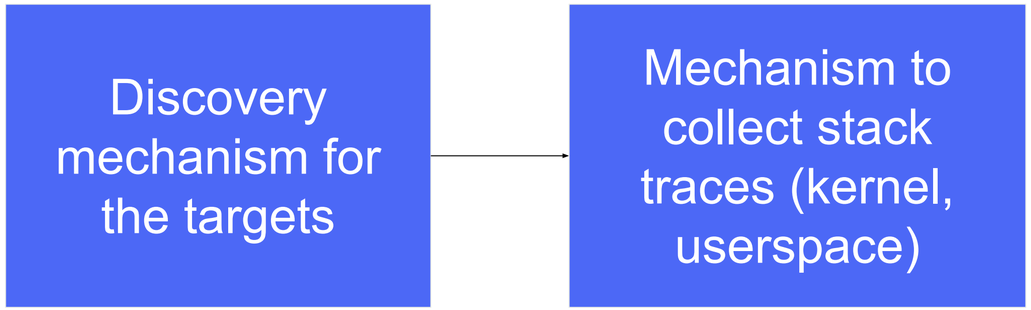
eBPF Profilers: Userspace and Kernelspace

Profilers for the cloud native environment

Discovery mechanism for the targets

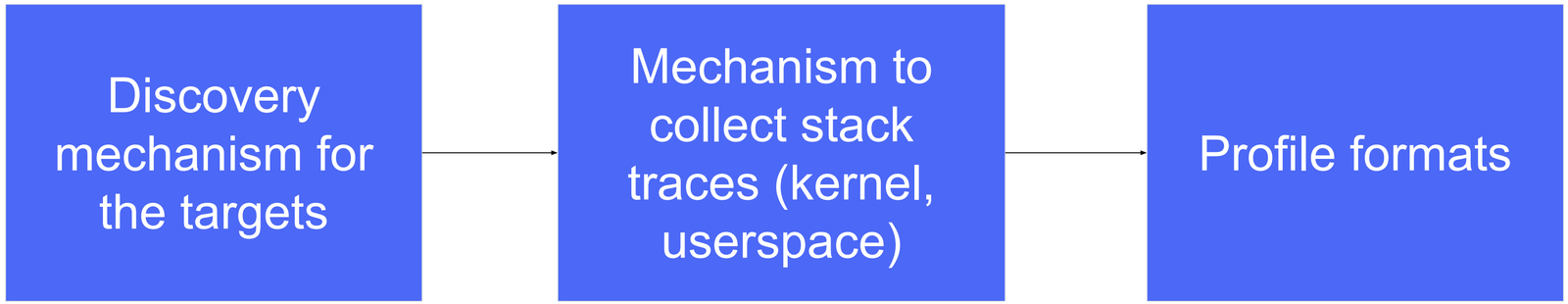
Polar Signals

Profilers for the cloud native environment



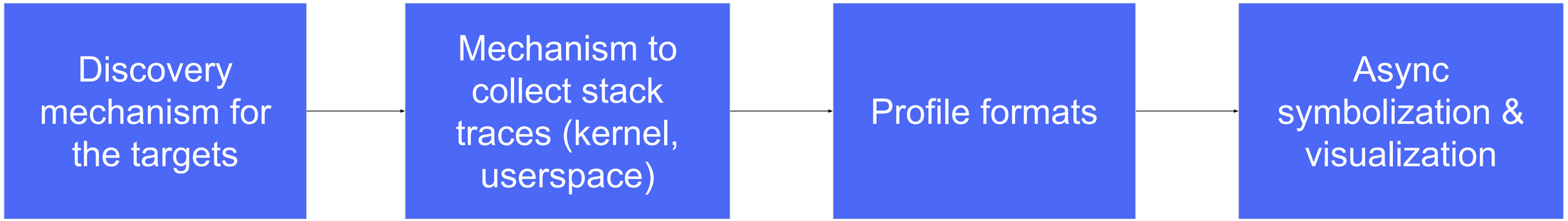
Polar Signals

Profilers for the cloud native environment



Polar Signals

Profilers for the cloud native environment



Polar Signals

Profilers for the cloud native environment

Discovery mechanism for the targets

→

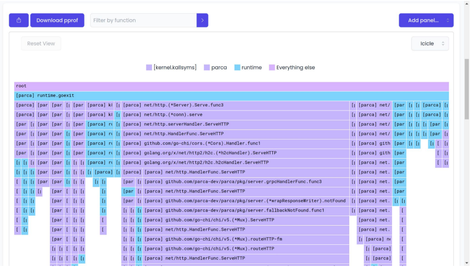
Mechanism to collect stack traces (kernel, userspace)

→

Profile formats

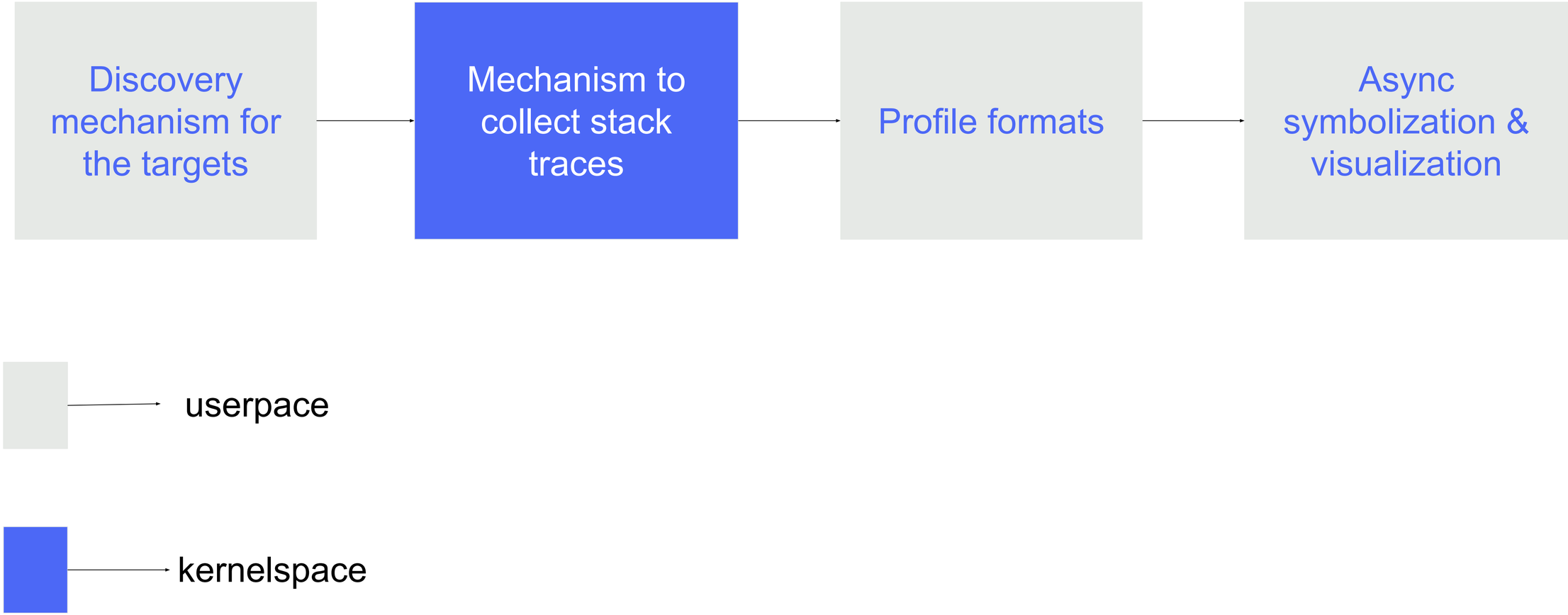
→

Async symbolization & visualization

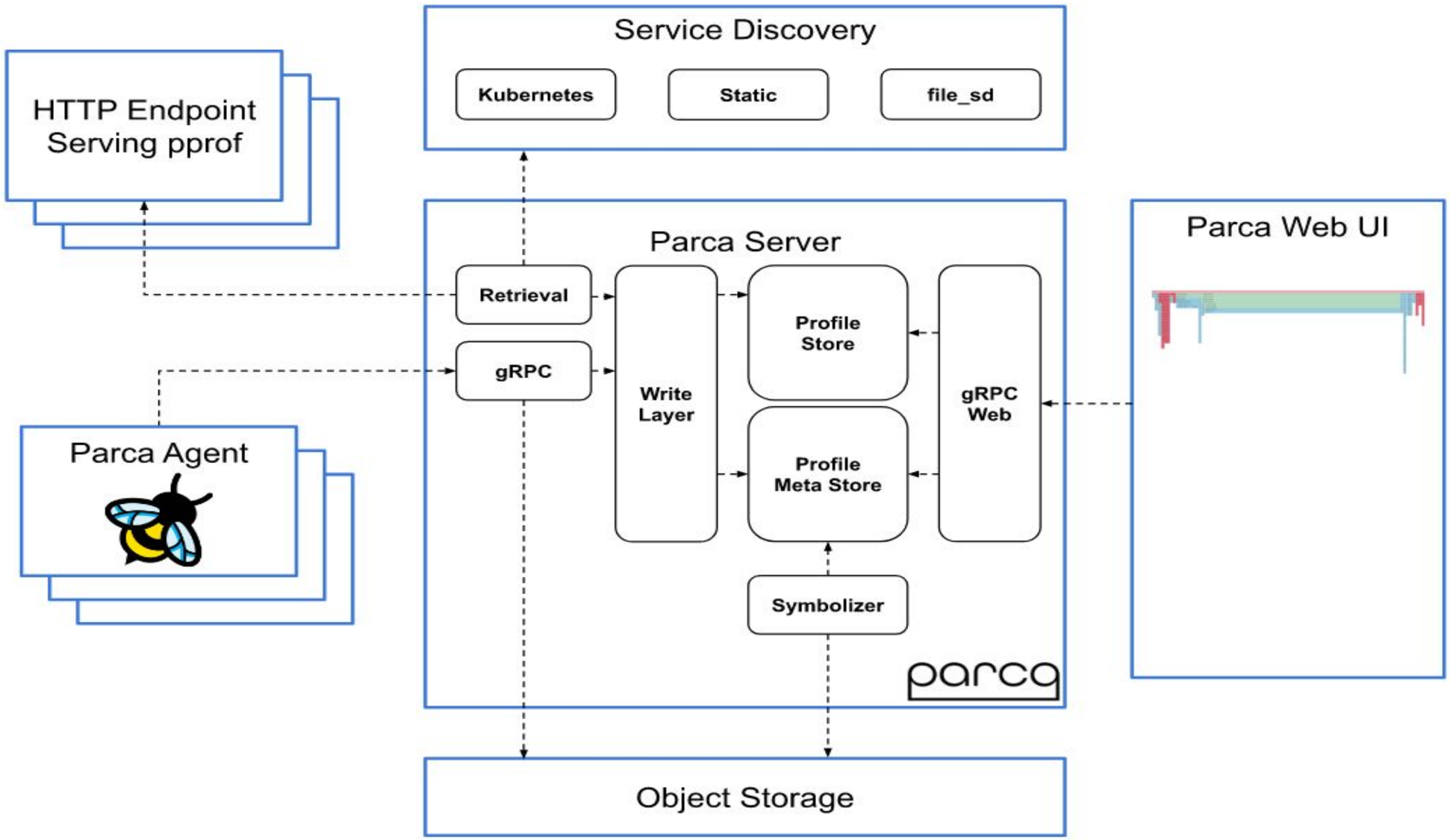


Polar Signals

Profilers for the cloud native environment



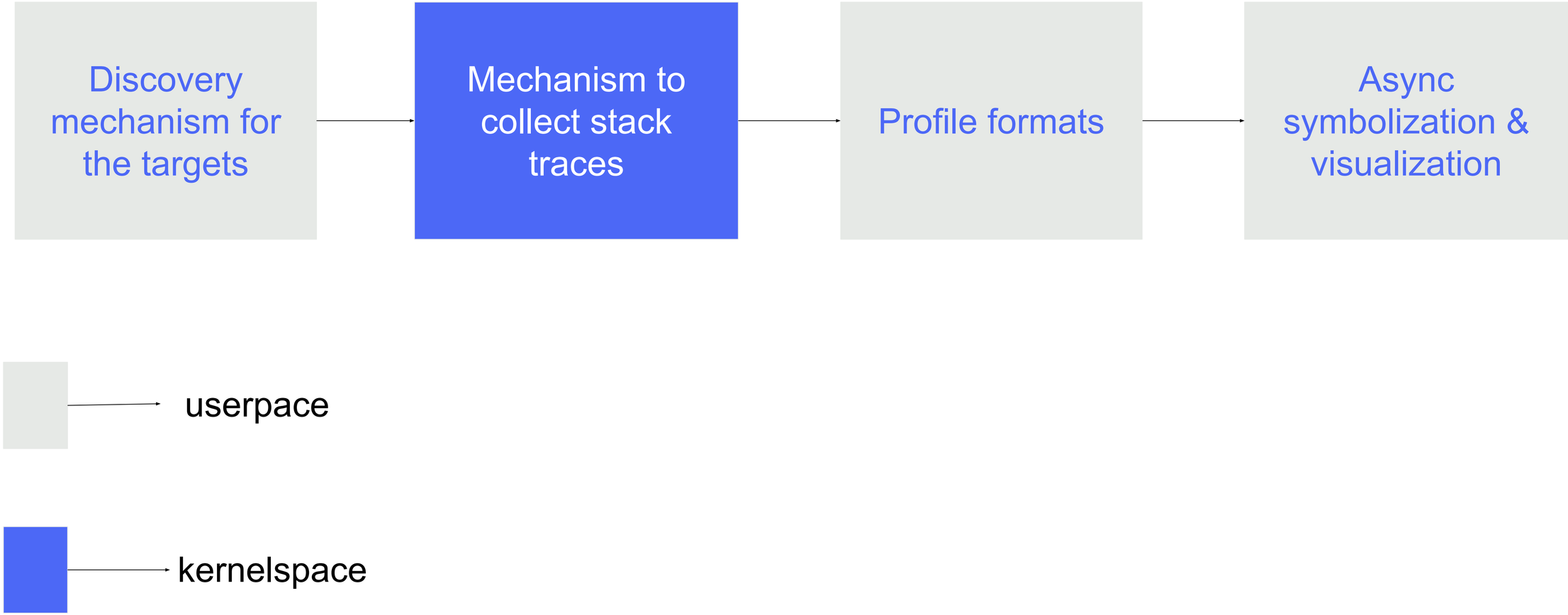
Polar Signals



Polar Signals

Userspace

Profilers for the cloud native environment



Polar Signals

Userspace

- Discovery of targets
  - pods
  - binaries
- Debuginfo extraction
  - can be uploaded separately

Userspace

- Discovery of targets
- Debuginfo extraction
- Extract Unwind information from DWARF
  - **Executable Linkable format - ELF**: For obj file, executable program, shared object etc
  - **DWARF - widely used debugging format**: CIE - Common Information Entry
  - **Tools to read ELF and/or DWARF information**: readelf, objdump, elfutils, llvm-dwarfdump

```
// UnwindTableRow represents a single row in the unwind table.
// x86_64: rip (instruction pointer register), rsp (stack pointer register), rbp (base pointer/frame pointer register)
// aarch64: lr(link register), sp(stack pointer register), fp(frame pointer register)
Francisco Javier Honduvilla Coto, 13 months ago | 2 authors (Francisco Javier Honduvilla Coto and others)
type UnwindTableRow struct {
    // The address of the machine instruction.
    // Each row covers a range of machine instruction, from its address (Loc) to that of the row below.
    Loc uint64
    // CFA, the value of the stack pointer in the previous frame.
    CFA frame.DWRule
    // The value of the RBP register.
    RBP frame.DWRule
    // The value of the saved return address. This is not needed in x86_64 as it's part of the ABI but is necessary
    // in arm64.
    RA frame.DWRule
}
```

Polar Signals

Userspace: DWARF Unwind Tables

```go
// UnwindTableRow represents a single row in the unwind table.
// x86_64: rip (instruction pointer register), rsp (stack pointer register), rbp (base pointer/frame pointer register)
// aarch64: lr(link register), sp(stack pointer register), fp(frame pointer register)
Francisco Javier Honduvilla Coto, 13 months ago | 2 authors (Francisco Javier Honduvilla Coto and others)
type UnwindTableRow struct {
    // The address of the machine instruction.
    // Each row covers a range of machine instruction, from its address (Loc) to that of the row below.
    Loc uint64
    // CFA, the value of the stack pointer in the previous frame.
    CFA frame.DWRule
    // The value of the RBP register.
    RBP frame.DWRule
    // The value of the saved return address. This is not needed in x86_64 as it's part of the ABI but is necessary
    // in arm64.
    RA frame.DWRule
}
```

```go
// CompactUnwindTableRows encodes unwind information using 2x 64 bit words.
// `lrOffset` is the link register for arm64; it is initialized to 0 for x86.
You, 3 months ago | 2 authors (You and others)
type CompactUnwindTableRow struct {
    pc        uint64
    lrOffset  int16
    cfaType   uint8
    rbpType   uint8
    cfaOffset int16
    rbpOffset int16
}
```

compact unwind table

Polar Signals

Userspace

- DWARF information is included in (ELF) binaries
- Extract Unwind tables from DWARF info

```
=> Function start: 293c0, Function end: 29427
    pc: 293c0 cfa_type: 2  rbp_type: 0  cfa_offset: 8    rbp_offset: 0
    pc: 293c5 cfa_type: 2  rbp_type: 0  cfa_offset: 16   rbp_offset: 0
    pc: 293cb cfa_type: 2  rbp_type: 0  cfa_offset: 32   rbp_offset: 0
    pc: 29408 cfa_type: 2  rbp_type: 0  cfa_offset: 16   rbp_offset: 0
    pc: 29409 cfa_type: 2  rbp_type: 0  cfa_offset: 8    rbp_offset: 0
    pc: 2940e cfa_type: 2  rbp_type: 0  cfa_offset: 32   rbp_offset: 0
```

compact unwind table snippet for `libc` on x86

Polar Signals

```
=> Function start: 293c0, Function end: 29427
    pc: 293c0 cfa_type: 2  rbp_type: 0  cfa_offset: 8    rbp_offset: 0
    pc: 293c5 cfa_type: 2  rbp_type: 0  cfa_offset: 16   rbp_offset: 0
    pc: 293cb cfa_type: 2  rbp_type: 0  cfa_offset: 32   rbp_offset: 0
    pc: 29408 cfa_type: 2  rbp_type: 0  cfa_offset: 16   rbp_offset: 0
    pc: 29409 cfa_type: 2  rbp_type: 0  cfa_offset: 8    rbp_offset: 0
    pc: 2940e cfa_type: 2  rbp_type: 0  cfa_offset: 32   rbp_offset: 0
```

compact unwind table snippet for `libc` on x86

x86

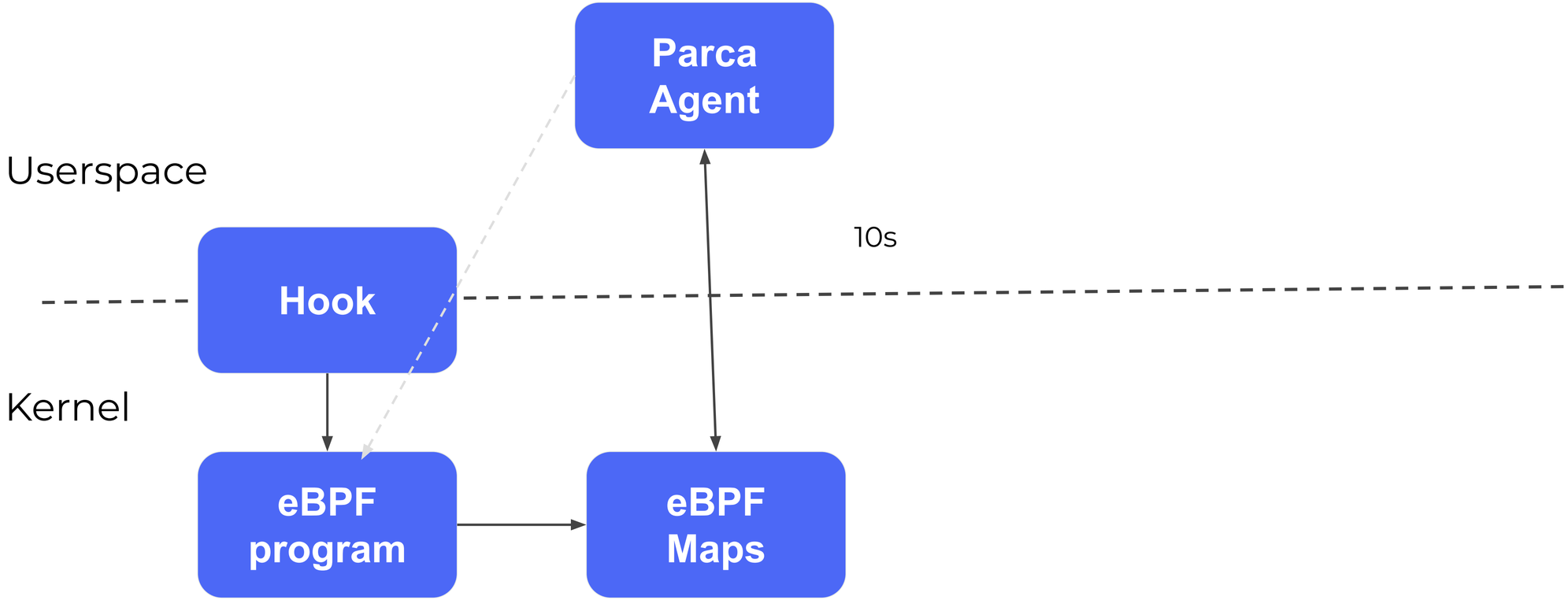
Aarch64

```
=> Function start: 26c00, Function end: 26c80
    pc: 26c00 cfa_type: 2  rbp_type: 0  cfa_offset: 0    rbp_offset: 0   lr_offset: 0
    pc: 26c04 cfa_type: 2  rbp_type: 1  cfa_offset: 48   rbp_offset: -48 lr_offset: -
    pc: 26c14 cfa_type: 2  rbp_type: 1  cfa_offset: 48   rbp_offset: -48 lr_offset: -
    pc: 26c6c cfa_type: 2  rbp_type: 0  cfa_offset: 0    rbp_offset: 0   lr_offset: 0
    pc: 26c70 cfa_type: 2  rbp_type: 1  cfa_offset: 48   rbp_offset: -48 lr_offset: -
=> Function start: 26a00, Function end: 26a0c
    pc: 26a00 cfa_type: 2  rbp_type: 0  cfa_offset: 0    rbp_offset: 0   lr_offset: 0
    pc: 26a04 cfa_type: 2  rbp_type: 1  cfa_offset: 16   rbp_offset: -16 lr_offset: -
```

compact unwind table snippet for `libc` on Arm64

Kernelspace

Communicating with Userspace



Polar Signals

Communicating with Userspace

```
        }
        r
    }

    mapsH
    if er
        r
    }
    m.pro
    retur
}

// writeU
    You
```

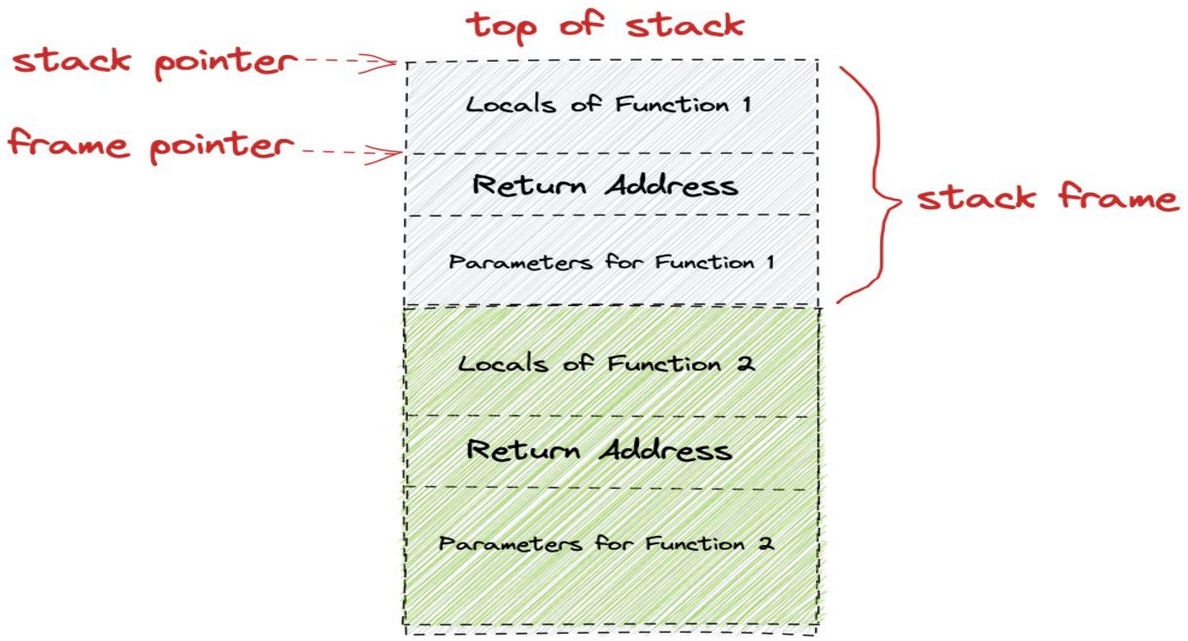
```
    pythonPIDToProcessInfo       *libbpf.BPFMap
    pythonVersionSpecificOffsets *libbpf.BPFMap
    pythonVersionToOffsetIndex   map[string]uint32

    unwindShards *libbpf.BPFMap
    unwindTables *libbpf.BPFMap
    programs     *libbpf.BPFMap
    processInfo  *libbpf.BPFMap

    // Unwind stuff 🔬
    processCache      *processCache
    mappingInfoMemory profiler.EfficientBuffer
```

```go
func (m *Maps) writeUnwindTableRow(rowSlice *profiler.EfficientBuffer, row unwind.CompactUnwindTableRow, arch elf.Machine) {
    // .pc
    rowSlice.PutUint64(row.Pc())
    if arch == elf.EM_AARCH64 {
        // .lr_offset
        rowSlice.PutInt16(row.LrOffset())
    }
    // .cfa_type
    rowSlice.PutUint8(row.CfaType())
    // .rbp_type
    rowSlice.PutUint8(row.RbpType())
    // .cfa_offset
    rowSlice.PutInt16(row.CfaOffset())
    // .rbp_offset
    rowSlice.PutInt16(row.RbpOffset())
}
```

Polar Signals



Kernelspace:

Walking the Stacks

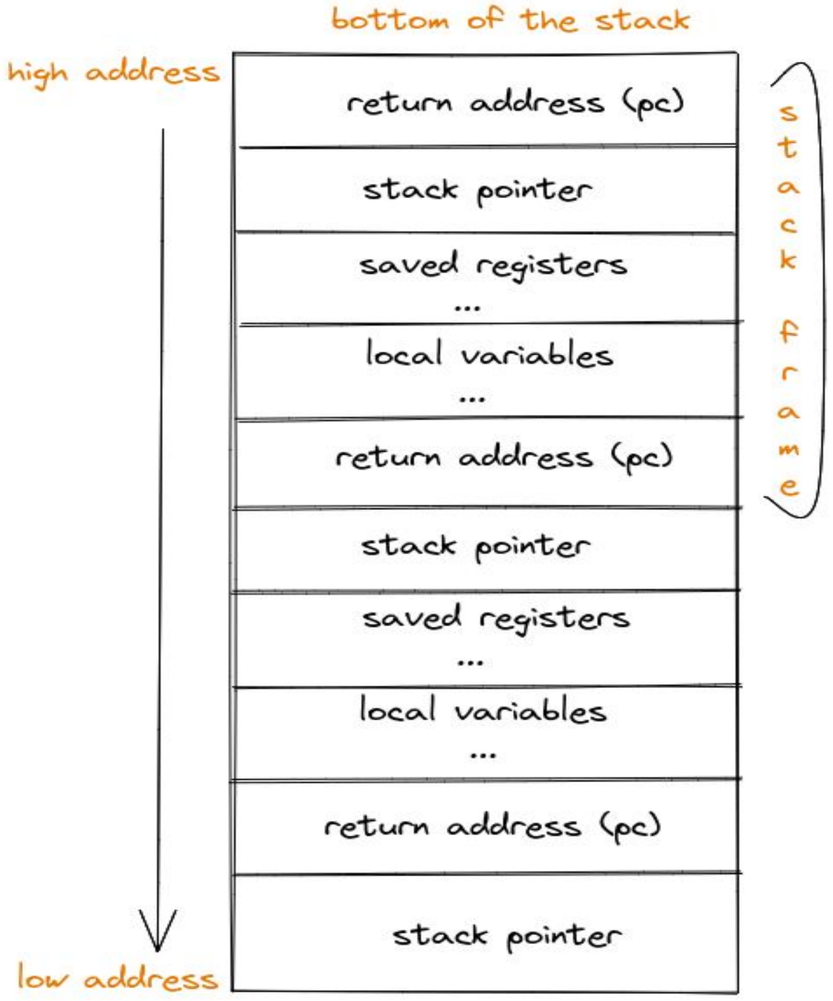
Polar Signals

Stacktraces

- What collecting stack traces involve
  - Kernel stacks
  - Application stacks
- Direction of stack growth
- So what are stack pointers, where do they come from

Figure 3.3: Stack Frame with Base Pointer

Position	Contents	Frame
8n+16(%rbp)	argument eightbyte n	Previous
	...	
16(%rbp)	argument eightbyte 0	
8(%rbp)	return address	Current
0(%rbp)	previous %rbp value	
-8(%rbp)	unspecified	
	...	
0(%rsp)	variable size	
-128(%rsp)	red zone	



From: x86_64 ABI specification

Polar Signals

Userspace

```
=> Function start: 293c0, Function end: 29427
    pc: 293c0 cfa_type: 2  rbp_type: 0  cfa_offset: 8    rbp_offset: 0
    pc: 293c5 cfa_type: 2  rbp_type: 0  cfa_offset: 16   rbp_offset: 0
    pc: 293cb cfa_type: 2  rbp_type: 0  cfa_offset: 32   rbp_offset: 0
    pc: 29408 cfa_type: 2  rbp_type: 0  cfa_offset: 16   rbp_offset: 0
    pc: 29409 cfa_type: 2  rbp_type: 0  cfa_offset: 8    rbp_offset: 0
    pc: 2940e cfa_type: 2  rbp_type: 0  cfa_offset: 32   rbp_offset: 0
```

x86

compact unwind table snippet for `libc` on x86

```
=> Function start: 26c00, Function end: 26c80
    pc: 26c00 cfa_type: 2  rbp_type: 0  cfa_offset: 0    rbp_offset: 0   lr_offset: 0
    pc: 26c04 cfa_type: 2  rbp_type: 1  cfa_offset: 48   rbp_offset: -48 lr_offset: -
    pc: 26c14 cfa_type: 2  rbp_type: 1  cfa_offset: 48   rbp_offset: -48 lr_offset: -
    pc: 26c6c cfa_type: 2  rbp_type: 0  cfa_offset: 0    rbp_offset: 0   lr_offset: 0
    pc: 26c70 cfa_type: 2  rbp_type: 1  cfa_offset: 48   rbp_offset: -48 lr_offset: -
=> Function start: 26a00, Function end: 26a0c
    pc: 26a00 cfa_type: 2  rbp_type: 0  cfa_offset: 0    rbp_offset: 0   lr_offset: 0
    pc: 26a04 cfa_type: 2  rbp_type: 1  cfa_offset: 16   rbp_offset: -16 lr_offset: -
```

Aarch64

compact unwind table snippet for `libc` on Arm64

Polar Signals

DWARF Unwinding in x86

```
pc: 293c0 cfa_type: 2  rbp_type: 0  cfa_offset: 8    rbp_offset: 0
pc: 293c5 cfa_type: 2  rbp_type: 0  cfa_offset: 16   rbp_offset: 0
pc: 293cb cfa_type: 2  rbp_type: 0  cfa_offset: 32   rbp_offset: 0
pc: 29408 cfa_type: 2  rbp_type: 0  cfa_offset: 16   rbp_offset: 0
pc: 29409 cfa_type: 2  rbp_type: 0  cfa_offset: 8    rbp_offset: 0
pc: 2940e cfa_type: 2  rbp_type: 0  cfa_offset: 32   rbp_offset: 0
```

compact unwind table snippet for `libc` on x86

Unwind table in x86

```
// HACK(javierhonduco): This is an architectural shortcut we can take. As we
// only support x86_64 at the minute, we can assume that the return address
// is *always* 8 bytes ahead of the previous stack pointer.
#if __TARGET_ARCH_x86
    u64 previous_rip_addr = previous_rsp - 8;
    int err = bpf_probe_read_user(&previous_rip, 8, (void *)(previous_rip_addr));
    if (err < 0) {
      LOG("\n[error] Failed to read previous rip with error: %d", err);
    }
    LOG("\tprevious ip: %llx (@ %llx)", previous_rip, previous_rip_addr);
#endif
```

Polar Signals

DWARF Unwinding in Aarch64

```
=> Function start: 26c00, Function end: 26c80
    pc: 26c00 cfa_type: 2  rbp_type: 0  cfa_offset: 0    rbp_offset: 0   lr_offset: 0
    pc: 26c04 cfa_type: 2  rbp_type: 1  cfa_offset: 48   rbp_offset: -48 lr_offset: -
    pc: 26c14 cfa_type: 2  rbp_type: 1  cfa_offset: 48   rbp_offset: -48 lr_offset: -
    pc: 26c6c cfa_type: 2  rbp_type: 0  cfa_offset: 0    rbp_offset: 0   lr_offset: 0
    pc: 26c70 cfa_type: 2  rbp_type: 1  cfa_offset: 48   rbp_offset: -48 lr_offset: -
=> Function start: 26a00, Function end: 26a0c
    pc: 26a00 cfa_type: 2  rbp_type: 0  cfa_offset: 0    rbp_offset: 0   lr_offset: 0
    pc: 26a04 cfa_type: 2  rbp_type: 1  cfa_offset: 16   rbp_offset: -16 lr_offset: -
```

compact unwind table snippet for `libc` on Arm64

```
#if __TARGET_ARCH_arm64
    // For the leaf frame, the saved pc/ip is always be stored in the link register i
    if (found_lr_offset == 0) {
      previous_rip = PT_REGS_RET(&ctx->regs);
    } else {
      u64 previous_rip_addr = previous_rsp + found_lr_offset;
      int err = bpf_probe_read_user(&previous_rip, 8, (void *)(previous_rip_addr));
      if (err < 0) {
        LOG("\n[error] Failed to read previous rip with error: %d", err);
      }
      LOG("\tprevious ip: %llx (@ %llx)", previous_rip, previous_rip_addr);
    }
#endif
```

Aarch64

Polar Signals

Compilers and Runtimes

Parca Agent Language Support

Compiled Languages

- C
- C++
- Rust
- Go
- and more!

With or without frame pointers!

JIT (Just in Time Compiled) Languages

With Perf map or jitdump

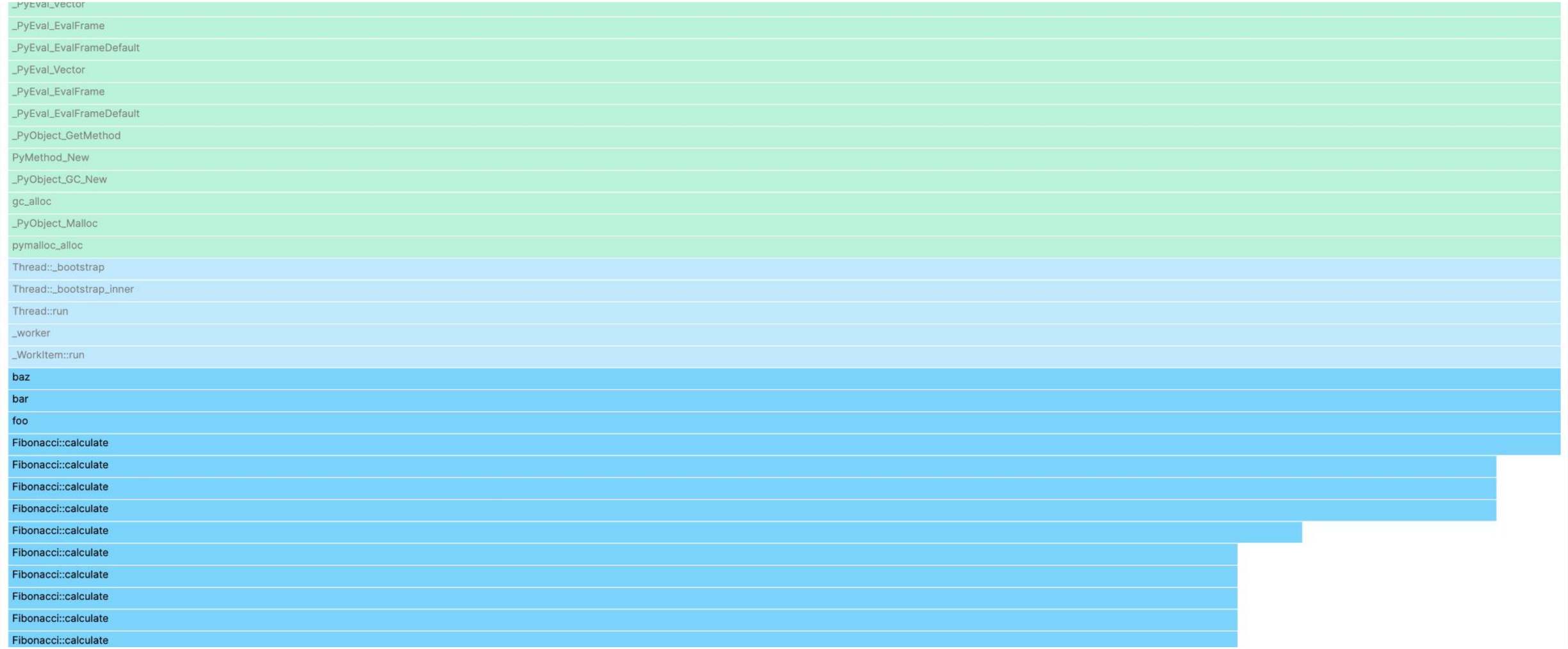
- C#
- Erlang
- JVM(with async-profiler)
- Julia
- NodeJS

Interpreted Languages

- Python
- Ruby

Polar Signals

Interpreted Languages: Python



(3.11)

Polar Signals

Interpreted Languages: Ruby

Filter by function | Preferences | Download pprof | Add panel

Group: Function Name | Sort: Function | Show legend | Reset View | Icicle

root
__libc_start_main_impl
__libc_start_call_main
main
ruby_run_node
rb_ec_exec_node
rb_vm_exec
vm_exec_core
vm_sendish
vm_call_cfunc_with_frame
range_each
range_each_fixnum_loop
rb_yield
rb_yield_0
vm_yield
invoke_block_from_c_bh
<main>

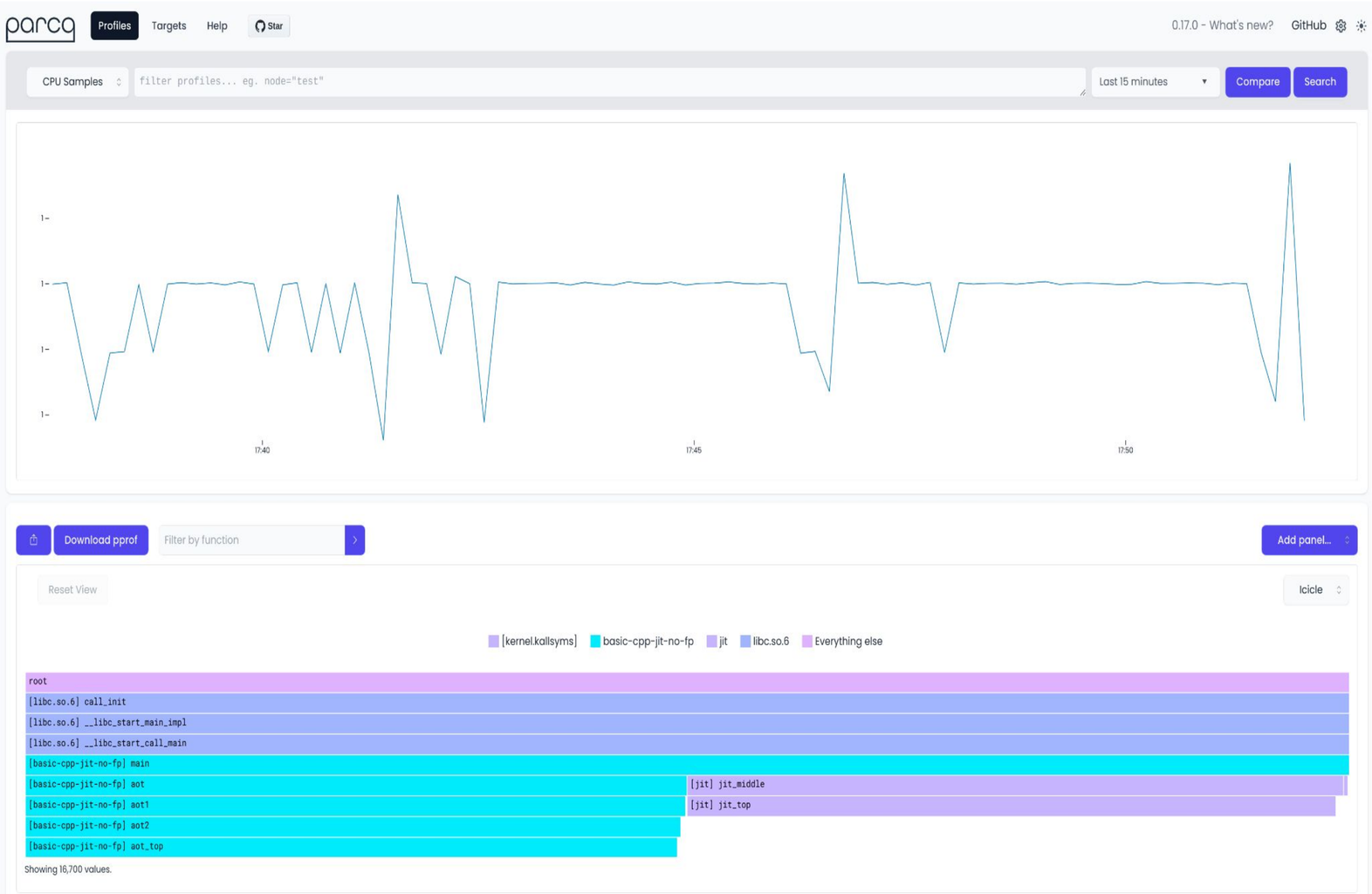
a1	a2	a3
b1	b2	b3
c1	c2	c3
cpu	hog	program

Showing 12,183 values.

(3.0.4)

Polar Signals

Parca Agent Language Support: JITed Runtimes and Stacks




```c
// This implements a simple JIT for x86_64 with support for symbolizatio
// with perfmap. We don't use any JIT framework or assembler for the sak
// of simplicity.

// Some tools have heuristics to unwind the stack even
// with frame pointers *and* unwind information omitted.
#define ENABLE_FRAME_POINTERS_IN_JIT true
// Amount of items to push into the stack to simulate a
// more realistic stack usage. Otherwise stack unwinding with
// frame pointers might work by accident.
#define STACK_ITEMS 30

int __attribute__((noinline)) aot_top() {
  for (int i = 0; i < 1000; i++) {
  }

  return 0;
}

// ahead of time
int __attribute__((noinline)) aot2() { return aot_top(); }

int __attribute__((noinline)) aot1() { return aot2(); }

int __attribute__((noinline)) aot() { return aot1(); }

void add_preamble(char **mem) {
  if (!ENABLE_FRAME_POINTERS_IN_JIT) {
    return;
  }

  *(*mem)++ = 0x55; // push   %rbp
  *(*mem)++ = 0x48; // mov    %rsp,%rbp
  *(*mem)++ = 0x89;
  *(*mem)++ = 0xe5; //   < difference between this and 0xec?
```

Polar Signals

JITed Runtimes: Symbolising VSCode

Polar Signals

- Bottom of the Stack in C++
- Correctly Profiling NodeJS
- Am I contained ?

How Compilers affect Icicle Graphs:

the cool edition

Polar Signals

Clouds and Kernels

Kernels in the Cloud

- GKE by Google Cloud
- Graviton by AWS
- Custom kernels
- Custom cloud kernels

.... and others

Linux Kernel v5.3+ with BTF

Kernels versions we support

Polar Signals

- Bad kernel bug causing a CPU lockup in Linux Kernel >= **v5.19** && **< v6.1**
  - https://github.com/parca-dev/parca-agent/issues/1675

- Fix backported to ~stable for 6.1 and 6.3

Kernels versions we prefer and recommend

Linux Kernel >= **v6.4**

Polar Signals

BUT....

- Most OS Distros don’t update as frequently as upstream
- Custom kernels don’t backport fixes
- Cloud providers largely use v5.xx and <v6.2x kernels

Future Work

Future Roadmap

- Language Support
  - JVM
  - PHP
  - LuaJIT
  - Python
- Improve Memory Usage in the Agent
- More code coverage in Testing:
  - Kernel tests with QEMU
  - Integration tests to ensure correctness
  - Using ASAN and TSAN while developing
  - Snapshot testing of the unwind tables

Resources

- https://www.polarsignals.com/blog/posts/2023/10/17/profiling-arm64-with-ebpf-in-parca-agent
- https://www.polarsignals.com/blog/posts/2023/10/04/profiling-python-and-ruby-with-ebpf
- https://www.polarsignals.com/blog/posts/2022/11/29/dwarf-based-stack-walking-using-ebpf
- https://www.polarsignals.com/blog/posts/2023/03/28/how-to-read-icicle-and-flame-graphs/
- https://www.polarsignals.com/blog/posts/2022/01/13/fantastic-symbols-and-where-to-find-them/
- https://demo.parca.dev

Polar Signals

Thank you for listening!

by Sumera | @sylfrena | @SumoOfShinovar

Polar Signals