Polar Signals

Profiling in the Cloud Native Era

by Matthias Loibl | @metalmatze

About me

@metalmatze

- Senior Software Engineer at Polar Signals
- Open Source Maintainer
  - Parca
  - Thanos
  - Prometheus
  - Prometheus Operator
  - Pyrra

Polar Signals

caddy-response-latency

We want our demo to be fast and therefore we want 90% of our responses to be faster than 50ms as seen by Caddy.

Objective

90.000% in 4w

faster than 50ms

Availability

97.093%

Slow: 9,863

Total: 339,334

Error Budget

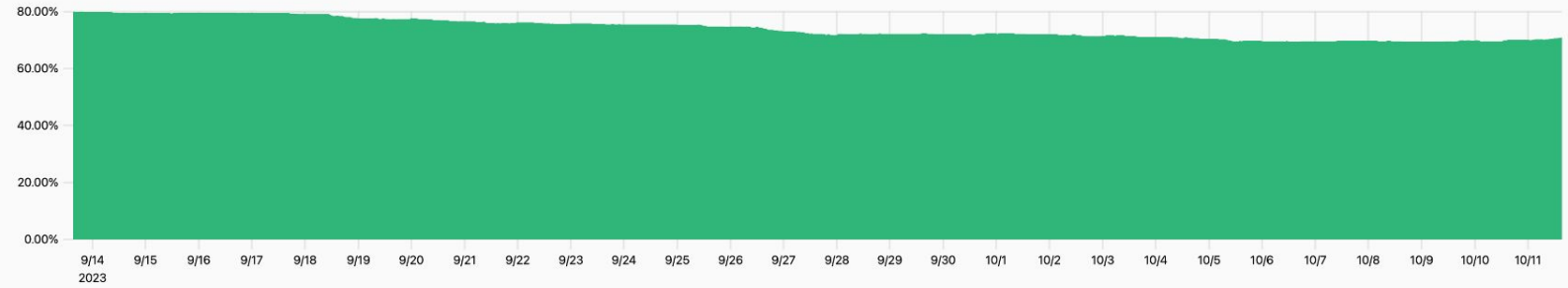
70.934%

4w | 1w | 1d | 12h | 1h | 40m | Absolute | Relative

Error Budget

What percentage of the error budget is left over time?

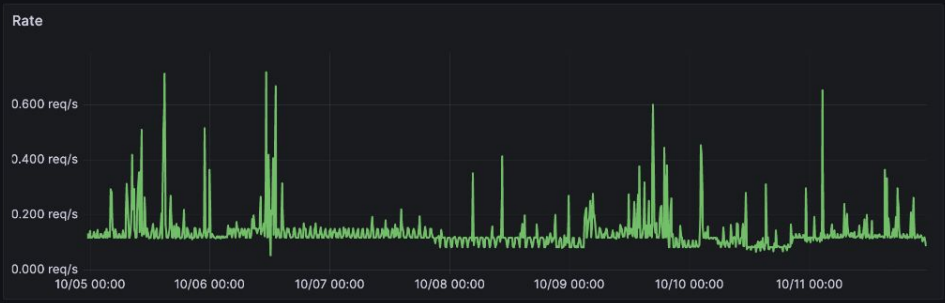
Prometheus



Time: -- Value: -

Prometheus: Prometheus SLO: caddy-response-latency

Objective	Window	Availability	Error Budget
90.000%	4 weeks	97.074%	70.738%



Overview

PromCon EU 2023 is the eighth conference fully dedicated to the Prometheus monitoring system. It will take place 2023-09-28 & 2023-09-29 (Thu & Fri) in Berlin as a single-track event with space for 300 attendees.

PromCon aims to connect Prometheus users and developers from around the world in order to exchange knowledge, best practices, and experience gained around using Prometheus. We also want to collaborate to build a community and grow professional connections around systems and service monitoring.

Get an impression of PromCon EU 2019:

Profiling

Profiling

As old as programming

What?

Profiling is a form of dynamic program analysis that **measures** resource consumption, for example:

- the **space** (memory)
- **time complexity** of a program (CPU),
- **usage of instructions**,
- **frequency** and **duration** of function calls

https://en.wikipedia.org/wiki/Profiling_(computer_programming)

Tracing

- Recording each and every event constantly
- High costs

Polar Signals

Sampling

- Sample for a certain duration
  - Eg. 10 seconds
- Periodically observe function call stack
  - Eg. 100x per second
- Low overhead*
  - <0.5% CPU
  - ~4MB memory

*https://cloud.google.com/blog/products/management-tools/in-tests-cloud-profiler-adds-negligible-overhead

Why?

Improve Performance!

Save Money

30%

Many organizations have 20-30% of resources wasted with easily optimized code paths.

How to profile Go programs?

pprof

pprof descends from the Google Performance Tools suite.

pprof profiling is built into the Go runtime.

Polar Signals

Open Standards

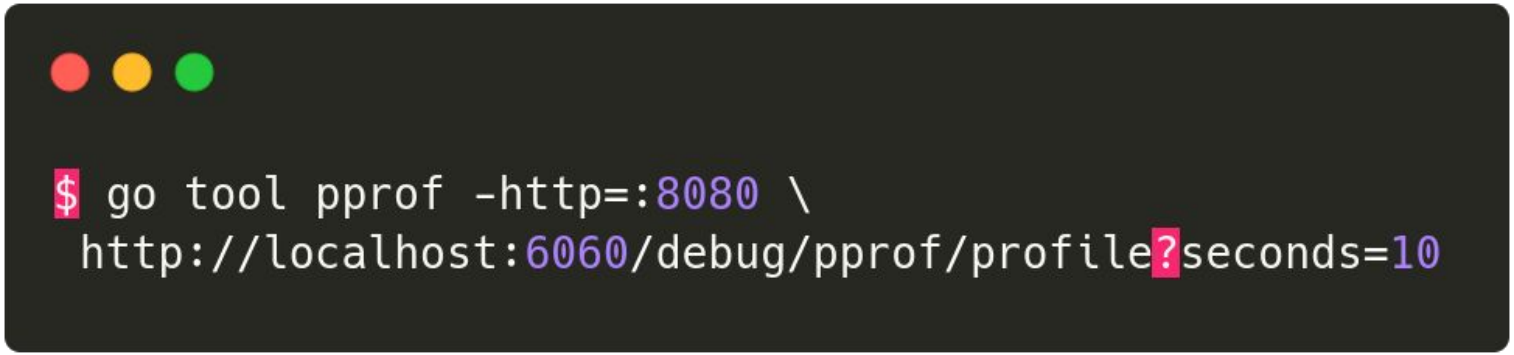
Supports any pprof formatted profiles allowing for wide language adoption and interoperability with existing tooling.

pprof for other languages

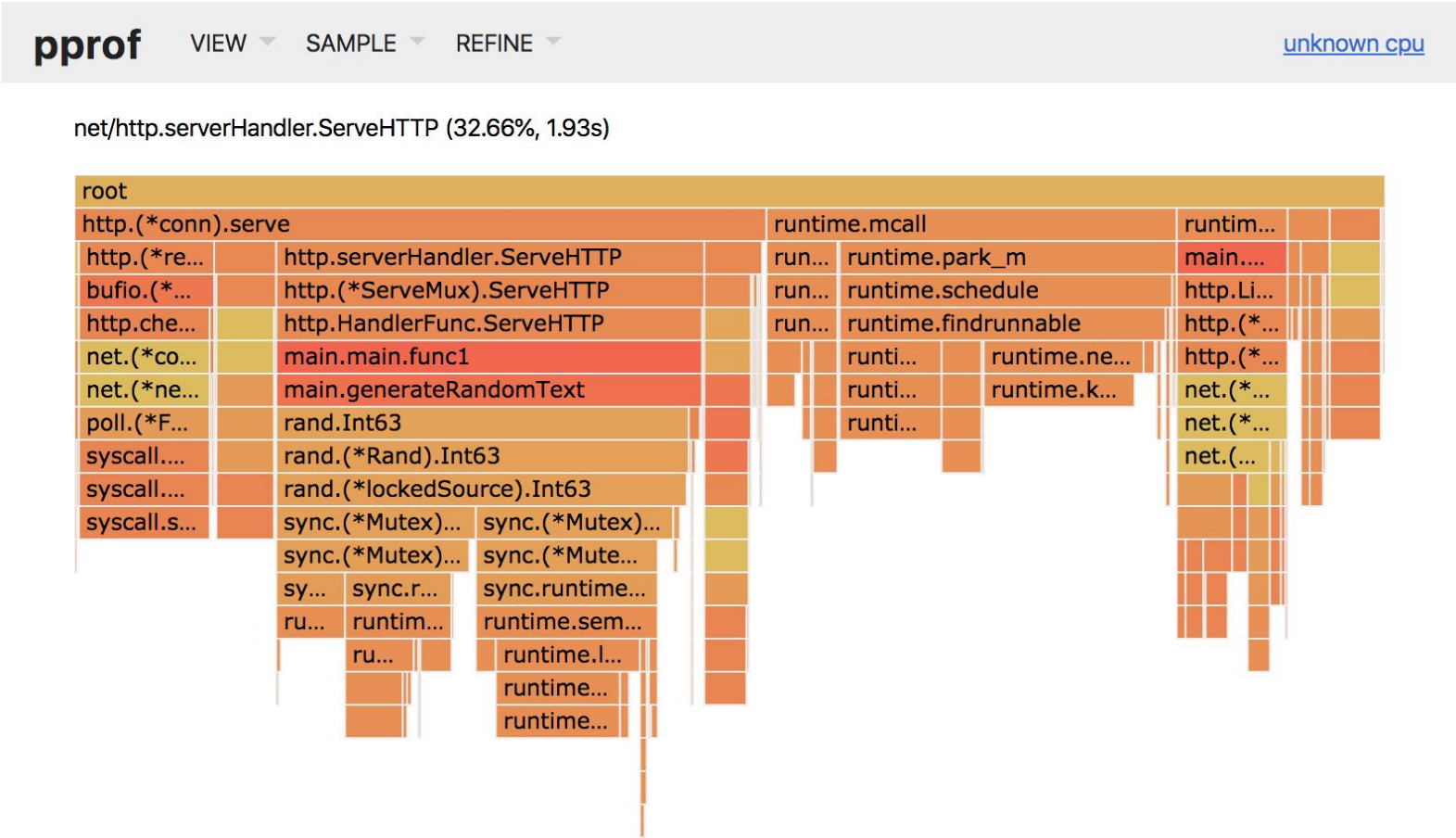
Language/Runtime	CPU	Heap	Allocations	Blocking	Mutex Contention	Extra
Go	✓	✓	✓	✓	✓	goroutine, fgprof
Rust	✓	✗	✗	✗	✗	
Python	✓	✓	✗	✗	✗	
NodeJS	✓	✓	✗	✗	✗	
JVM	✓	✗	✗	✗	✗	

https://www.parca.dev/docs/ingestion

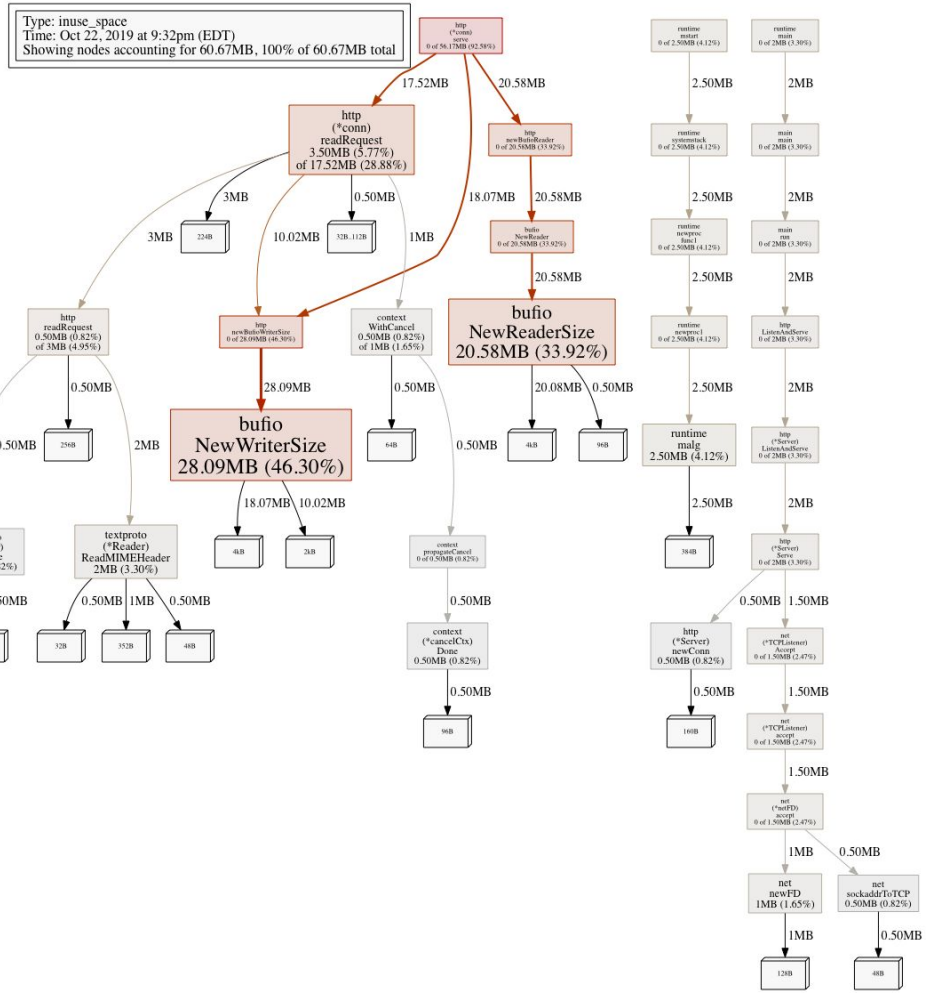
Polar Signals



Polar Signals



Polar Signals



Polar Signals

Code

```
package main

func main() {
    iterateLong()
    iterateShort()
}

func iterateLong() {
    iterate(10_000_000_000)
}

func iterateShort() {
    iterate(1_000_000_000)
}

func iterate(iterations int) {
    for i := 0; i < iterations; i++
{   }
}
```

Folded stack-trace

```
iterate;iterateLong;main
iterate;iterateLong;main
iterate;iterateLong;main
iterate;iterateLong;main
...
iterate;iterateShort;main
iterate;iterateShort;main
```


Polar Signals

Folded stack-trace

```
iterate;iterateLong;main
iterate;iterateLong;main
iterate;iterateLong;main
iterate;iterateLong;main
...
iterate;iterateShort;main
iterate;iterateShort;main
```

pprof

```
$ protoc --decode perftools.profiles.Profile
cpuprofile.pb

sample {
  location_id: 1
  location_id: 2
  location_id: 3
  value: 253
}
sample {
  location_id: 1
  location_id: 13
  location_id: 3
  value: 26
}
```

+ a bunch of metadata to resolve the location

```go
package main

import (
    "log"
    "net/http"
    "net/http/pprof"
)

func main() {
    mux := http.NewServeMux()
    mux.HandleFunc("/debug/pprof/", pprof.Index)
    mux.HandleFunc("/debug/pprof/cmdline", pprof.Cmdline)
    mux.HandleFunc("/debug/pprof/profile", pprof.Profile)
    mux.HandleFunc("/debug/pprof/symbol", pprof.Symbol)
    mux.HandleFunc("/debug/pprof/trace", pprof.Trace)
    log.Fatal(http.ListenAndServe(":8080", mux))
}
```

Profile-guided optimization

GA since Go v1.21

https://go.dev/blog/go1.21

Profiling is an incredible tool

but...

Polar Signals

The problem

Momentary

We can only explain the moment (if at all). Not the change.

Eg. rollout a new version, why is it faster/slower or using more/less resources?

Manual

We only start profiling after detecting a problem.

We don't have the profile of the time of detection or of the incident itself.

Homebrew workflows

"I SSH to a production machine and take a profile."

Not automated, not auditable, lots of room for mistakes.

Continuous Profiling

Google-Wide Profiling: A Continuous Profiling Infrastructure for Data Centers

Google-Wide Profiling (GWP), a continuous profiling infrastructure for data centers, provides performance insights for cloud applications. With negligible overhead, GWP provides stable, accurate profiles and a datacenter-scale tool for traditional performance analyses. Furthermore, GWP introduces novel applications of its profiles, such as application-platform affinity measurements and identification of platform-specific,

Why?

Development isn’t Production

Data and Context Overtime

When is continuous profiling useful?

Saving Money

Statistically significant insight into what code causes the most resources to be used, allows engineers to optimize those pieces and be confident, that resource usage will be lower after optimizing.

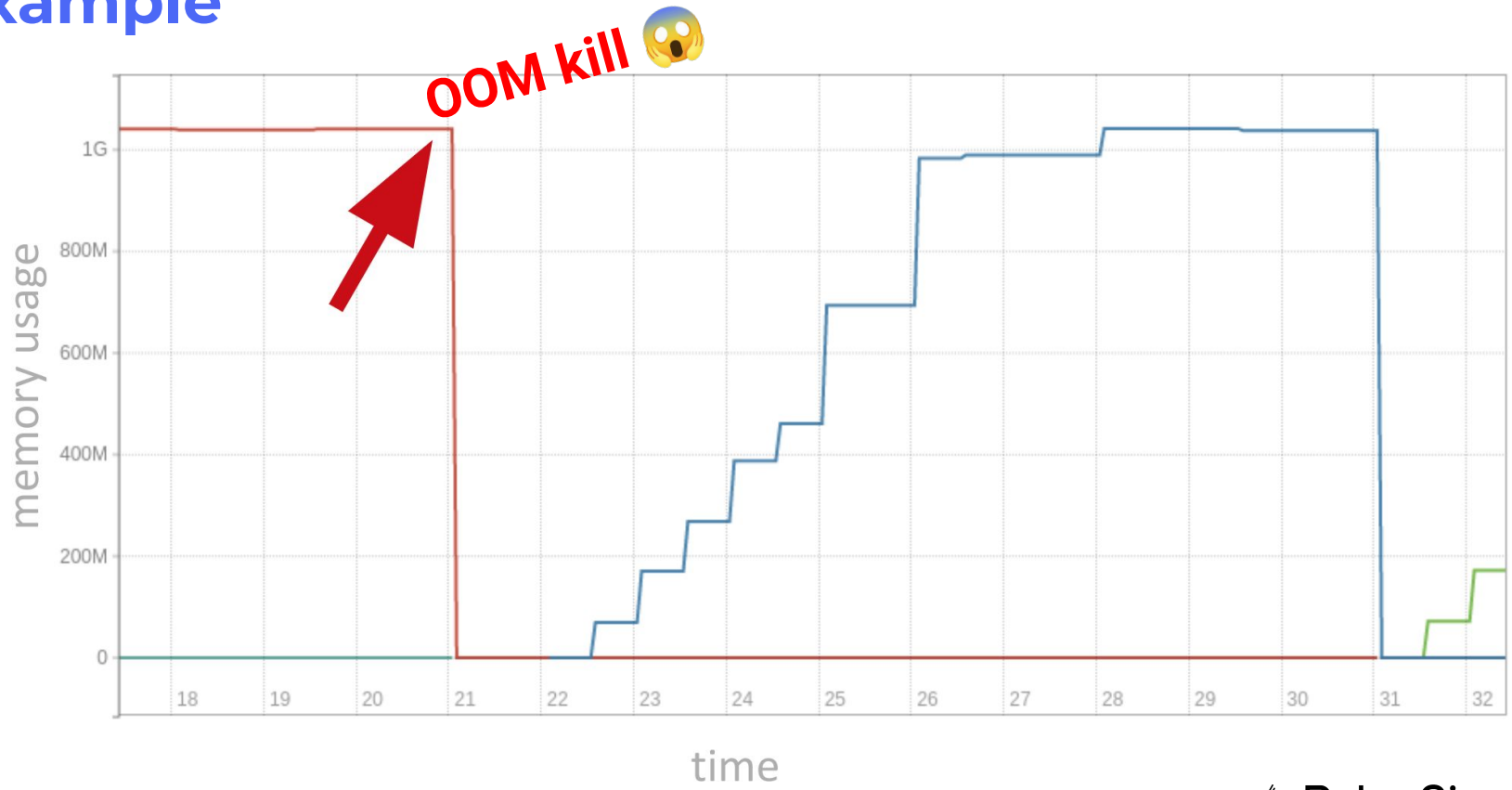
Understand difference

Always collecting data from all processes allows comparing why execution of code was different in time, across processes or even versions of code.

Understand incidents

Collecting data in the past allows us to understand incidents even after they have happened and without manual capturing of profiling data.

Example



Polar Signals

Continuous Profiling: How?

Sampling is cheap

Sampling profiling is cheap, so do it always, and store all the observed data.

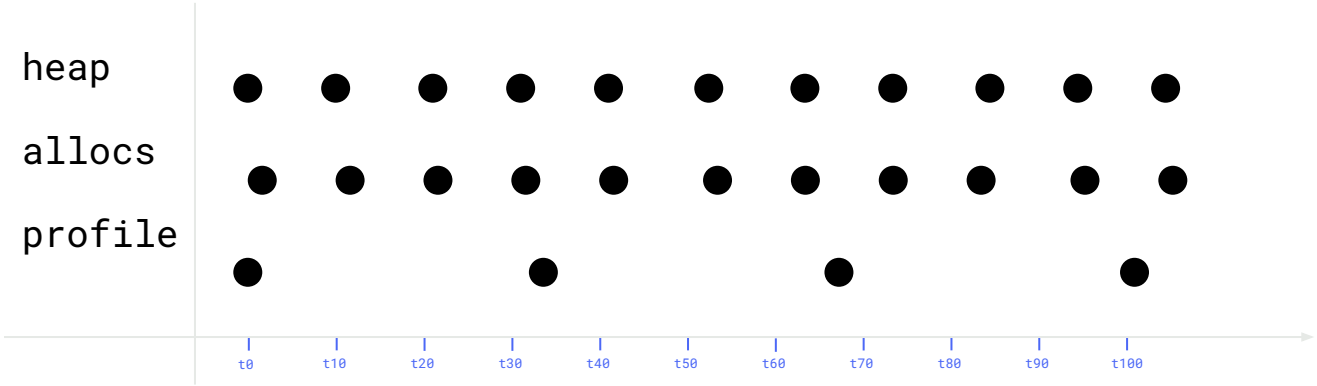
Index data

Index collected profiles over time and with workload metadata, so we can search for container's profiles over time.

Query

With new data collected over time, we unlock new workflows that were impossible before.

Continuous Profiling



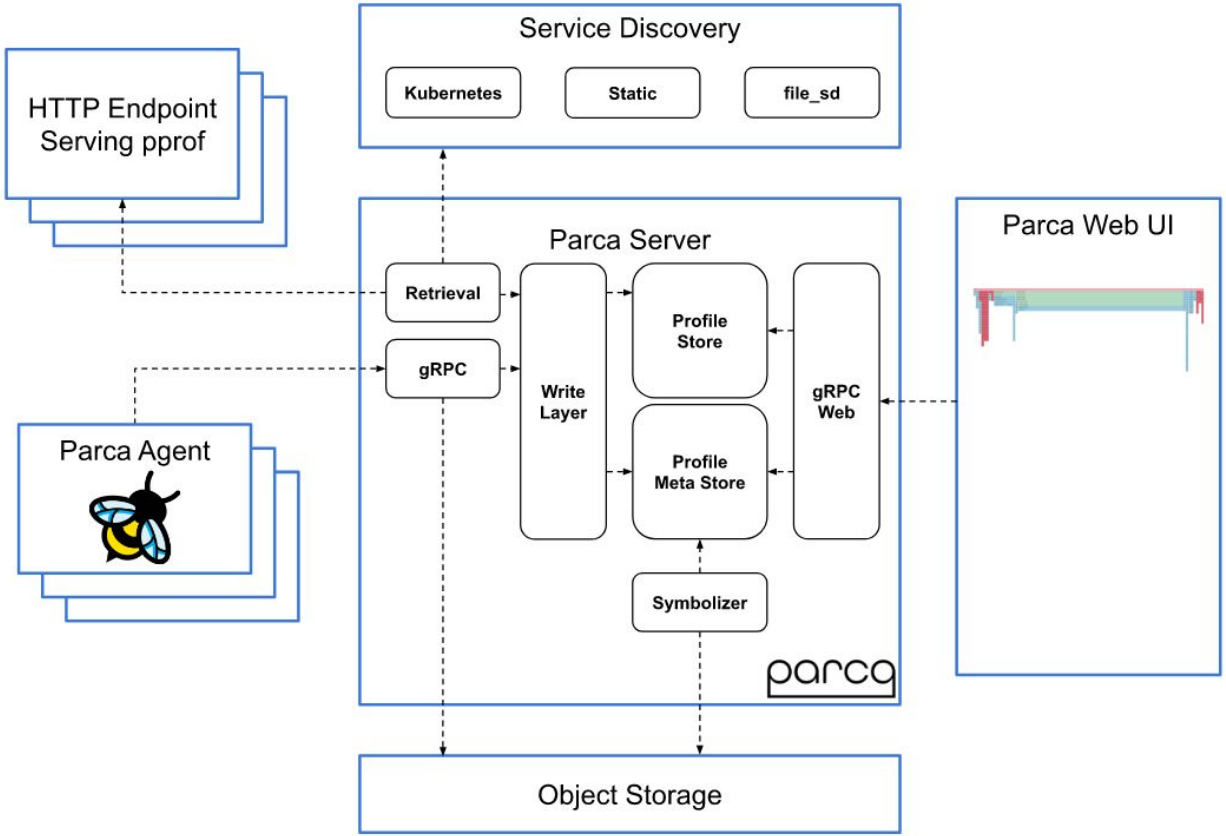
Polar Signals

Gist!

It’s possible to profile in production all the time!

Parca

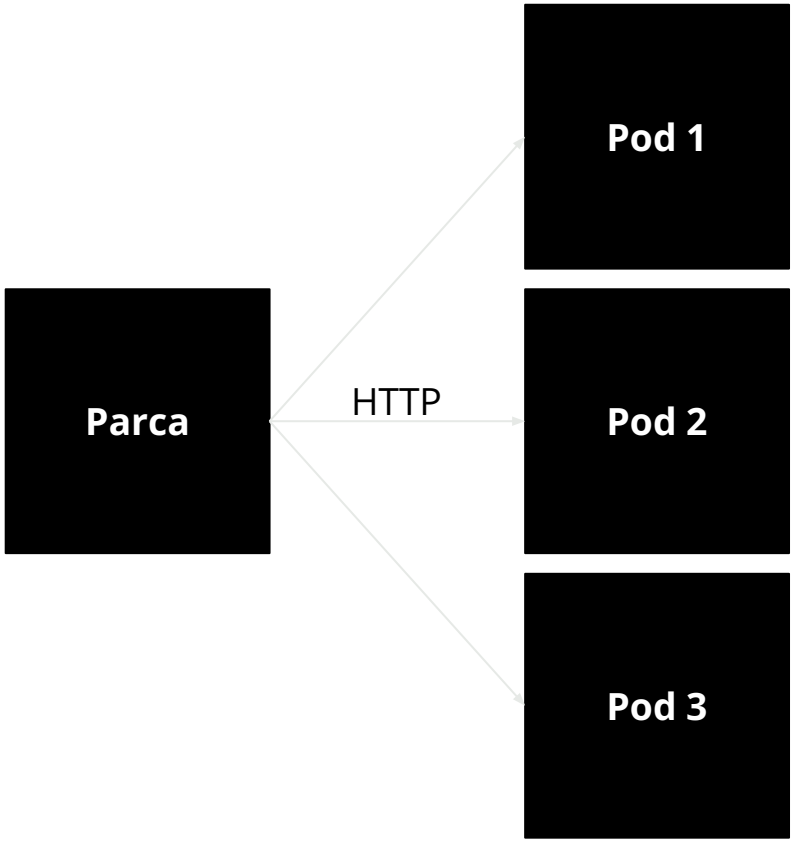
https://parca.dev/



Polar Signals

Parca

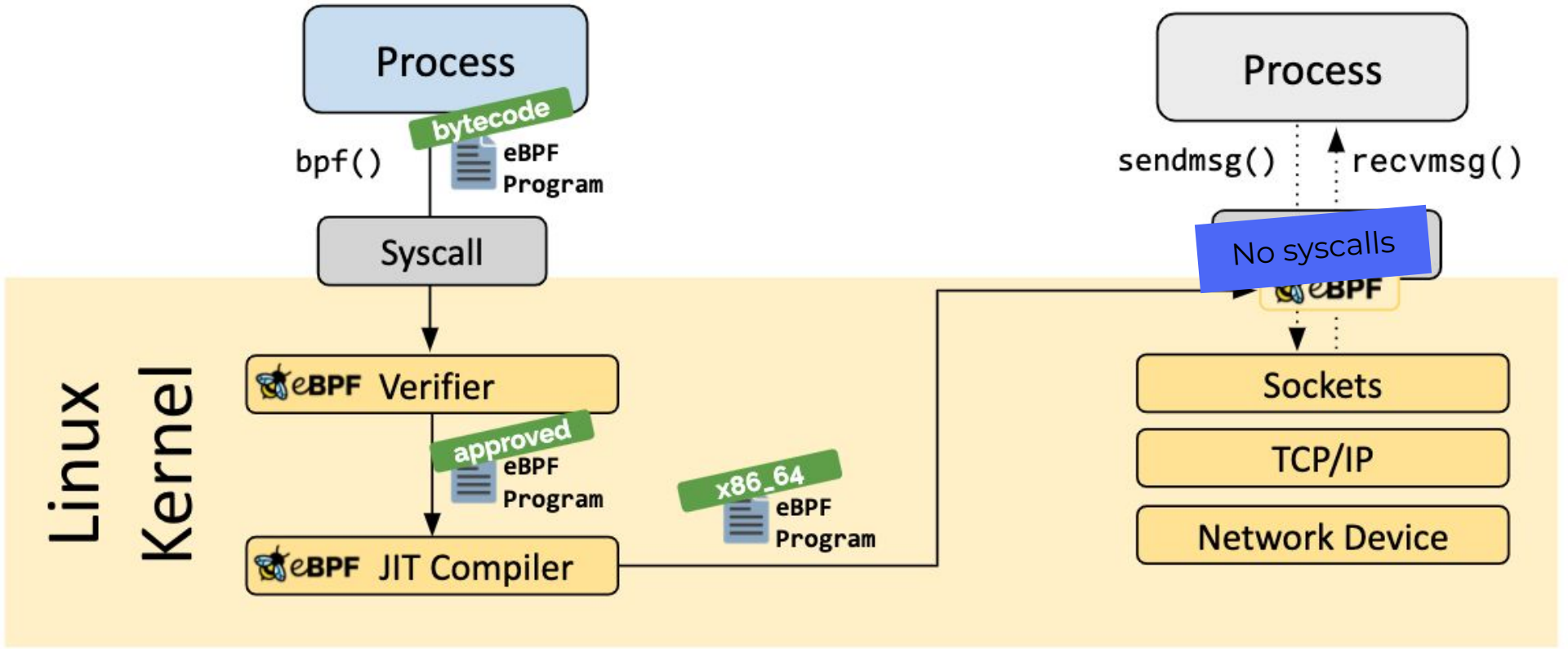
- Open Source: github.com/parca-dev/parca
  - Neutral governance/org
  - Contributions welcome!
- Inspired by Prometheus
  - Single statically linked binary
  - Multi-dimensional label model
  - Service discovery
  - Built-in storage



Polar Signals

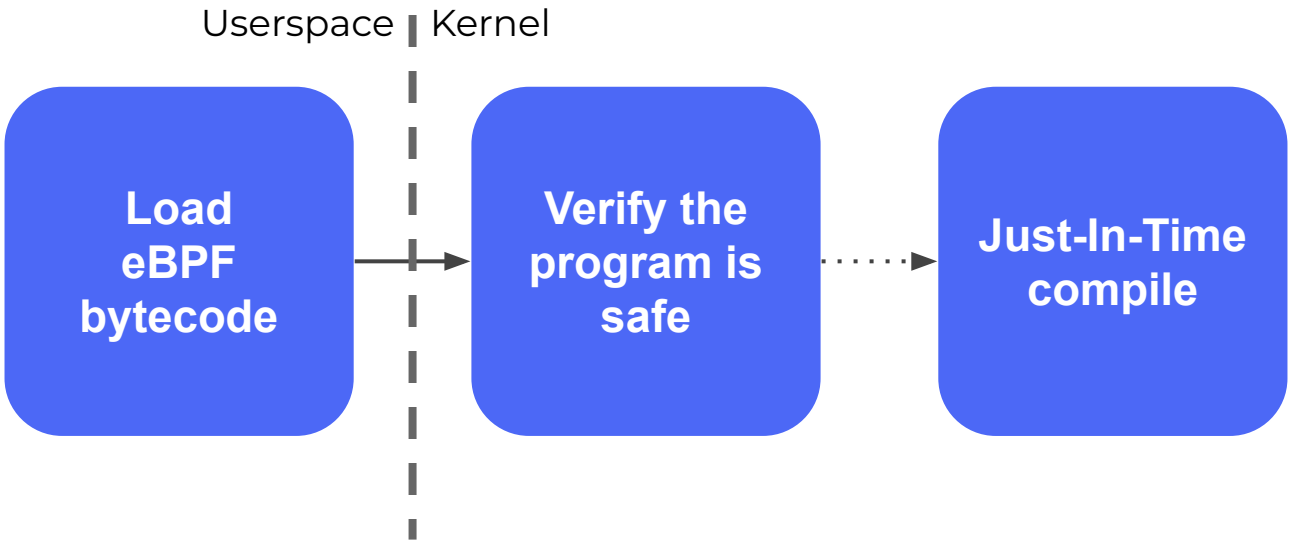
Parca Agent

eBPF 🐝

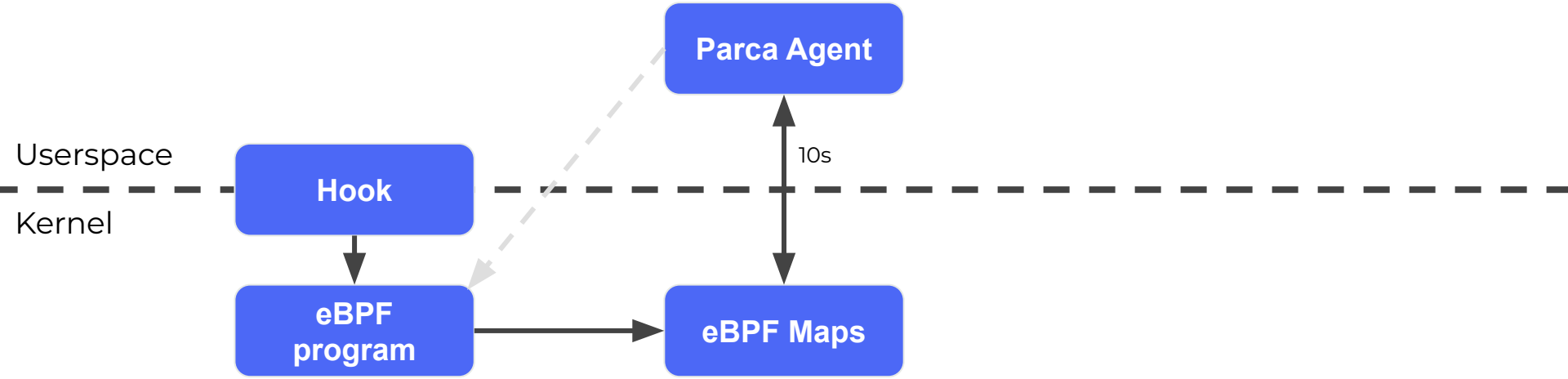


Polar Signals

Ensuring it’s safe to load



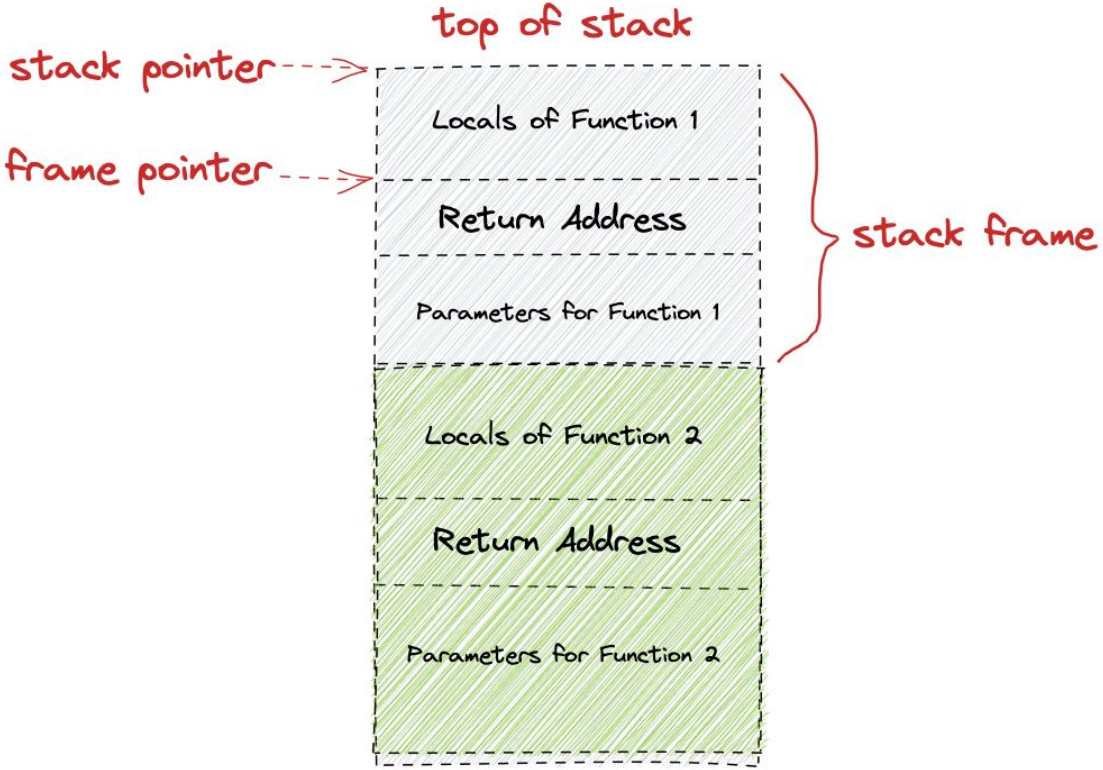
Communicating with Userspace



Hooks

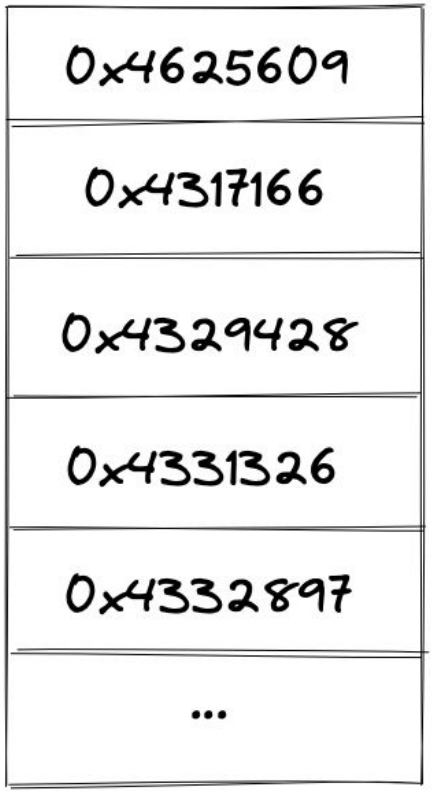
- Pre-defined (syscalls, kernel-functions, tracepoints, network events...)
- Custom hooks
  - kprobe
  - uprobe
  - **perf_event**

Stack Unwinding

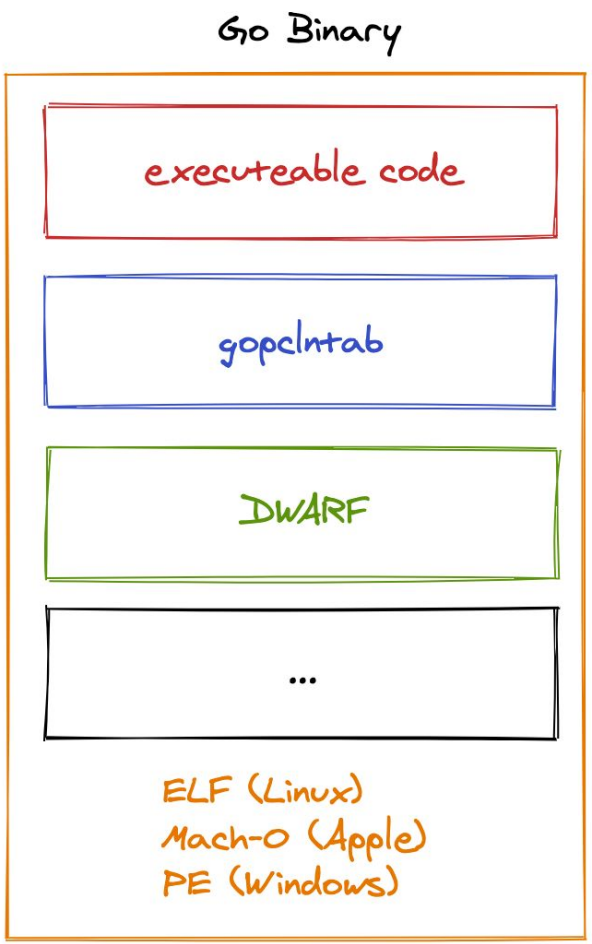


Polar Signals

Result of Stack Unwinding



Polar Signals

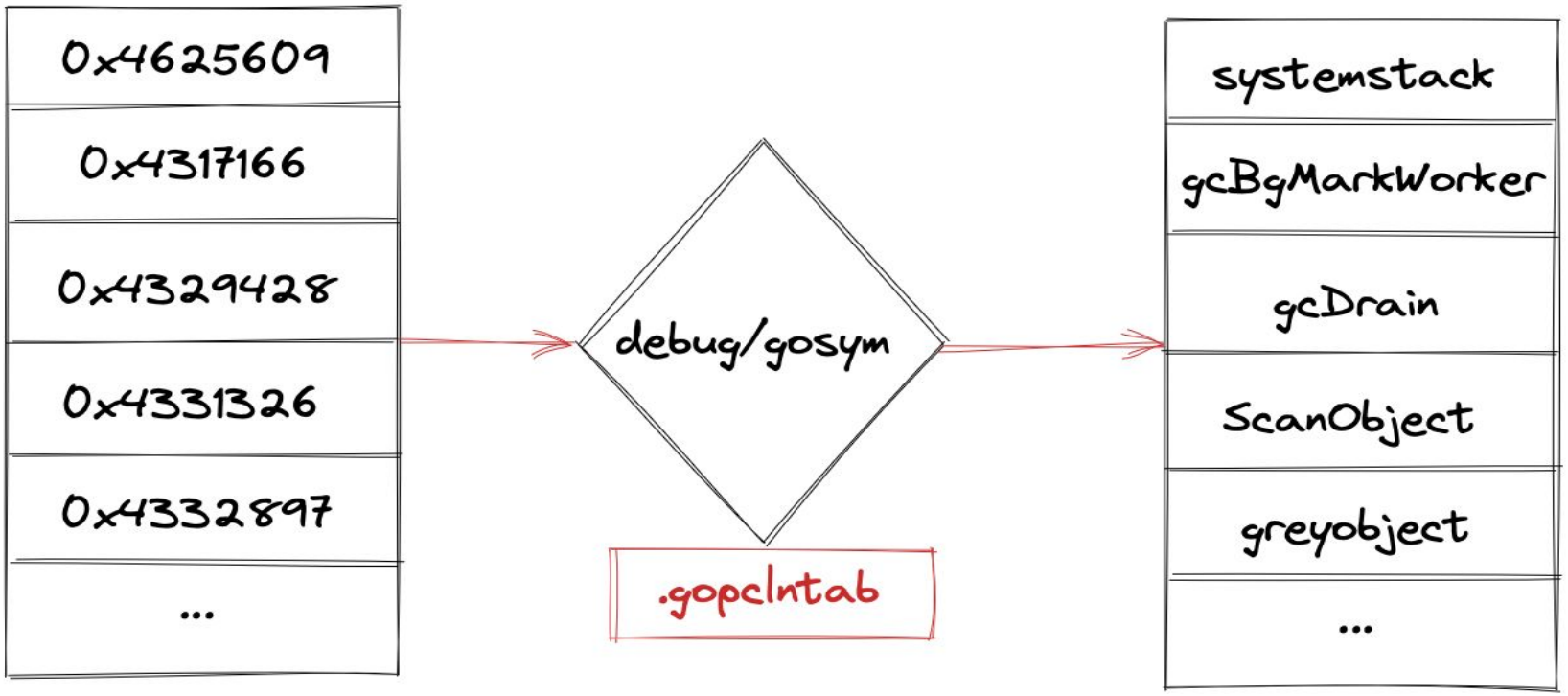


*https://github.com/DataDog/go-profiler-notes/blob/main/go-binary.png

Made with Excalidraw

Polar Signals

Symbolizing using debug/gosym



Easy to integrate

eBPF Profiler

A single profiler, using eBPF, profiles targets across the entire infrastructure with very low overhead.

Additionally, it automatically adds metadata for e.g. Kubernetes and systemd.

Polar Signals

SRE Concerns

Running privileged

Security

- Fewer dependencies and statically-linked binaries, CO:RE
- Reproducible builds
- Signed containers
  - https://www.parca.dev/docs/faq#how-can-i-verify-the-container-image-signature-with-cosign
- Automated PRs for our dependencies
- Only uploads symbols (no executable code ever send)

Performance

- Low-overhead (~1%)
  - 19 per second sampling (default)
  - Every 10s reads from kernel

Metadata

- Correlation through additional labels
  - Tracing via pprof labels

Zero Instrumentation!

Profiling Python and Ruby using eBPF

Dive into the internals of profiling interpreted Python and Ruby code using eBPF

Kemal Akkoyun & Javier Honduvilla Coto

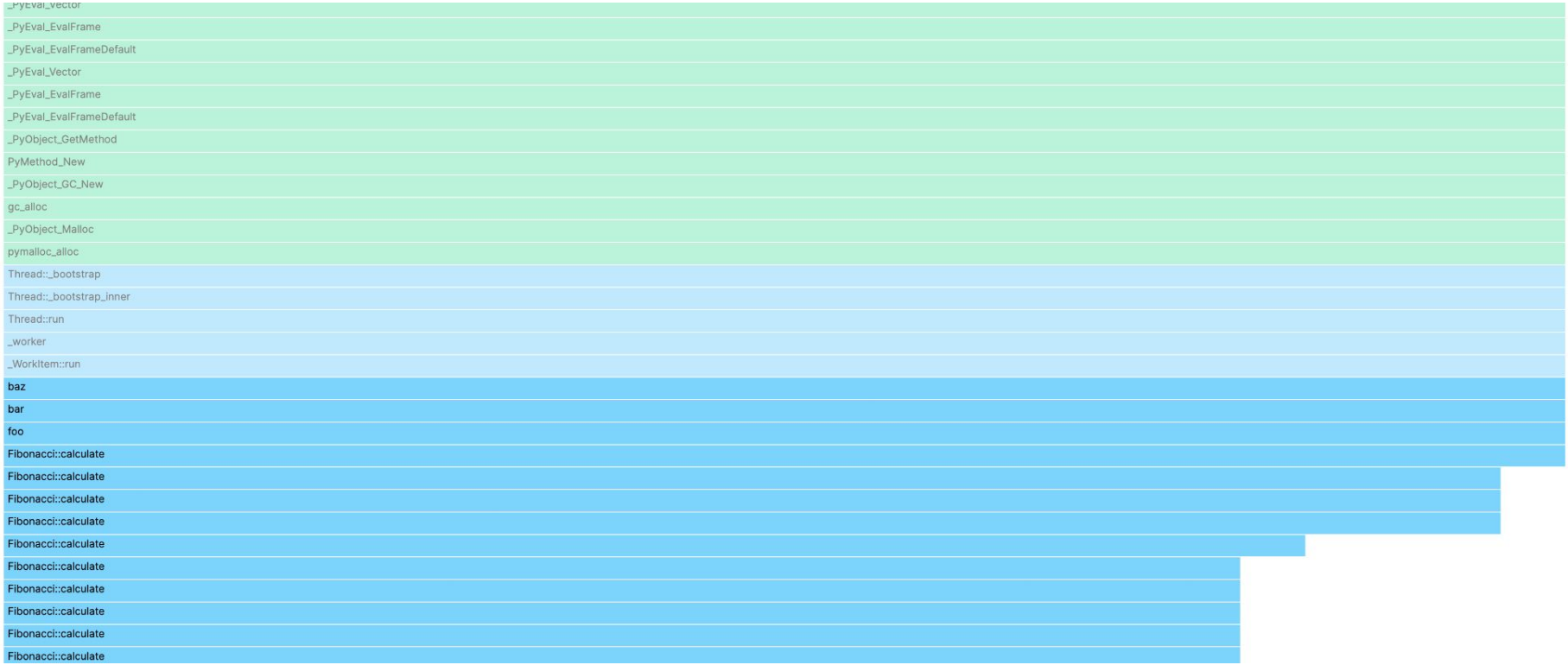
October 04, 2023

EBPF Interpreter Ruby Python

`tl;dr` In our continued efforts to expand and improve your profiling experience, we are excited to announce new additions to our language support: Ruby and Python. All the features that are described in this blog post have recently been released as part v0.26.0 of Parca Agent.

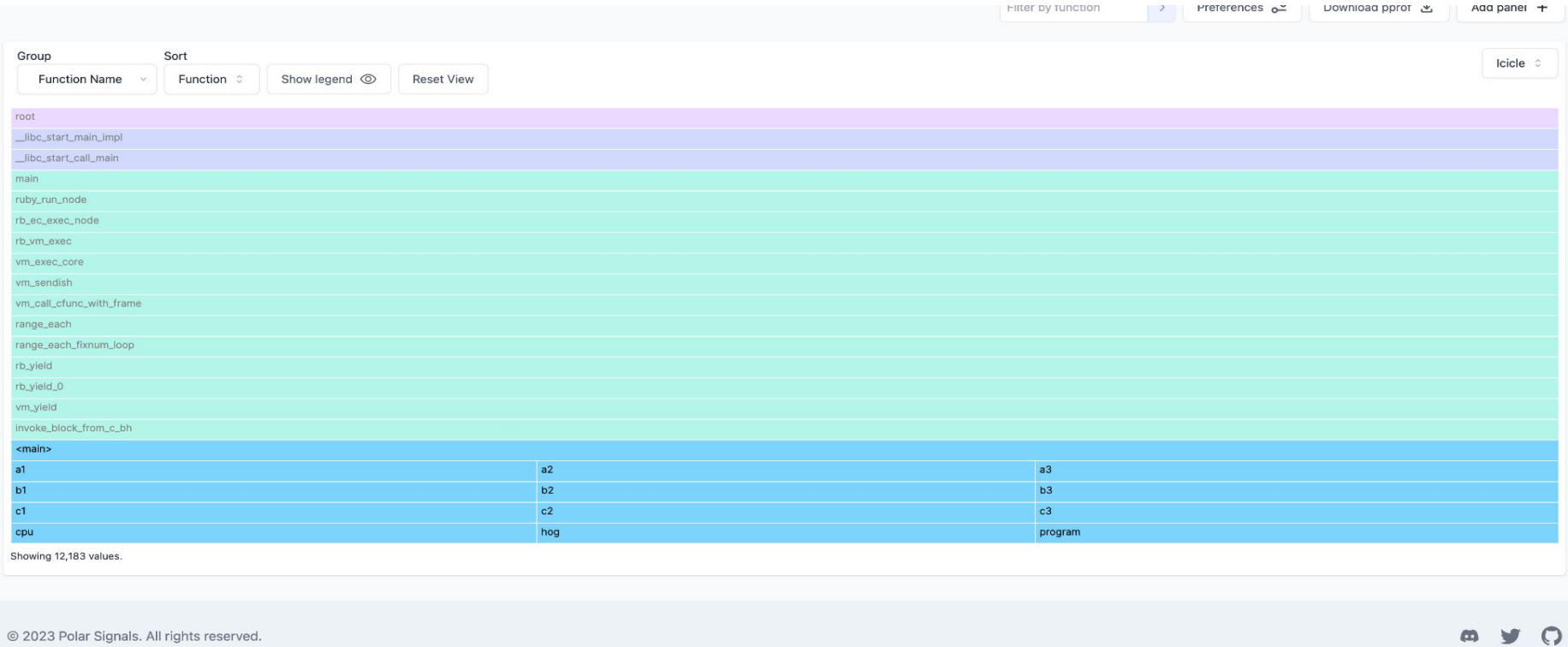
While currently these language supports are in beta, they can be enabled using the `--enable-ruby-unwinding` and `--enable-python-unwinding` flags when running the open-source Parca Agent. It's really easy to get started. You can use the following commands to run Parca and Parca Agent locally with Python

Python


_PyEval_Vector
_PyEval_EvalFrame
_PyEval_EvalFrameDefault
_PyEval_Vector
_PyEval_EvalFrame
_PyEval_EvalFrameDefault
_PyObject_GetMethod
PyMethod_New
_PyObject_GC_New
gc_alloc
_PyObject_Malloc
pymalloc_alloc
Thread::_bootstrap
Thread::_bootstrap_inner
Thread::run
_worker
_WorkItem::run
baz
bar
foo
Fibonacci::calculate
Fibonacci::calculate
Fibonacci::calculate
Fibonacci::calculate
Fibonacci::calculate
Fibonacci::calculate
Fibonacci::calculate
Fibonacci::calculate
Fibonacci::calculate
Fibonacci::calculate

(3.11)

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.
© 2023 Polar Signals. All rights reserved.

Parca Agent Language Support

Compiled Languages

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

With or without frame pointers!

JIT (Just in Time Compiled) Languages

- C#
- Erlang
- Java
- Julia
- NodeJS

Interpreted Languages

- Python
- Ruby

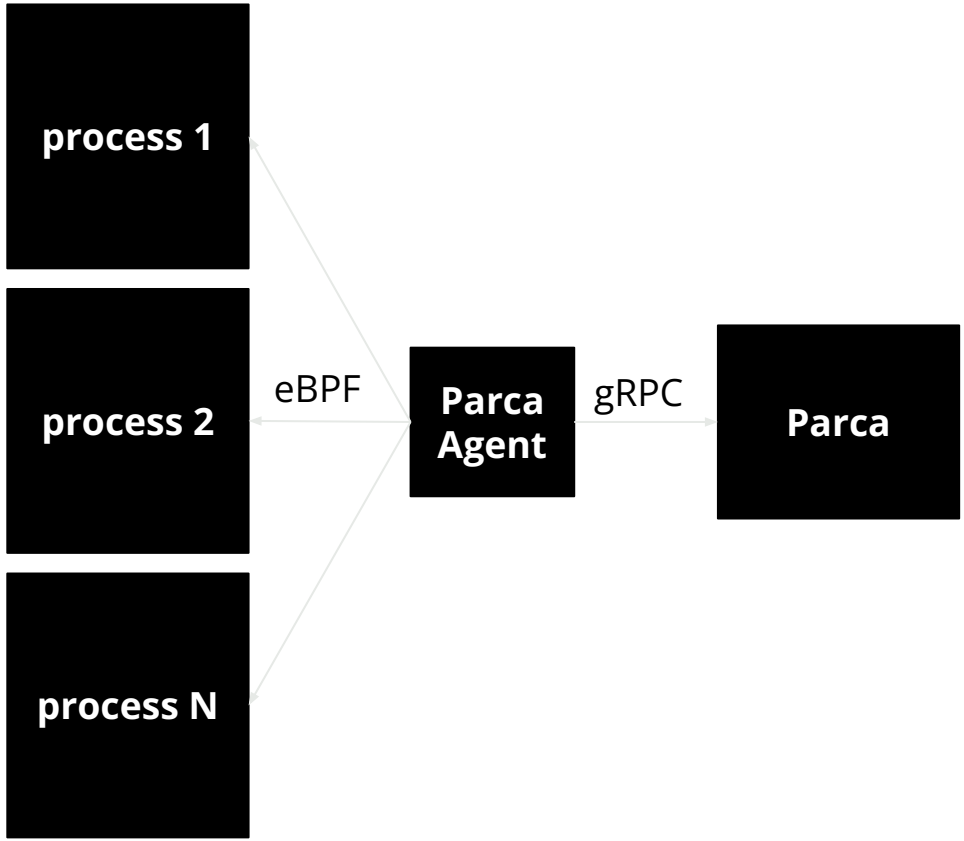
Parca Agent

- Open Source:

  github.com/parca-dev/parca-agent

- Enrich the metadata of the processes by associating them with their cgroups and corresponding metadata of the container runtimes
- Uses eBPF
- Understands where CPU resources are being spent
- Captures “current” stack trace X times per second to create statistical analysis off of

  No code changes required!



Polar Signals

Visualization

parca

Profiles | Targets | Help | Star

v0.17.0-05cd248d GitHub

Select profile... | Select a profile first to enter a filter... | Last 15 minutes | Search

Fgprof Samples Total
CPU profile samples observed regardless of their current On/Off CPU scheduling status

Fgprof Samples Time Total
CPU profile measured regardless of their current On/Off CPU scheduling status in nanoseconds

Goroutine Created Total
Stack traces that created all current goroutines.

Memory Allocated Objects Total
A sampling of all past memory allocations by objects.

Memory Allocated Bytes Total
A sampling of all past memory allocations in bytes.

Memory In-Use Objects
A sampling of memory allocations of live objects by objects.

Memory In-Use Bytes
A sampling of memory allocations of live objects by bytes.

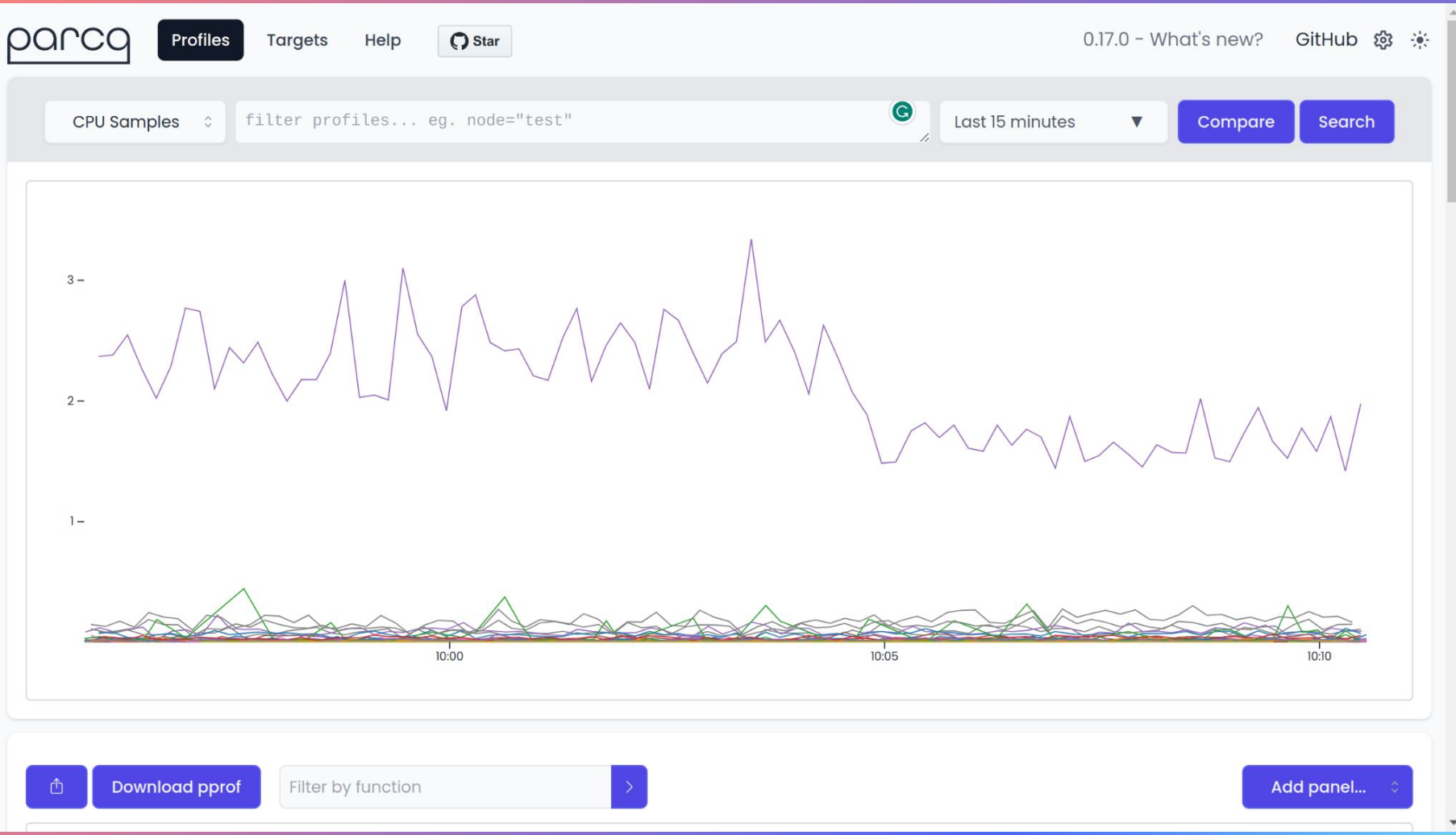
Mutex Contentions Total
Stack traces of holders of contended mutexes.

Mutex Contention Time Total
Time delayed stack traces caused by contended mutexes.

CPU Samples
CPU profile samples observed by Parca Agent.

Process CPU Nanoseconds

be displayed here.


parca
Profiles
Targets
Help
Star
0.17.0 - What's new?
GitHub
CPU Samples
filter profiles... eg. node="test"
Last 15 minutes
Compare
Search
3
2
1
10:00
10:05
10:10
Download pprof
Filter by function
Add panel...

Download pprof

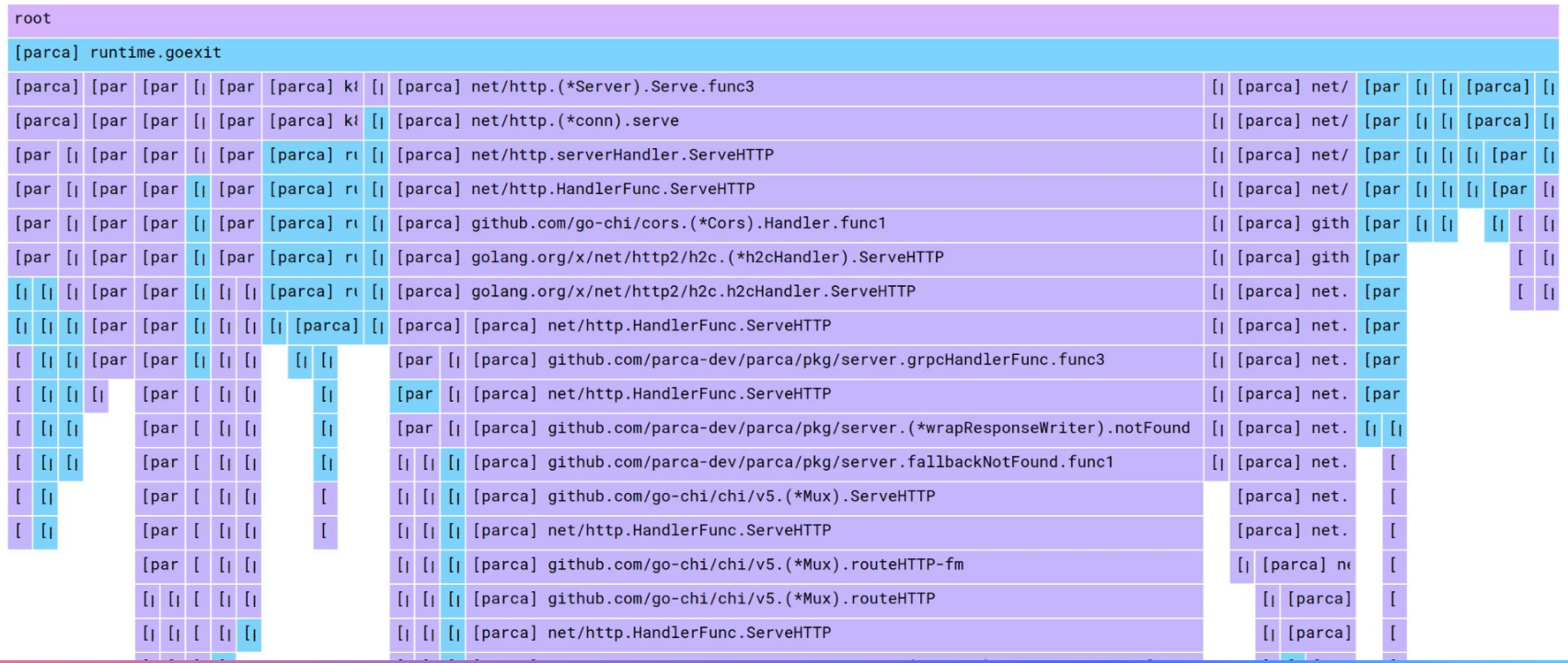
Filter by function

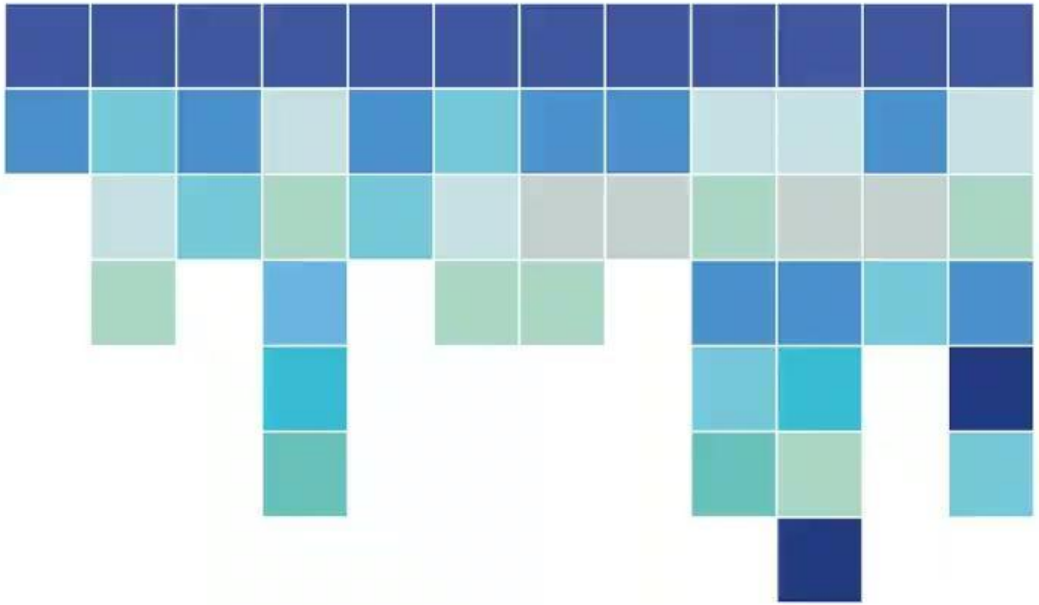
Add panel...

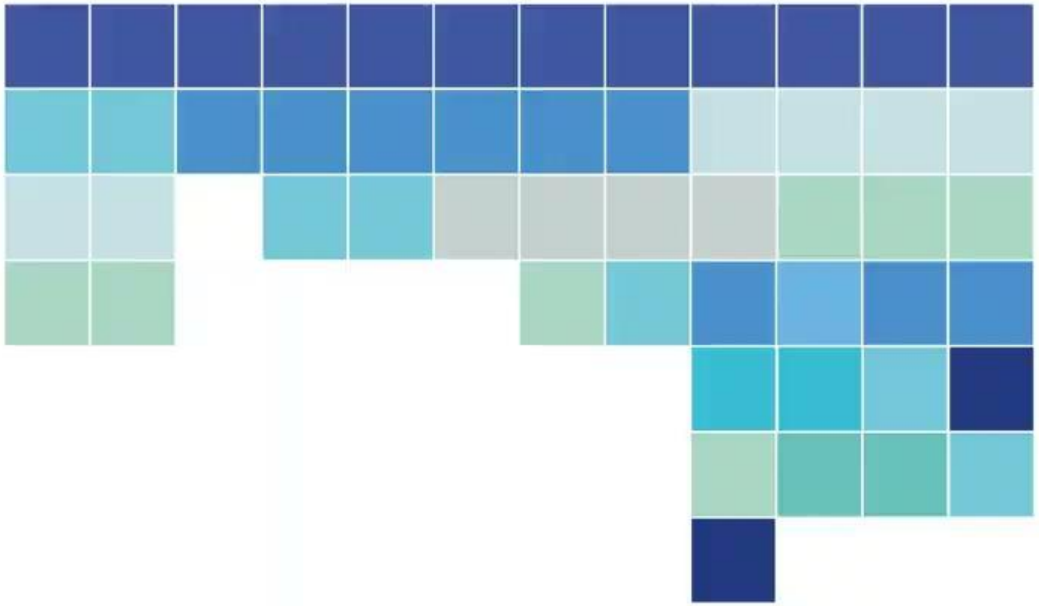
Reset View

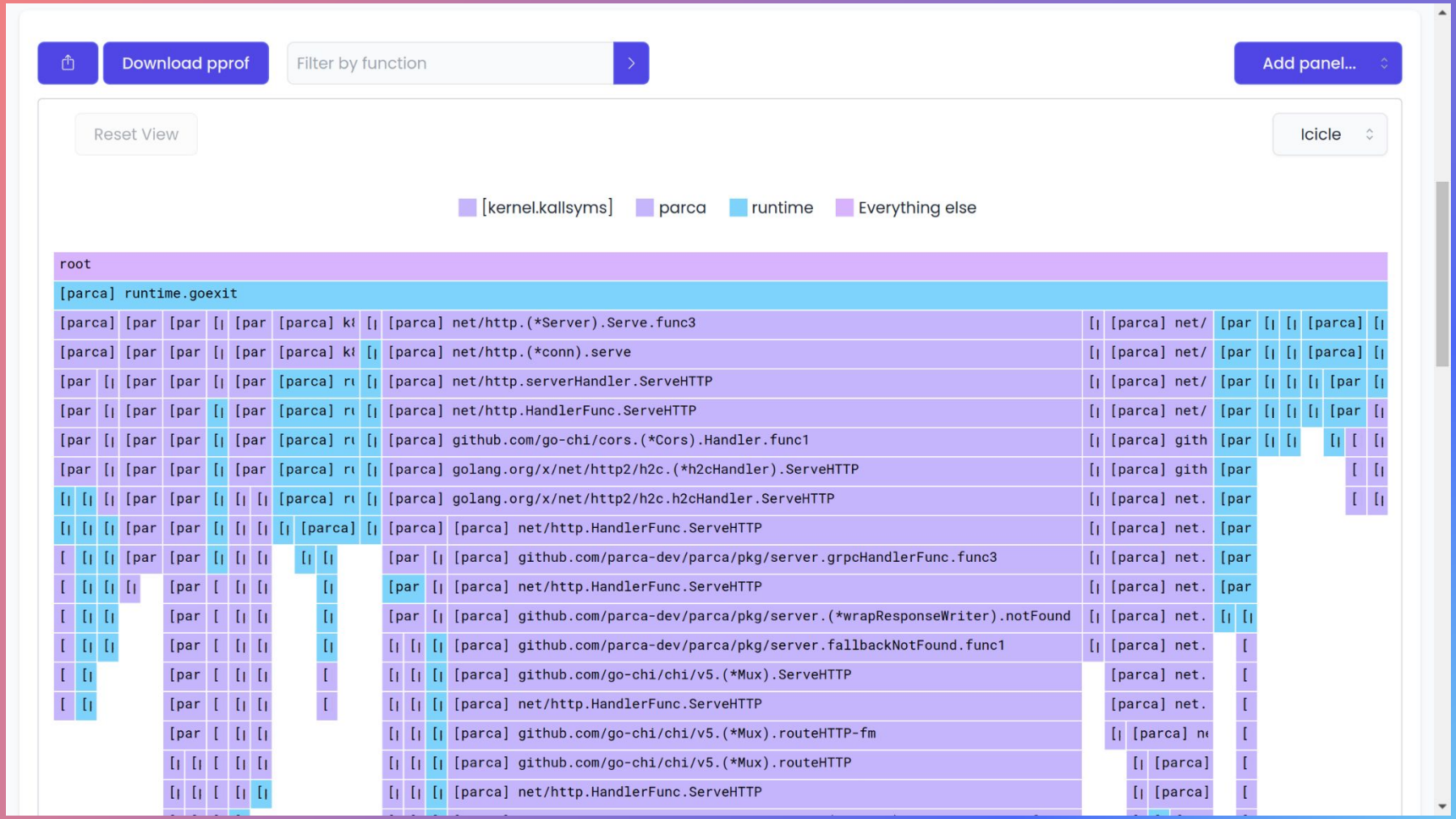
Icicle

[kernel.kallsyms] parca runtime Everything else



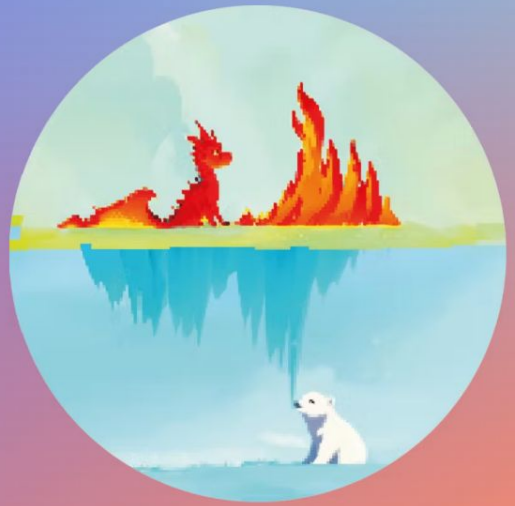





Download pprof
Filter by function
Add panel...
Reset View
Icicle
[kernel.kallsyms]
parca
runtime
Everything else
root
[parca] runtime.goexit
[parca] net/http.(*Server).Serve.func3
[parca] net/http.(*conn).serve
[parca] net/http.serverHandler.ServeHTTP
[parca] net/http.HandlerFunc.ServeHTTP
[parca] github.com/go-chi/cors.(*Cors).Handler.func1
[parca] golang.org/x/net/http2/h2c.(*h2cHandler).ServeHTTP
[parca] golang.org/x/net/http2/h2c.h2cHandler.ServeHTTP
[parca] net/http.HandlerFunc.ServeHTTP
[parca] github.com/parca-dev/parca/pkg/server.grpcHandlerFunc.func3
[parca] net/http.HandlerFunc.ServeHTTP
[parca] github.com/parca-dev/parca/pkg/server.(*wrapResponseWriter).notFound
[parca] github.com/parca-dev/parca/pkg/server.fallbackNotFound.func1
[parca] github.com/go-chi/chi/v5.(*Mux).ServeHTTP
[parca] net/http.HandlerFunc.ServeHTTP
[parca] github.com/go-chi/chi/v5.(*Mux).routeHTTP-fm
[parca] github.com/go-chi/chi/v5.(*Mux).routeHTTP
[parca] net/http.HandlerFunc.ServeHTTP

Ice and Fire: How to read icicle and flame graphs

Flame graphs and icicle graphs are a great way to visualize performance profiles. In this post, we will learn how to read and interpret them.

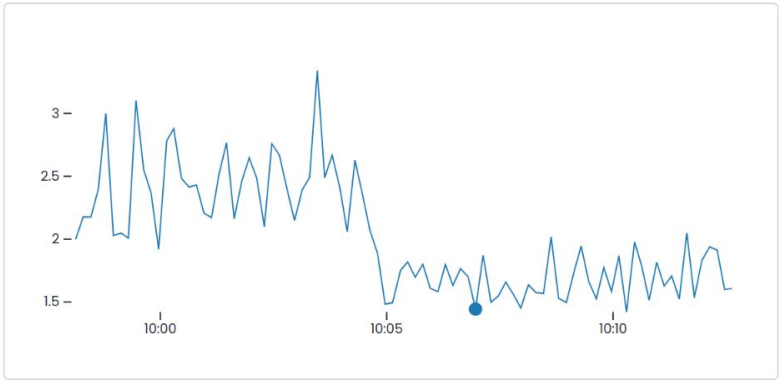


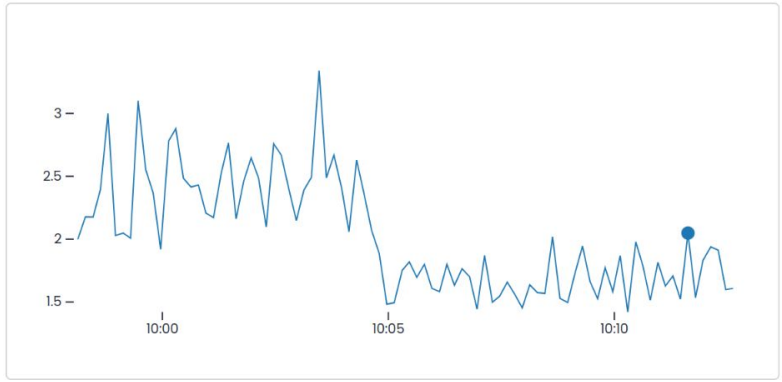
Kemal Akkoyun
March 28, 2023

Continuous Profiling
Profiling
Icicle Graphs
Flame Graphs

In this blog post, we'll dive into the world of performance profiling, flame graphs, and icicle graphs, exploring their impact on optimizing application performance.

But before we get started, I have a fun little tidbit to share with you about the inspiration behind the title of this post. To unravel the connection between the popular book series "A Song of Ice and Fire" and performance visualization, be sure to check out the final section of this post.





Download pprof

Filter by function

Add panel...

Reset View

Icicle

Better Worse

root

[parca] runtime.goexit

[parca] github.com/felixge/fgpro [p [parca] github.com/parca-dev/parca/pkg/scrape.(* [parca] github.com/polarsignals/frostdb. [[pa [parca] net/http.([pa

[parca] runtime.GoroutineProfil [p [parca] github.com/parca-dev/parca/pkg/scrape.(* [parca] github.com/polarsignals/frostdb. [[pa [parca] crypto/tls [pa

[parca] github.com/felixge/fgpr [p [parca] github.com/parca-dev/parca/pkg/profilest [parca] github.com/polarsignals/frostdb. [[pa [parca] crypto/tls [pa

[parca] runtime.goroutineProfil [p [parca] github.com/parca-dev/parca/pkg/parcacol. [parca] github.com/google/btree.(*BTreeG [pa [parca] crypto/tls [p

[parca] runtime.goroutineProfil [p [parca] [parca] github.com/parca-dev/parca/pkg/ [parca] github.com/google/btree.(*node[g [[pa [parca] cryp

[parca] runtime.forEachGRace [p [parca] [p [parca] github.com/polar: [parca] [parca] github.com/googl [parca] github. [p [parca] cry

[parca] runtime.goroutineProfil [p [parc [parca] github.com/polar: [parca] [parca] github.com/googl [parca] github. [p [parca] cr

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

Merged Profiles Of Query
from Aug 31, at 11:57:1 AM (UTC) to Aug 31, at 12:12:1 PM (UTC)

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

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

[kernel.kallsyms] main runtime Everything else

root
runtime.goexit
runtime.main
main.main
main.Fibonacci
math/big.(*Int).Add
math/big.addVV

main.Fibonacci

Cumulative	2.81k (97.47%)
File	...s/sources-example/main.go +16 open
Address	0x49c906
Binary	main
Build Id	714d324d5635707169702d74764b...
Labels	thread_id="0"

Hold shift and click on a value to copy.

Showing 0 values.

...rc/github.com/polarsignals/sources-example/main.go | Source

Line	Cumulative	Flat	Source
1			`package main`
2			
3			`import (`
4			`"log"`
5			`"math/big"`
6			`)`
7			
8			`func Fibonacci(n uint) *big.Int {`
9			`if n <= 1 {`
10			`return big.NewInt(int64(n))`
11			`}`
12			
13			`var n2, n1 = big.NewInt(0), big.NewInt(1)`
14			
15			`for i := uint(1); i < n; i++ {`
16	2810		`n2.Add(n2, n1)`
17			`n1, n2 = n2, n1`
18			`}`
19			
20			`return n1`
21			`}`
22			
23			`func main() {`
24			`i := uint(1000000)`
25			`for {`
26	2842		`log.Println("fibonacci number", i, Fibonacci(i))`
27			`i++`
28			`}`
29			`}`
30			

Demo

demo.parca.dev

Polar Signals

@PolarSignalsIO 136 subscribers 19 videos

More about this channel >

Subscribed

HOME VIDEOS LIVE PLAYLISTS COMMUNITY CHANNELS ABOUT

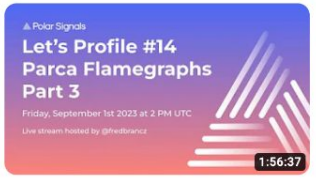
Latest Popular Oldest

Let's Profile Kubernetes Kubelet Part 2 - Ep16

59 views • Streamed 4 days ago

Let's Profile Kubernetes Kubelet - Ep15

110 views • Streamed 3 weeks ago



Let's Profile Parca Flamegraphs Part 3 - Ep14

47 views • Streamed 1 month ago

Let's Profile Parca Flamegraphs Part 2 - Ep13

50 views • Streamed 1 month ago

Let's Profile Parca Flamegraphs (Episode 12)

90 views • Streamed 2 months ago

Let's Profile #11 - xz compression

97 views • Streamed 2 months ago

Let's Profile: FluxCD! (Episode 10)

75 views • Streamed 3 months ago

Let's Profile: Kubernetes Metrics Server! (Episode 9)

117 views • Streamed 3 months ago

Storage

We needed something better

Parca's DB is written from scratch

Polar Signals

Parca's Storage

- Still inspired by Prometheus
- Separate meta data storage
- Handles stack traces in the storage
- FrostDB - Embeddable column database written in Go.
  - Column Database
  - First Class Wide-Columns
  - Apache Arrow
  - Apache Parquet

FrostDB

labels.pod	labels.node	stacktrace	timestamp	value
mypod1	mynode1	main;func1;func2	t1	2
mypod1	mynode1	main;func1;func2	t4	3
mypod1	mynode1	main;func1;func3	t1	23
mypod1	mynode1	main;func1;func3	t2	10
mypod1	mynode1	main;func1;func3	t3	12
mypod1	mynode1	main;func1;func3	t5	234

FrostDB

labels.pod	labels.node	stacktrace	timestamp	value
6x mypod1	6x mynode1	2x main;func1;func2	t1	2
			t4	3
		4x main;func1;func3	t1	23
			t2	10
			t3	12
			t5	234

FrostDB

labels.pod	labels.node	stacktrace	timestamp	value
6x mypod1	6x mynode1	2x main;func1;func2	t1	2
			t4	3
		4x main;func1;func3	t1	23
			t2	10
			t3	12
			t5	234

FrostDB - Merging (SIMD)

labels.pod	labels.node	stacktrace	timestamp	value
6x mypod1	6x mynode1	2x main;func1;func2	t1-t4	5
		4x main;func1;func3	t1-t5	279

Querying?

Time

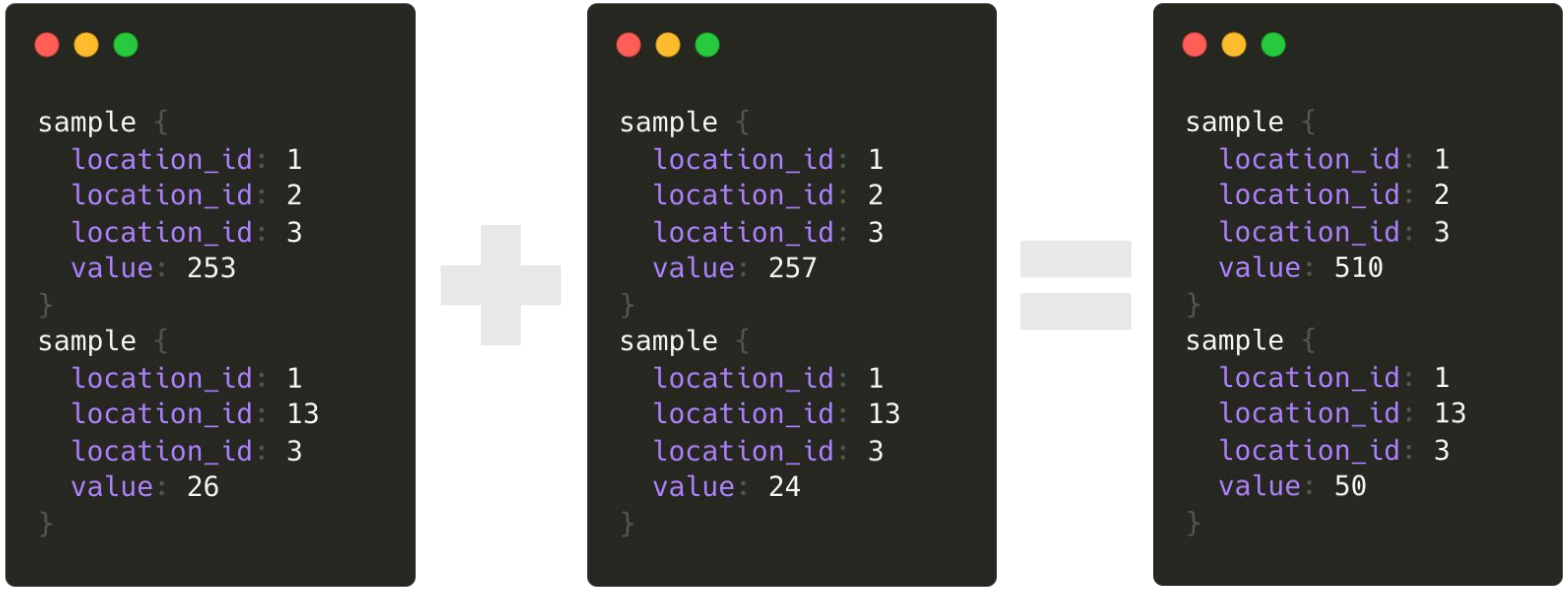
1631263917

Query

```
cpu:samples{job="parca",instance="localhost:7070"}
```

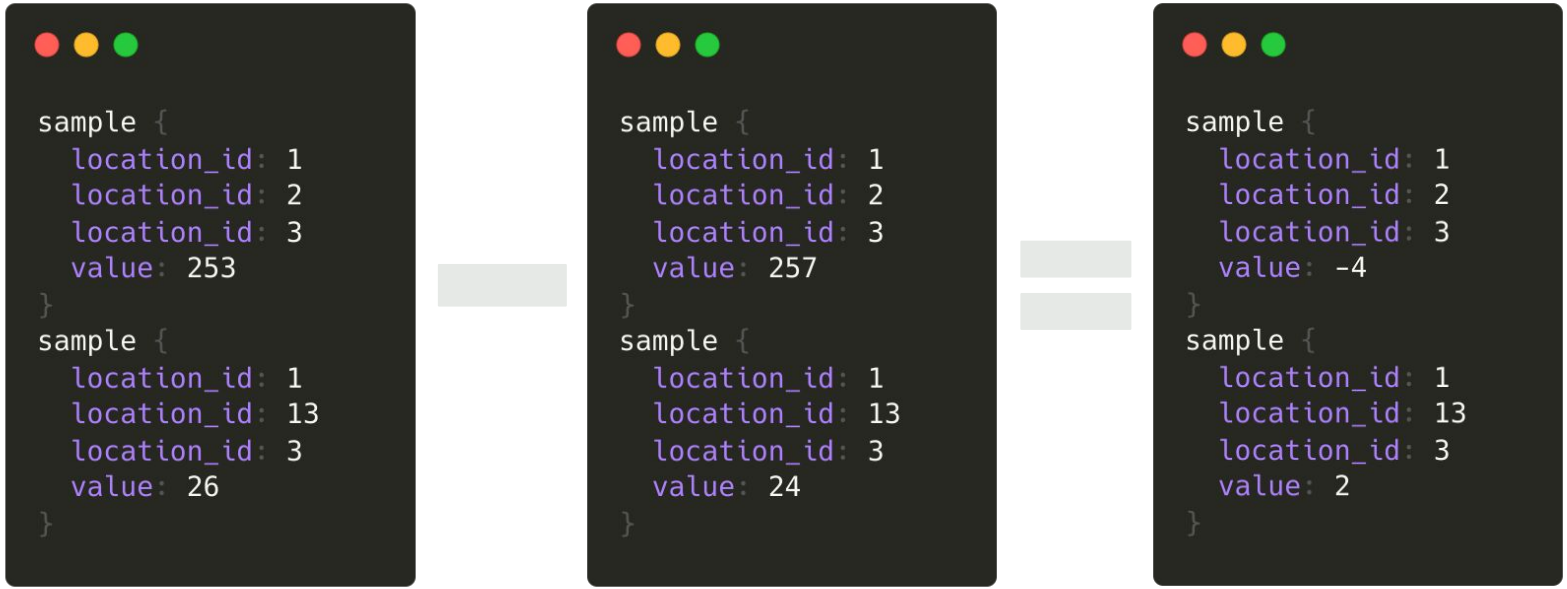

Polar Signals

Combine/Merge Profiles



Polar Signals

Compare/Diff Profiles



Polar Signals

GA

Parca's Roadmap

- Persistence on disk
- Querying parts of stack traces only
- Improve language and runtime support
- Add additional profiles (heap, alloc, i/o ...)

Most importantly:
Build a community with **YOU**!

We invite you to

- Join the Parca Discord
- Attend the Parca Office Hours

Resources

- 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://www.polarsignals.com/blog/posts/2022/01/27/fantastic-symbols-and-where-to-find-them-part-2/
- https://github.com/DataDog/go-profiler-notes
- https://www.infoq.com/presentations/cotinuous-profiling-production
- https://github.com/davecheney/presentations/blob/master/seven.slide
- https://parca.dev

Polar Signals

Thank you for listening!

Matthias Loibl

@metalmatze