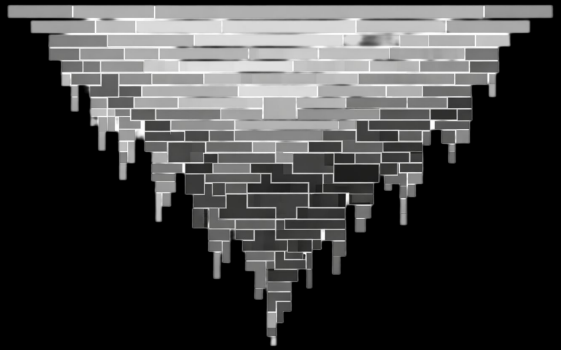
groundcover

Profiling



groundcover

Who

Noam Levy

Field CTO

groundcover

groundcover

What

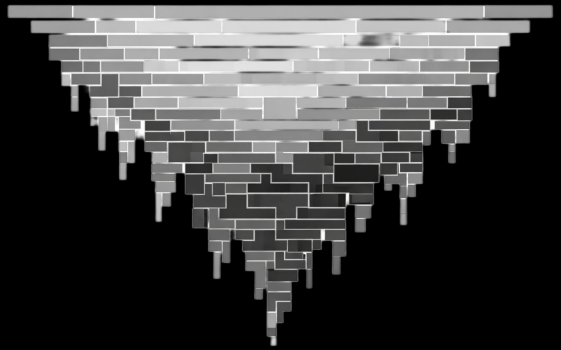
Profiling is
powerful

It's (Becoming)
accessible

And soon enough,
~frictionless

groundcover

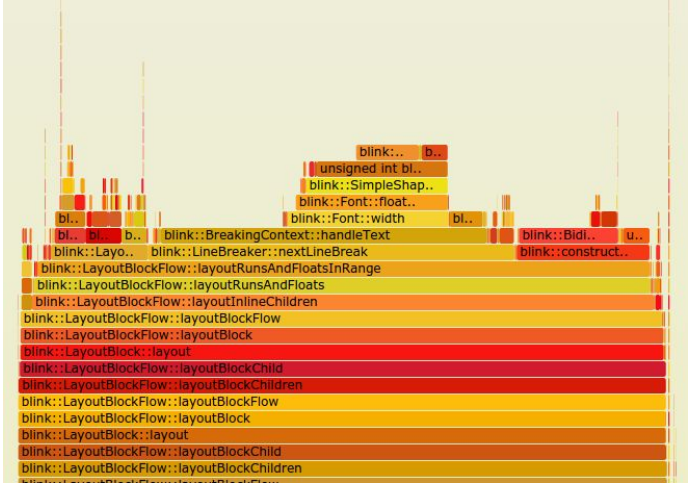
Powerful



groundcover

101

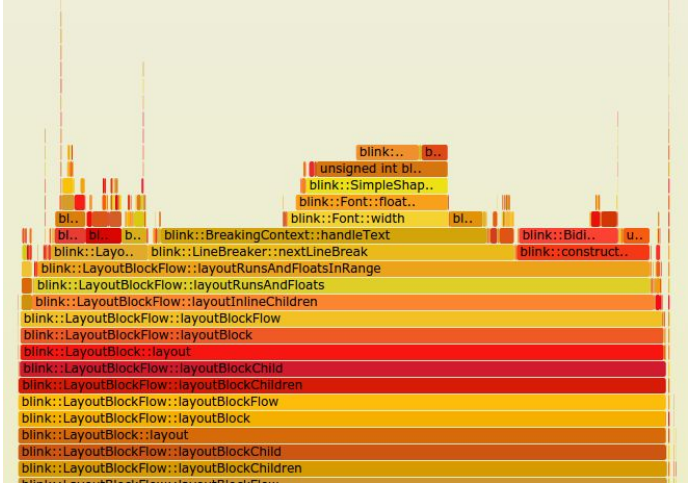
- Resources Allocation Behavior Tracking
- CPU or Memory
- Continuous/Ad Hoc
- Down to the function



groundcover

Why

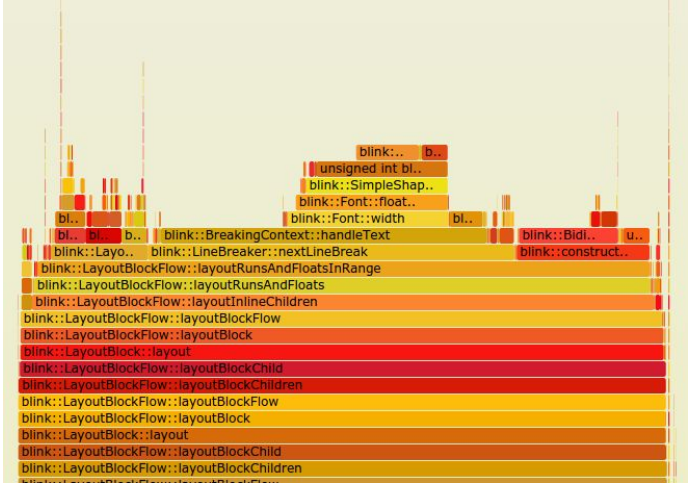
- “Where CPU time is spent”
- “Where do allocations occur”
- “Oh its the GC” (it’s very rarely is)



groundcover

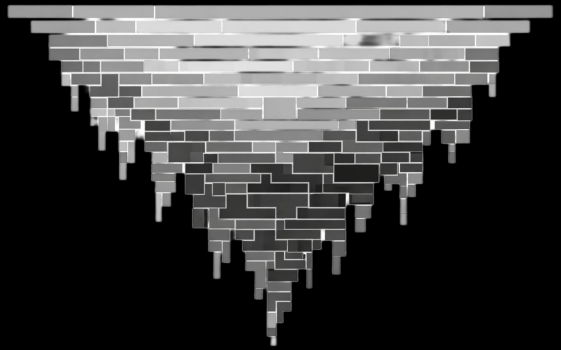
Words

- Flame graphs
- Winding/Walking
- Symbolization
- **eBPF**



groundcover

Accessible



groundcover

Accessibility

- How easy it is to **implement?**
- How easy it is to **consume?**

Implement

- Profiling was a fragmented field
- Runtime sensitive
- Usually operated separately
- Binaries/OS were not always compatible
- **Isolated pillar**

parca

elastic

Grafana Pyroscope

groundcover

Implement

- Framework for standardizing telemetry delivery
- Spec and Collector
- Metrics, Logs, Traces
- **Profiling signal released**

groundcover

Implement

- Portable / Vendor neutral
- Popular, **referable**
- Multiple runtimes
- Consistent

groundcover

Implement

- Receiver
- Processor
- Exporter

```yaml
receivers:
  otlp:
    protocols:
      grpc:
        endpoint: 0.0.0.0:4317

processors:
  batch: {}

exporters:
  otlp/backend:
    endpoint: <>

service:
  pipelines:
    traces:
      receivers: [otlp]
      processors: [batch]
      exporters: [otlp/backend]
```

groundcover

Implement

- SDK (instrumentation)
- *not always needed

```
func (c *clickHouseHttpClient) Do(ctx context.Context, req *http.Request) (*http.Response, []byte, error) {
    ctx, span := serverTracer.Start(ctx, clickhouseServerDoSpan)
    defer span.End()
```

groundcover

Consume

- Profiles also contain information about **the logic peripherals**
- Its is hard to understand
- Only few (**humans**) could make decisions based of it.

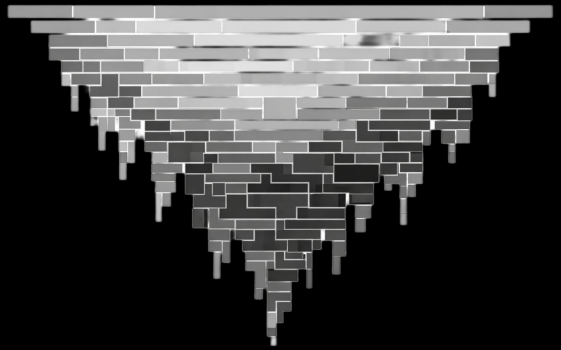
groundcover

Consume

- If Agents can read profiles
- Agents can take you straight to conclusions
- **The bar for benefiting from profiling has been lowered**

groundcover

Friction(less?)



eBPF

- Kernel sandbox for loading programs with **access to the entire os and running apps**
- Allows to **hook on** kernel and user programs at entry/exit/custom places.
- *”This is like putting Javascript into the kernel.”* - Brendan Gregg

groundcover

eBPF 101

- We hook
- We work (safely)
- We leave

```c
SEC("kprobe/do_unlinkat")
int BPF_KPROBE(do_unlinkat, int dfd, struct filename *name)
{
    pid_t pid;
    const char *filename;

    pid = bpf_get_current_pid_tgid() >> 32;
    filename = BPF_CORE_READ(name, name);
    bpf_printk("KPROBE ENTRY pid = %d, filename = %s\n", pid, filename);
    return 0;
}

SEC("kretprobe/do_unlinkat")
int BPF_KRETPROBE(do_unlinkat_exit, long ret)
{
    pid_t pid;

    pid = bpf_get_current_pid_tgid() >> 32;
    bpf_printk("KPROBE EXIT: pid = %d, ret = %ld\n", pid, ret);
    return 0;
}
```

groundcover

eBPF+011Y

- eBPF unblocks **zero instrumentation techniques**
- We can automate signal generation **without relying on engineers**
- Used for **generating** traces, metrics
- And now - **profiling**

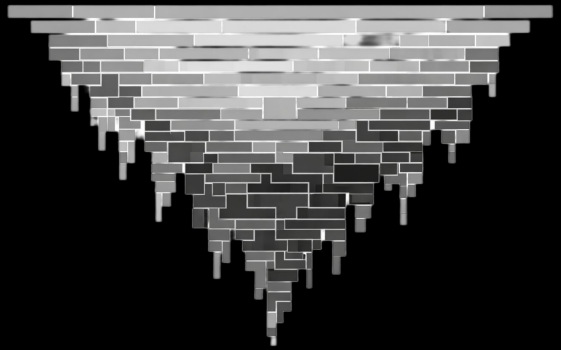
groundcover

eBPF+Profiling

- Opentelemetry-ebpf-profiler
- Wide support, including (kinda) stripped binaries
- **No need for instrumentation**

groundcover

Getting Started



groundcover

Getting started

- eBPF agent
- OpenTelemetry Collector
- Profiling backend
- *K8s

groundcover

Profiling Agent

- eBPF agent is **privileged**
- hostPID: true
- securityContext.privileged = true
- /sys, /proc, /lib volume mounts

```yaml
spec:
  hostPID: true
  containers:
    - name: profiler
      image: otel/opentelemetry-collector-ebpf-profiler:latest
      args: …
      env: …
      resources: …
      securityContext:
        runAsUser: 0
        runAsGroup: 0
        privileged: true
      volumeMounts:
        - name: config
          mountPath: /etc/otel/config.yaml
          subPath: config.yaml
        - name: sys-kernel
          mountPath: /sys/kernel
          readOnly: true
        - name: tracefs
          mountPath: /sys/kernel/tracing
          readOnly: true
        - name: lib-modules
          mountPath: /lib/modules
          readOnly: true
        - name: proc
          mountPath: /proc
          readOnly: true
```

groundcover

Collector

- Receiver
  - **OTLP**
- Processors
  - **Metadata enriching**
- Exporter
  - **Profiling backend**

```
receivers:
  otlp:
    protocols:
      grpc:
        endpoint: 0.0.0.0:4317

processors:
  k8sattributes:
    auth_type: "serviceAccount"
    passthrough: false
    extract:⋯
    pod_association:
      - sources:
          - from: resource_attribute
            name: container.id
exporters:
  otlp_http/pyroscope:
    endpoint: http://pyroscope.profiling-lab.svc.cluster.local:4040

service:
  pipelines:
    profiles:
      receivers: [otlp]
      processors: [k8sattributes]
      exporters: [otlp_http/pyroscope]
```

groundcover

Backend

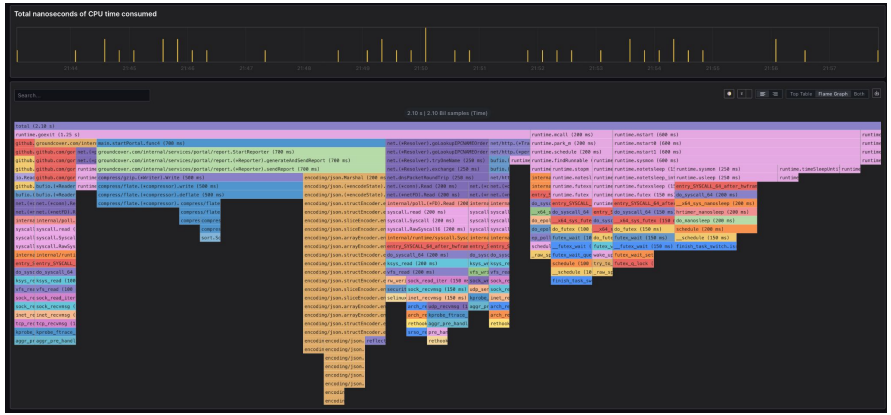
- Needs storage
- StatefulSet

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: pyroscope
  namespace: profiling-lab
spec:
  serviceName: pyroscope
  replicas: 1
  selector: ⋯
  template:
    metadata: ⋯
    spec:
      containers:
        - name: pyroscope
          image: grafana/pyroscope:latest
          args: ⋯
          ports: ⋯
          volumeMounts: ⋯
  volumeClaimTemplates:
```

groundcover

How it looks like

- Per service profile
- Complete metadata
- **No need for code changes**



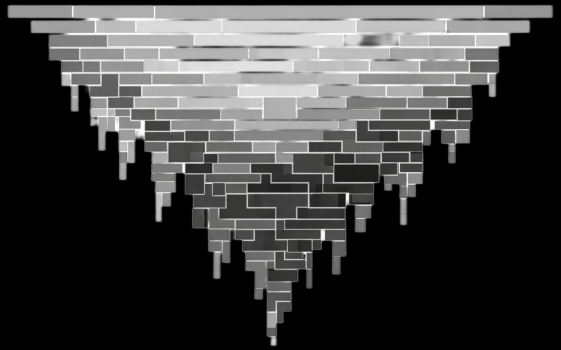
groundcover

Not Perfect

- Still early
- Symbolization
- Making sense out of it

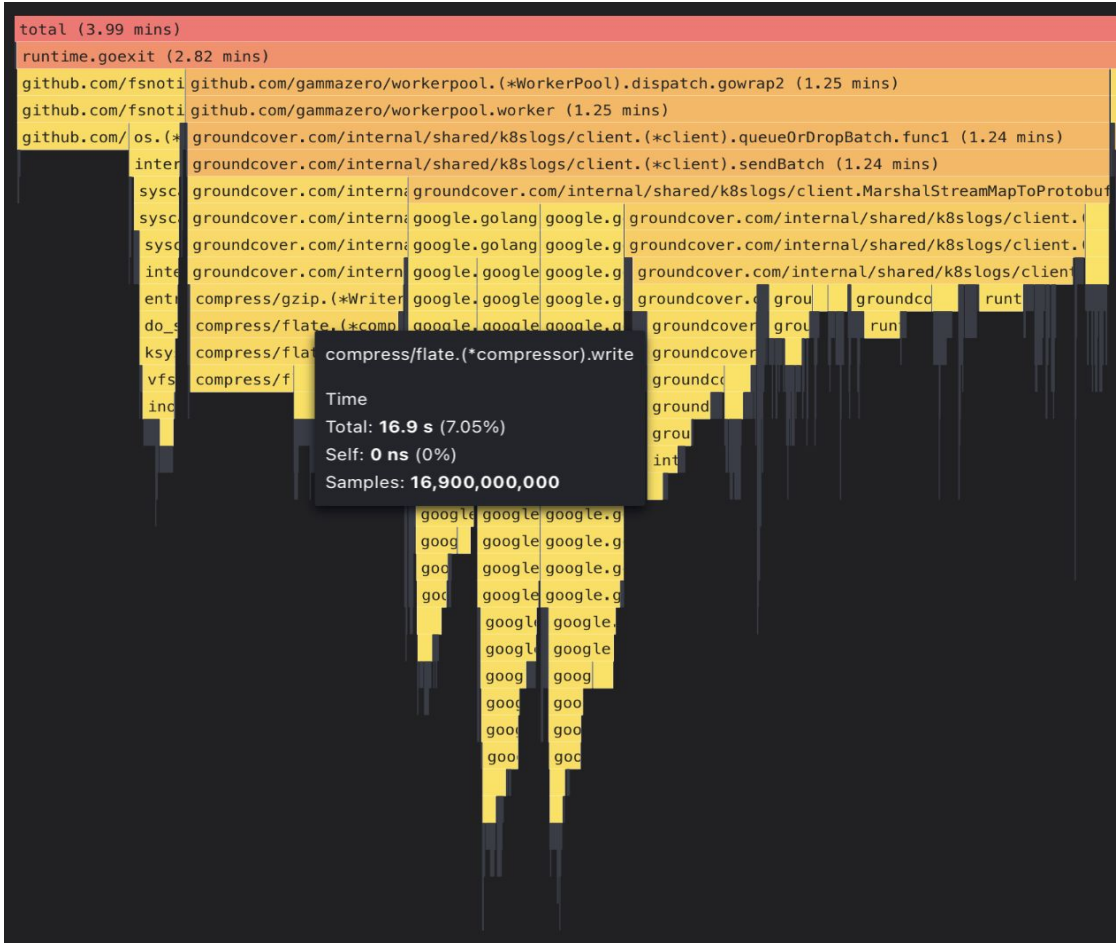
groundcover

Examples



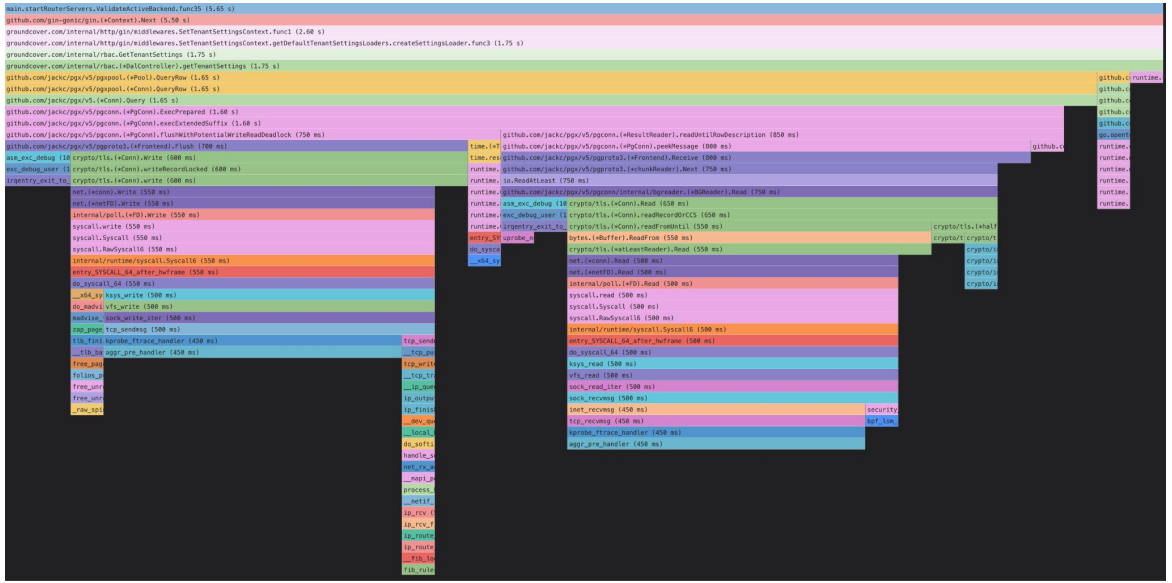
groundcover

Time spent on serialization



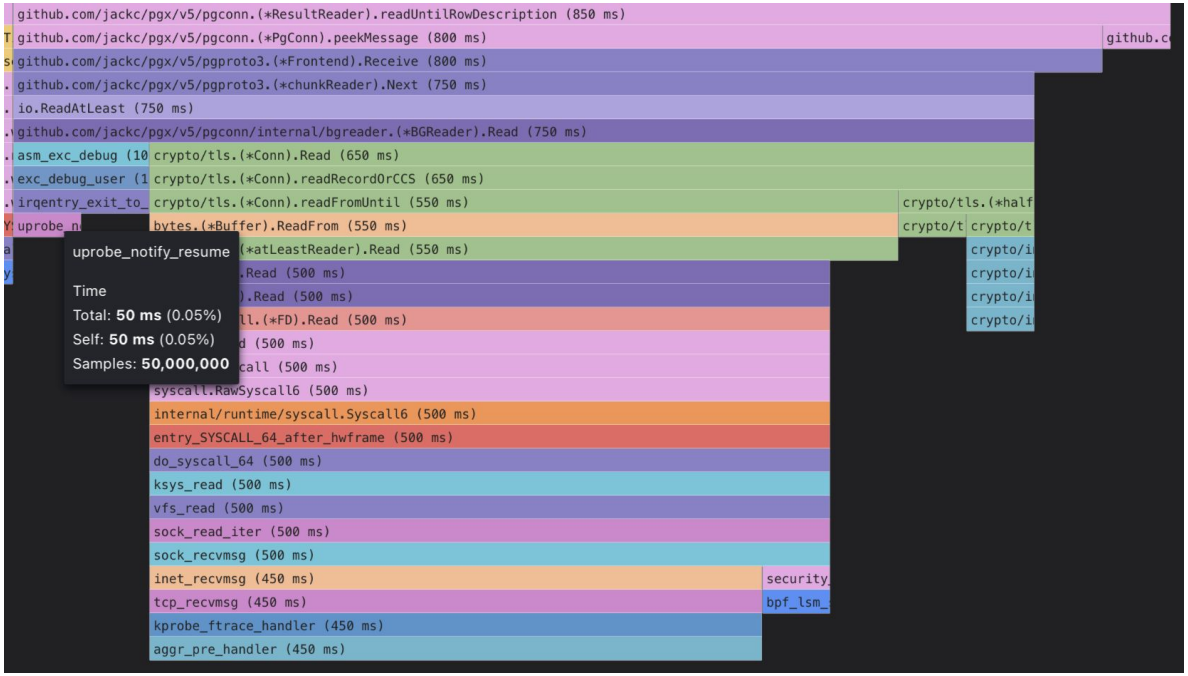
groundcover

Blocking Operations



groundcover

eBPF overhead!



groundcover

Much more

- 3rd party effects
- OS
- Memory profiling

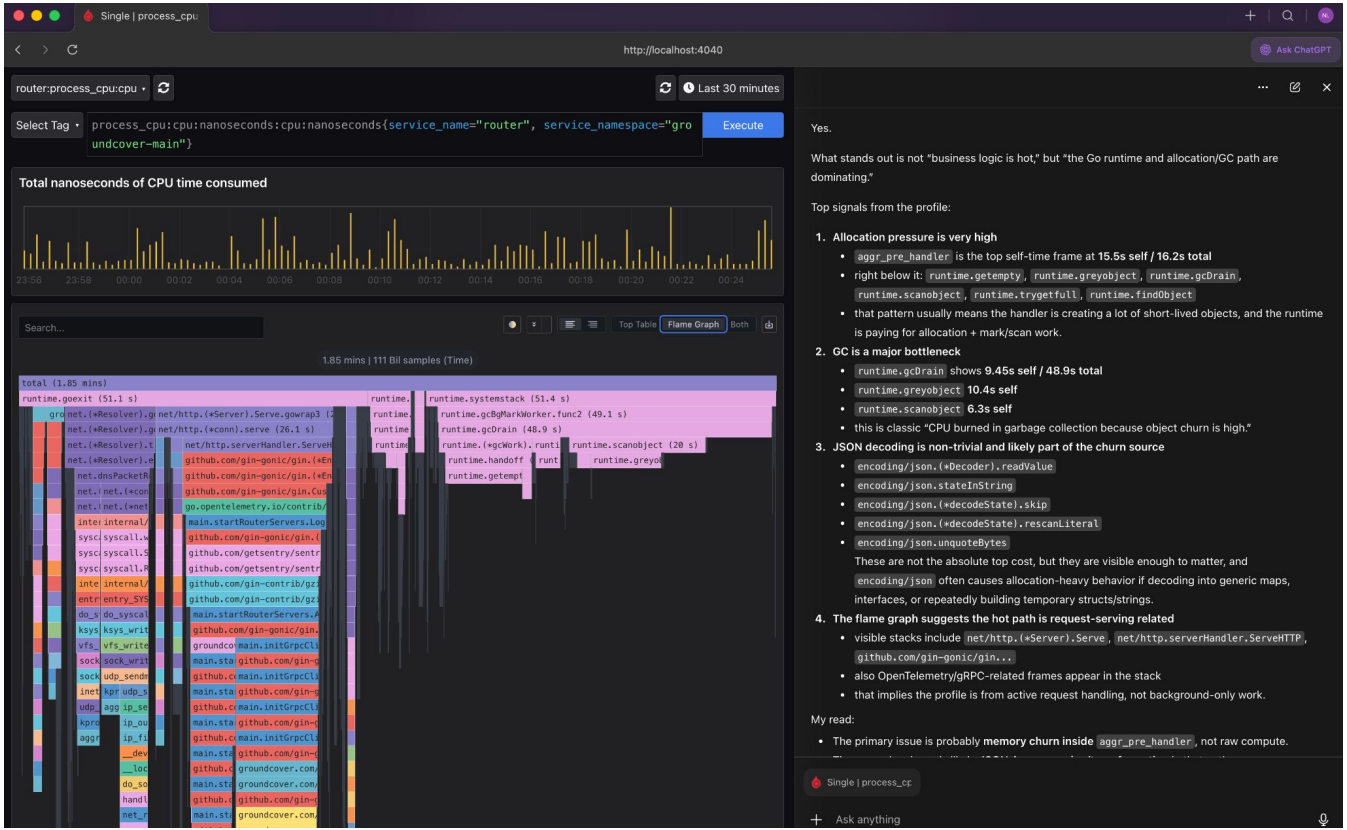
groundcover

The 4th Pillar

- Profiling insights **complements MLT**
- Latency from metric - drill down to a span - bottleneck from profile
- Suspicion from a profile - added instrumentation - clearer signal going forward
- **Single pane of glass**

groundcover

AI



groundcover

AI+eBPF

- AI will analyze profiles **without/alongside humans**
- eBPF **is** generating signals without humans
- They empower each other
- Unblocks human-on-the-loop

groundcover

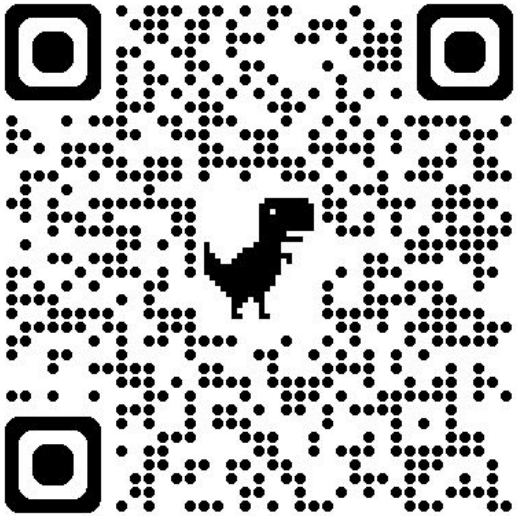
Takeaways

- Introducing profiling becomes **easier**
- tooling/approach is **standardized**
- It's only getting **better**
- There is so much your apps are doing **and you don't know**
- eBPF unblocks human-agnostic observability.
- https://www.brendangregg.com/

groundcover

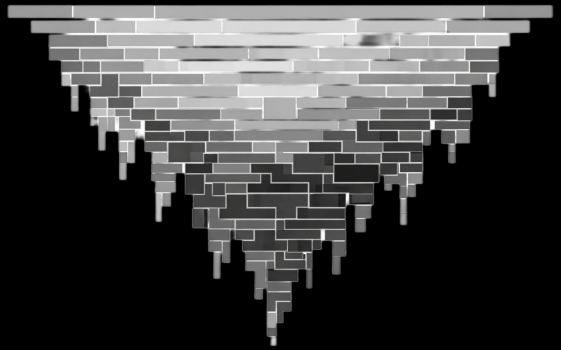
Try it yourself

- One command
- Full setup - agent,collector,backend



groundcover

?



groundcover

- eBPF based 011Y platform
- OpenTelemetry native
- BYOC Architecture
- AI packed
- Profiling - soon