

eBPF instrumentation for RGW: when Ceph's native tracing isn't enough



Ondřej Kukla

WHO WE ARE

CDN77 in numbers

A top-5 global CDN — the only one from Europe.

40 000

physical servers

390+

data centers in 200+ locations worldwide

330 Tbps

network capacity

85 Tbps

daily peak traffic

50 Tbps

global private backbone

~230 PB

raw capacity across 45+ Ceph clusters

3 AM: S3 is slow — and this is all you have.

RGW OPS LOG

one line, no breakdown

```
operation=get_obj method=GET path=/my-bucket/report.pdf http_status=200
object_size=2489331 bytes_sent=2489331 total_time=49
trans_id=tx0000034ef12bc78d90a1-006a573543-9876543-zone1
```

PERF COUNTERS

averages over every
request — not this one

```
rgw.req · avgcount 1 284 311
rgw.get_initial_lat · avgtime 0.0031 s
rgw.put_initial_lat · avgtime 0.0098 s
rgw.qlen · 12
```

OSD LOG

a PG and an
op — no trans_id

```
osd.23 [WRN] slow request 0.012 s
osd_op(client.4321.0:987654 5.3c [call version.read,read] ondisk+read)
currently waiting for rw locks
```

49 ms to fetch one 2.5 MB object. Where did it go?

Doesn't Ceph have tracing?

It exists — but the setup doesn't pay for itself.

Not enough detail

It gives you trace id that's transfered between daemons, however the spans have very little detail.

Needs its own infrastructure

Spans go nowhere until you deploy a Jaeger agent and collector next to the cluster.

Little movement since it landed

Tracer code: last functional change Oct 2023 — formatting-only since.

Docs page: last content change Apr 2023.

It's there. It's not where the effort goes. And what comes out doesn't justify the setup.

ACT 1 · THE PIVOT

So what can we do?

ON A PRODUCTION RADOSGW

We don't want to restart.

We don't want to recompile.

We don't want to run debug logging at volume.

WHERE THE ANSWER COMES FROM

built on top of

cephtrace

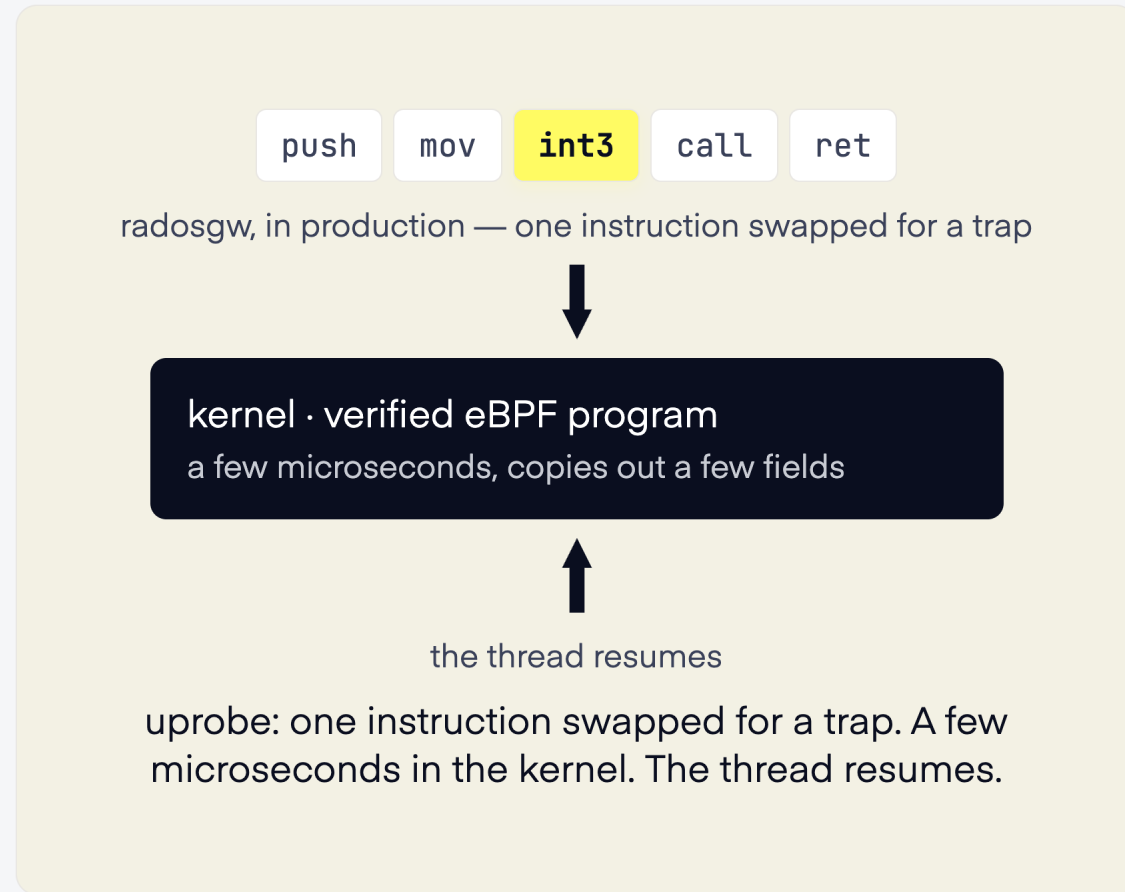
The existing eBPF tracer for RADOS clients — code lineage, GPL-2.0.

inspired by

Grafana Beyla

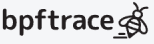
Zero-code eBPF instrumentation of running processes — conceptual inspiration.

A breakpoint that never stops the process.



What runs in that dip is eBPF

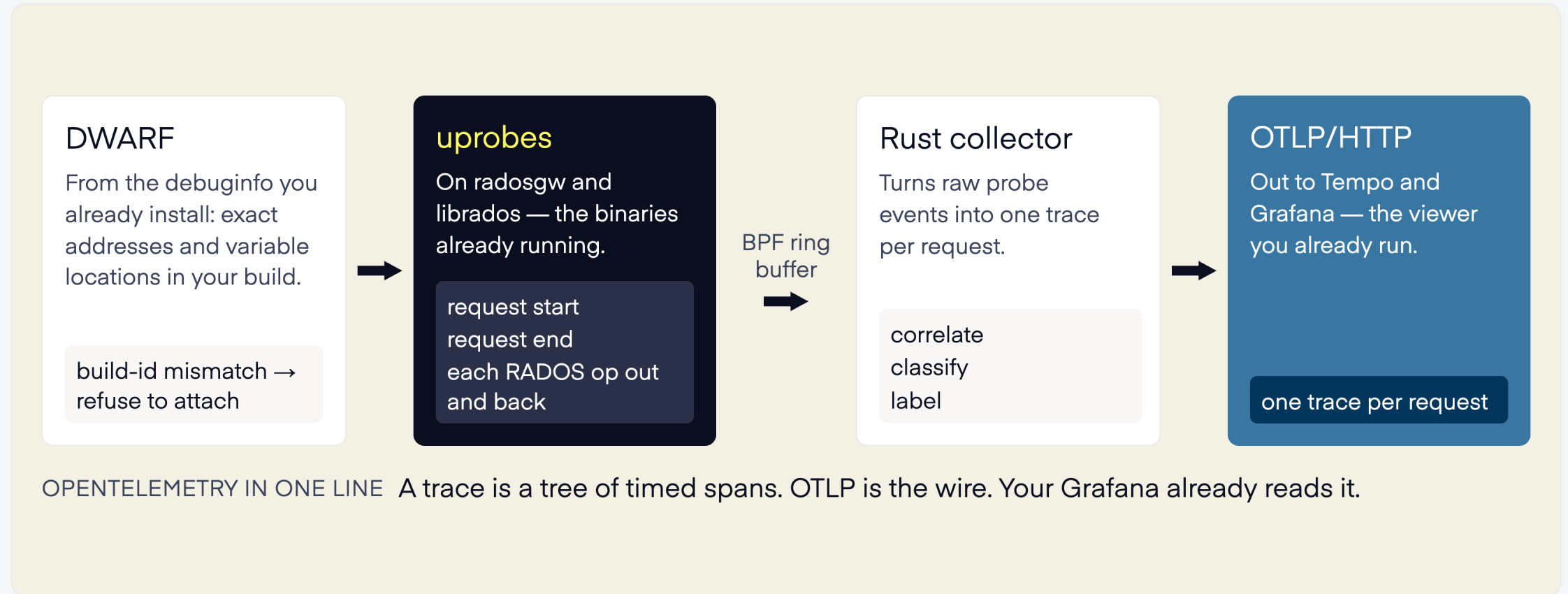
A tiny sandboxed VM inside the Linux kernel you already run. Before a program may attach, the verifier proves it is safe — so an attached probe cannot take the kernel down.

you already run eBPF:    cilium

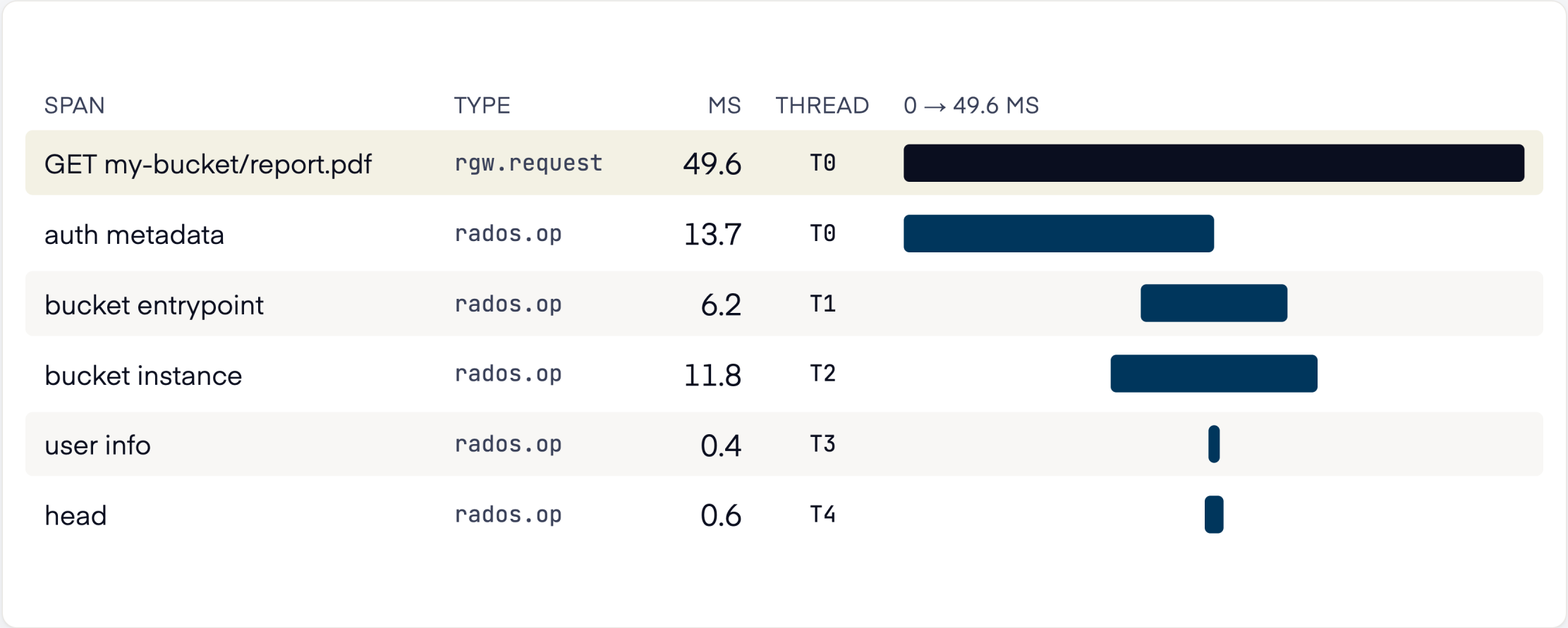
THE VERIFIER GATE

- ✓ terminates
- ✓ memory-safe
- ✓ can't crash the kernel

From your debuginfo to Grafana



One request, as rgwtrace sees it.



Every RADOS op tells you where it went.

rados.op · bucket instance		11.8 ms
attribute	value	
ceph.rados.object	.bucket.meta.my-bucket:e7c1a2b3...12345678.1	
ceph.rados.ops	call(version.read), read	
ceph.rados.pool	default.rgw.meta	
ceph.rados.pg	5.3c	
ceph.rados.acting	[23, 41, 8]	
ceph.rados.target_osd	23	
ceph.rados.rw	read	
ceph.rados.duration_ns	11 800 000	
rgw.rados_role	bucket metadata	

A slow request
names its slow OSD.

ACT 4 · MEASURED, MODELED, OPEN

What it costs.

5.46 μ s

per probe hit

~158 μ s

added per request at 10 RADOS ops

One request at 4 RADOS ops, all probes

96.5 μ s

trap · 79 %

bodies · 21 %

76.5 μ s is the trap itself — the cost of stopping at a breakpoint at all. 20 μ s is our code. There is little left to optimize in the probes.

Measured on an 8-vCPU VM with PTI and MDS mitigations. Bare metal literature: 1.5–4 μ s per hit.

Try it on your test cluster.

```
# build-id check, every probe resolved — before anything is touched
sudo rgwtrace --dwarf-source native --preflight
# live run against the Tempo you already have; --pid for containerized RGW
sudo rgwtrace --dwarf-source native --pid $(pidof radosgw)
--output otlp [OTLP endpoint]
```

KNOW BEFORE YOU RUN

- ✓ Needs root and debuginfo for your Ceph build.
- ✓ Verified on reef 18.2.x and squid 19.2.x.

BUILT ON CEPHTRACE

Dongdong Tao and contributors. The BPF core and the DWARF approach derive from its radostrace.



github.com/kuklaondrej/rgwtrace

Questions?

Let's connect.

Ondřej Kukla

▶ ondrej.kukla@cdn77.com

🔗 linkedin.com/in/ondrej-kukla

RGWTRACE ON GITHUB



github.com/kuklaondrej/rgwtrace