Vector: One Agent to Rule Them All?

One agent. One community. All your observability.

Pavlos Rontidis
Software Engineering Lead
github.com/pront

Today's talk

01 Many tools, many problems

02 One solution: Vector

03 Story time

04 State of the project

05 How to get involved

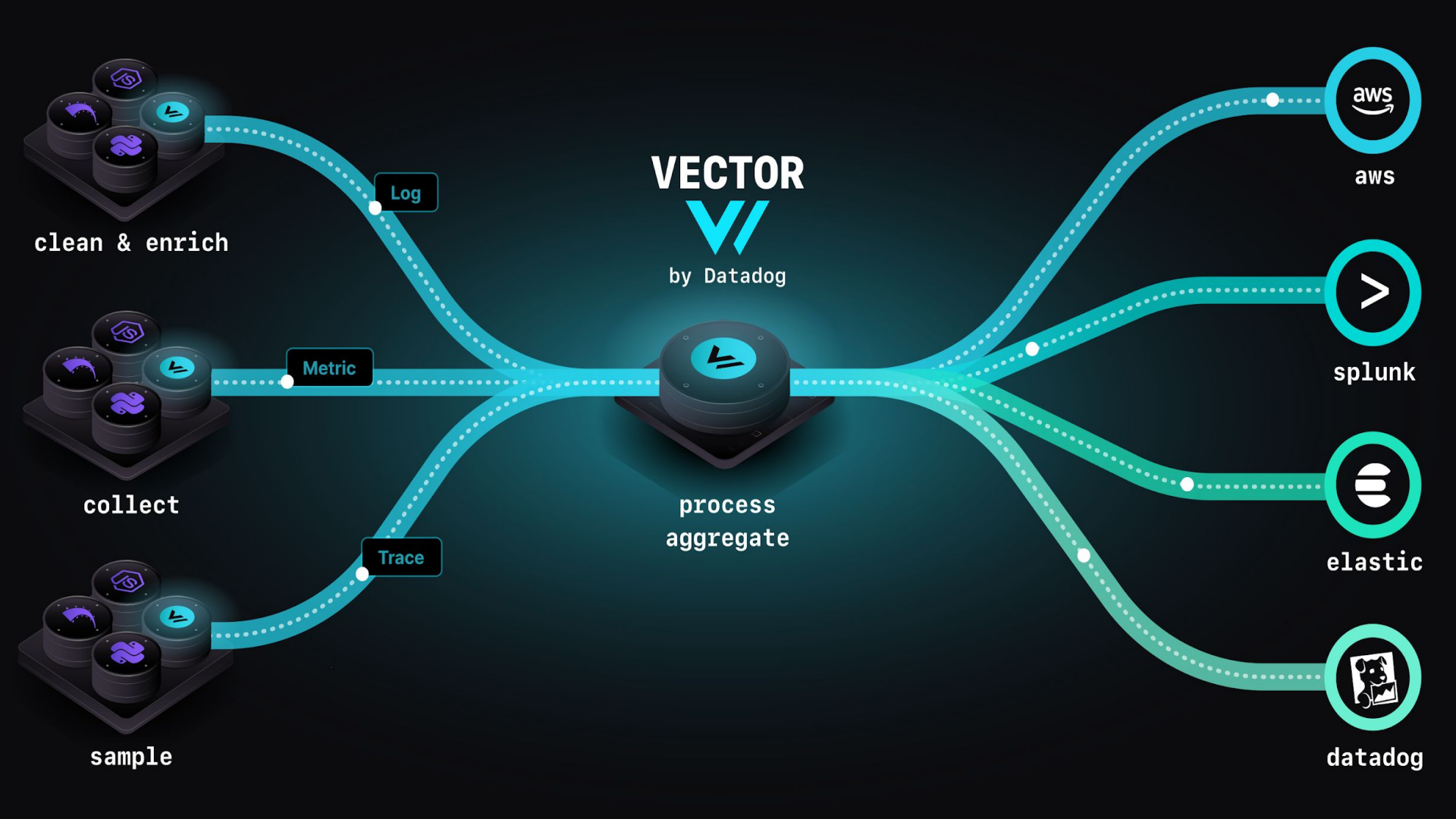
DATADOG

N agents → N problems

🤦 *If you are using more than one...*

- Different Configs
- Different Failure Modes
- Different Resource Profiles

DATADOG


clean & enrich
collect
sample
Log
Metric
Trace
VECTOR
by Datadog
process
aggregate
aws
aws
splunk
elastic
datadog

Vector, in a nutshell

Written in Rust
Memory safe, high performance

Config-driven
Declarative and high level

Unified Pipeline
Logs, metrics, and traces

Deploy Anywhere
Kubernetes, containers, VMs, bare metal

100+ sources, transforms, and sinks

Open Source, Truly

GitHub Repository
github.com/vectordotdev/vector

Zero closed-source components
Fork and ship your own

Stewardship
Community Open Source Engineering team (COSE)

Processing Solutions

Parse and enrich
remap
Vector Remap Language offers 200+ functions

Aggregate over time
aggregate, reduce

Sample and rate-limit
sample, throttle, dedupe

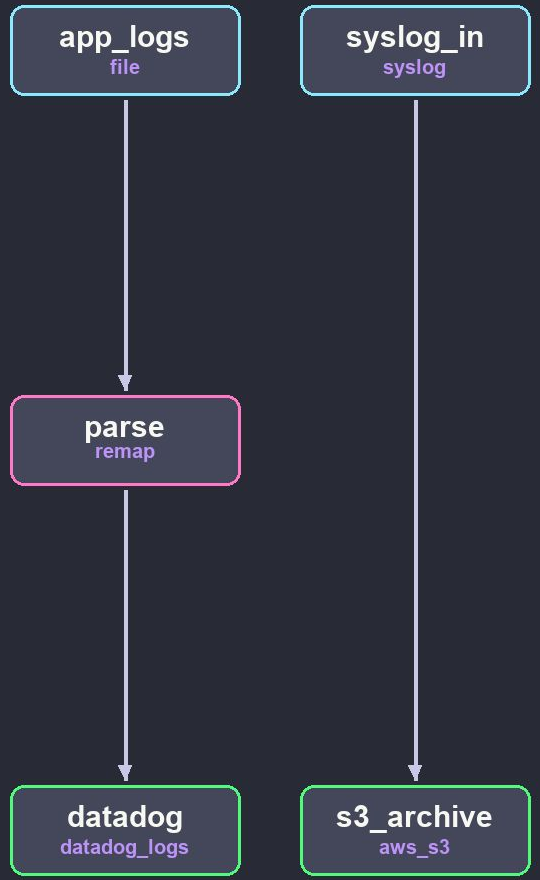
Split by content
route, filter, exclusive_route

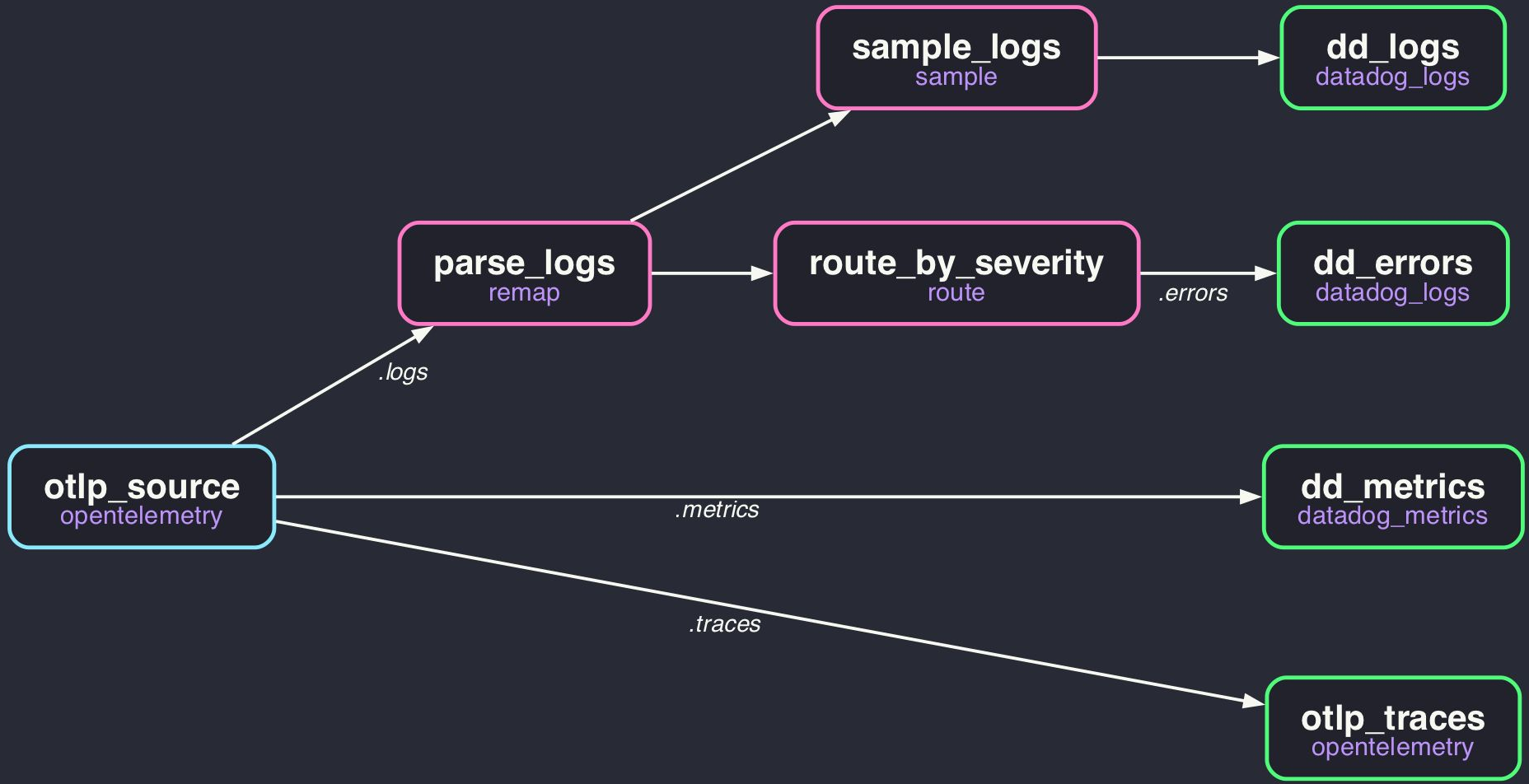
Tame cardinality
tag_cardinality_limit

Convert data types
log to metric, trace to log, etc.

All in one Vector pipeline

```yaml
sources:
  app_logs:
    type: file
    include: ["/var/log/app/*.log"]

  syslog_in:
    type: syslog
    address: "0.0.0.0:514"
    mode: tcp

transforms:
  parse:
    type: remap
    inputs: ["app_logs"]
    source: |
      . = parse_json!(.message)

sinks:
  datadog:
    type: datadog_logs
    inputs: ["parse"]

  s3_archive:
    type: aws_s3
    inputs: ["syslog_in"]
    bucket: "logs-archive"
    region: "us-east-1"
```




sample_logs
sample
dd_logs
datadog_logs
parse_logs
remap
route_by_severity
route
.errors
dd_errors
datadog_logs
.logs
otlp_source
opentelemetry
.metrics
dd_metrics
datadog_metrics
.traces
otlp_traces
opentelemetry

```yaml
sources:
  otlp:
    type: opentelemetry
    grpc:
      address: 0.0.0.0:4317
    http:
      address: 0.0.0.0:4318

sinks:
  dd_logs:
    type: datadog_logs
    inputs: ["sample_logs"]
  dd_errors:
    type: datadog_logs
    inputs: ["route_by_severity.errors"]
  dd_metrics:
    type: datadog_metrics
    inputs: ["otlp.metrics"]
  otlp_traces:
    type: opentelemetry
    inputs: ["otlp.traces"]
    protocol:
      type: http
      uri: "http://tempo:4318/v1/traces"
```

```yaml
transforms:
  parse_logs:
    type: remap
    inputs: ["otlp.logs"]
    source: '. = parse_json!(.message)'

  sample_logs:
    type: sample
    inputs: ["parse_logs"]
    rate: 1000

  route_by_severity:
    type: route
    inputs: ["parse_logs"]
    route:
      errors: '.severity == "error"'
```

Case Study: Quad9

Before Vector
4 Go repos · custom dnstap tooling · 2012 hardware

Contributing
They assigned one engineer to implement various Vector features

Case Study: Quad9

Before Vector
4 Go repos · custom dnstap tooling · 2012 hardware

Contributing
They assigned one engineer to implement various Vector features

After

- Single Vector pipeline
- 5x faster ingestion
- New features now available for all Vector users

Vector is alive and thriving

Release Schedule

every 6 weeks

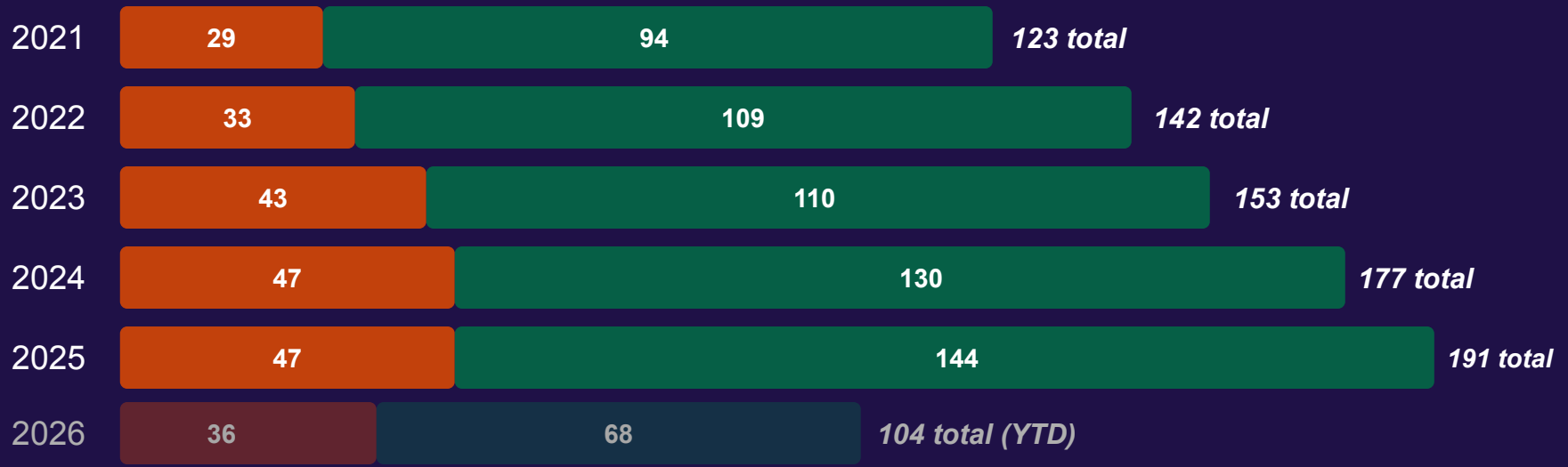
Broad Adoption

Used by thousands of orgs in prod 💜

Vibrant Community

~160 PR contributors in the last 12 months

Unique Contributors



Get involved!

pront.github.io/maintainer-month-vector-2026

DATADOG

Thank You!

We're saving questions for the networking session right after the talks. Please come find us there.

We are hiring 🎉 Scan to apply:

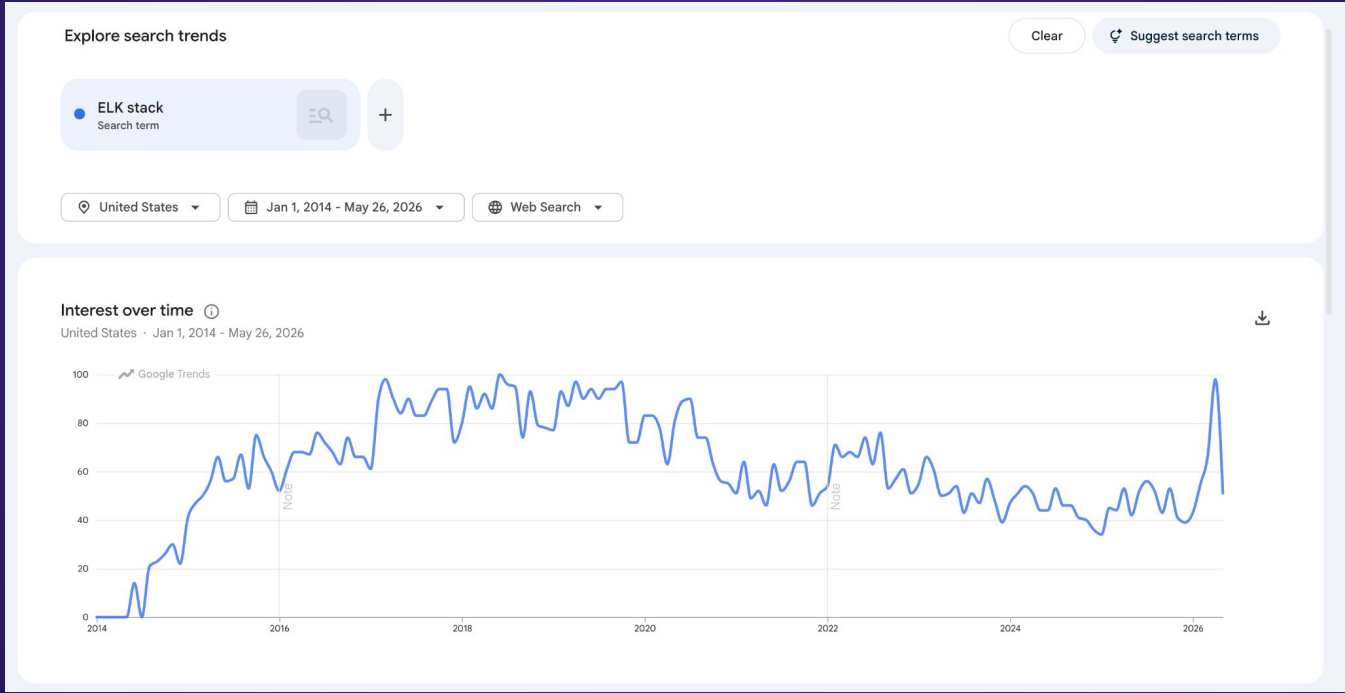
DATADOG

Quickwit

Subsecond Full-Text Search on Object Storage

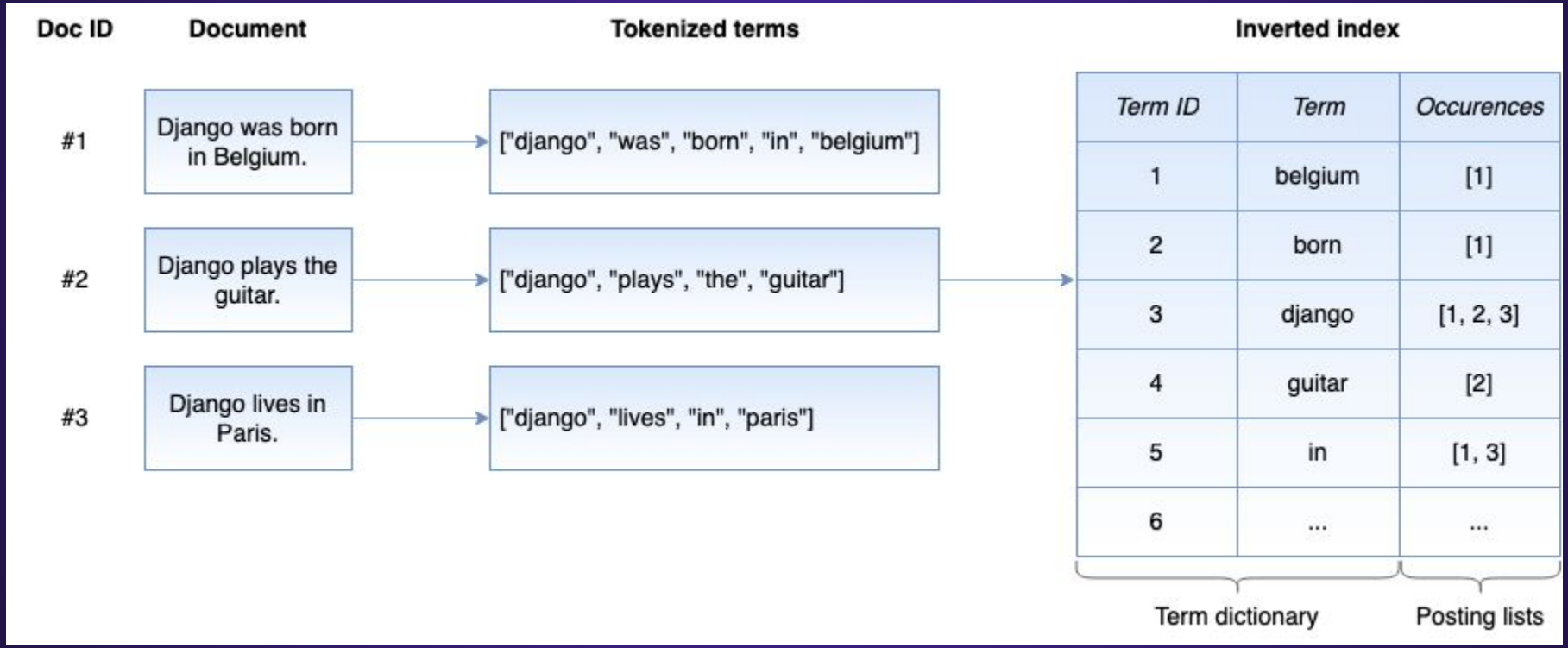
Adrien Guillo
Software Engineer
github.com/guilload

ELK Stack Popularity



DATADOG

Inverted Index 101



DATADOG

Inverted Index Trade-Offs

- Fast and powerful full-text search queries

BUT

- Expensive to build
- Expensive too store
- Hostile to object storage

→ *“Pay” for CPU at **indexing** time*

“Index-Free” Mouvement

How Humio Index-free Log Management Searches 1 PB in Under a Second

June 10, 2021 | Humio Staff | Next-Gen SIEM & Log Management

DATADOG

“Index-Free” Way

- No or very little indexing (metadata-only)
- Fast compression algorithms (LZ4, Snappy)
- Massively parallelized distributed *grep*

“Index-Free” Trade-Offs

- Lower Total Cost of Ownership (TCO)
- “Simpler” architecture

BUT

- Slow “needle in haystack” queries (trace ID, user ID, …)
- Large scans if can’t narrow down scope of queries

→ *“Pay” for CPU at **query** time*

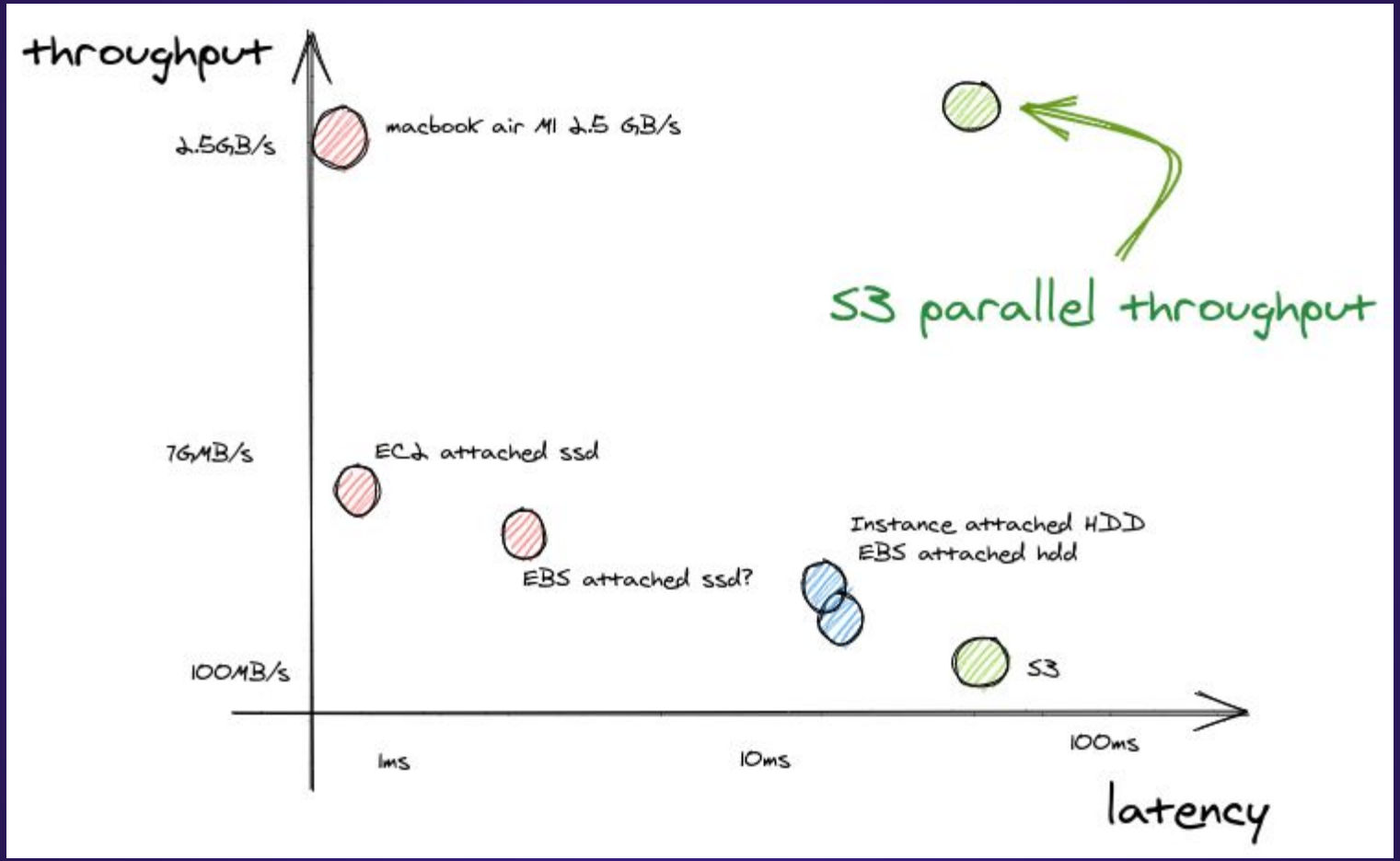
Introducing Quickwit

- A new take on indexing for logs and traces
- Optimized for object storage

Optimizing for Object Storage

Optimizing for Low Throughput

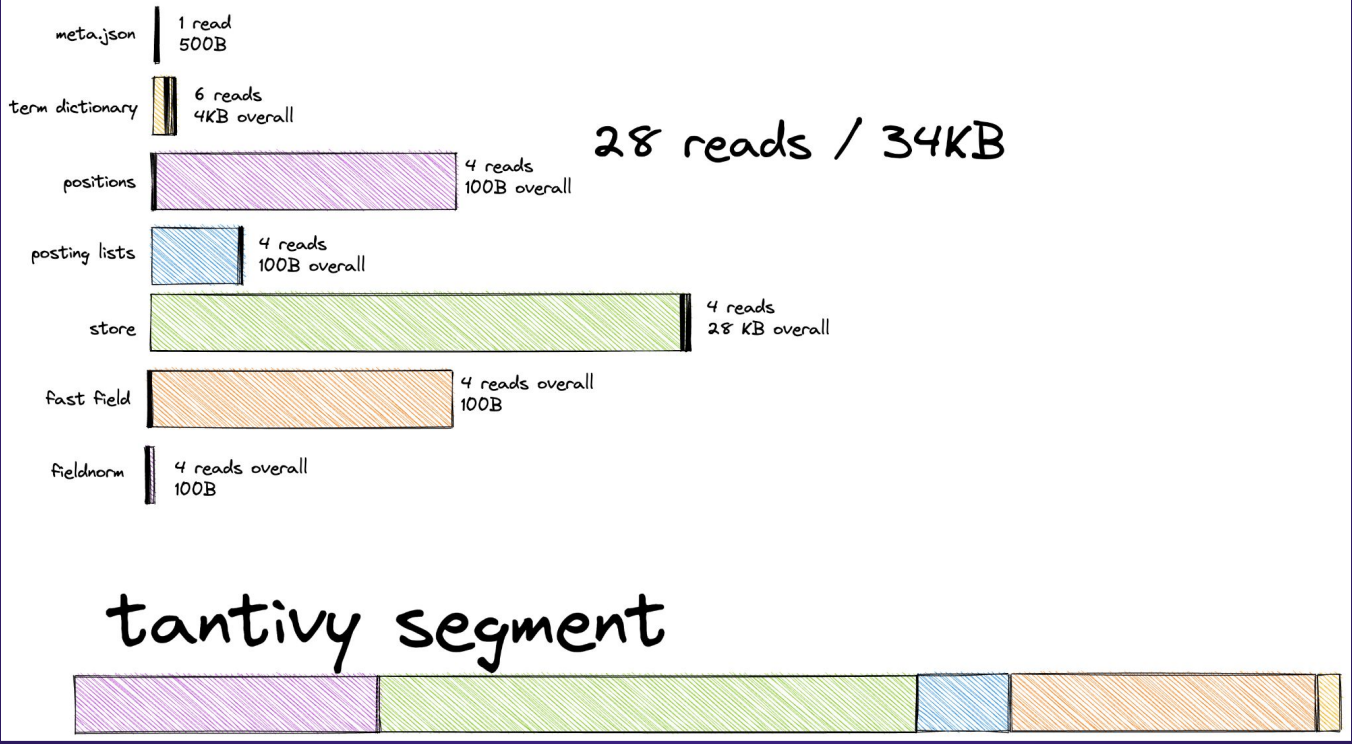
- Throughput per TCP connection: 80–100 MiB/s


throughput
2.5GB/s
macbook air M1 2.5 GB/s
S3 parallel throughput
7GMB/s
EC2 attached ssd
Instance attached HDD
EBS attached hdd
EBS attached ssd?
100MB/s
S3
1ms
10ms
100ms
latency

Optimizing for High Latency

- Time To First Byte (TTFB): 50–200 ms

IO operations to open a tantivy segment



IO operations to open a Quickwit split

turbo index 1 read / 34KB

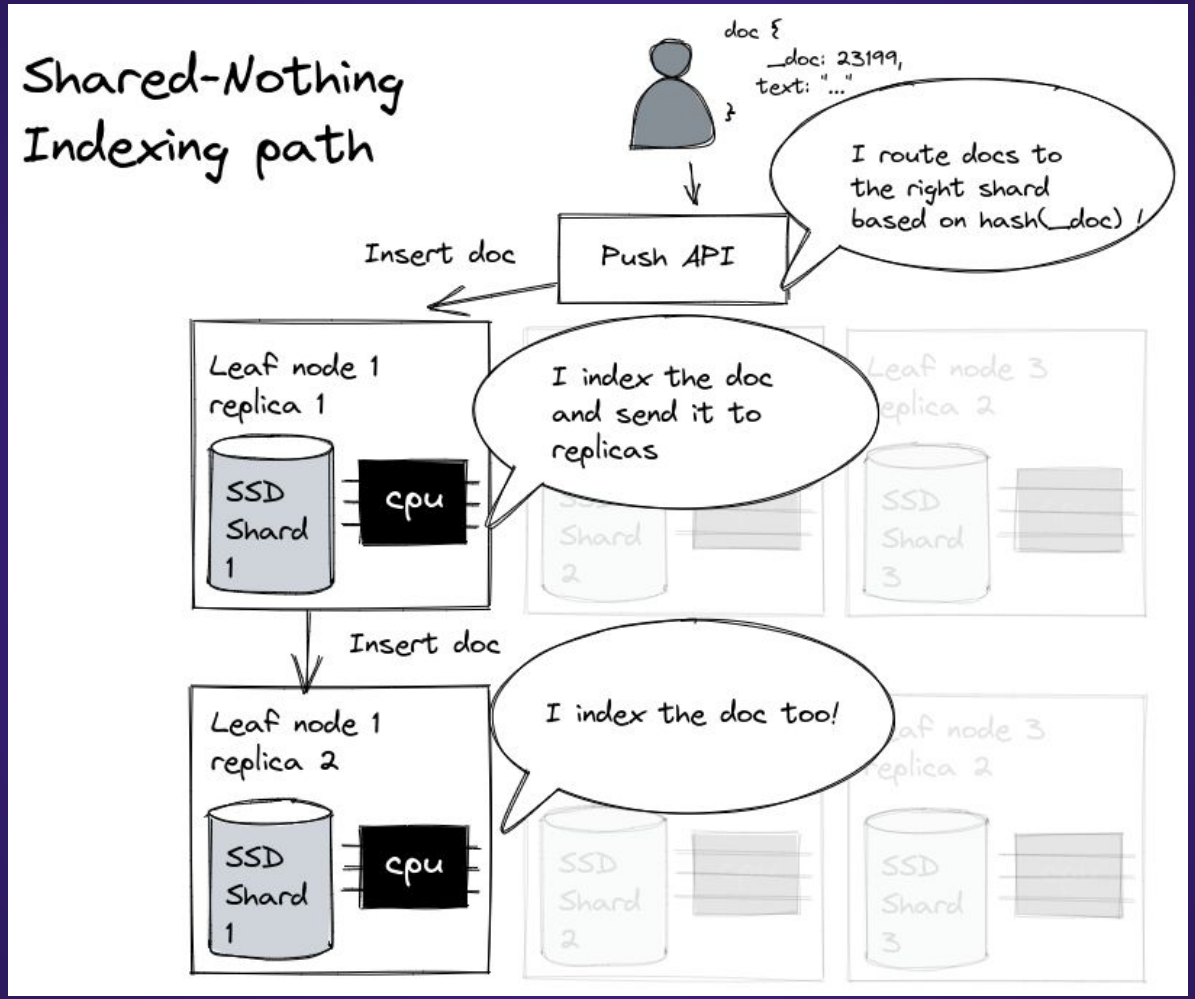
Quickwit split

Optimizing for High Latency

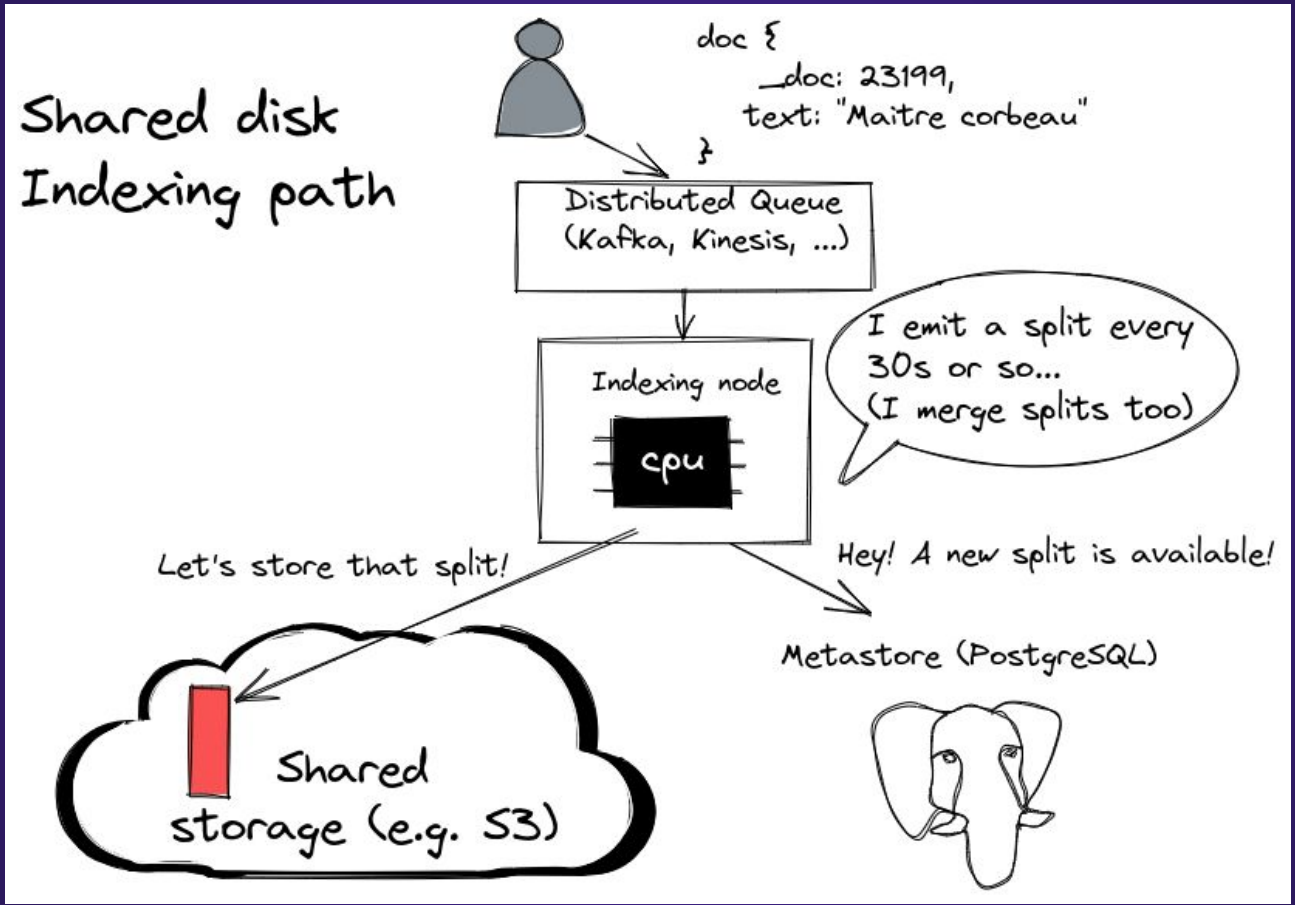
- Fetch large ranges
- Use data structures that can be split in large blocks
- Coalesce contiguous GET requests
- Hedge GET requests

Architecture

Shared-Nothing vs. Shared Disk


Shared-Nothing
Indexing path
doc {
_doc: 23199,
text: "..."
}
I route docs to the right shard based on hash(_doc) !
Push API
Insert doc
Leaf node 1
replica 1
SSD
Shard
1
cpu
I index the doc and send it to replicas
Insert doc
Leaf node 1
replica 2
SSD
Shard
1
cpu
I index the doc too!

DATADOG


Shared disk
Indexing path
doc {
_doc: 23199,
text: "Maitre corbeau"
}
Distributed Queue
(Kafka, Kinesis, ...)
Indexing node
cpu
I emit a split every
30s or so...
(I merge splits too)
Let's store that split!
Hey! A new split is available!
Metastore (PostgreSQL)
Shared
storage (e.g. S3)

Problem Solved?

- Decoupled compute and storage ✅
- Fast search queries on object storage ✅

Problem Solved?

- High indexing throughput? -ish (5–20 MiB/s per core)
- High compression ratio? -ish (3–5x)

CLP: Efficient and Scalable Search on Compressed Text Logs

Kirk Rodrigues, Yu Luo, Ding Yuan
University of Toronto & YScope Inc.

Abstract

This paper presents the design and implementation of CLP, a tool capable of losslessly compressing unstructured text logs while enabling fast searches directly on the compressed data. Log search and log archiving, despite being critical problems,

generated over three years. As a result, the log management industry has grown incredibly large.

Currently, Elastic [2] and Splunk [4] are two of the largest companies in the industry. In just their last fiscal year, Elastic reported revenue of $428 million with a total of 11,300 cus-

Quickwit Today

- Under Active Development
- Still pre 1.0 and some rough edges
- Plenty of opportunities to contribute
- github.com/quickwit-oss/quickwit

Thank You!

We're saving questions for the networking session right after the talks. Please come find us there.

We are hiring 🎉 Scan to apply:

DATADOG