Modern application
Observability with
Grafana & Quickwit
Who am I?

- François Massot @francoismassot on Twitter / Fosstodon
- Core dev on Quickwit engine (Rust) and the Grafana datasource plugin
- Cofounder of Quickwit
Agenda

1. Anatomy of observability
2. The cardinality curse
3. Logs and traces storage engine: Quickwit
4. Demo time
Anatomy of observability

Author: Ben Sigelman, [link](#)
The cardinality curse

Distributed systems can fail for a large number of reasons.

What if we have metrics with labels: version, host, customer_id, service,

Cardinality = 10 * 1k * 100k * 10 = 10 billion

=> We should control the cardinality for metrics.
=> Let's keep all the attributes in traces.
Logs and traces storage engine: Quickwit

- Distributed search engine for logs and traces
- Decoupled compute & storage (like Loki/Tempo).
- Optimized (sub-second) for query on object storage.
Engine architecture

Architecture in its simplest form.
Anatomy of a split

- **44.86B** Size of JSON docs
- **15.16B** Split (10^6 docs)
  - Doc store (27.2%)
  - Inverted index (24.5%)
  - Columnar store (43.1%)
  - Hotcache (0.07%)

Compressed block of 1M docs

Contains:
- Term dictionary (term -> posting list) - 53%
- Posting list - 47%

Columns as compressed indexable bitset

Contains metadata on the 3 datastructures
Span data model in Quickwit

Based on the OTEL data model. `resource_attributes` and `span_attributes` are schemaless fields.

```json
{
    "trace_id": "b31ab9dda41afde6b4ac992ea56afc89",
    "parent_span_id": "7505182209aa02d1",
    "span_id": "6bf90d1ffe7b9b5c",
    "span_kind": 2, // SERVER
    "service_name": "postgres",
    "span_name": "query-articles",
    "span_start_timestamp_nanos":1706984714411104000,
    "span_duration_millis":164,
    "span_attributes": {
        "net.sock.host.addr":"192.168.13.102",
        "k6.OK4pWn6reJjwtVH": "PwWJNACBiM0Kvwv2RMuykgsx0AySXl",
        ...
    },
    "resource_attributes": {
        ...
    }
}
```
Demo time!

Setup:

- xk6 tracing jobs to generate traces
- Quickwit deployed on a Kubernetes cluster
- Grafana instance with Quickwit datasource
Generating and sending traces to Quickwit

- **tracegen**: simple but spans are too simple
- **xk6 tracing**: YES!

```javascript
const traceTemplate = {
  spans: [
    {service: "shop-backend", name: "list-articles", duration: {min: 200, max: 900}},
    {service: "article-service", name: "select-articles", attributeSemantics: tracing.SEMANTICS_DB},
    {service: "postgres", name: "query-articles", attributeSemantics: tracing.SEMANTICS_DB, randomAttributes: {count: 5}},
    ...
  ],
}

const client = new tracing.Client({
  endpoint,
  exporter: tracing.EXPORTER_OTLP
});

export default function () {
  const gen = new tracing.TemplatedGenerator(traceTemplate);
  client.push(gen.traces())
}
```
## Demo: Quickwit traces index

![Index details screenshot](image)

<table>
<thead>
<tr>
<th>SUMMARY</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Created at:</td>
<td>2024/02/03 17:20</td>
</tr>
<tr>
<td>URI:</td>
<td>s3://quickwit-indexes/otel-traces-v0_7</td>
</tr>
<tr>
<td>Number of published documents:</td>
<td>1,283,476,852</td>
</tr>
<tr>
<td>Size of published documents (uncompressed):</td>
<td>105,680.32 MB</td>
</tr>
<tr>
<td>Number of published splits:</td>
<td>21</td>
</tr>
<tr>
<td>Size of published splits:</td>
<td>15,215.47 MB</td>
</tr>
<tr>
<td>Number of staged splits:</td>
<td>1</td>
</tr>
<tr>
<td>Number of splits marked for deletion:</td>
<td>514</td>
</tr>
</tbody>
</table>

API URL: [http://localhost:7280/api/v1/indexes/otel-traces-v0_7](http://localhost:7280/api/v1/indexes/otel-traces-v0_7)
Demo: Quickwit indexer dashboard

Indexing throughput

13.8 MB/s

Documents throughput

16806 docs/s
Demo: Exploring traces in Grafana
Demo: Zoom on a specific trace
Demo: APM dashboard in Grafana
Future work

- More aggregations (cardinality, rate) (Q2)
- Pipe-based query language (Q3)
- Metrics support (Q4)
Thank you!

francois @ quickwit.io | @francoismassot on Twitter/Fosstodon