Telemetry and APL

Exploring telemetry solutions in distributed systems and an implementation in Dyalog APL

Gilgamesh Athoraya

TIAMATICA

Telemetry ChatGPT

What the computer says about Telemetry:

Telemetry in distributed applications refers to the process of collecting and transmitting data about the performance, usage, and health of various components across the system for monitoring and analysis purposes.

Telemetry
3 Pillars

The three pillars of telemetry:

1. Logs
2. Metrics
3. Traces

Telemetry Logs

Structured logs describing discrete events

- Timestamp
- Source
- Description
- Other useful data

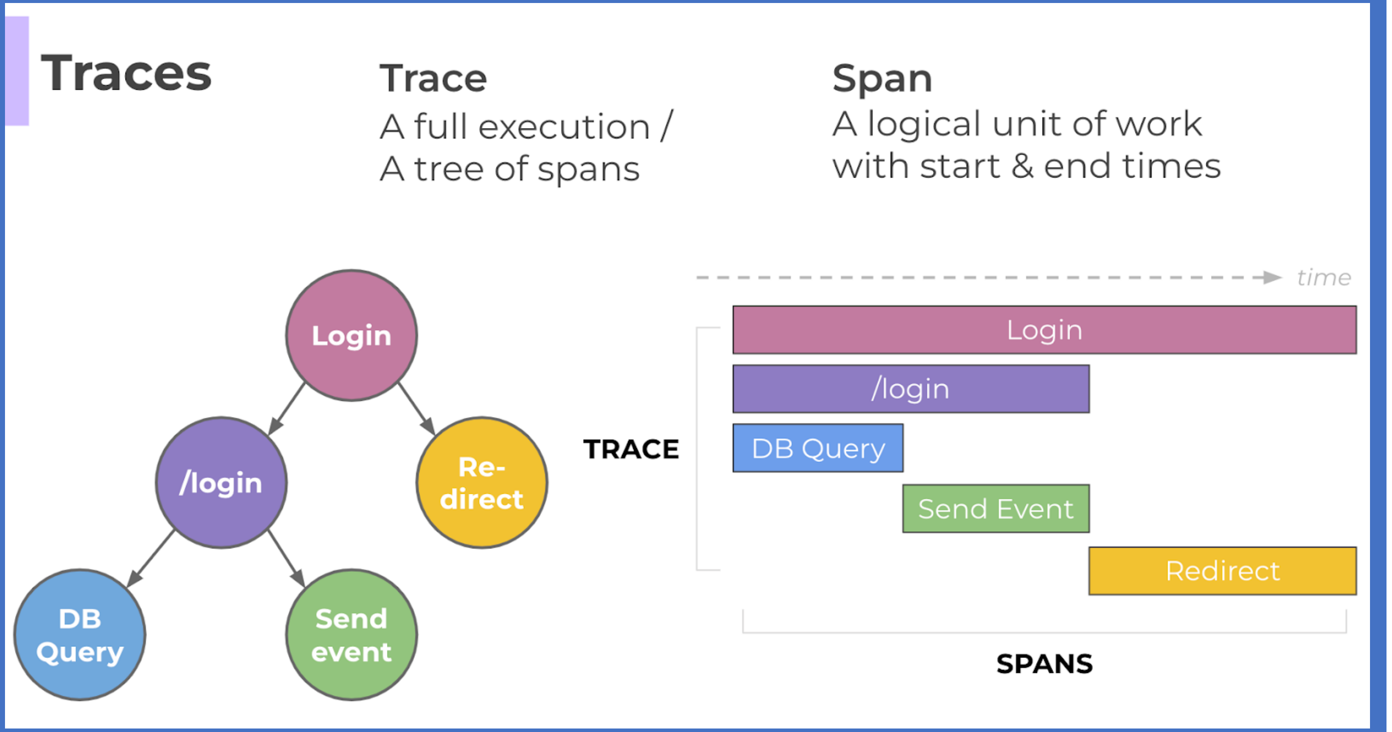
Telemetry Metrics

High level aggregations, counts and measures of various indicators:

- CPU
- Memory
- Jobs/Requests handled
- Etc.

Telemetry Traces

A trace represents the complete path through the system when handling a request or executing a job.



Telemetry Setup

- Applications and services emit telemetry data
- Need a backend to store the data
- Need a frontend to visualize
- Maybe a monitoring tool to alert on certain triggers?

Telemetry Standard

OpenTelemetry

- Observability framework
- Vendor- and tool-agnostic
- Open source
- Collection of tools, APIs, SDKs and protocols

OpenTelemetry Vendors

Long list of vendors that support OpenTelemetry:

- AWS
- Azure
- Google Cloud Platform
- Jaeger
- SigNoz

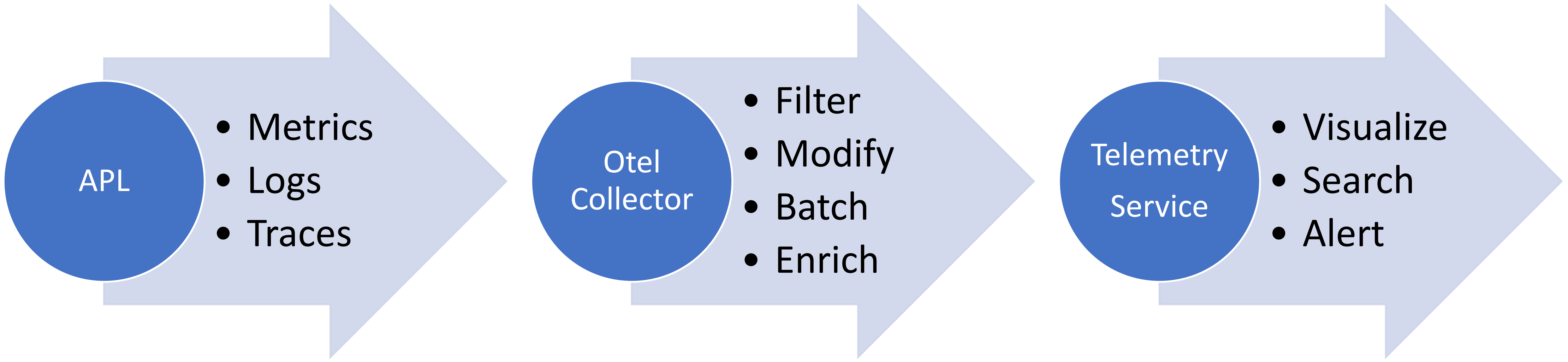
https://opentelemetry.io/ecosystem/vendors/

OpenTelemetry Collector

Vendor-agnostic implementation of how to receive, process and export telemetry data.

- Local agent receives telemetry data from application
- Receiver supports multiple forms of the OpenTelemetry Protocol (OTLP)
  - grpc
  - http + protobuf
  - http + json

Process flow using OtelCollector



OpenTelemetry with APL

To use OpenTelemetry from APL we need an APL SDK that implements:

- the specification
- APIs
- Emits telemetry data

OpenTelemetry with APL

- Send telemetry data using OTLP over HTTP+JSON to local Otel Collector agent
- Configure Otel Collector to batch messages and export to one or more backends
- Use local backend during test/development

Demo

- Simple example app that emits telemetry
- Use docker to start a local OtelCollector and backend

Summary

- OpenTelemetry is adopted by a large number of vendors
- SDKs available for many languages (APL is missing on the list)
- Recommendation is to use a local OtelCollector:
  - Low latency
  - Many extensions available as contribution plugins
  - It supports HTTP+JSON, meaning no need to implement grpc and protobuf support in APL

Thanks for listening

TIAMATICA