Tracing tools for low latency microservices

Eya-Tom Augustin SANGAM

Polytechnique Montréal

DORSAL Laboratory

Agenda

- Context and goal
- Related work
- Considerations
- Proposed solution
- Current and future work
- Conclusion

Context and goal

We have :

- C Microservices communicating with each other using ZeroMQ

We want to **collect telemetry data (TD)** :

- Application logs
- Application and host metrics
- Requests traces (aka spans)

Related Work : LTTng and LTTng-UST

- We can use LTTng to collect host metrics (CPU usage, RAM usage etc.)
- We can use LTTng-UST to collect applications metrics, applications logs and requests traces
  - One big advantage is the ability to record or not specific telemetry data at runtime
  - We need to define a protocol over the standard LTTng-UST logging library for trace collection, metrics collection and context propagation. Which OpenTelemetry Specification already does.

Related Work : OpenTelemetry

- OpenTelemetry (OTel) is becoming the industry standard of creating and collecting TD
- OTel specification describes cross-language requirements and expectations for all OTel implementations.
- Lot of visualisations tools like TraceCompass, Jaeger support OTel data schemas out of the box
- OTel created the OTel Collector which is a vendor-agnostic way to receive, process and export TD

Considerations (1/2)

- We want to do cross-hosts TD analysis
  - We need to bring all TD together at some point

- Some hosts have limited hard drive storage. A filtering mechanism should be possible at runtime to help minimize the amount of data saved on the disk. For example, we should be able to decide at runtime whether we want to save heartbeat traces or not.

Considerations (2/2)

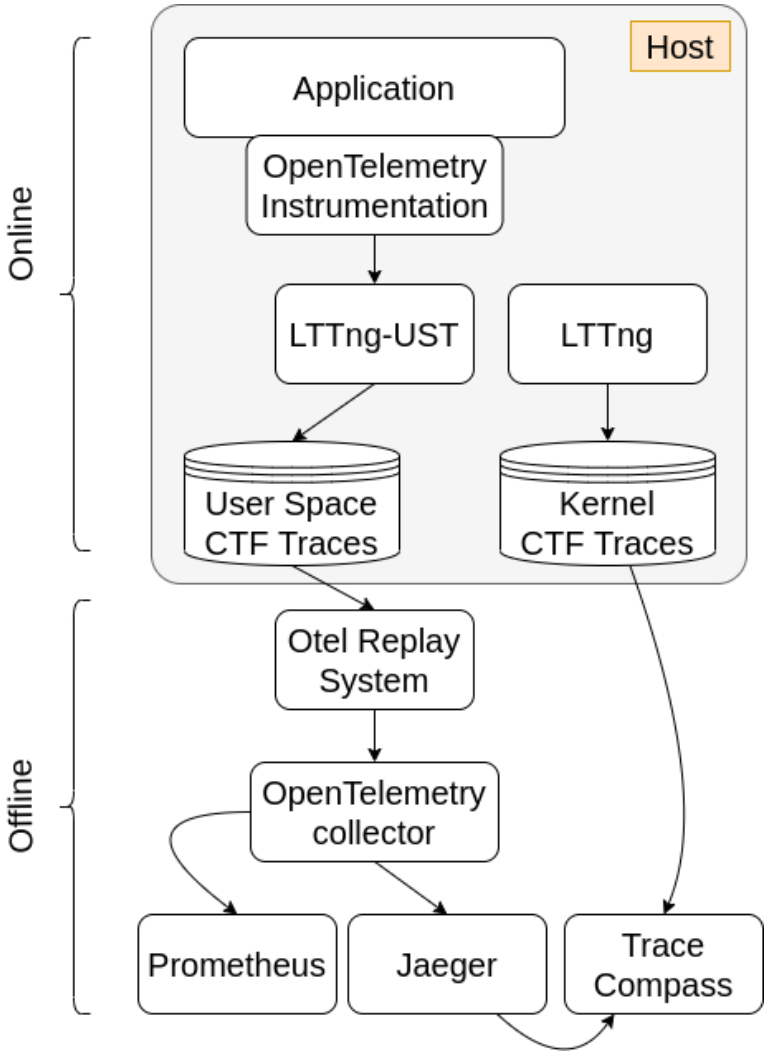
- Some applications run on hosts with limited resources. Installing OTel collector or an observability backend on those hosts may highly affect the application behaviour.

- Live monitoring is not required.

Proposed solution (1/2)

Online part
(When application runs)

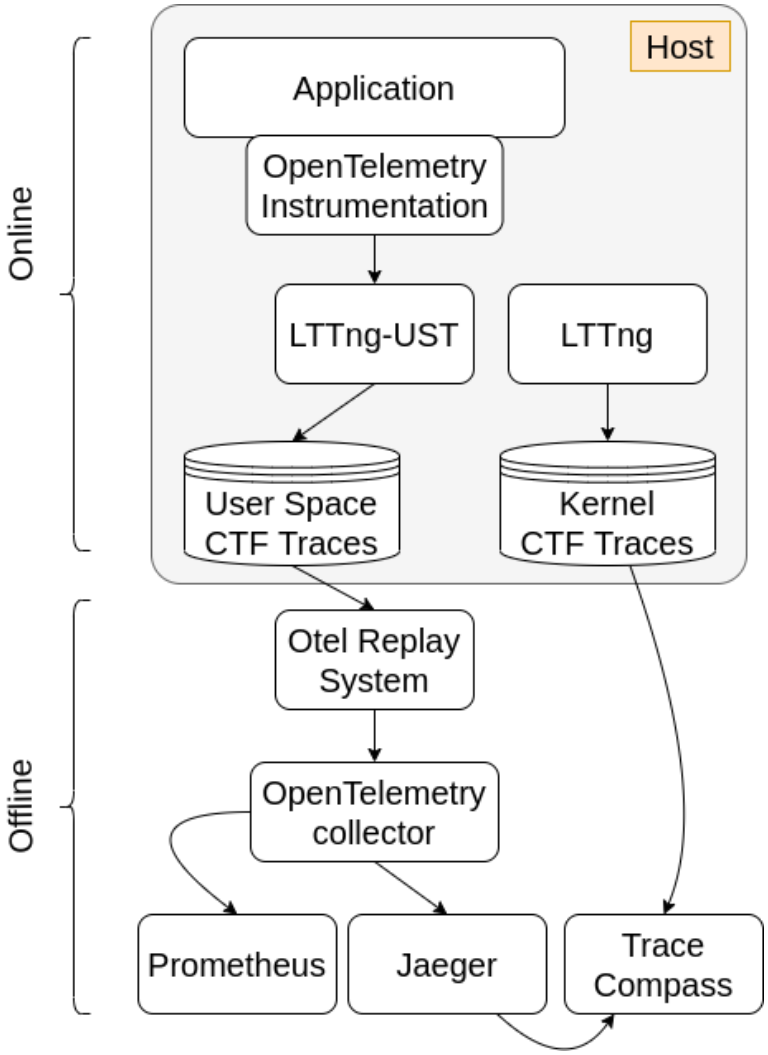
- LTTng is used to collect host metrics
- We use OTel instrumentation
- Telemetry Data generated is logged to LTTng-UST and saved in CTF files.
  - We can control are runtime data we save this way



Proposed solution (2/2)

Offline part
(Only when we want to do analysis)

- CTF files are copied from the host
- Host metrics could be viewed in Trace Compass directly
- The OTel Replay System reads the telemetry data and send them to the OTel collector which will send them later to observability backends (Jaeger, Prometheus, etc.)



Current and future work

- Working on a project using the previous solution
  - Created a client, proxy and server in C application using ZeroMQ
  - Working on a OTel C API/SDK bindings over the OTel C++ API/SDK
  - Working on the OTel Replay System
- Next
  - Integrate ZeroMQ instrumented version of Pierre-Frédérick Denys [1]
  - Start working on traces analysis

Thanks you !

Questions, ideas, remarks ?

References

[1] Pierre-Frédérick DENYS, Michel R. Dagenais, Martin Pepin et al. Advanced Tracing Methods for Container Messaging Systems Analysis, 08 November 2021, PREPRINT (Version 1) available at Research Square [https://doi.org/10.21203/rs.3.rs-722452/v1]