Distributed Tracing

OBSERVABILITY

Problem Statement

- Suppose, your company has developed a complex, distributed system that consists of multiple microservices.
- Each microservice runs on its own server and communicates with other microservices over the network.
- Suppose you receive a report of slow performance and errors, how do you troubleshoot ?
- Each microservice may have their own logs but how do you correlate the logs of all microservices for the same request.

Solution

Distributed Tracing

Distributed Tracing

Distributed tracing is a technique used to track and understand the behavior of complex, distributed systems.

Tracing allows you to see the complete end-to-end flow of a request, including which microservices it touches, how long it takes to process, and any errors that may have occurred.

Advantages

- OBSERVABILITY

- It allows you to see the complete end-to-end flow of a request.
- Provides visibility into the inner workings of the system, which can be critical for ensuring that it is running smoothly and efficiently.
- Helps you to identify bottlenecks and performance issues, debug errors,

Implementation

Typically we use a combination of logging and instrumentation (adding log statements or other code to your application to capture information)

To record key events, we add log statements, such as when a request is received and when a response is sent.

You might also use a tracing library or tool to automatically capture this information and associate it with a unique trace ID. This trace ID will be a part of trace context that will be propagated to the callee.

Thus allowing traces from different services to correlate.

Trace Context
(defined by W3C Trace Context specification)

Trace Context is split into two individual propagation fields.

- Traceparent header
- Tracestate header (Optional)

This headers needs to be propagated by the components participating in the distributed trace

Traceparent header

- Describes incoming request in a tracing system in a common format, understood by all the vendors.
- Format: {version}-{trace_id}-{parent_id}-{trace_flags}

version

- 2 Hexadecimal Digits in lower case. (size: 8 bit)
- Current specification version is set to 00.
- Version ff is forbidden

trace_id

- 32 Hexadecimal digits in lowercase. (size: 16 bytes)
- All traces belonging to the same transaction/request will share this trace_id.
- It is used to correlate all the traces from the distributed components.
- All zeroes is forbidden (E.x: 00000000000000000000000000000000)

parent_id/span_id

- 16 Hexadecimal digits in lowercase. (size: 8 bytes)
- It is the id of the caller.
- All zeroes forbidden.
- In some tracing system it may also be known as span_id.
- Every participating system, component or step in your request processing can be a parent/span thus their unique parent_id/span_id.
- These components should update the parent_id/span_id before propagating the traceparent header to another.

trace_flags

- 2 hexadecimal digits in lowercase. (size: 8 bits)
- The current version specification (00) only supports a single flag called sampled .
- sampled flag indicates whether the tracing system should record and persist the trace data.
- 0x01 (00000001) caller has recorded data, 0x00 (00000000) caller did not record the data.
- flags such as (Sampling rate, Error flag, Debug flag) are also used depending on the context and tracing system in use.

Example

- 00-4bf92f3577b34da6a3ce929d0e0e4736-00f067aa0ba902b7-01
  - base16(version) = 00
  - base16(trace-id) = 4bf92f3577b34da6a3ce929d0e0e4736
  - base16(parent-id) = 00f067aa0ba902b7
  - base16(trace-flags) = 01 // sampled
- 00-4bf92f3577b34da6a3ce929d0e0e4736-00f067aa0ba902b7-00
  - base16(version) = 00
  - base16(trace-id) = 4bf92f3577b34da6a3ce929d0e0e4736
  - base16(parent-id) = 00f067aa0ba902b7
  - base16(trace-flags) = 00 // not sampled

Tracestate header (Optional)

- **Provides additional vendor-specific trace identification information across different distributed tracing systems.**
- **It is a list of list-members separated by commas (,).**
- **list-member is a key value pair separated by equals sign (=).**
- **A list can have maximum 32 list-members.**
- **Examples:**

tracestate: rojo=00f067aa0ba902b7 (single tracing system: rojo)

tracestate: rojo=00f067aa0ba902b7,congo=t61rcWkgMzE (multiple tracing system: rojo, congo)

Demo

Tracing Vender OpenTelemetry (https://opentelemetry.io/docs/)

Trace Aggregation (Otel-collector, Zipkin, Jaeger)

Trace visualization (Zipkin, Jaeger)

Trace Sampling (Otel-collector)

References:

- https://www.w3.org/TR/trace-context-1
- https://opentelemetry.io/docs/
- https://www.jaegertracing.io/
- https://zipkin.io/
- https://github.com/open-telemetry/opentelemetry.io/blob/main/content/en/docs/collector/_index.md