ORACLE

Service Communication Patterns

Prasenjit Sarkar
Oracle Cloud Infrastructure
February 2020

Safe harbor statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Objectives

After completing this lesson, you should be able to;

- Understand the Service Communication Patterns
- Protocols and their benefits in the communication pattern

ORACLE

Service Communication

Service Communication

External communication (North-South)

Communication from/to external services

Internal communication (East-West)

Service-to-service communication (e.g. within a cluster)

ORACLE

Synchronous and Asynchronous

Integrating services

- Minimize the communication between internal services
- Try not to depend on sync communication
- Use async between services (propagate data asynchronously)
- Orchestration vs. choreography

Protocols (1/2)

HTTP

- Textual protocol
- Most popular, not the most performant

HTTP/2

- Binary protocol instead of textual.
- Designed for low latency
- It is fully multiplexed
- More client data transfer on the wire

Protocols (2/2)

WebSockets

- Persistent connection between client/server
- Based on HTTP
- Low-latency, for transferring large volumes of data

gRPC

- Binary format, small payloads
- Uses HTTP/2 as transport protocol
- Uses protocol buffers to define & serialize structured data into binary format

Messaging Protocols

Message Queue Telemetry Transport (MQTT)

- Simple and lightweight binary protocol
- Designed for low-bandwidth/high-latency environments (e.g. dial-up lines, embedded systems)
- Focuses on Pub/Sub messaging

Advanced Message Queuing Protocol (AMQP)

- Binary protocol with rich set of features
- Reliable queuing, topic-base pub/sub, routing, security
- Battle-tested and proven to be reliable

both use WebSockets over TCP

Publisher/Subscriber - Considerations

Message order is not guaranteed (default)

- Design for idempotent operations

If ordering is needed:

- Use messaging systems ordering functionality
- Priority queue pattern

Use poison message queue (for errors/crashes)

Service Communication - Idempotency (1/2)

Run an operation multiple times, without changing the result

Messages can be received and processed more than once

- Retry policies, failures etc.

Two approaches:

- Exactly-once approach is hard
- Use at-least-once approach

Service Communication - Idempotency (2/2)

Natural idempotency

- No need to do anything special

Not naturally idempotent

- Add unique identifier to the message
- Service checks if the message was processed or not

Service Communication - Serialization

JSON

- Readable, self-contained
- Large memory footprint
- Expensive serialization/deserialization with a lot of data

Protobuf

- Binary format - needs a generator
- Schema defined in .proto files

ORACLE

Oracle Cloud always free tier:
oracle.com/cloud/free/

OCI training and certification:
cloud.oracle.com/en_US/iaas/training
cloud.oracle.com/en_US/iaas/training/certification
education.oracle.com/oracle-certification-path/pFamily_647

OCI hands-on labs:
ocitraining.qloudable.com/provider/oracle

Oracle learning library videos on YouTube:
youtube.com/user/OracleLearning

Thank you