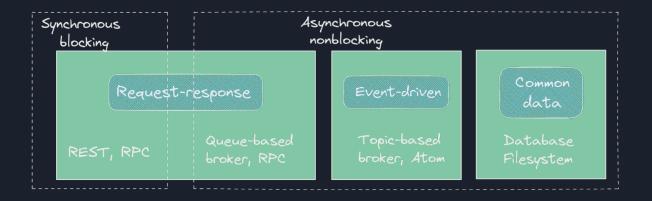


Motivation

Microservice Communication Styles



REST (Representational State Transfer)

Overview

Rest API Basics

HTTP GET /allUsers Rest API Recieves HTTP HTTP POST ENTS requests from /newUser Clients and does whatever request HTTP needs. i.e create PATCH users /updateUser_

Our Clients, send HTTP Requests and wait for responses

Typical HTTP Verbs:

GET -> Read from Database PUT -> Update/Replace row in Database PATCH -> Update/Modify row in Database POST -> Create a new record in the database DELETE -> Delete from the database

Database



Our Rest API queries the database for what it needs

Response: When the Rest API has what it needs, it sends back a response to the clients. This would typically be in JSON or XML format.

source: https://tutorialedge.net/software-eng/what-is-a-rest-api/

REST (Representational State Transfer)

Design Principles

Based on Protocol HTTP/1.1

Payload mostly **JSON**

Code generation **Swagger**

API contract optional

response format between an API and the client

Synchronous communication between client and server

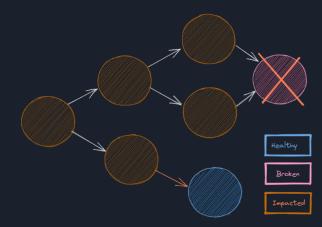
Advantages

- Simple to understand and implement
- Human readable payload
- Easy debuggable E.g. with curl or Postman
- Perfect fit for serverless (pay on execute)
- Ideal for public APIs

REST (Representational State Transfer)

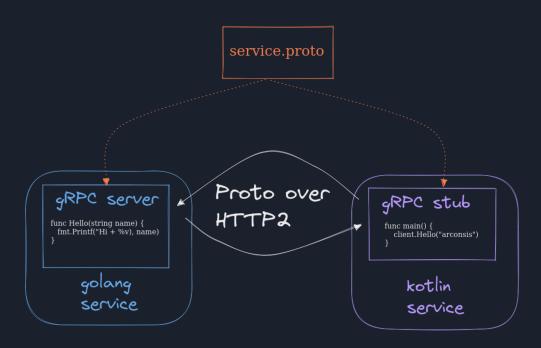
Disadvantages

- Payload can be a bottleneck in low latency or high throughput networks
- Does not support code generation for clients out of the box
- Unary & Synchronous by default



gRPC (gRPC Remote Procedure Calls)

Overview



gRPC

Design Principles

"gRPC is a modern, open source remote procedure call (RPC) framework that can run anywhere. It enables client and server applications to communicate transparently, and makes it easier to build connected systems."

Based on Protocol HTTP/2.0

Payload Agnostic but mostly **Protobuf**

Code generation build-in

API contract mandatory

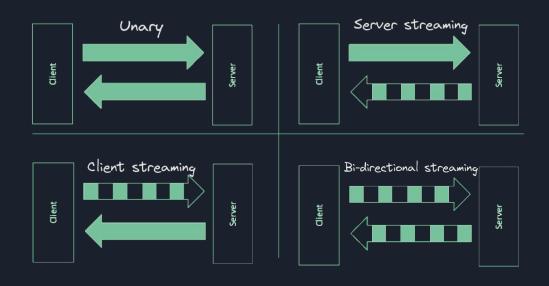
• Request and response format defined using Protocol Buffer

Blocking & Non-Blocking

 Support both asynchronous and synchronous processing of the sequence of messages exchanged by a client and server

gRPC

Streaming



gRPC

Advantages

- High performance due to compact message payload
- Streaming
- Code Generation
- Interoperability

Disadvantages

- Limited Browser Support
- Payload not human readable
- Harder to debug
- Steeper learning curve



When to use gRPC instead of REST?

- microservice-to-microservice communication
- When efficient communication is a goal (IoT)
- In multi-language environments
- Real-time communication services where you deal with streaming calls