Case Study

Airbnb

- Analytical architecture
    - Cost implications
- Corelation
- Actor , LMAX

Web Portal

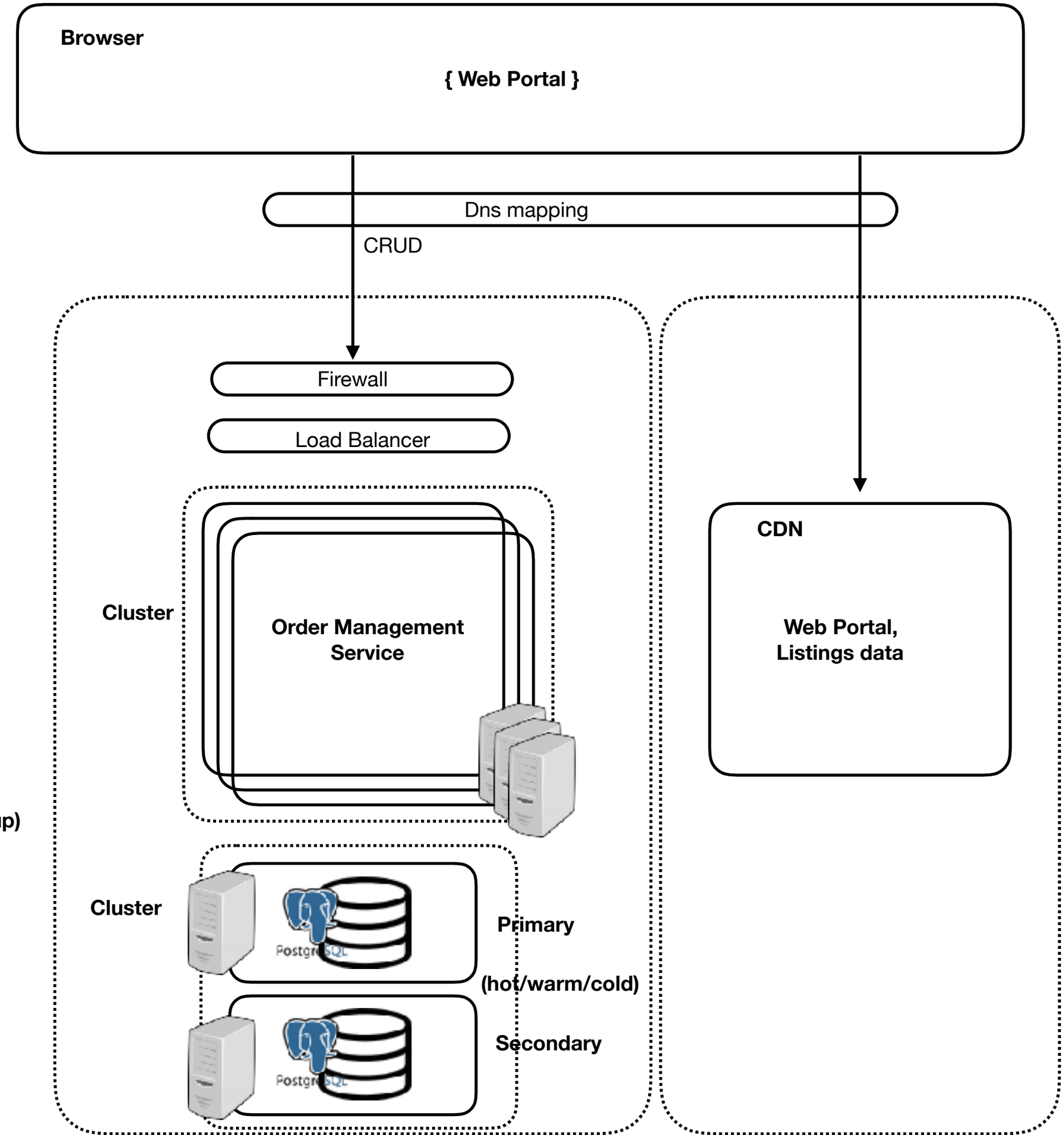
CRUD

Order Management Service

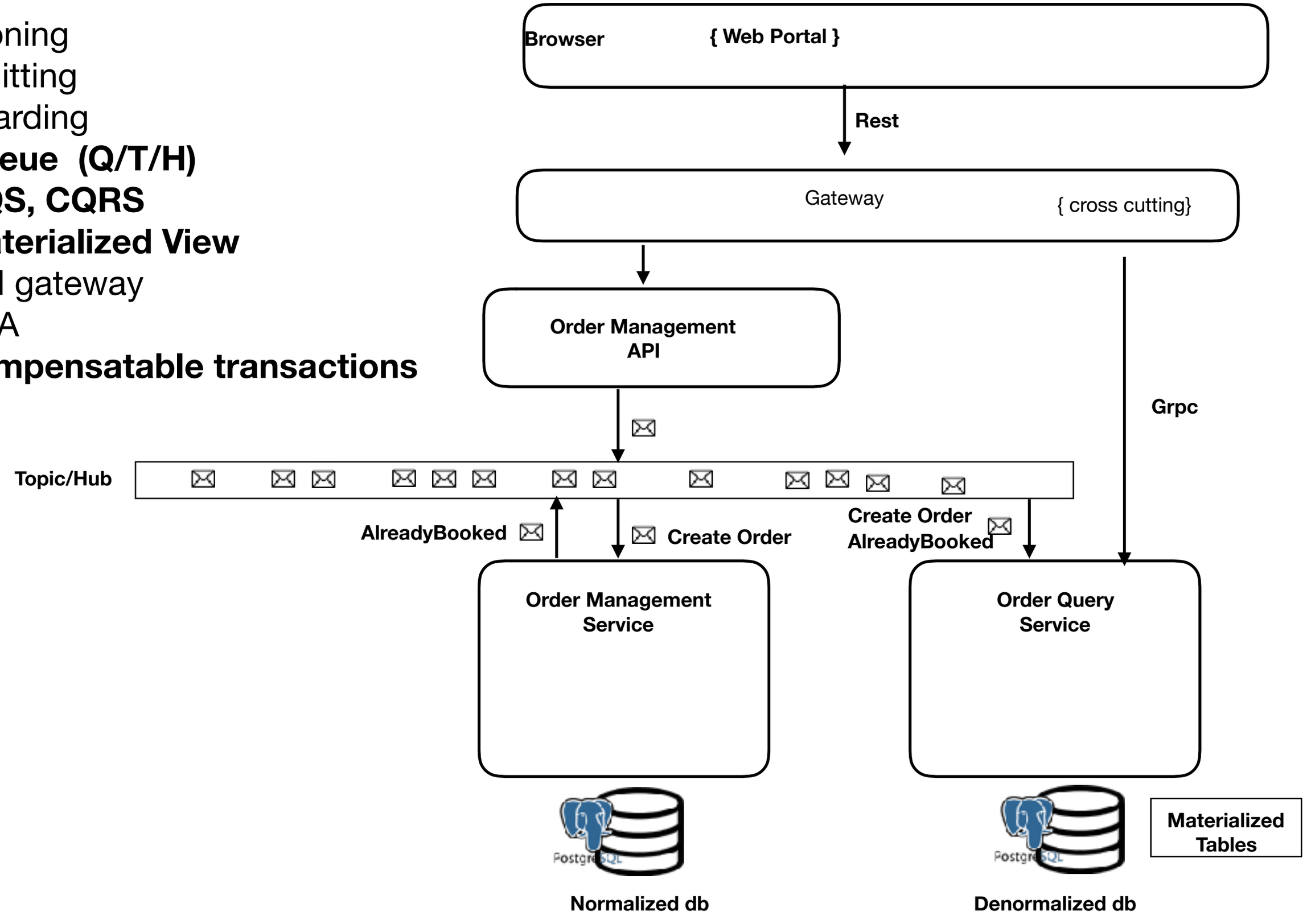
Cloning

@ Read heavy

* database will become the bottleneck
* Handling load spikes will be hard
* Database scaling issues (only scale up)
*



Cloning
Splitting
Sharding
 # Queue (Q/T/H)
 # CQS, CQRS
 # Materialized View
API gateway
EDA
 # compensatable transactions



CRUD

Order Management Service

API

Domain Layer

Data layer

CUD

R

API

Write (transaction)

Command (DOmain) Layer

Read

Query Layer

Data layer

CUD

R

API

Write
(transaction)

Command
(DOmain)
Layer

Read

Query Layer

Data layer

CUD

Write
(transaction)

API

Facade

Domain

Data Layer

CRUD

R

Read

API

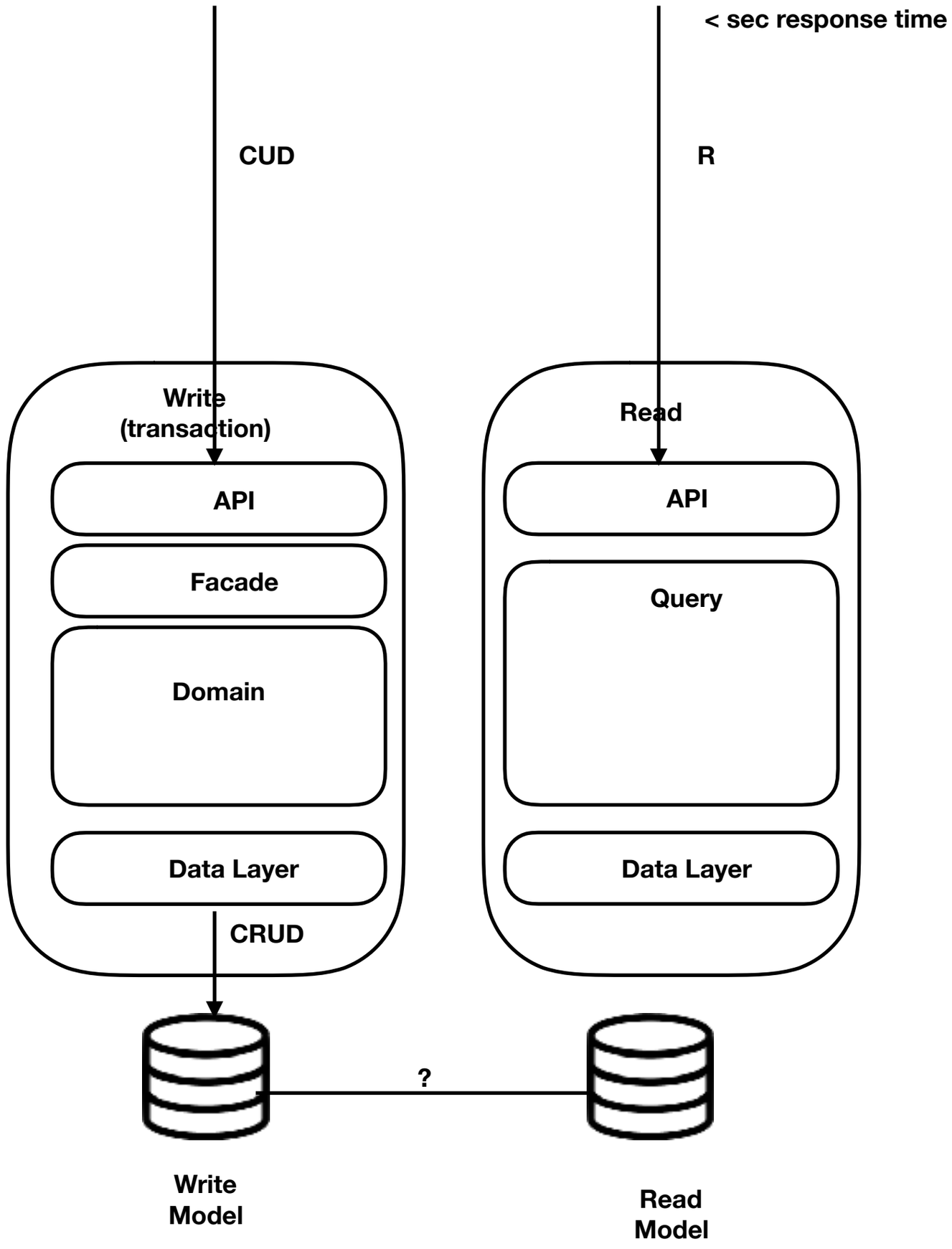
Query

Data Layer

?

Write
Model

Read
Model


< sec response time
CUD
R
Write
(transaction)
API
Facade
Domain
Data Layer
CRUD
Read
API
Query
Data Layer
?
Write
Model
Read
Model

Denormalized form

ID	NAME	SUBJECT	STATE	COUNTRY
29	Lalita	English	Gujrat	INDIA
33	Ramesh	English	Punjab	INDIA
49	Sarita	Mathematics	Gujrat	INDIA
78	Zayed	Mathematics	Punjab	INDIA

3rd form

ID	NAME	SUBJECT ID	STATE ID	COUNTRY ID
29	Lalita	1	1	1
33	Ramesh	1	2	1
49	Sarita	2	1	1
78	Zayed	2	2	1

Metadata

- Zookeepe
- etcd
- Consul

Text Database

- AWS **Elastic Search**
- AWS cloud Search
- Azure Cognitive Search
- Google Search API
- Elastic Search, **Solr**

[Unstructured]

[Analytical Reporting/ datawarehouse/ Manage business]

- AWS Redshit (max: 8 PB)
- **Azure Synapse**
- GCP BigQuery
- IBM Cognos Analytics
- Snow flakes
- Kudu, Druid, Pinot
- Click house
- Greenplum *
- Kylin

Columnar db (column database) Dimensional modeling

NewSql

- duckdb
-

[Multiple keys]

[Operational/ Run business]

[single Key]

- AWS Dynamo Db
- AWS Keyspaces (unlimited)
- Cosmos
- Google Big Table
- **Hbase, cassandra**
- Scylla *

[Single Value]

[Multiple values]

2-dimensional
(row key, cfamily key)

- **Very fast writes**
- **Limited secondary indexes**
- **Limited query patterns**
- **Less frequent updates**
- **Y and z split**
- **Book keeping**
- **Manage business**

Dictionary (key value)

App data

- Memdb
- Azure Cosmos -> Table
- Azure Table
- Google Big Table

- **Very fast read**
- **Slower writes**
- **z split**

[flexi schema]

[ACID, Simple Relationship]

[complex RelationShip]

< 10,000 tps

[strict schema]

Document

- AWS Document DB
- Azure Cosmos -> Document Db
- Firestore
- **Mongodb**, **Couchbase**,Solr

Rdbms

- AWS RDS
- Azure SQL
- Google Cloud Sql
- Postgres
- Mysql
-

NewSql

- Voltdb
- NuoDb
- Cockroach
- Google Cloud Spanner

Graph

- AWS Neptune
- Azure Cosmos -> Graph
- Google Janusgraph
- **Neo4j**
- Graph,orientdb

Wide Column db (column-family)

Time Series

- AWS Timestream
- InfluxDB, Scylla
- OpenTSDB,
- KairosDB,
- Prometheus

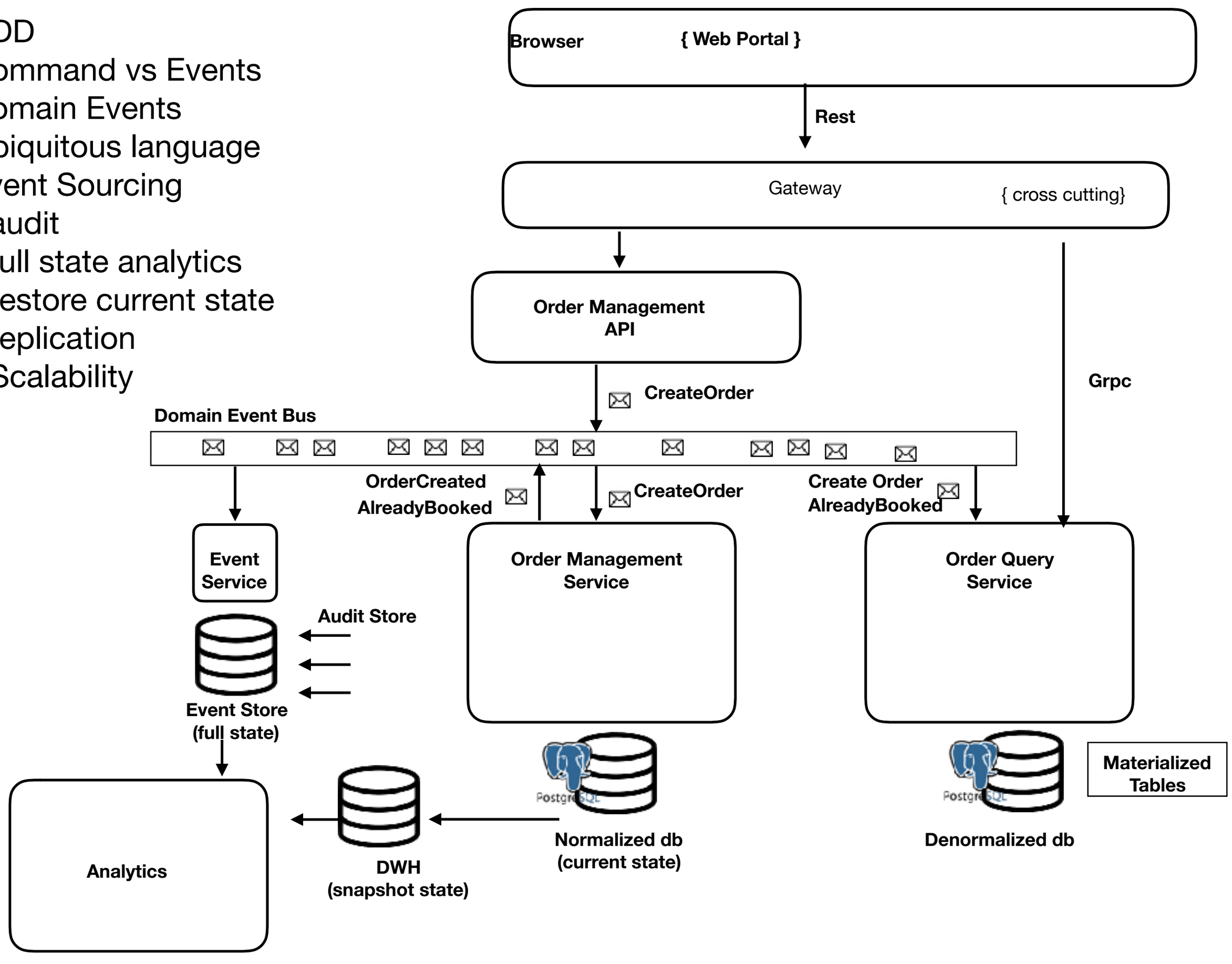
geo Spatial

- AWS Keyspaces
- Azure Cosmos
- Big Table, Big Query
- **Solr**, PostGIS
- Mongoldb (GeoJSON)

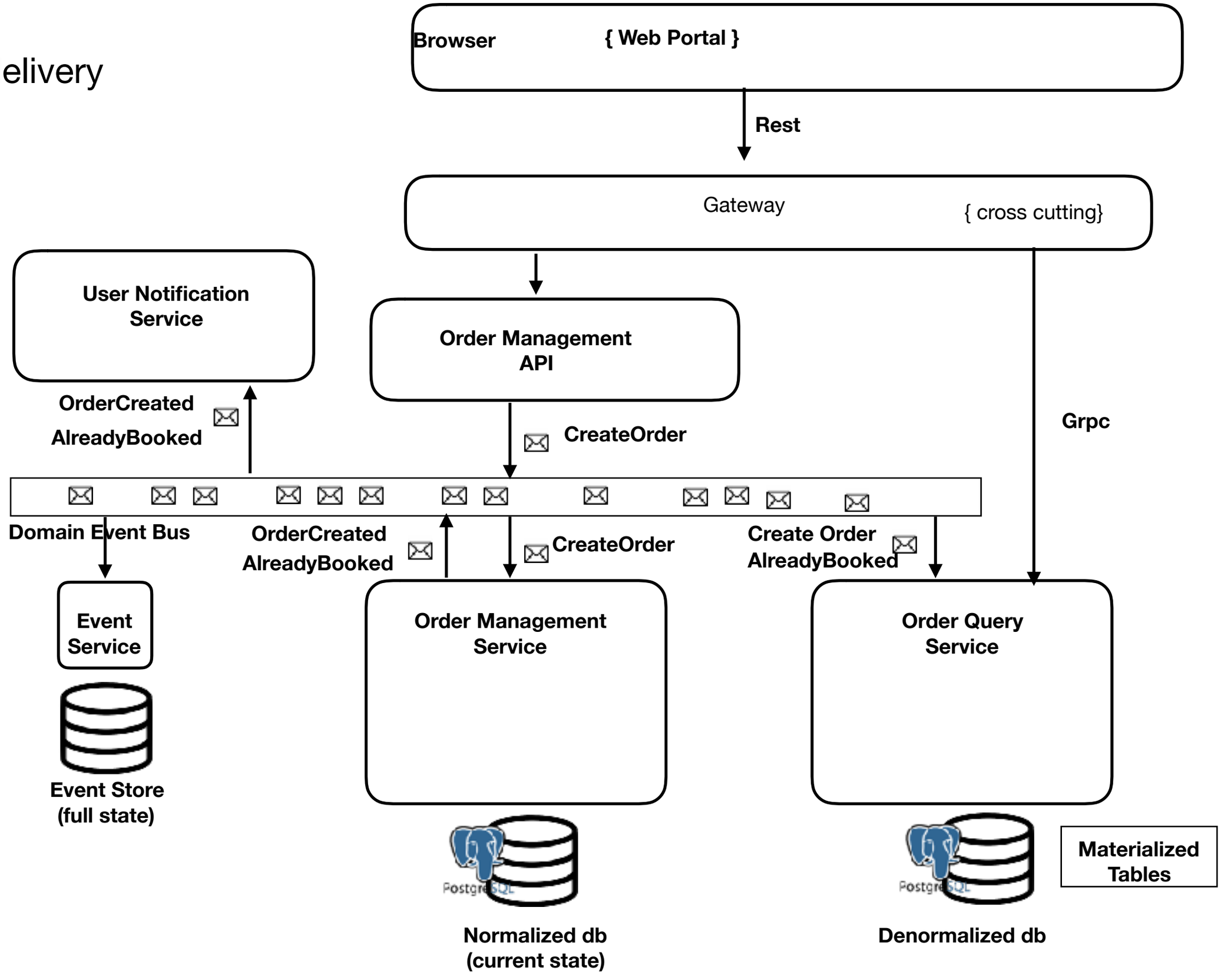
Immutable Logger

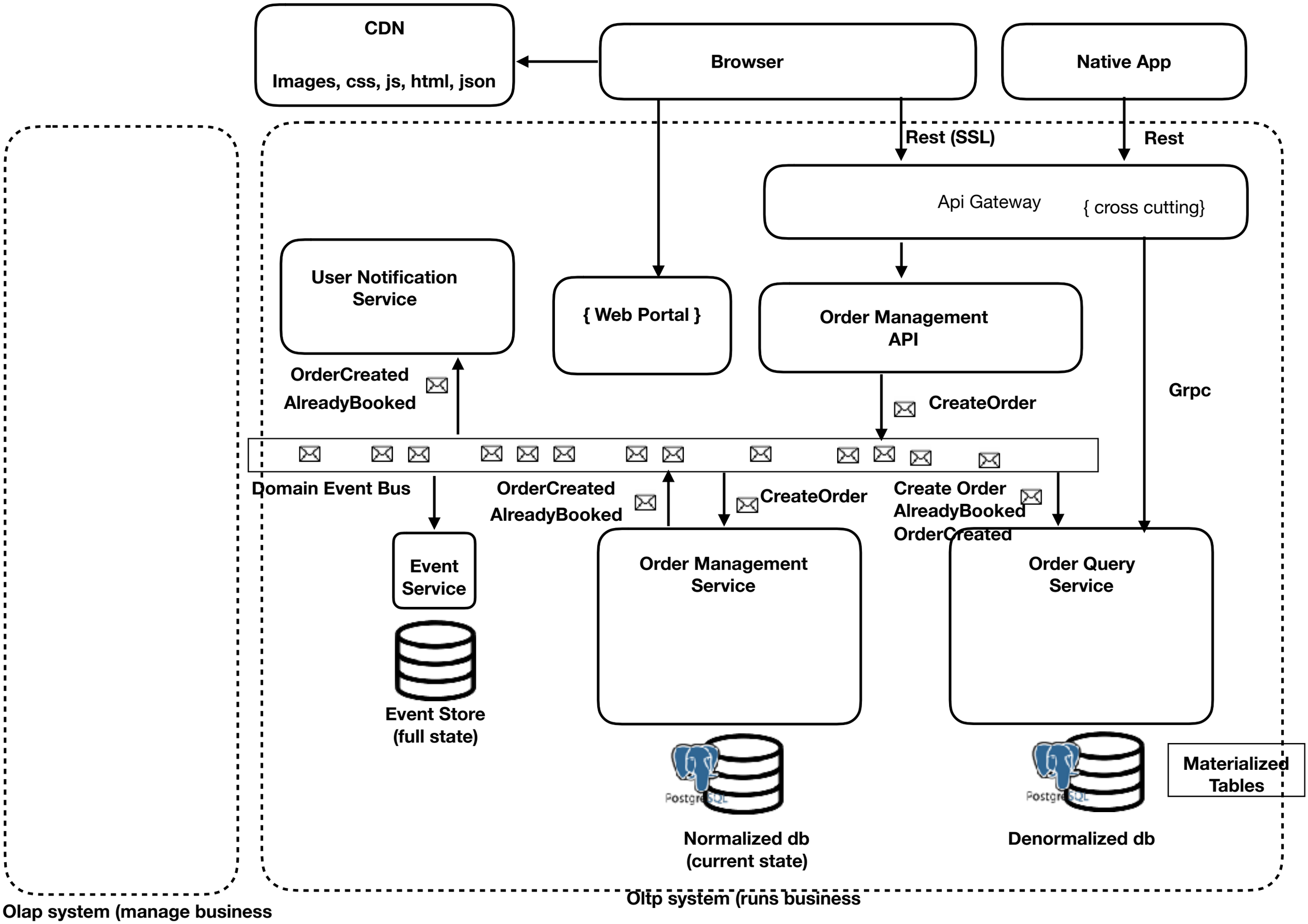
- AWS Quantum Ledger
- Azure SQL Database Ledge
- Google ?
- Hyper ledger Fabric

DDD
Command vs Events
Domain Events
Ubiquitous language
Event Sourcing
  - audit
  - full state analytics
  - restore current state
  - replication
  - Scalability



idempotent
unordered delivery




CDN
Images, css, js, html, json
Browser
Native App
Rest (SSL)
Rest
Api Gateway
{ cross cutting}
User Notification Service
{ Web Portal }
Order Management API
OrderCreated
AlreadyBooked
CreateOrder
Grpc
Domain Event Bus
OrderCreated
AlreadyBooked
CreateOrder
Create Order
AlreadyBooked
OrderCreated
Event Service
Order Management Service
Order Query Service
Event Store
(full state)
Normalized db
(current state)
Denormalized db
Materialized Tables
Olap system (manage business
Oltp system (runs business

CDN

Images, css, js, html, json

Browser

Native App

Rest (SSL)

Rest

Order Orchestrator

Api Gateway

{ cross cutting}

{ Web Portal }

Order Management
API

Commands *
Events *

OS Cmds

CreateOrder

Grpc

Domain Event Bus

OrderCreated
AlreadyBooked

OS Cmds

OS Cmds

Event
Service

Order Management
Service

do logic

undo logic

Order Query
Service

do logic

undo logic

Event Store
(full state)

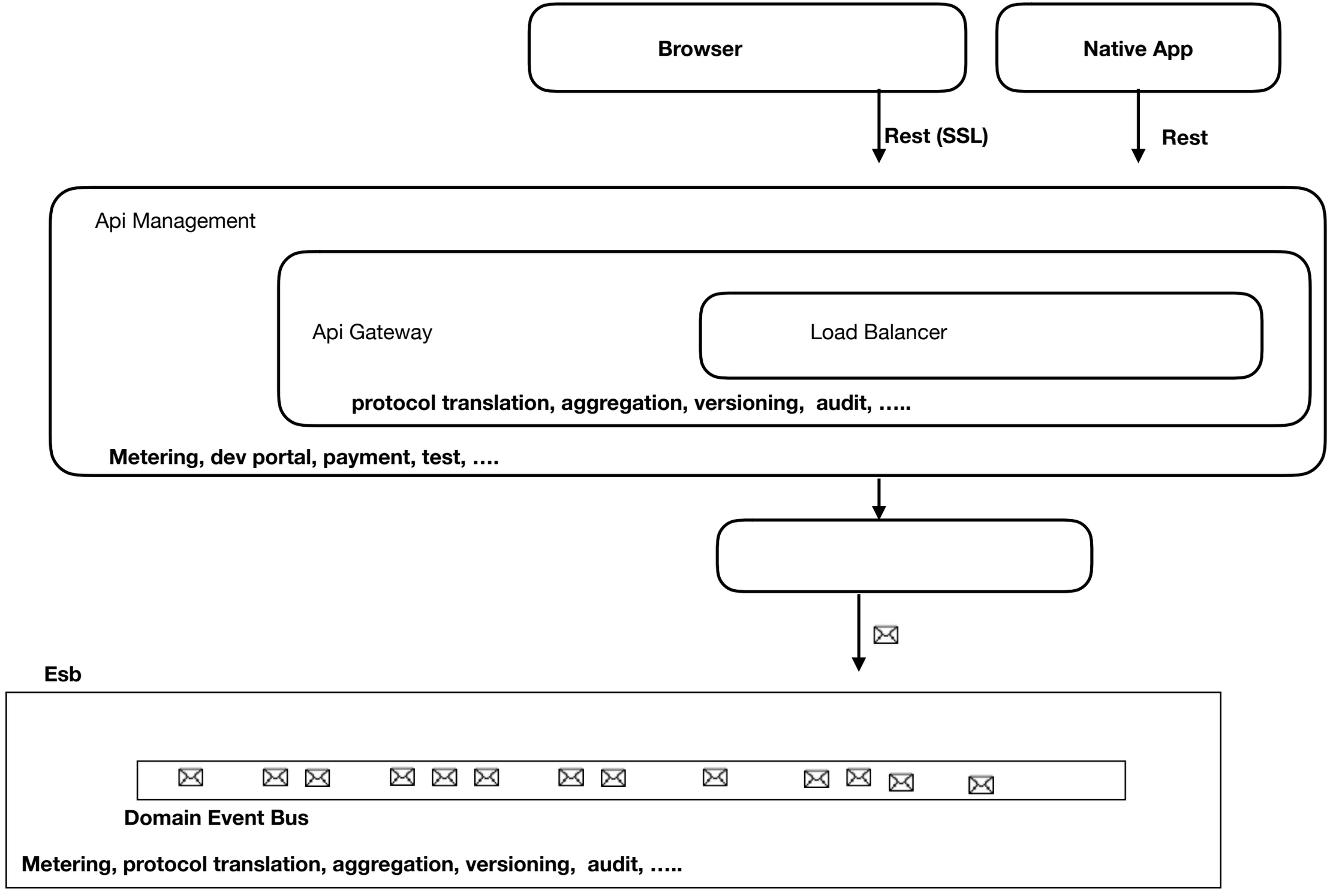
Normalized db
(current state)

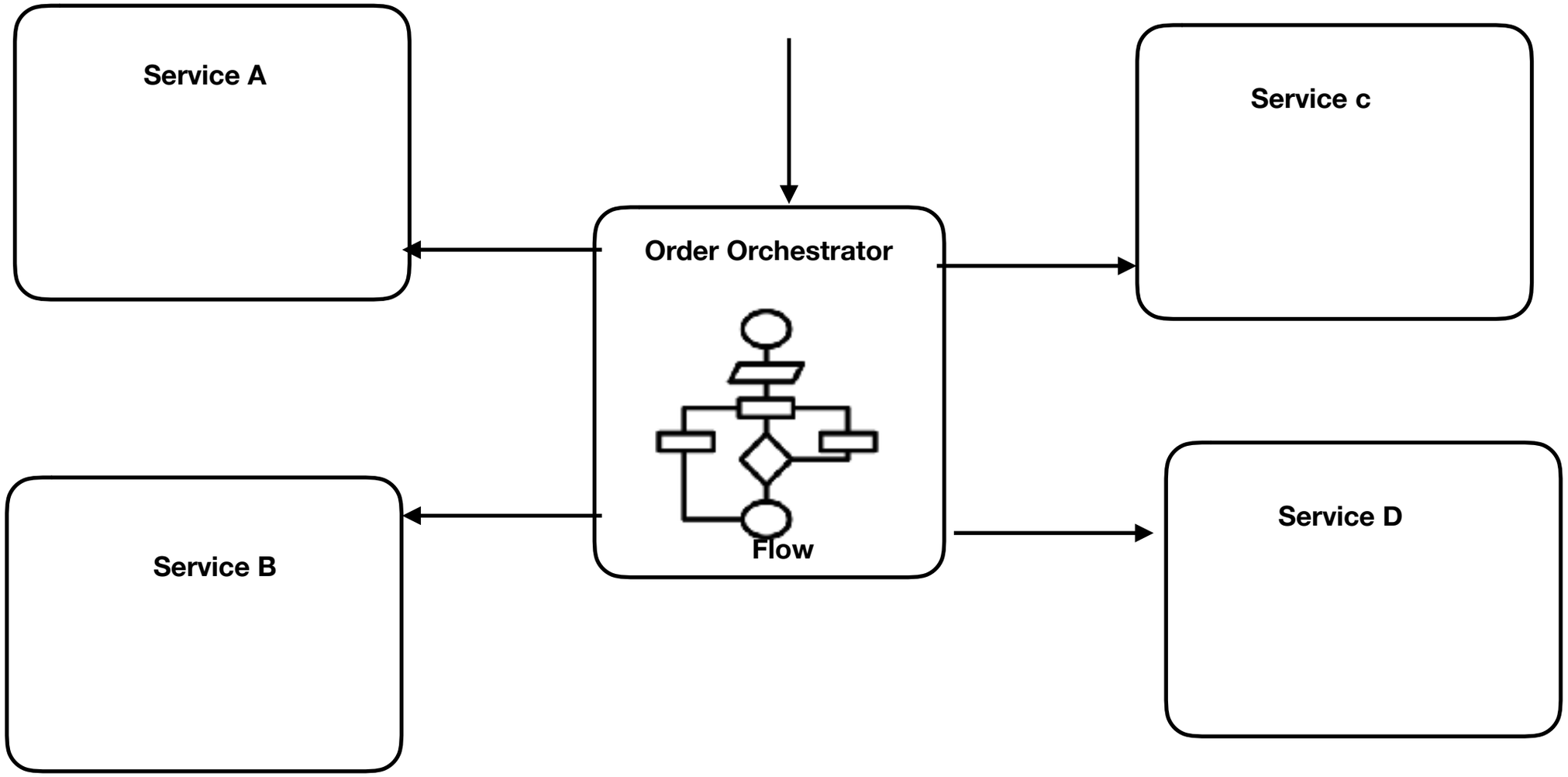
Denormalized db

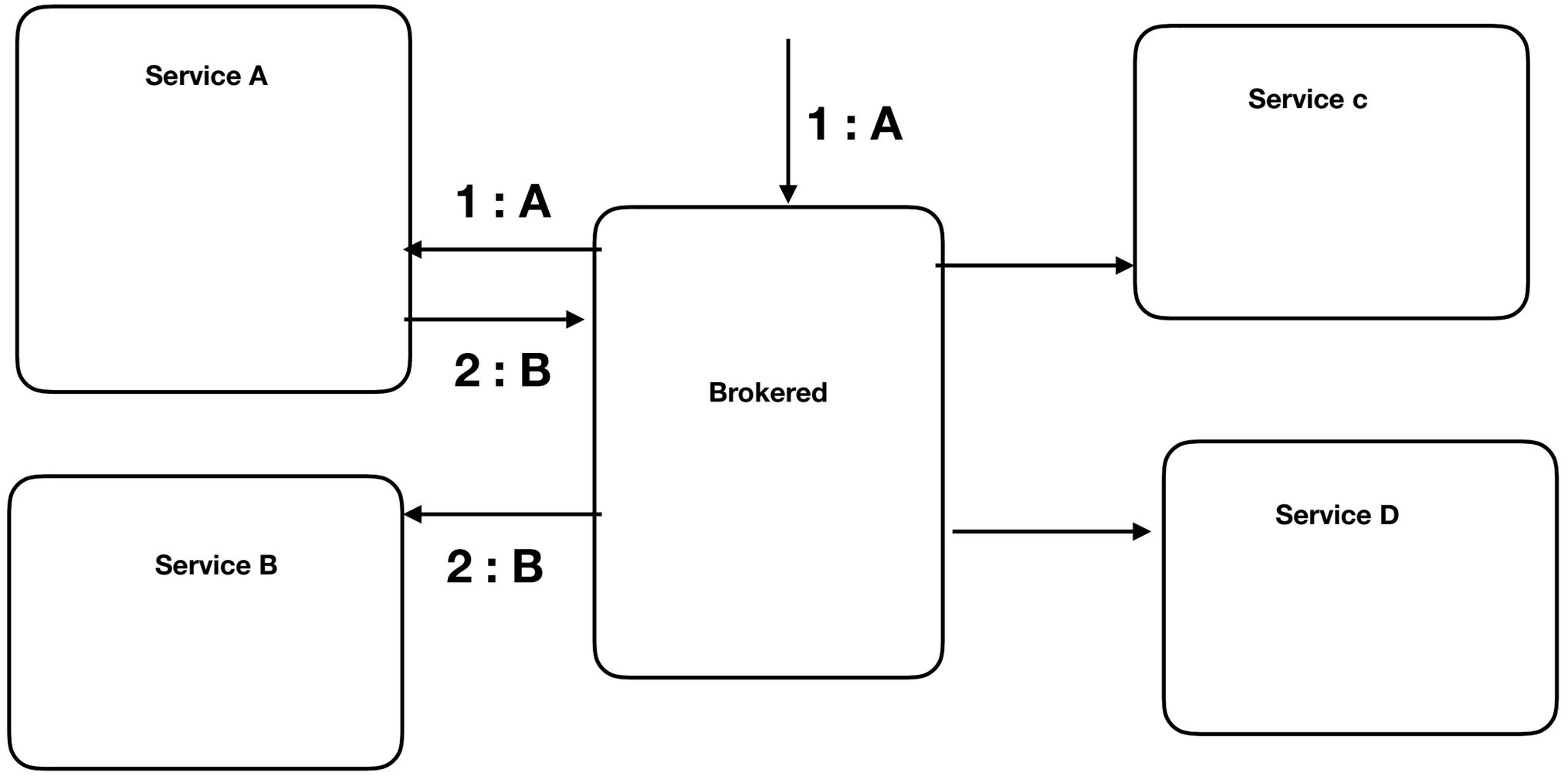
Materialized
Tables

Olap system (manage business

Oltp system (runs business


Browser
Native App
Rest (SSL)
Rest
Api Management
Api Gateway
Load Balancer
protocol translation, aggregation, versioning, audit, …..
Metering, dev portal, payment, test, ….
Esb
Domain Event Bus
Metering, protocol translation, aggregation, versioning, audit, …..


Service A
Order Orchestrator
Flow
Service c
Service B
Service D


Service A
1 : A
1 : A
Service c
2 : B
Brokered
Service D
Service B
2 : B

CDN

Images, css, js, html, json

Browser

Native App

Rest (SSL)

Rest

Api Gateway

{ cross cutting}

User Notification Service

#do Logic

#undo Logic

{ Web Portal }

Order Management API

OrderCreated

AlreadyBooked

ShippingFailed

CreateOrder

Grpc

Domain Event Bus

ShippingFailed

OrderCreated

OrderCreated

AlreadyBooked

OrderCanceled

CreateOrder

ShippingFailed

Create Order

AlreadyBooked

OrderCreated

OrderCanceled

Shipping Service

Event Service

Order Management Service

#do Logic

#undo Logic

Order Query Service

#do Logic

#undo Logic

Event Store (full state)

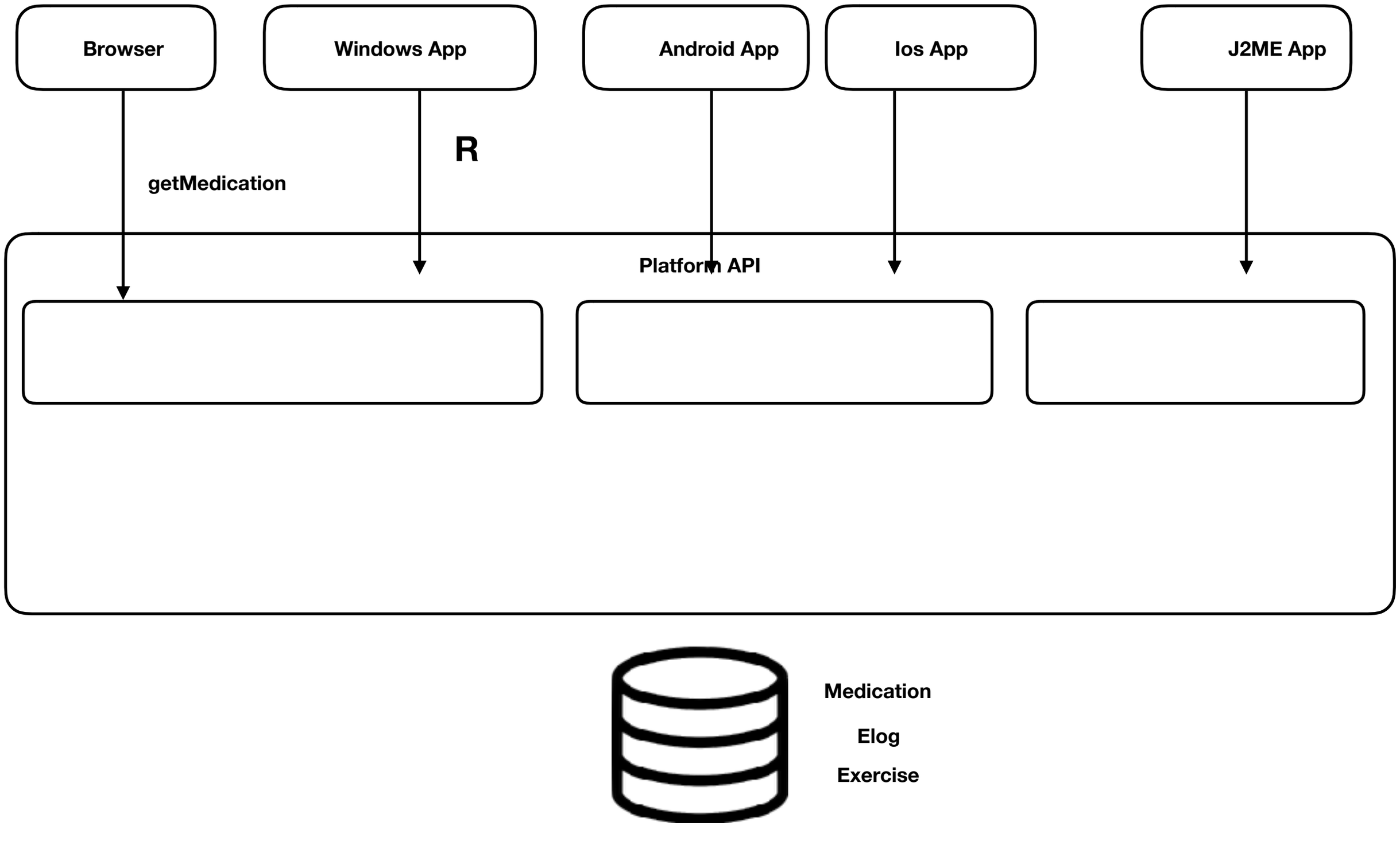
Normalized db (current state)

Denormalized db

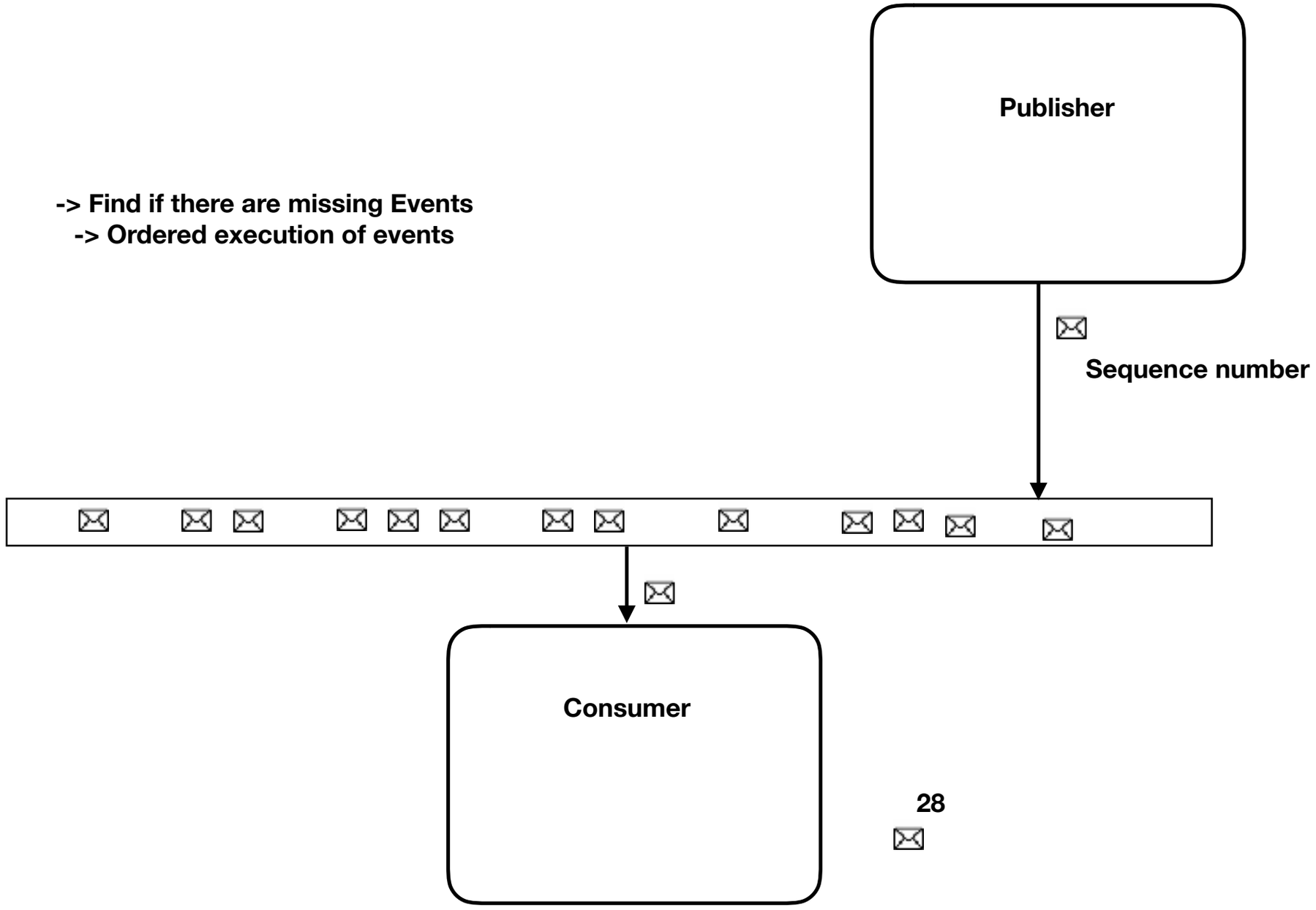
Materialized Tables

Olap system (manage business

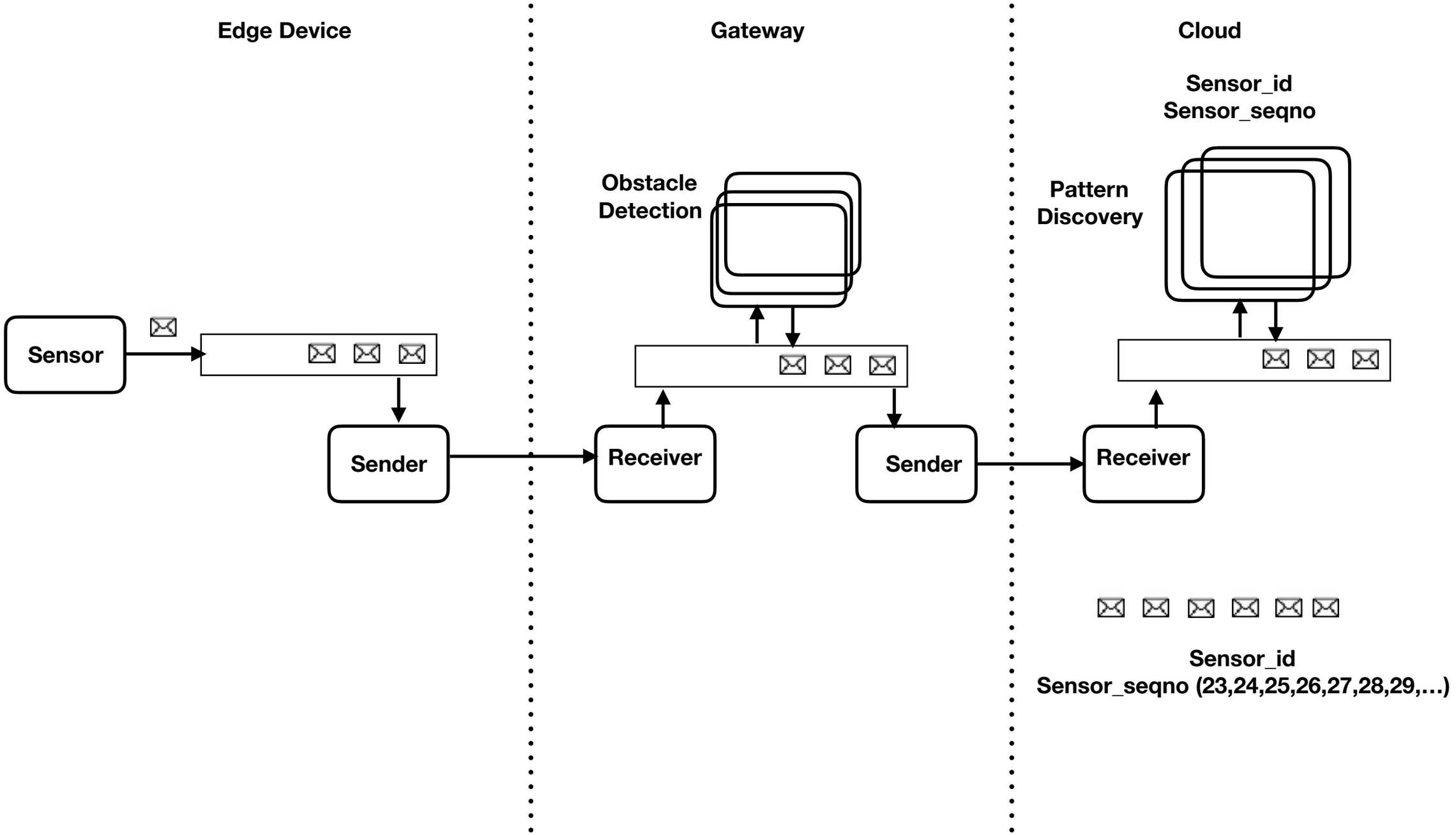
Oltp system (runs business

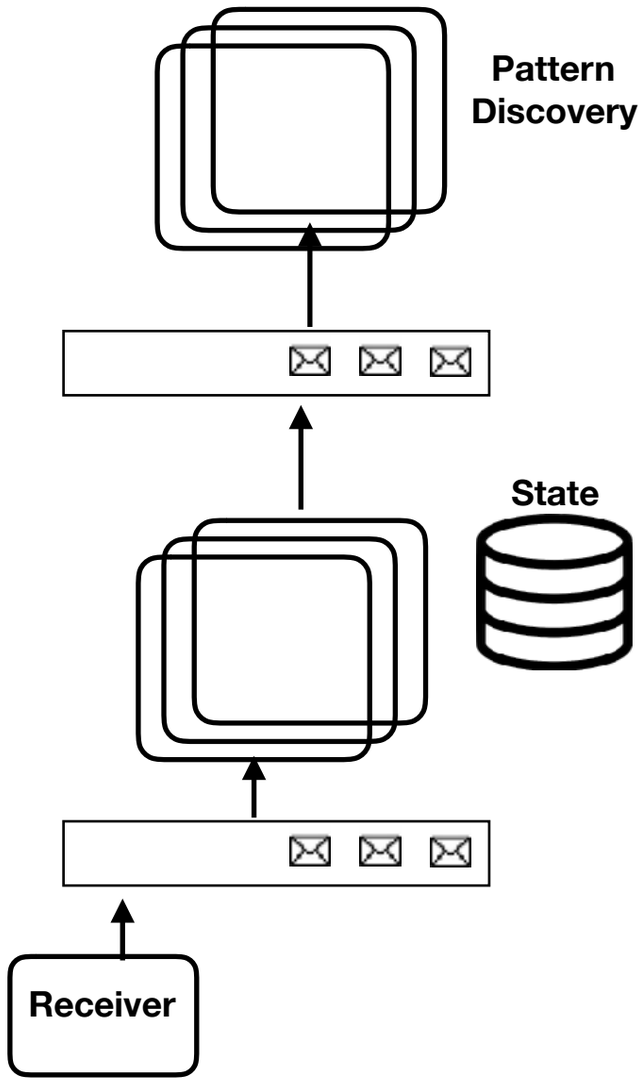

Browser
Windows App
Android App
Ios App
J2ME App
R
getMedication
Platform API
Medication
Elog
Exercise

-> Find if there are missing Events
-> Ordered execution of events

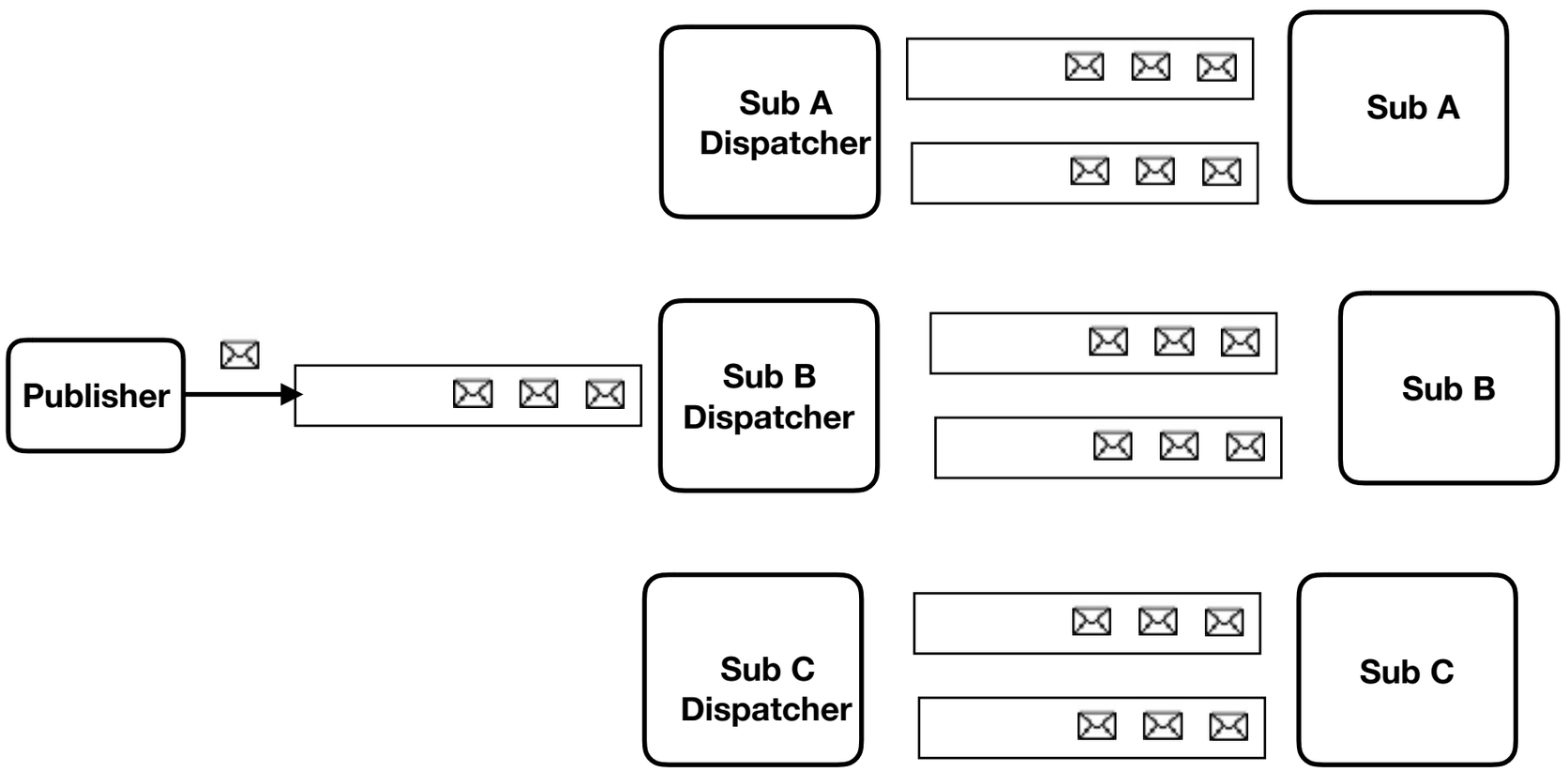


IOT


Edge Device
Gateway
Cloud
Sensor_id
Sensor_seqno
Obstacle
Detection
Pattern
Discovery
Sensor
Sender
Receiver
Sender
Receiver
Sensor_id
Sensor_seqno (23,24,25,26,27,28,29,…)



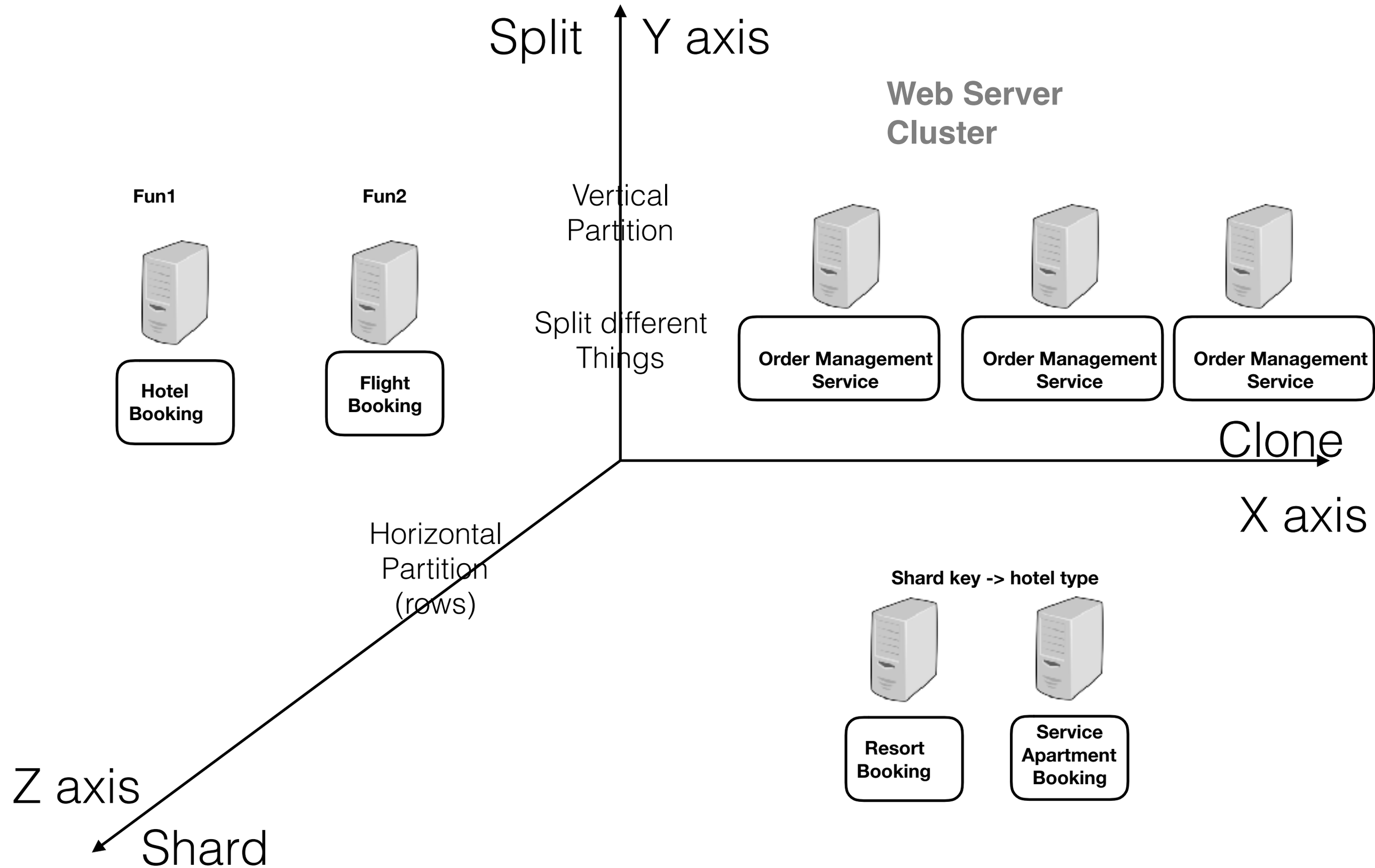
Sensor_id
Sensor_seqno

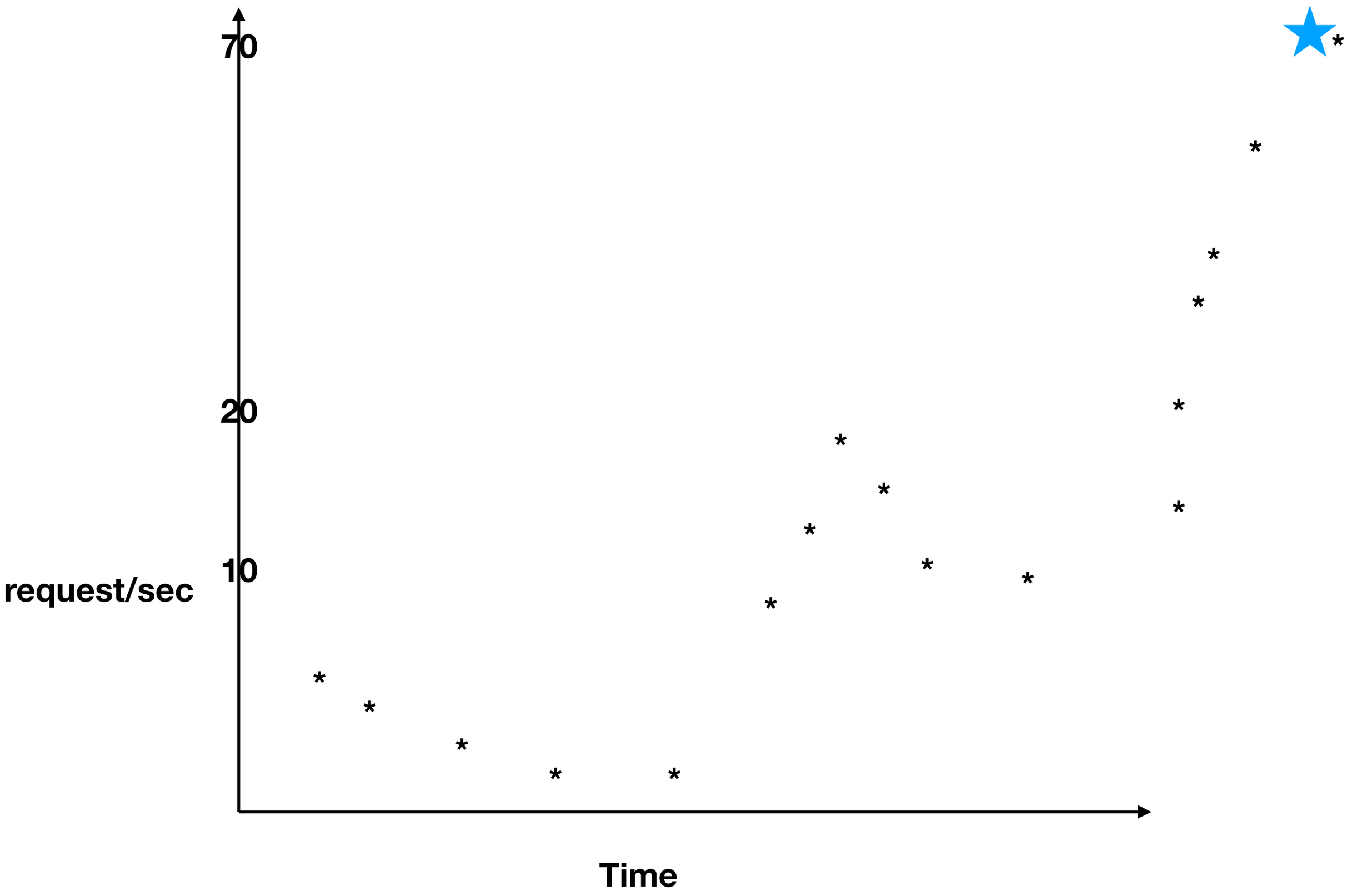

Sub A
Dispatcher
Sub A
Publisher
Sub B
Dispatcher
Sub B
Sub C
Dispatcher
Sub C

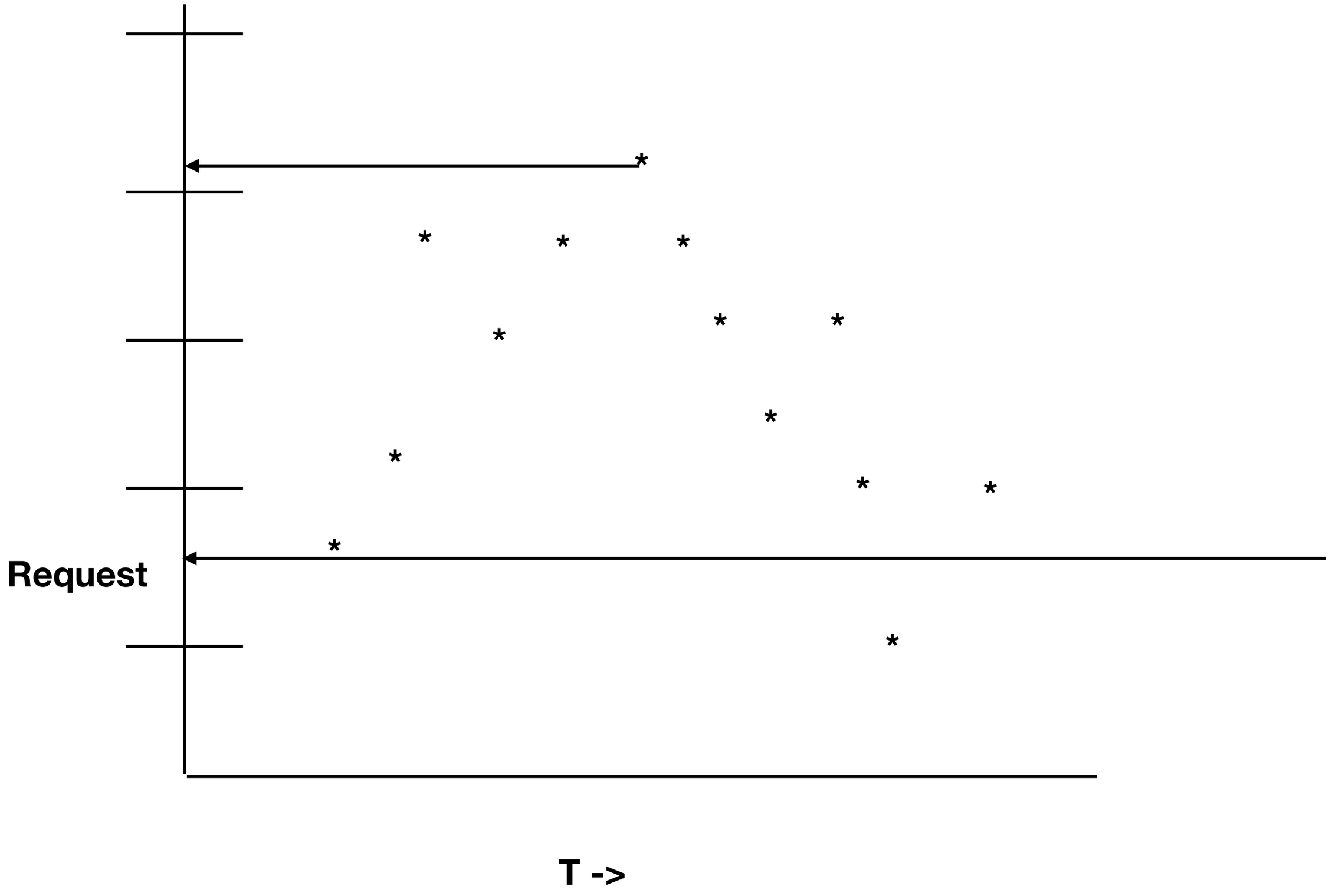
Cost of operation

- A + b -> 3 cpu cycles
- Fun call -> 10 ~
- Throw exception -> 3000 ~
- Create thread -> 200,000 ~
- Destroy thread -> 100,000 ~
- Write File -> 10,00,000 ~
- Write to db -> 40,00,000 ~

Applying cube pattern on Compute

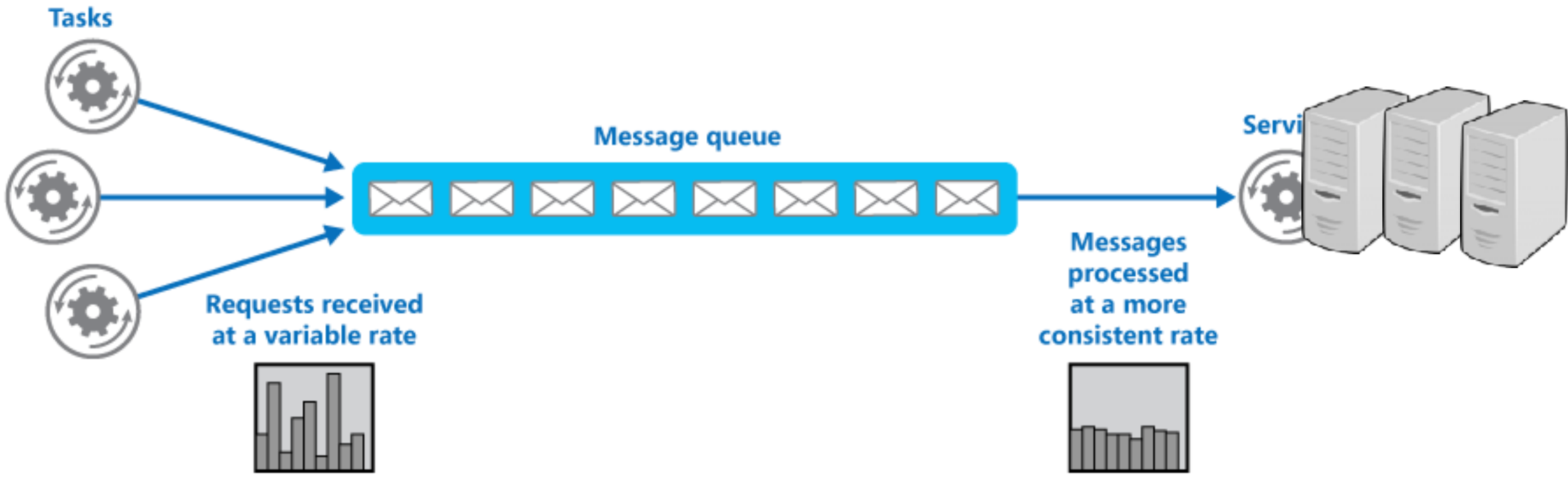



70
20
10
request/sec
Time


Request
T ->

Queue-Based Load Leveling pattern

20 tps * 2

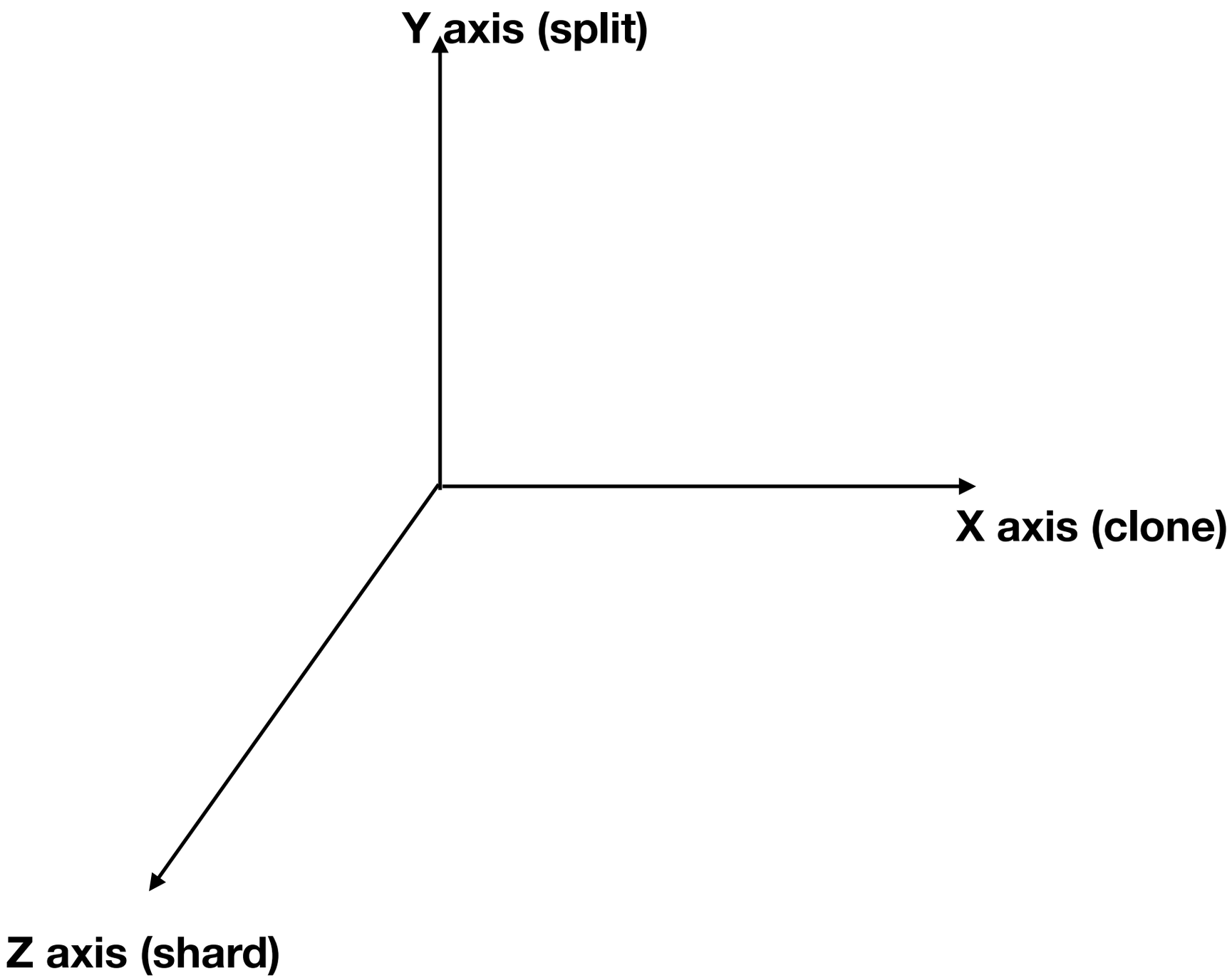


useful to any application that uses services that are subject to overloading.

- Queue
- **Sharding**
- **Auto scaling**
- CQRS - compute
- Horizontal scaling
- Read replicas
- Caching
- Sticky session on LB
- Multi region deployment (geode) { compute + db|}
- nosql ?
- Distributed locks, distributed cache
- Geo dns
- Split services to scale
- Data compression

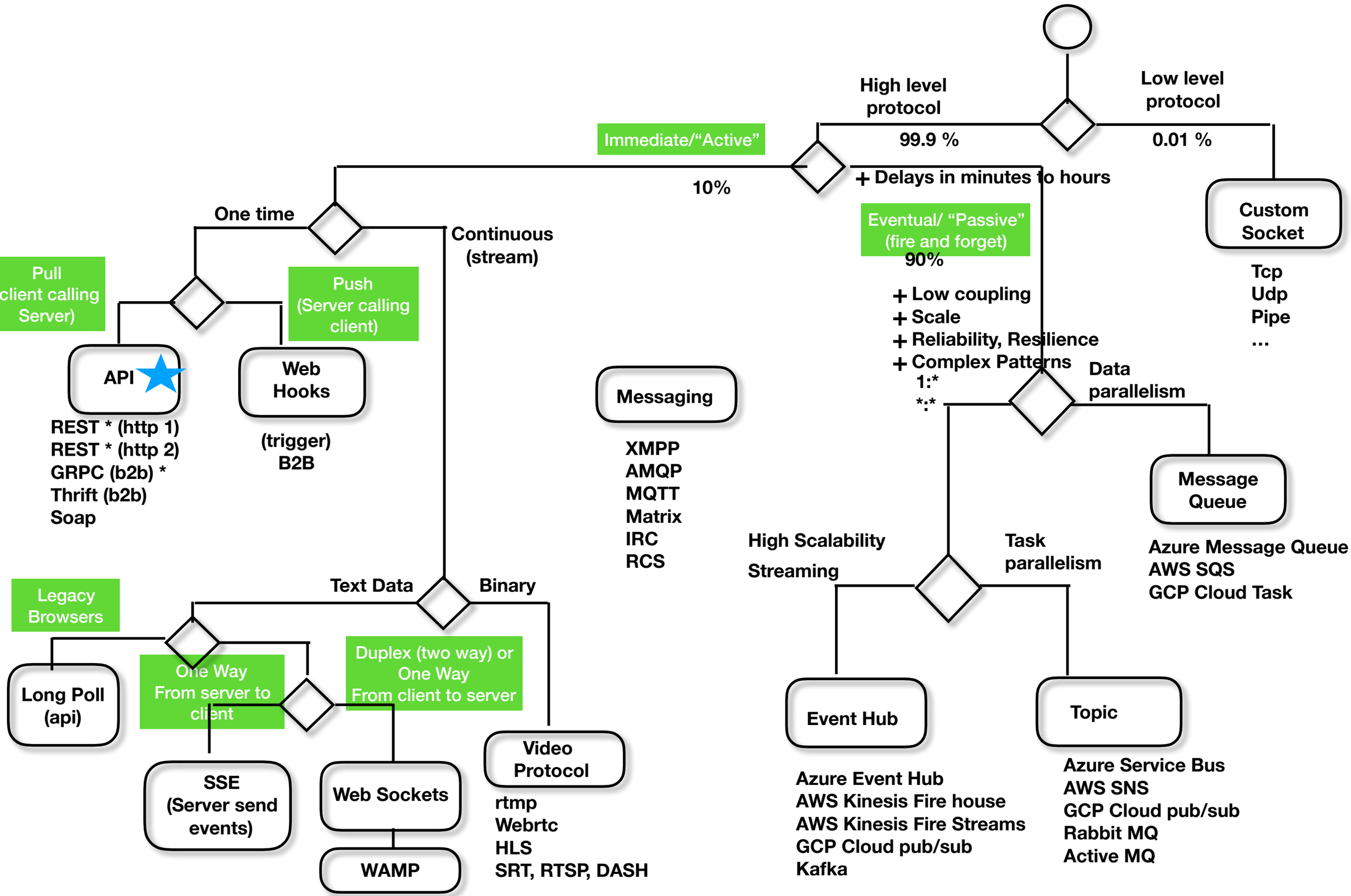
- Last write wins
-

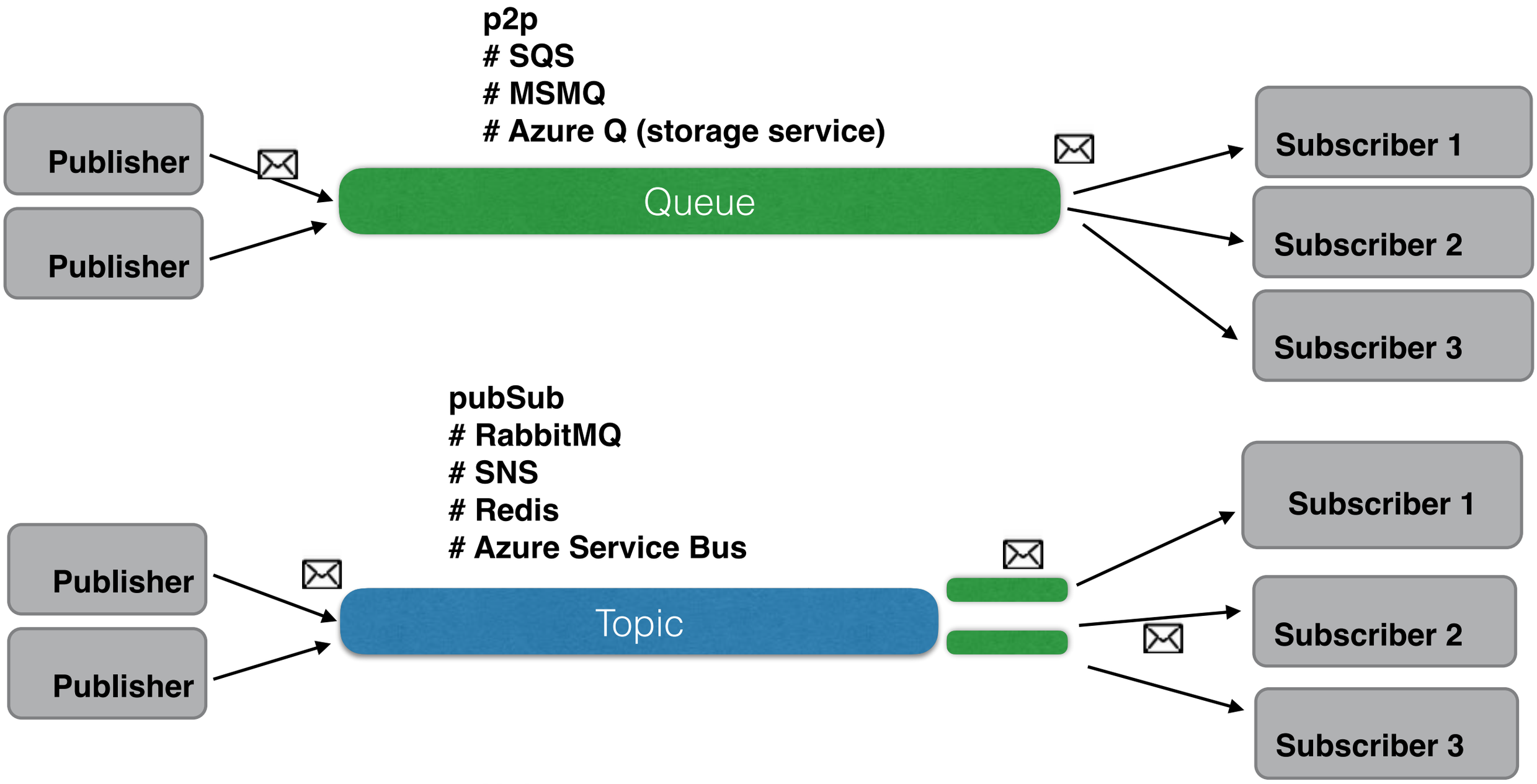
Scalability Cube



Load leveling

Choose Communication



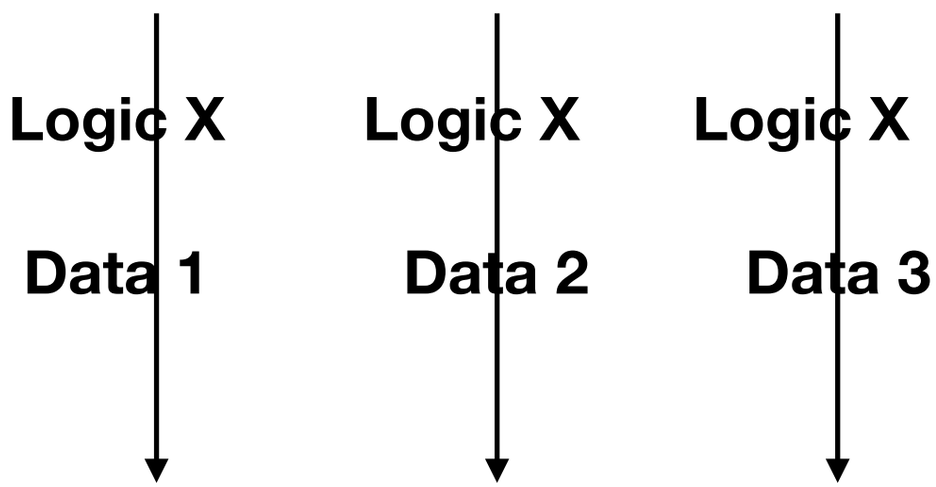

p2p
SQS
MSMQ
Azure Q (storage service)
Publisher
Publisher
Queue
Subscriber 1
Subscriber 2
Subscriber 3
pubSub
RabbitMQ
SNS
Redis
Azure Service Bus
Publisher
Publisher
Topic
Subscriber 1
Subscriber 2
Subscriber 3
Hub
Kafka
Event Hub
Kinesis
Publisher
Consumer
State
p2: time/offset
Consumer
State
P1:time/offset

Parallel

Data Parallelism

Logic X

data 1
data 2
data 3
data 4
...
data n

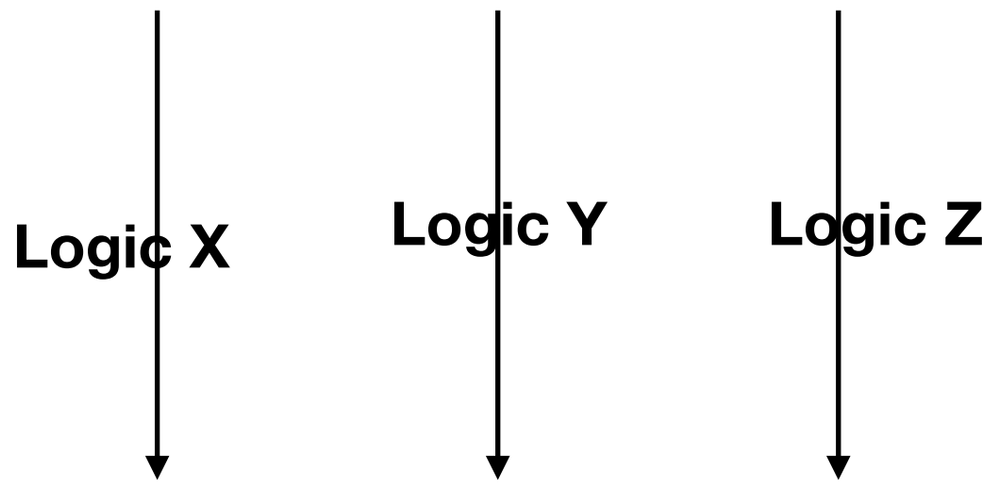


Task Parallelism

Logic Z

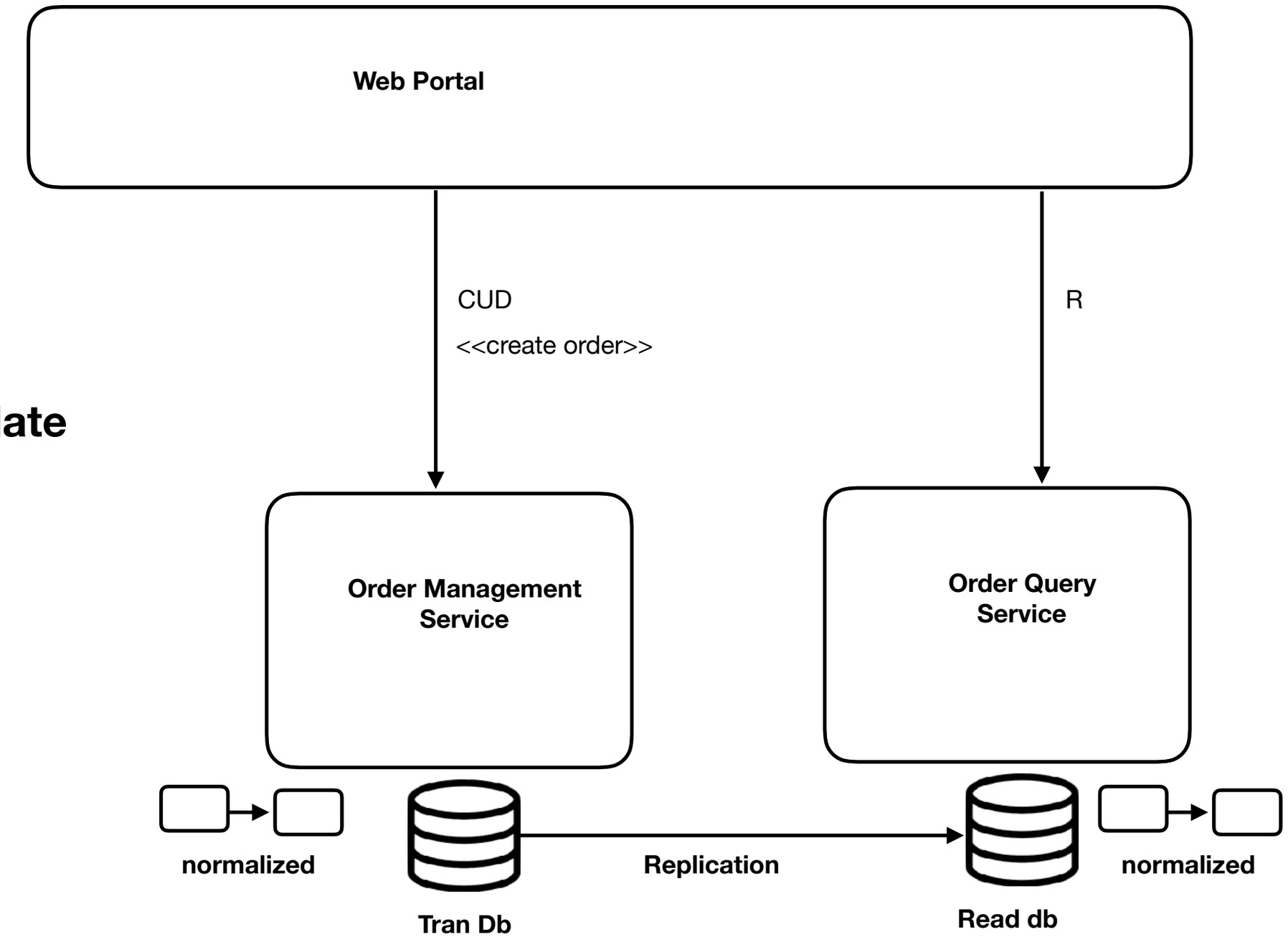
Logic X

Logic Y



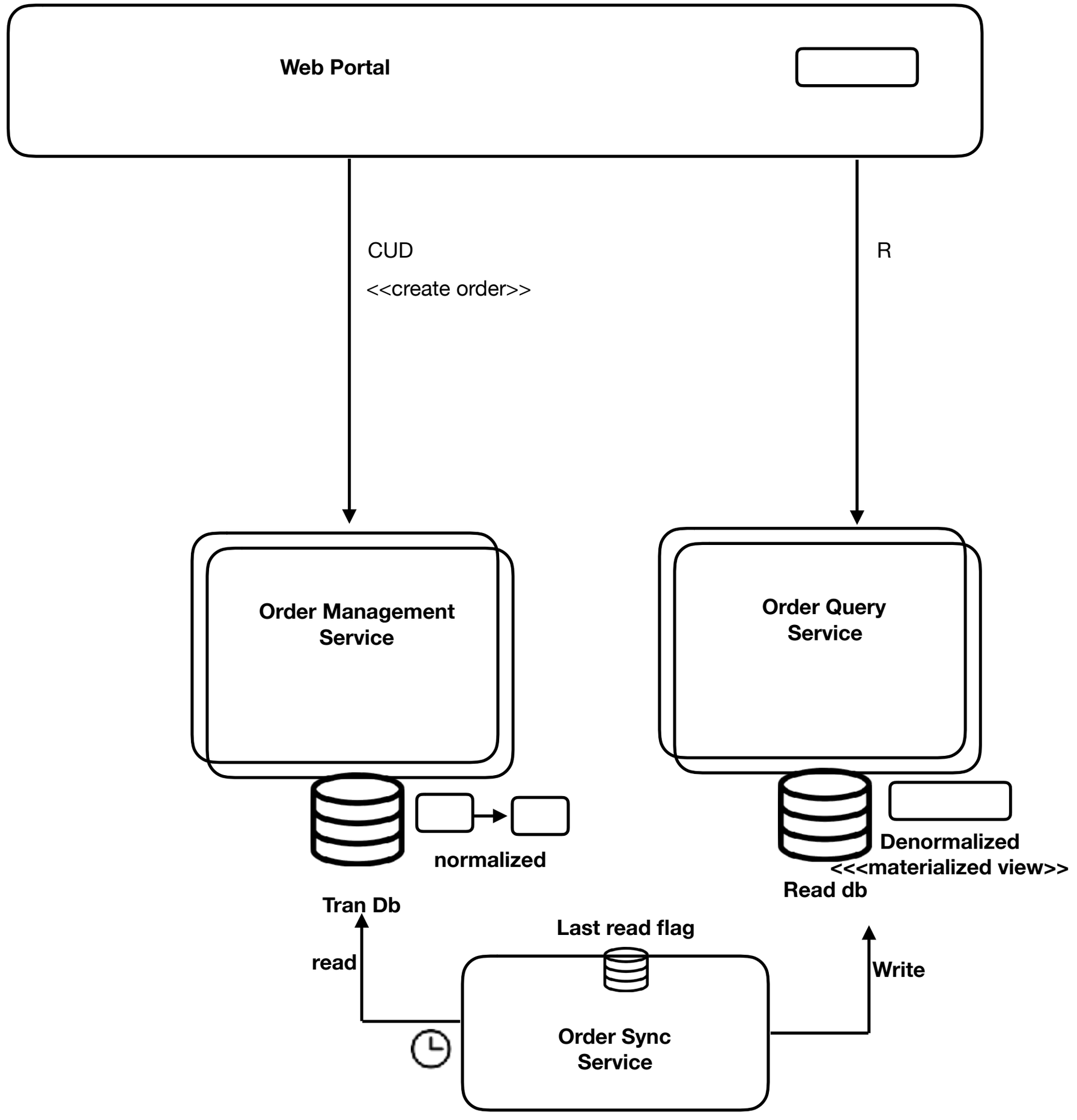
#0 db replication

@ performance of update

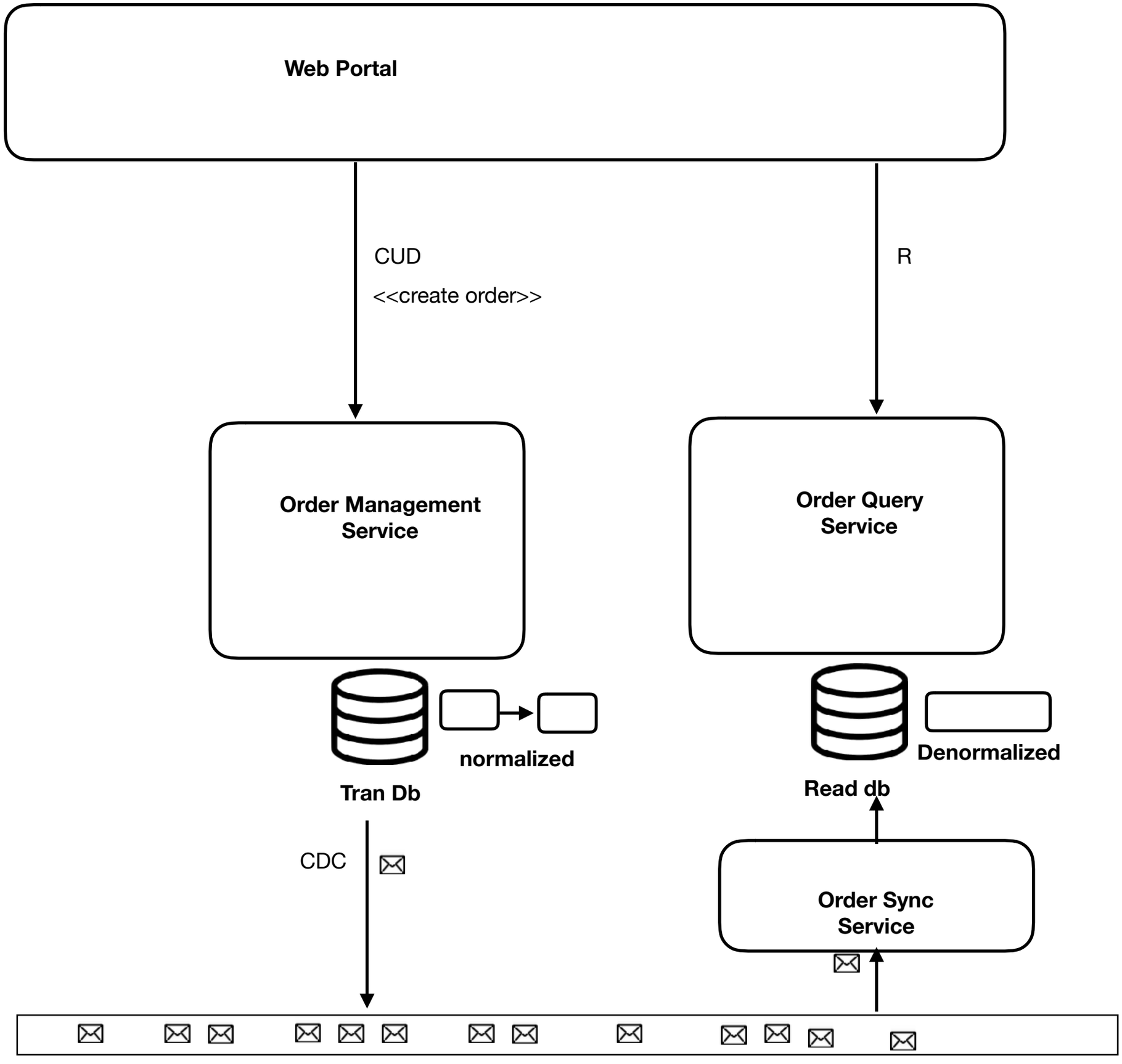


3rd normal form vs denormalized	
No duplicate	Lots of duplicate
Write performance	Write needs multiple updates
More joins	No joins
slow read	read performance

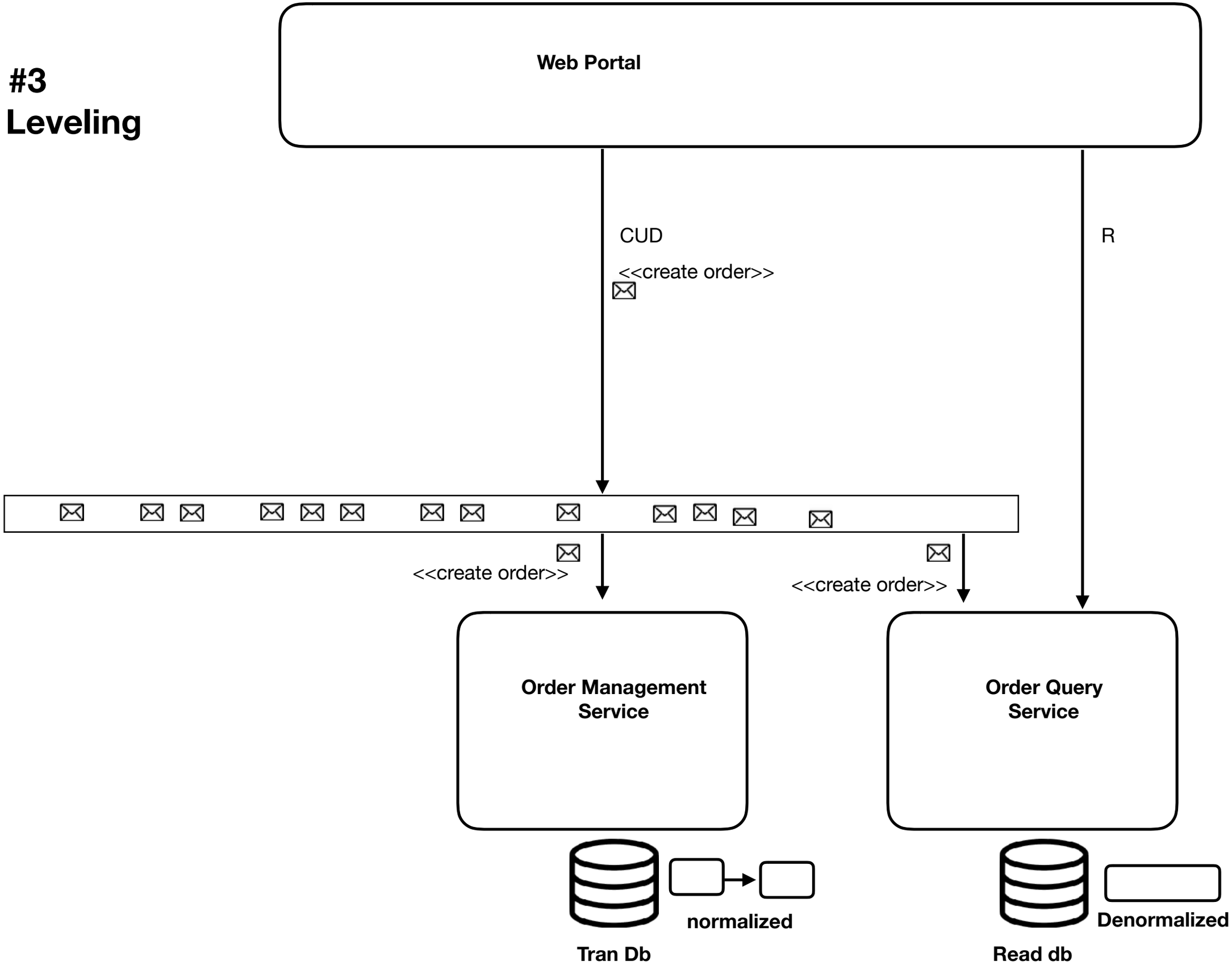
#1 polling


Web Portal
CUD
<<create order>>
R
Order Management Service
Order Query Service
normalized
Denormalized
<<<materialized view>>>
Read db
Tran Db
Last read flag
read
Write
Order Sync Service

2 CDC


Web Portal
CUD
<<create order>>
R
Order Management Service
Order Query Service
normalized
Tran Db
Denormalized
Read db
CDC
Order Sync Service

#3
Load Leveling



	Replication	Polling	CDC	Front end Queues
Performance of update	***	?		
Create Materialized views	? (create db views)	Yes (physical tables)	Yes (physical tables)	Yes (physical tables)
Infra / dev/ op cost	$	$	$$$	$$$
Read Performance	Slow	Fast	Fast	Fast
Version compatibility Effort	Less Effort	Medium Effort	More Effort	More Effort
Platform Capability	Mostly all platforms	All	Not supported for all stores	All
Materialized views in a different platform	No	Yes	Yes	`yes
Event Driven Architecture Support	No	No	No	Yes
High scalability	?	No no	Yes	Yes

Event Storming

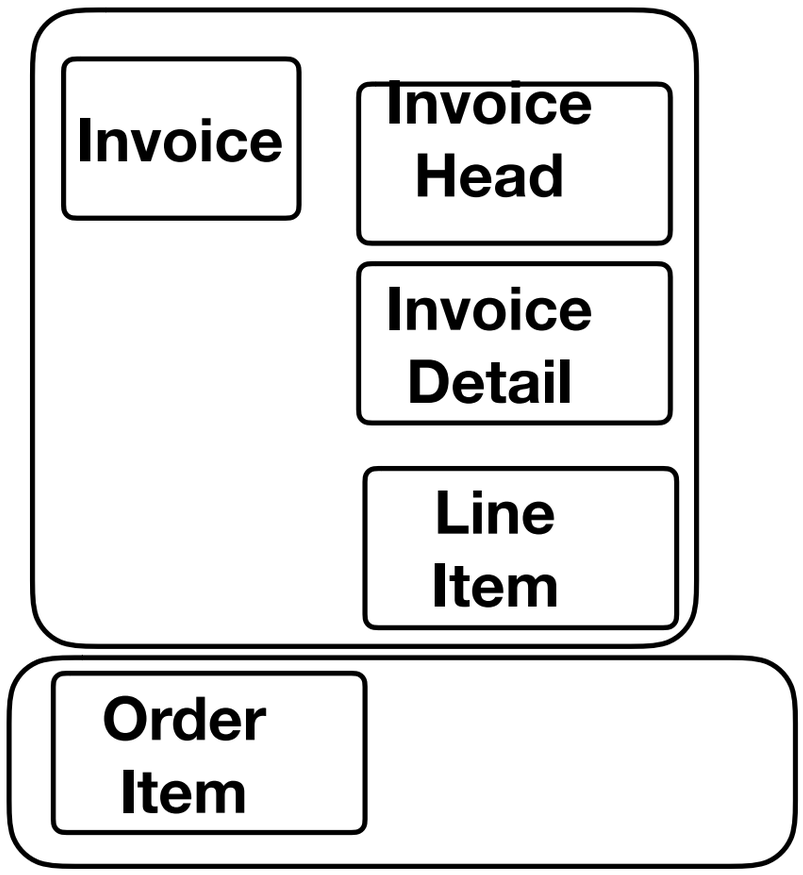
- Domain Driven Design
    - Domain Events
- Event Sourcing

Domain (ecom)

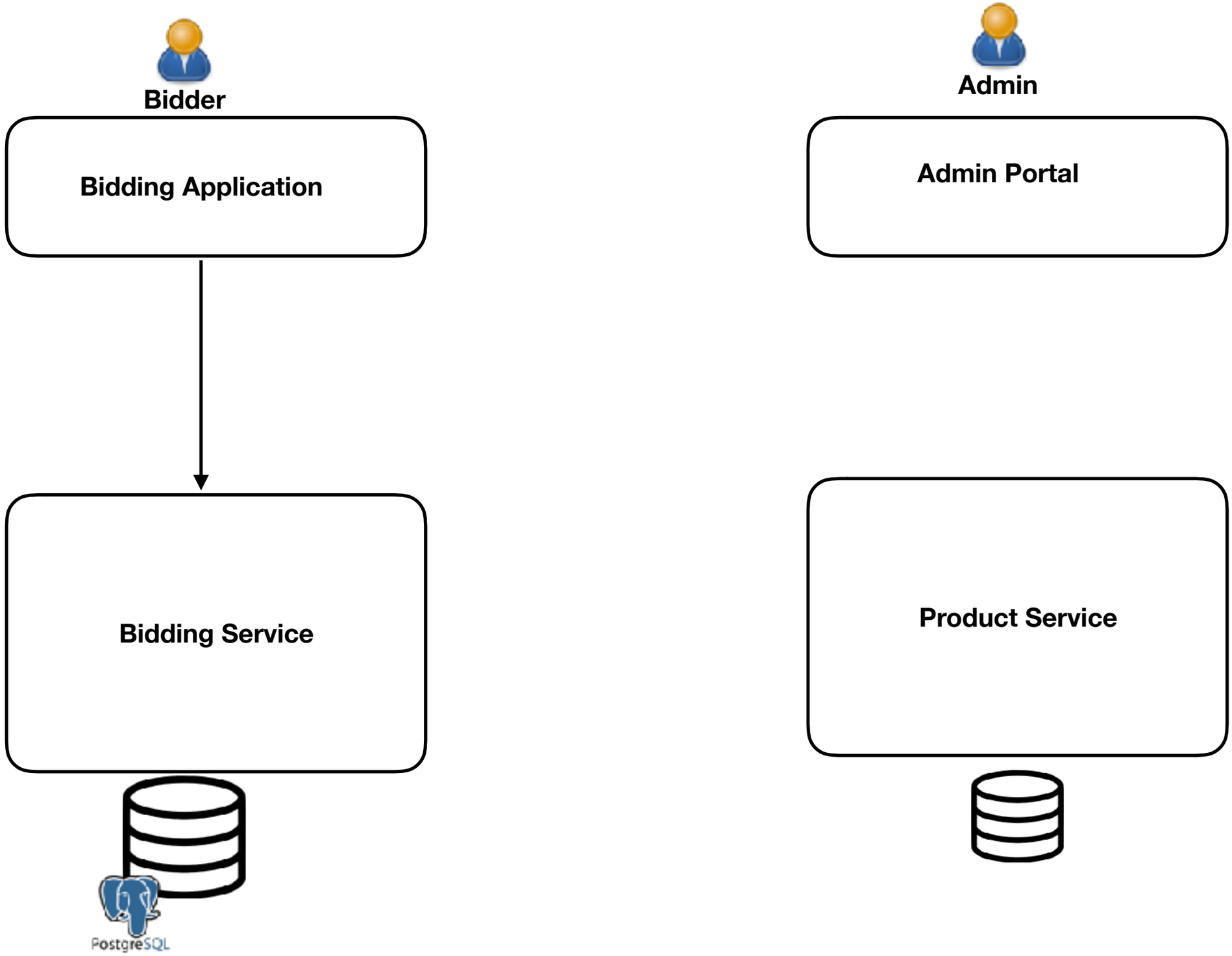
Sub Domain3 (Shipping)

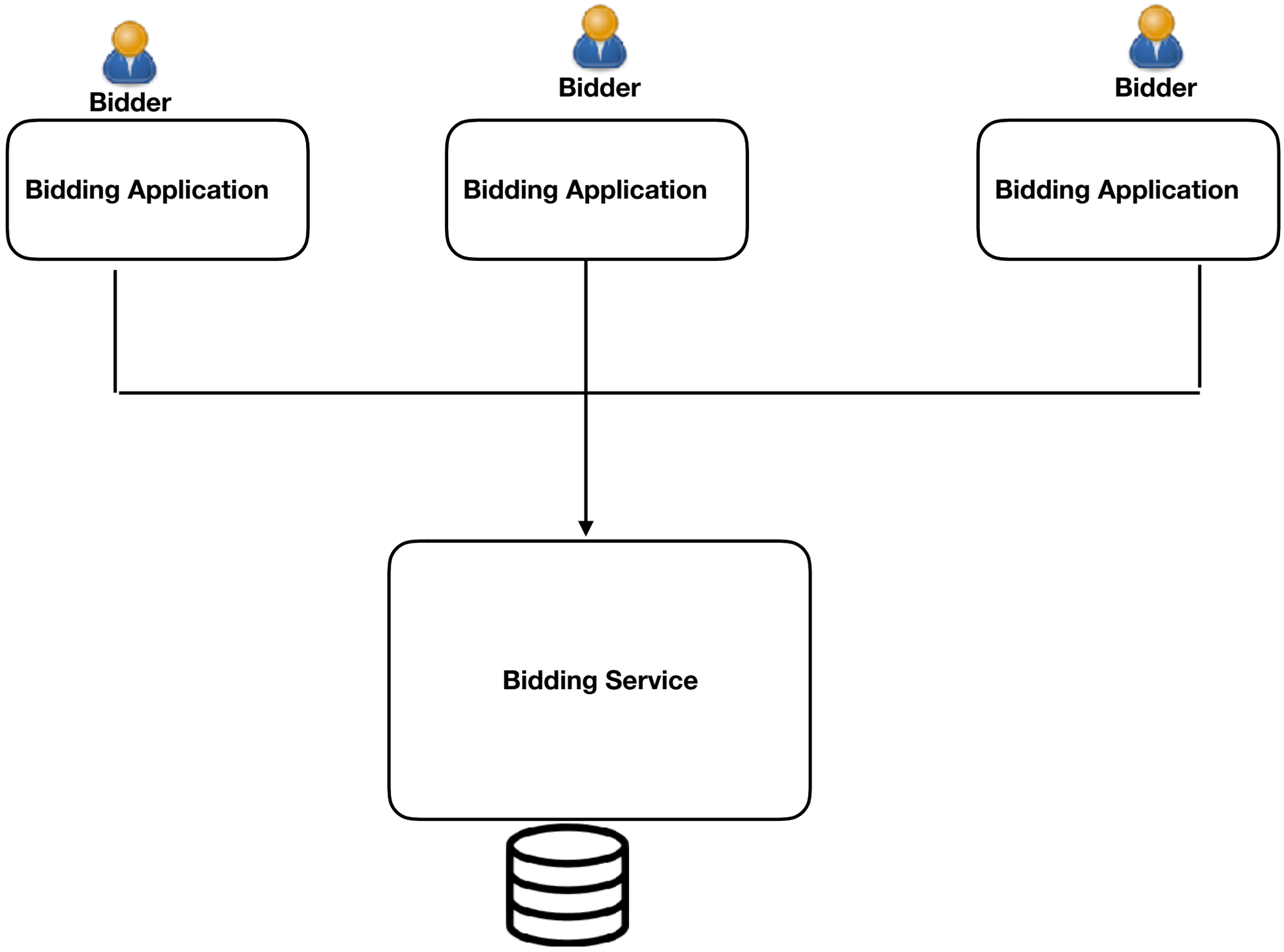
Sub Domain1 (Inventory)

Sub Domain2 (Accounting)

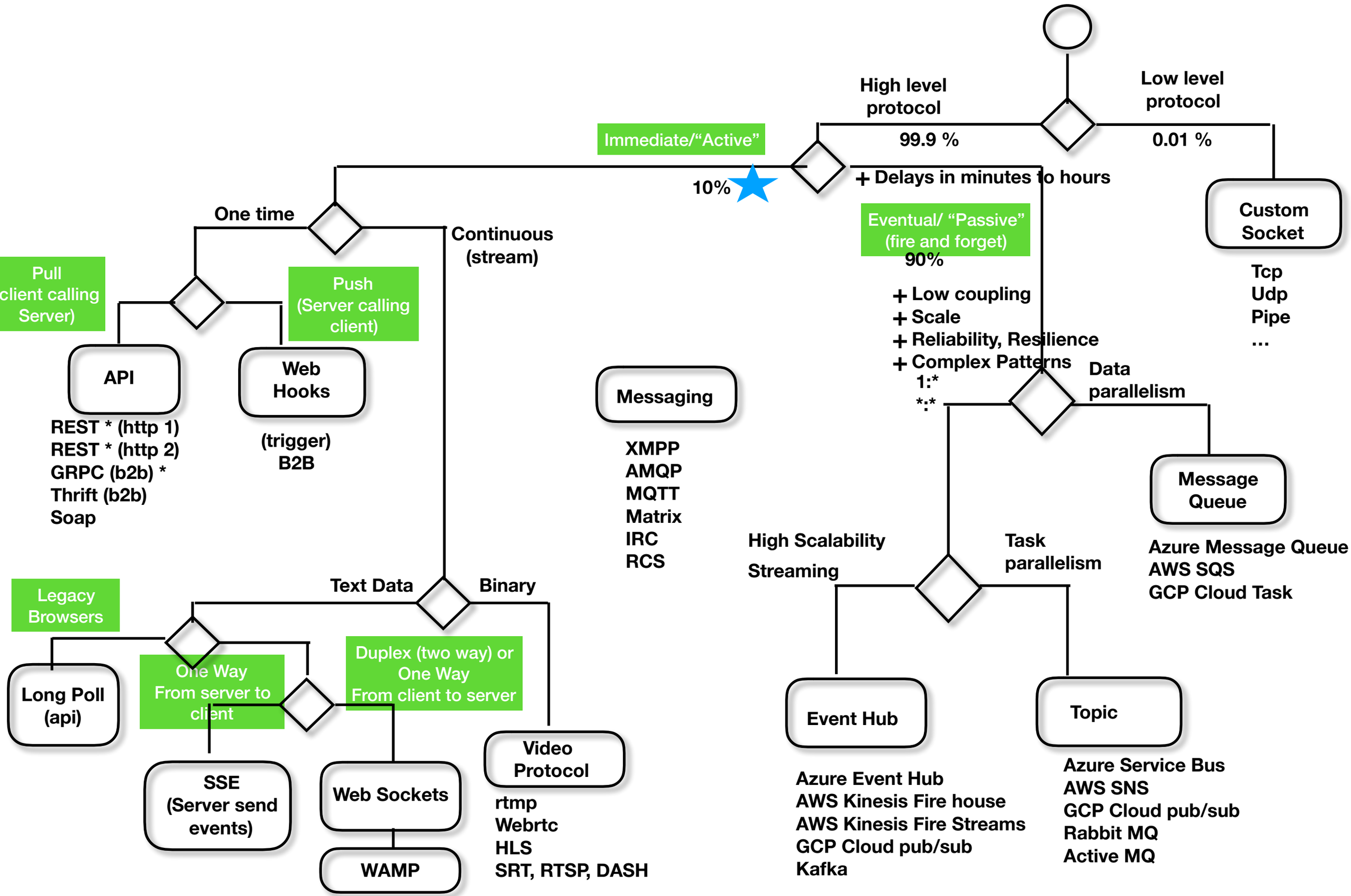


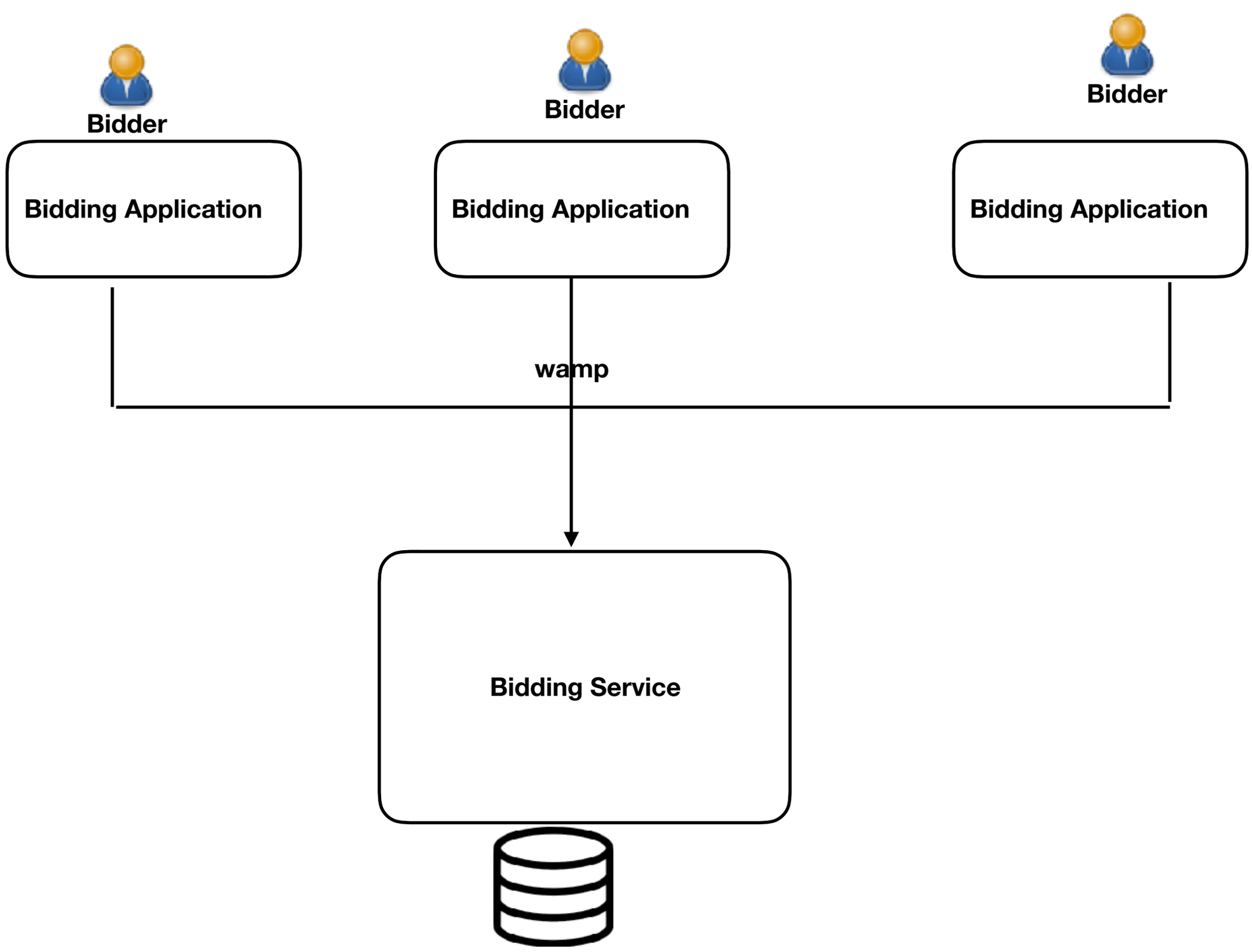
Bidding

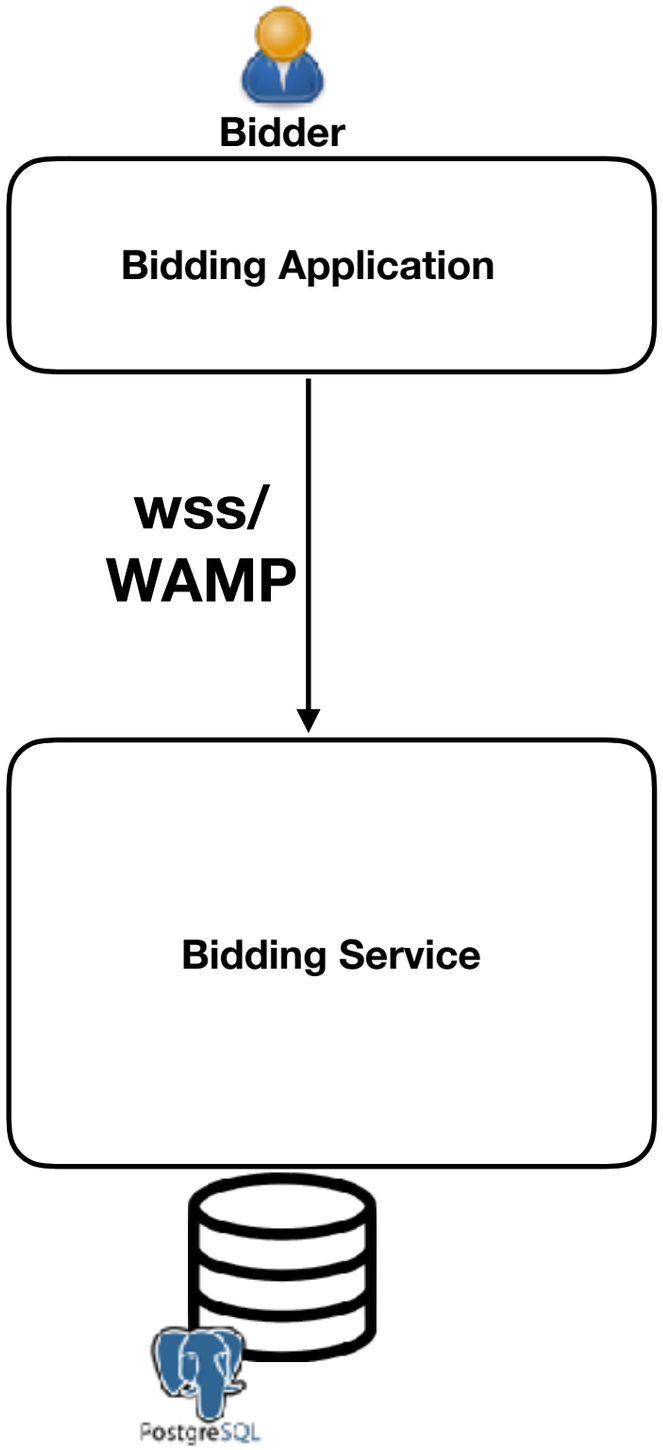

Bidder
Bidding Application
Bidding Service
PostgreSQL
Admin
Admin Portal
Product Service

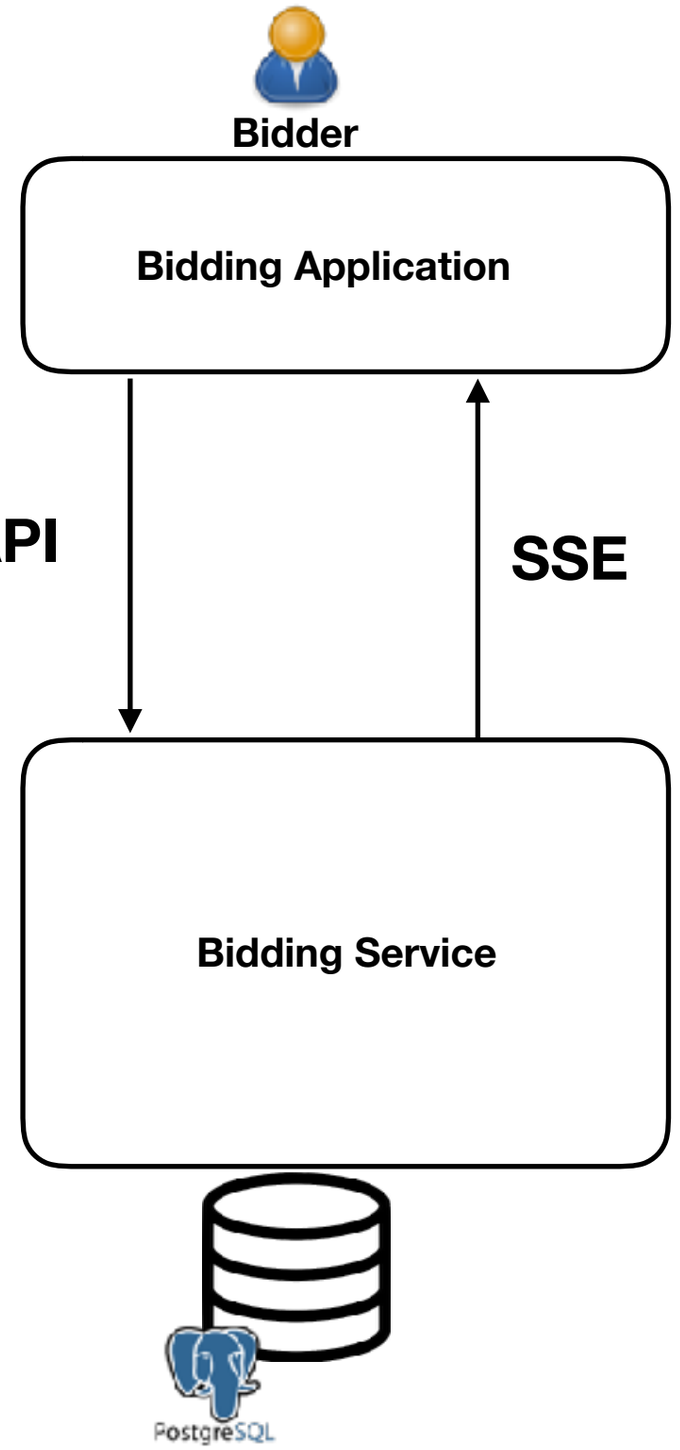

Bidder
Bidder
Bidder
Bidding Application
Bidding Application
Bidding Application
Bidding Service

Choose Communication

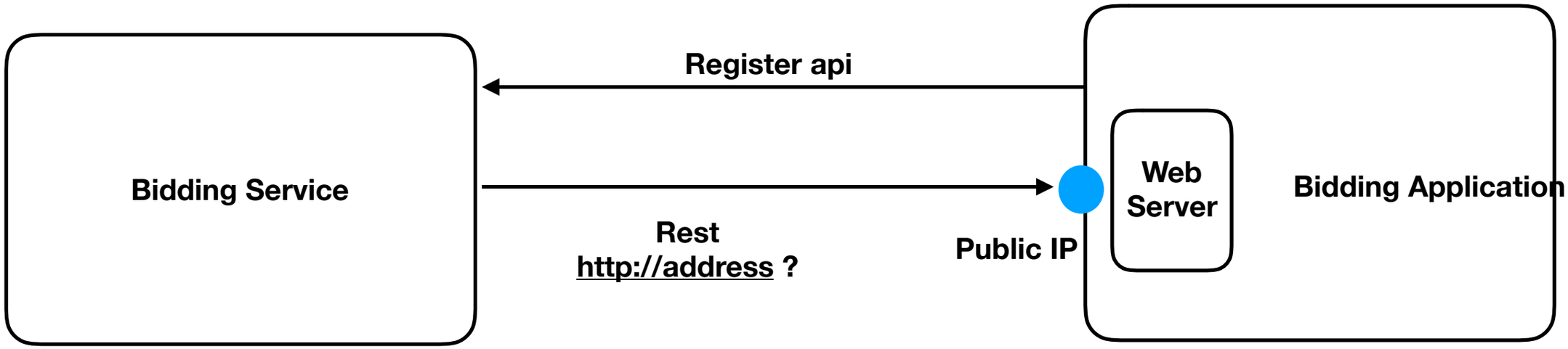



Bidder
Bidder
Bidder
Bidding Application
Bidding Application
Bidding Application
wamp
Bidding Service

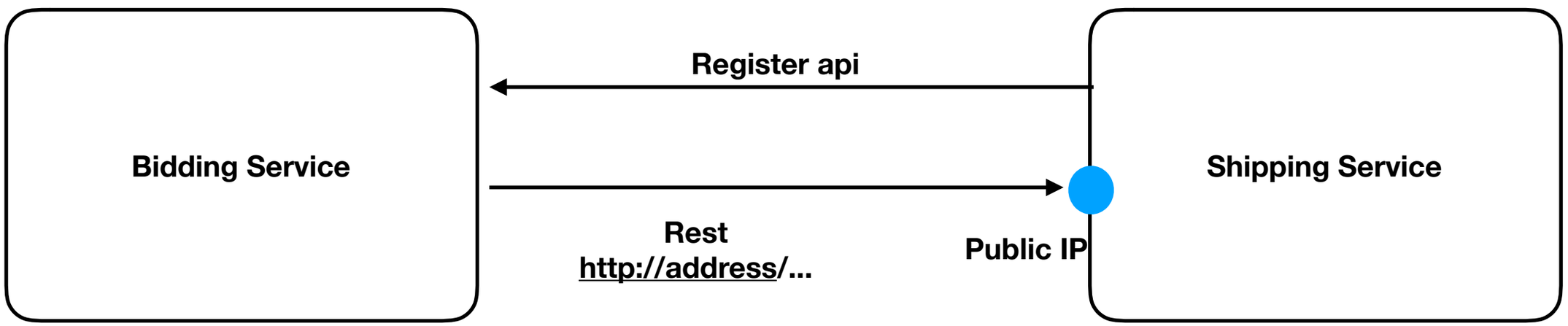

Bidder
Bidding Application
wss/
WAMP
Bidding Service
PostgreSQL


Bidder
Bidding Application
API
SSE
Bidding Service
PostgreSQL

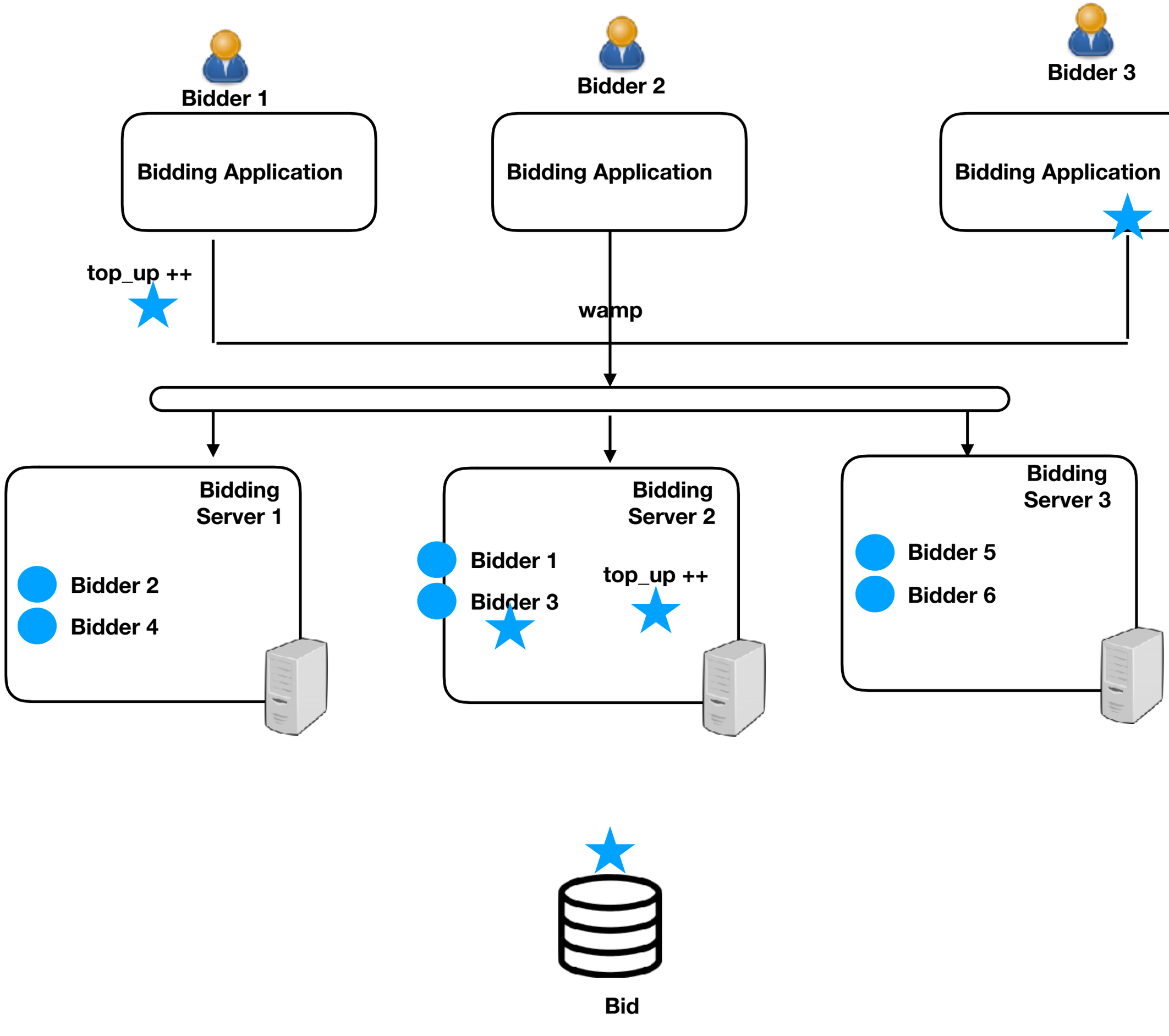
Bidding Application



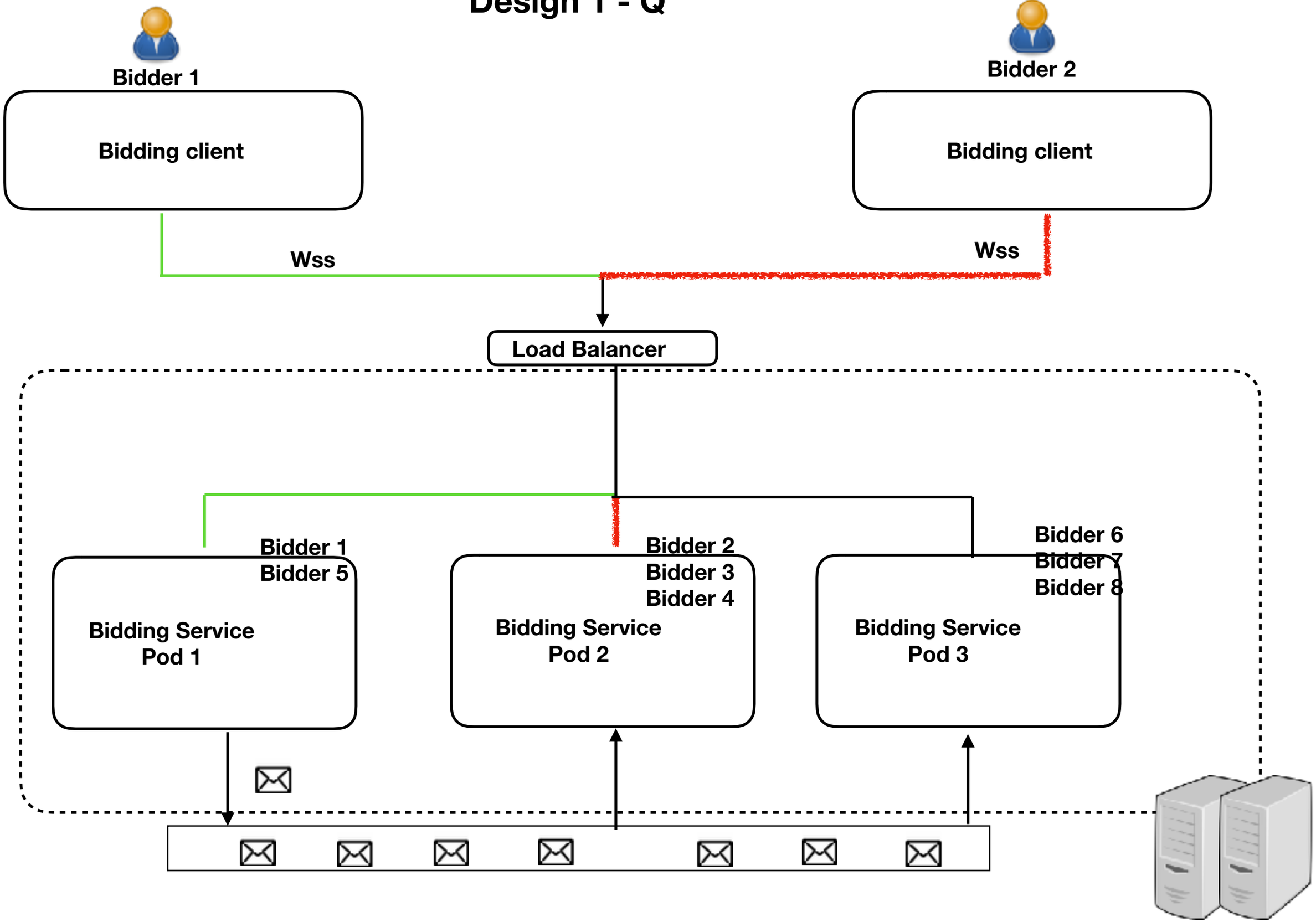
B2C



B2b


Bidder 1
Bidder 2
Bidder 3
Bidding Application
Bidding Application
Bidding Application
top_up ++
wamp
Bidding Server 1
Bidder 2
Bidder 4
Bidding Server 2
Bidder 1
Bidder 3
top_up ++
Bidding Server 3
Bidder 5
Bidder 6
Bid

Design 1 - Q


Bidder 1
Bidding client
Bidder 2
Bidding client
Wss
Wss
Load Balancer
Bidder 1
Bidder 5
Bidding Service
Pod 1
Bidder 2
Bidder 3
Bidder 4
Bidding Service
Pod 2
Bidder 6
Bidder 7
Bidder 8
Bidding Service
Pod 3

Design 2 - polling db

Bidder 1

Bidding client

Bidder 2

Bidding client

Wss

Wss

Load Balancer

Bidder 1
Bidder 5

Bidding Service
Pod 1

Bidder 2
Bidder 3
Bidder 4

Bidding Service
Pod 2

Bidder 6
Bidder 7
Bidder 8

Bidding Service
Pod 3

Poll

Poll

PostgreSQL

Design 3 - Cache

Bidder 1

Bidding Application

Bidder 2

Bidding Application

Wss

Wss

Load Balancer

Bidder 1
Bidder 5

Bidding Service
Pod 1

Bidder 2
Bidder 3
Bidder 4

Bidding Service
Pod 2

Bidder 6
Bidder 7
Bidder 8

Bidding Service
Pod 3

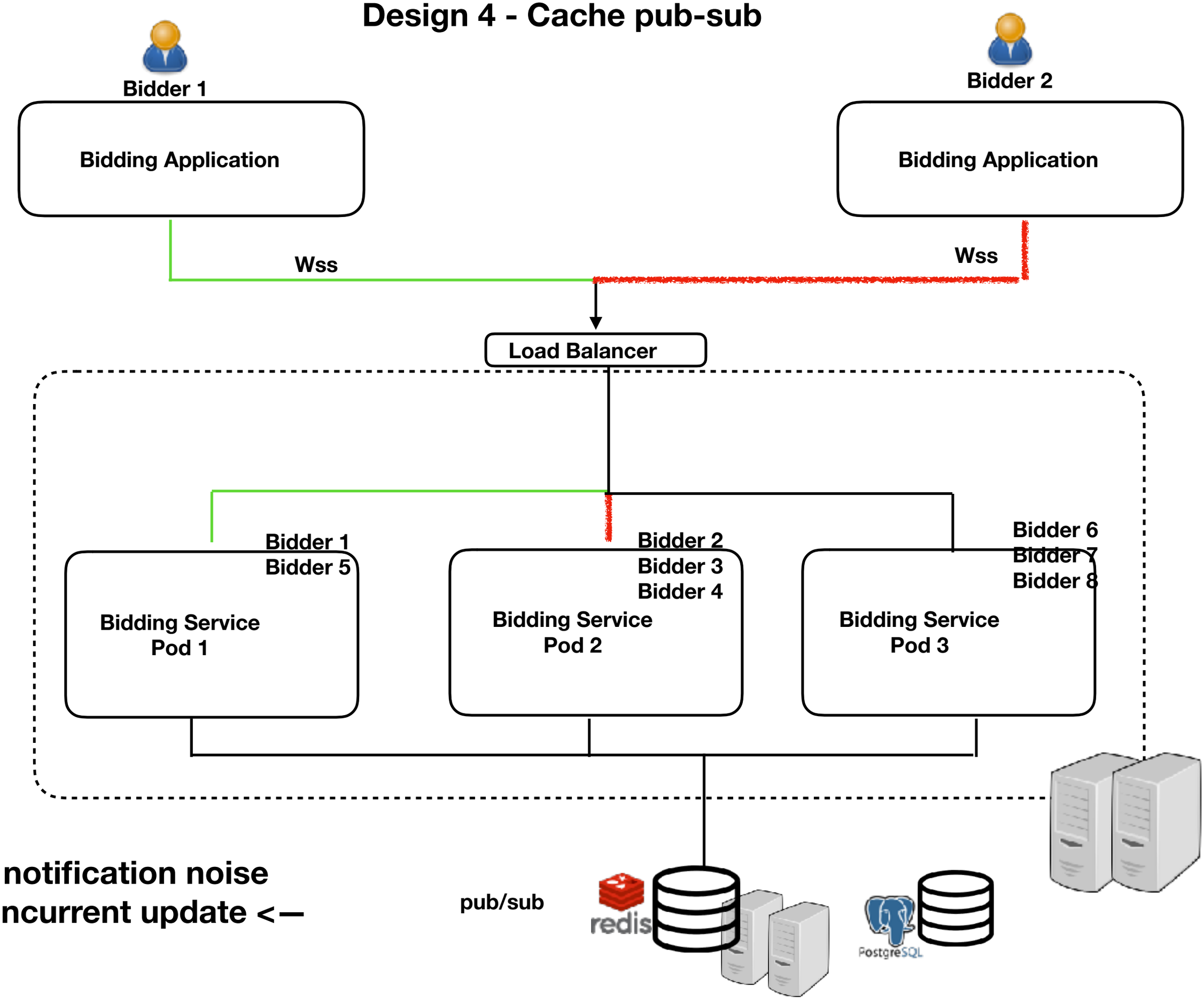
Poll

Poll

Cache
Server

PostgreSQL

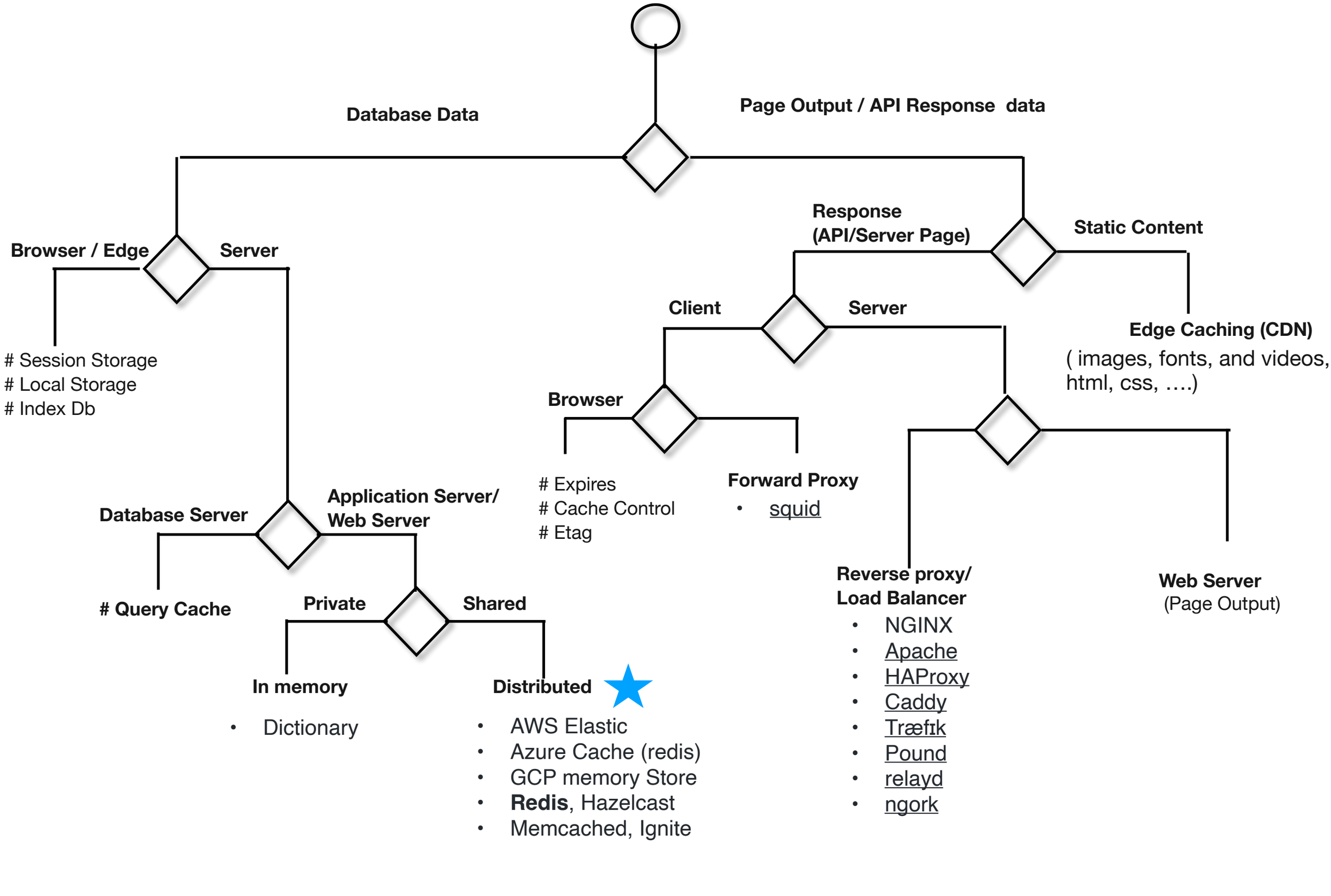
Design 4 - Cache pub-sub

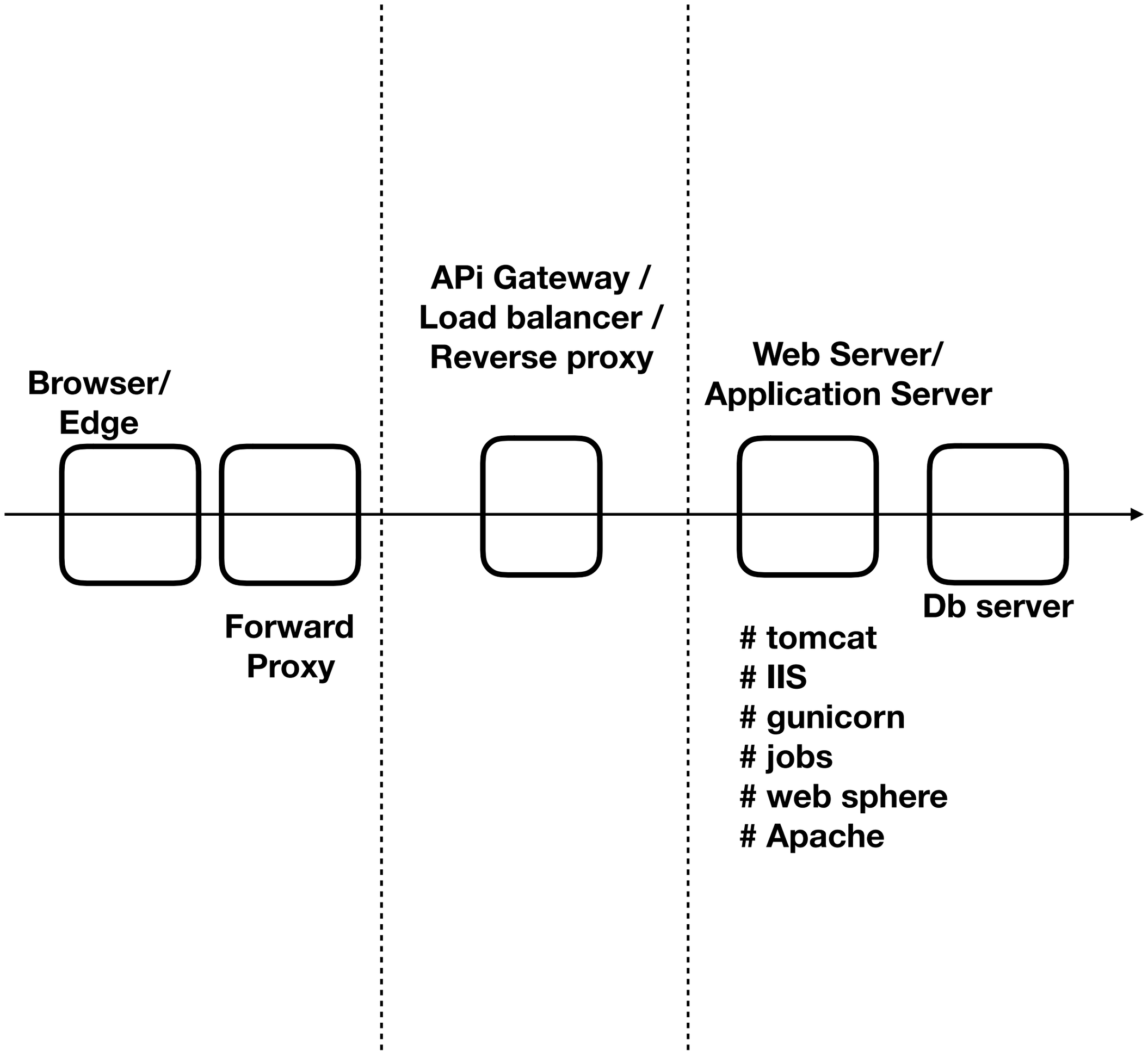


notification noise
concurrent update <—

pub/sub

Choose Cache




Browser/
Edge
APi Gateway /
Load balancer /
Reverse proxy
Web Server/
Application Server
Forward
Proxy
Db server
tomcat
IIS
gunicorn
jobs
web sphere
Apache

irctc

Tatkal booking

Principles

- YAGNI
- SOC

Reference

Manage Business

Run Business

Report

Report Server

Web Portal

<<rest>>

API gateway

Notification Service

Listing Search Service

Grpc

Grpc

PBI/ tableu

Stream Analytics

Booking API

Booking Created

OverBooked

Listing Created

elasticsearch

Create Booking

R

*

*

Event Service

Listing Created

Listing Service

Booking Created

OverBooked

Booking Service

Create Booking

Create Booking

Booking Created

Booking Query Service

OverBooked

Listing Created

Lake house

(Big Data)

Data Lake

Data Ware House

File Storage

mongoDB

Idempotent

PostgreSQL

Current

- normalized
Write friendly
db

PostgreSQL

Current

cassandra

History

- denormalized
Read friendly db
Matrialized View

Batch Analytics

Spark