Case Study

Airbnb

Web Portal

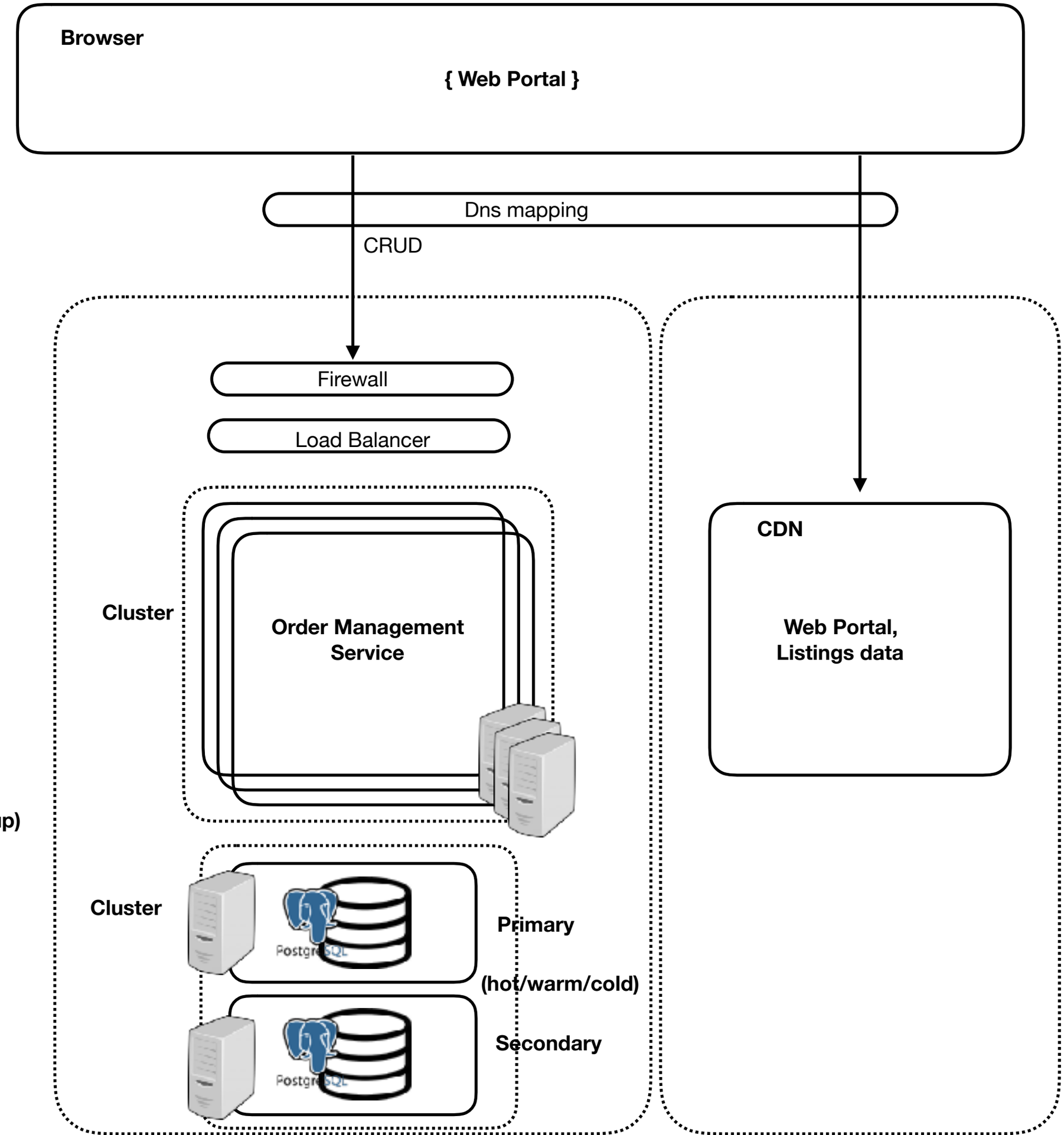
CRUD

Order Management Service

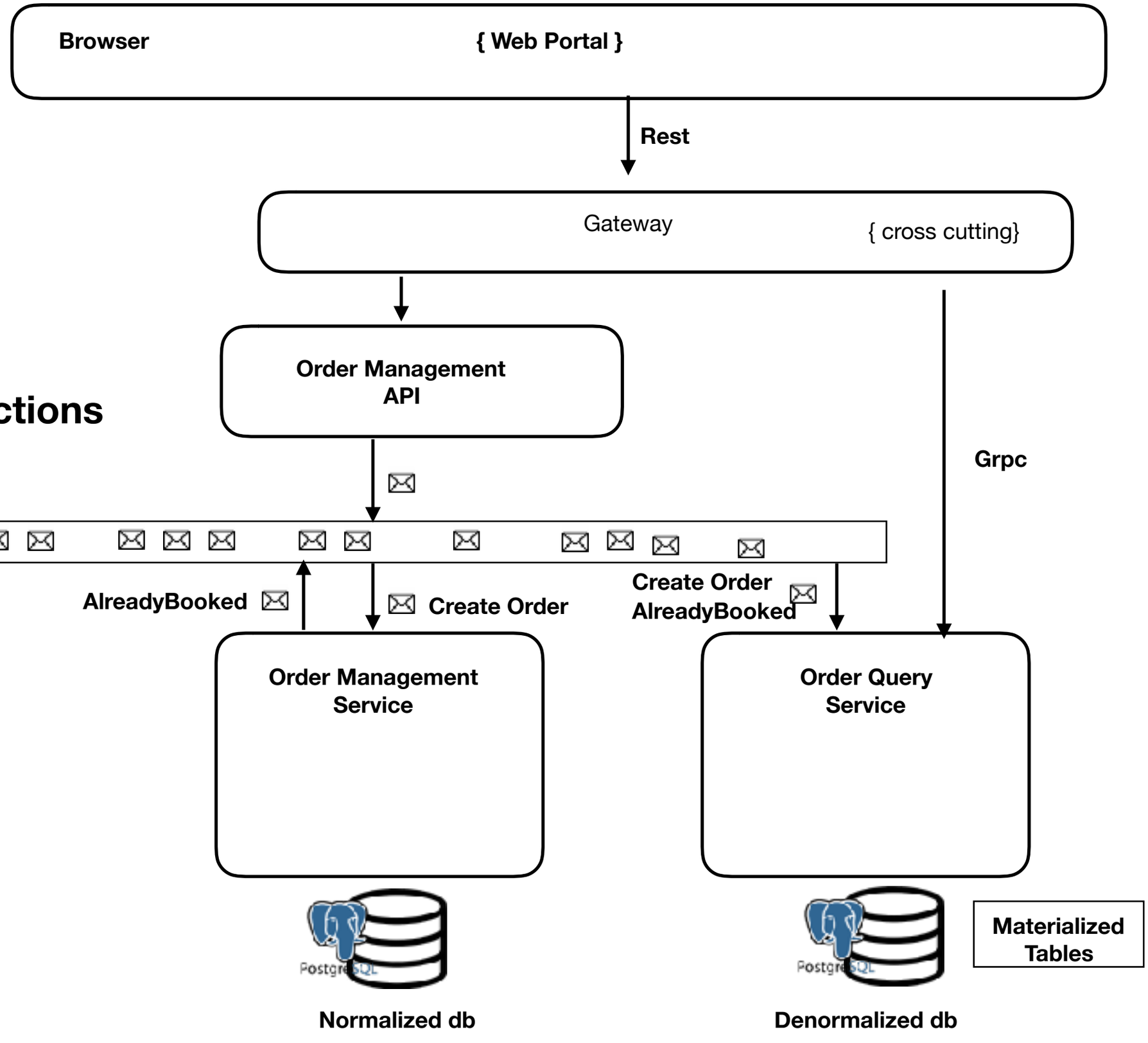
Cloning

@ Read heavy

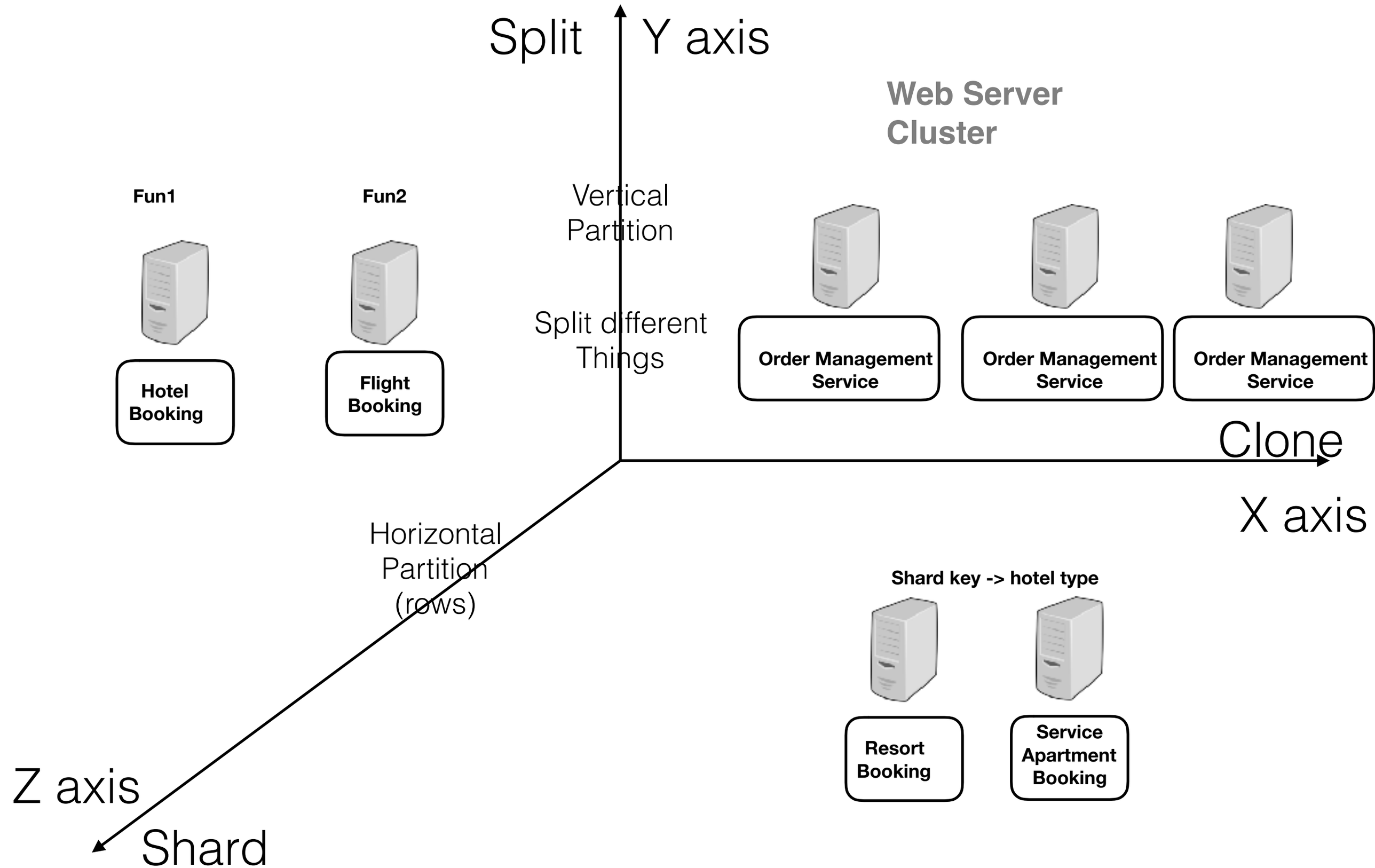
* database will become the bottleneck
* Handling load spikes will be hard
* Database scaling issues (only scale up)
*

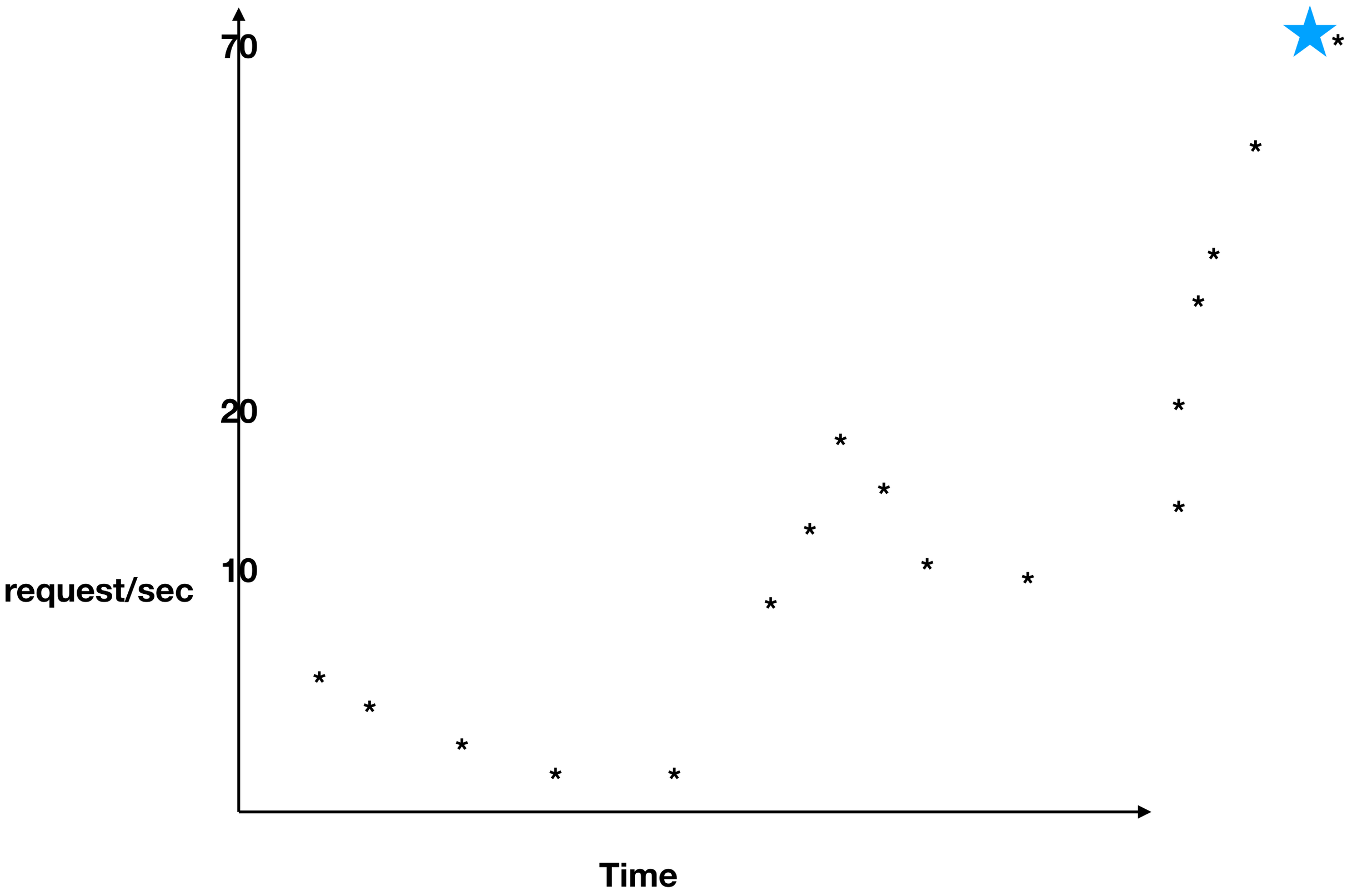


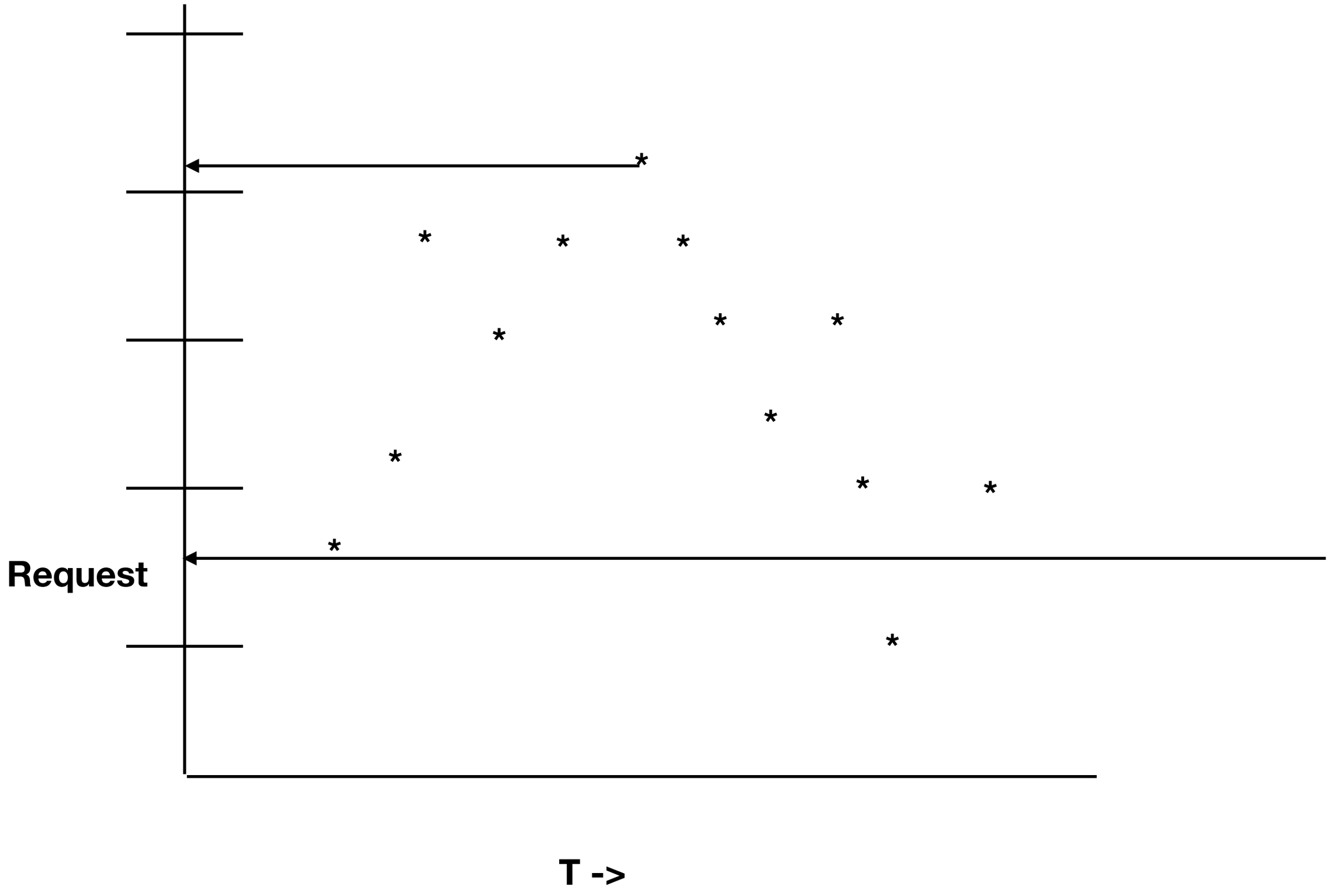
Cloning
Splitting
Sharding
 # Queue (Q/T/H)
 # CQS
 # Materialized View
API gateway
EDA
 # compensatable transactions



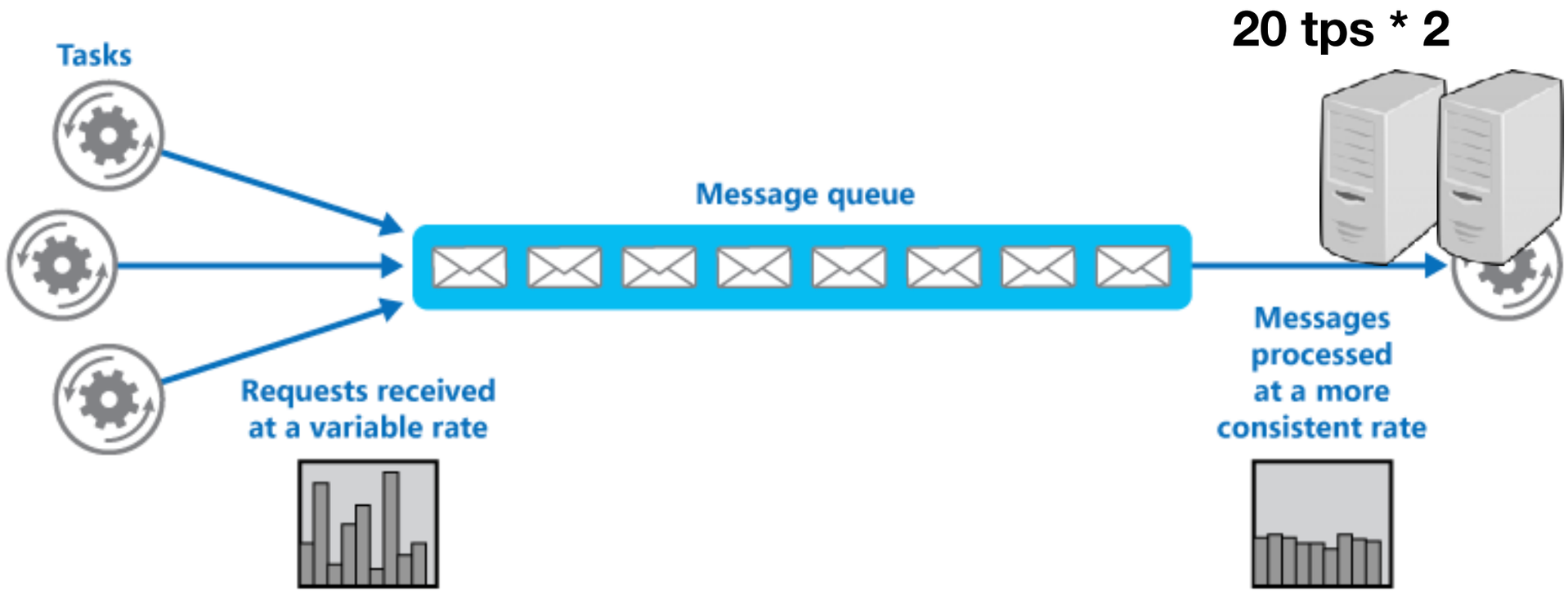
Applying cube pattern on Compute




70
20
10
request/sec
Time


Request
T ->

Queue-Based Load Leveling pattern

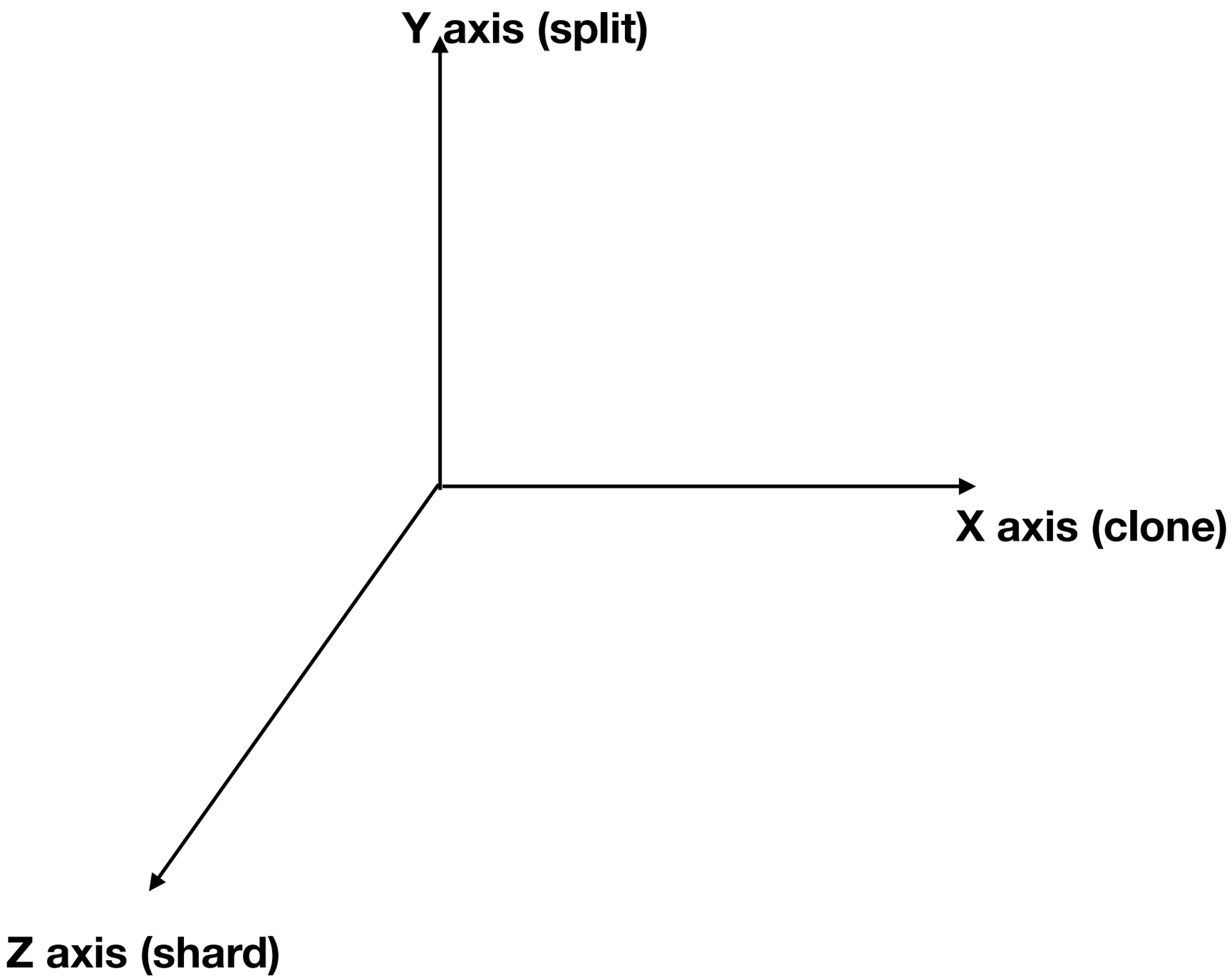


useful to any application that uses services that are subject to overloading.

- 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

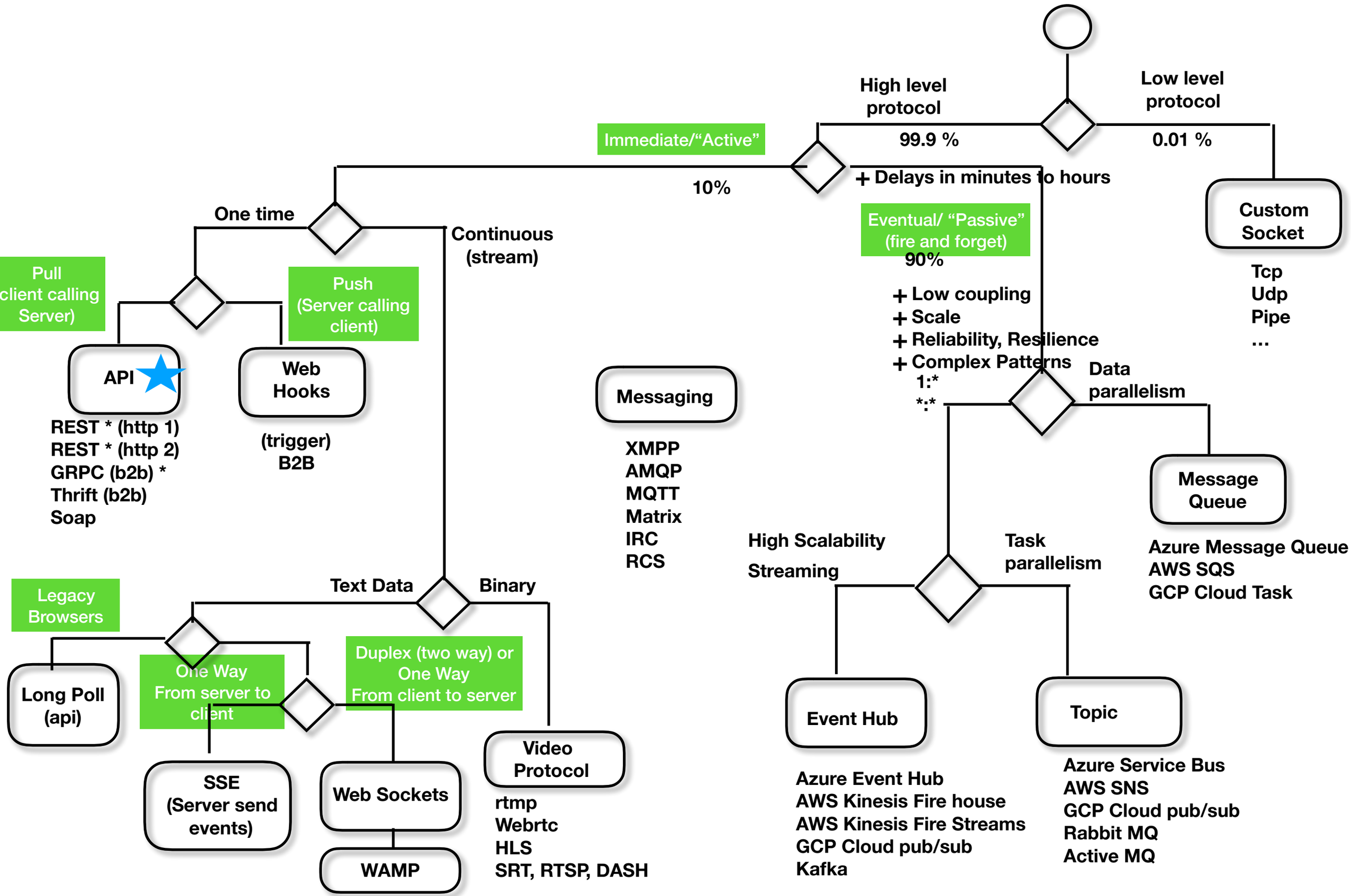
- Last write wins
-

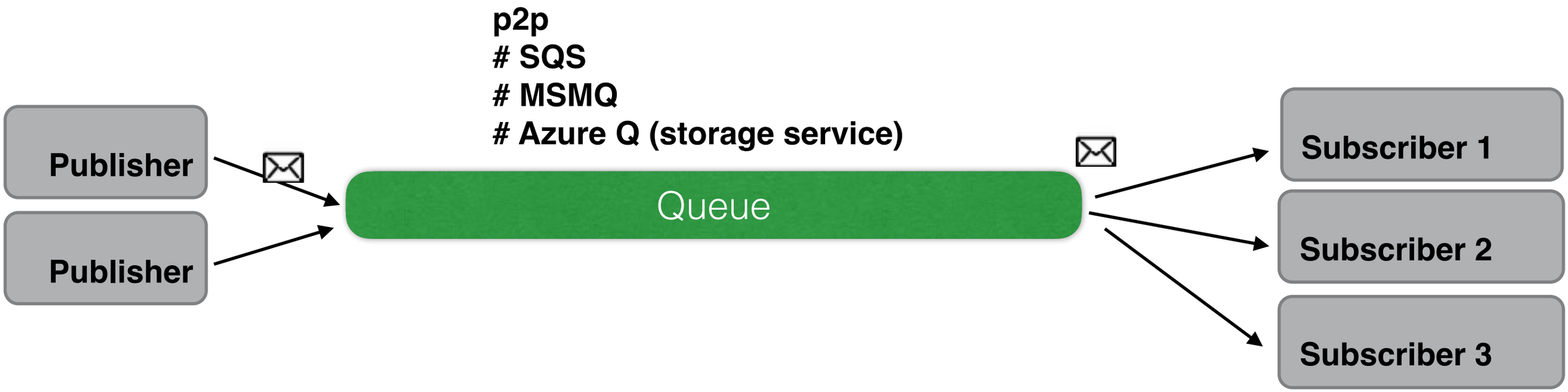
Scalability Cube

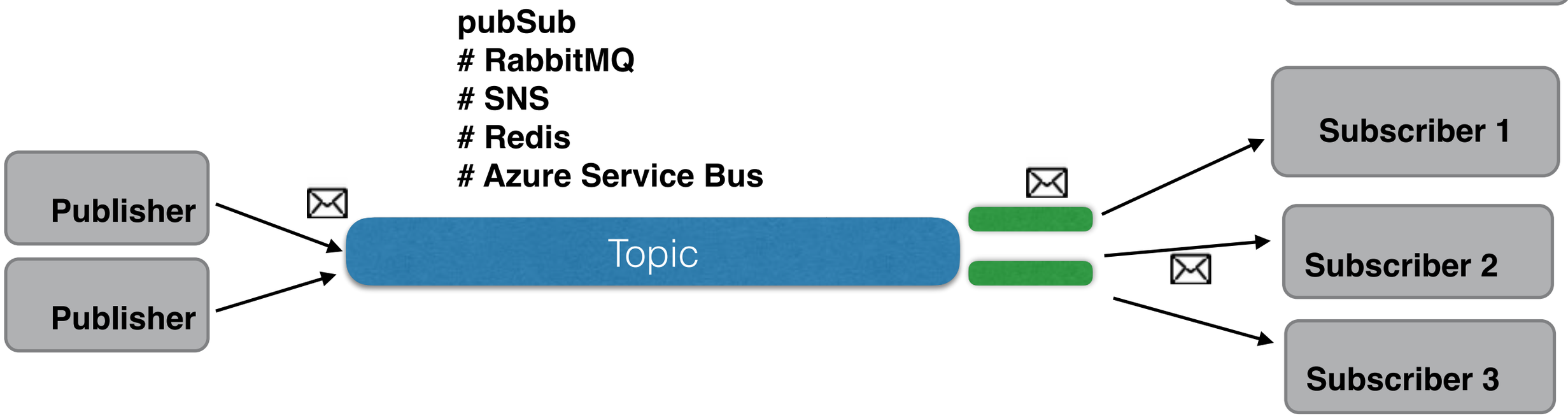


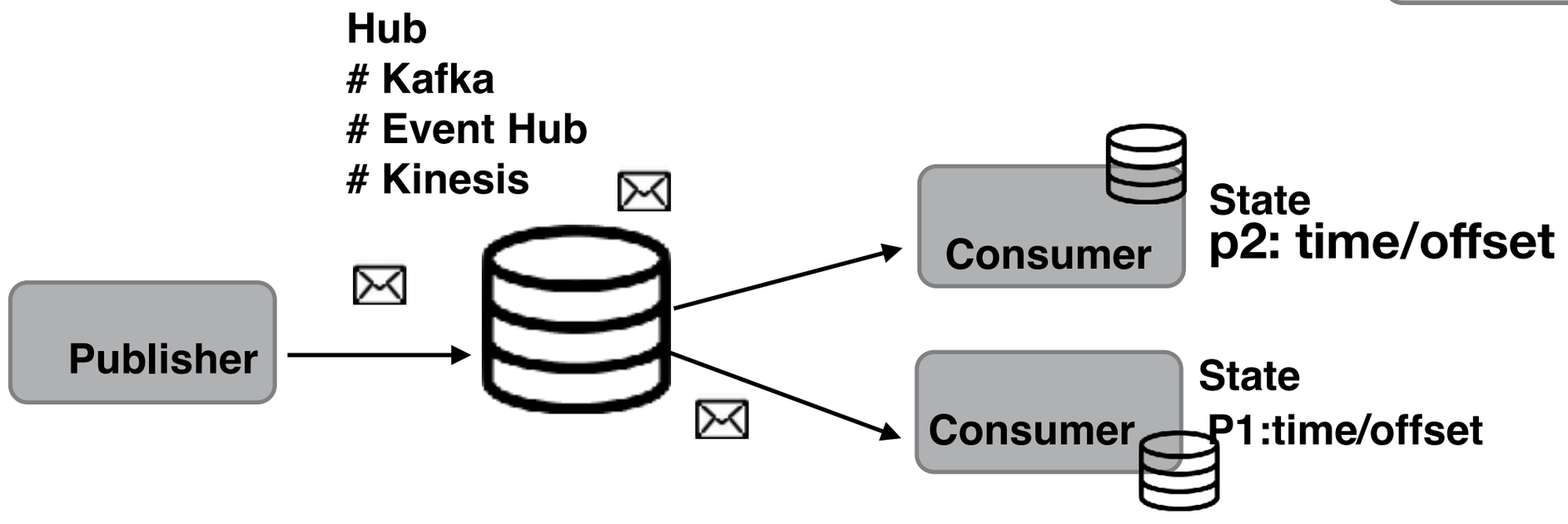
Load leveling

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

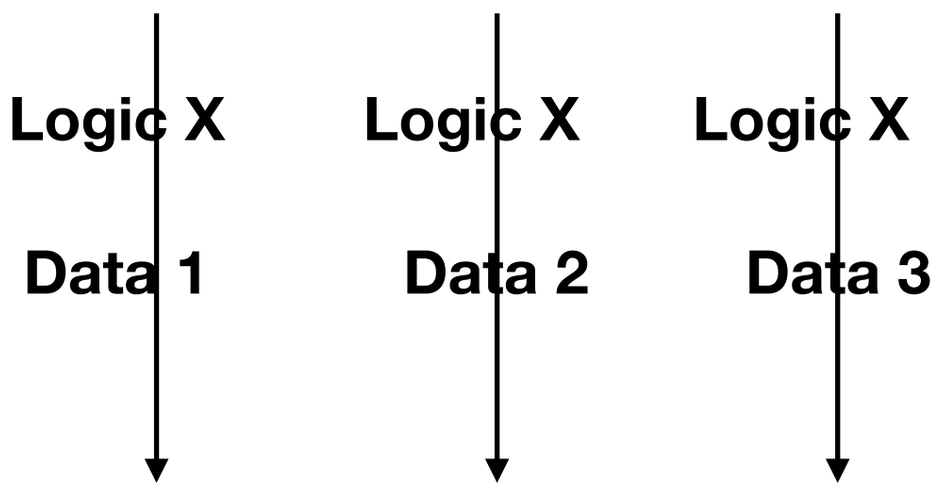

Hub
Kafka
Event Hub
Kinesis
Publisher
Consumer
State
p2: time/offset
Consumer
State
P1:time/offset

Parallel

Data Parallelism

Logic X

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

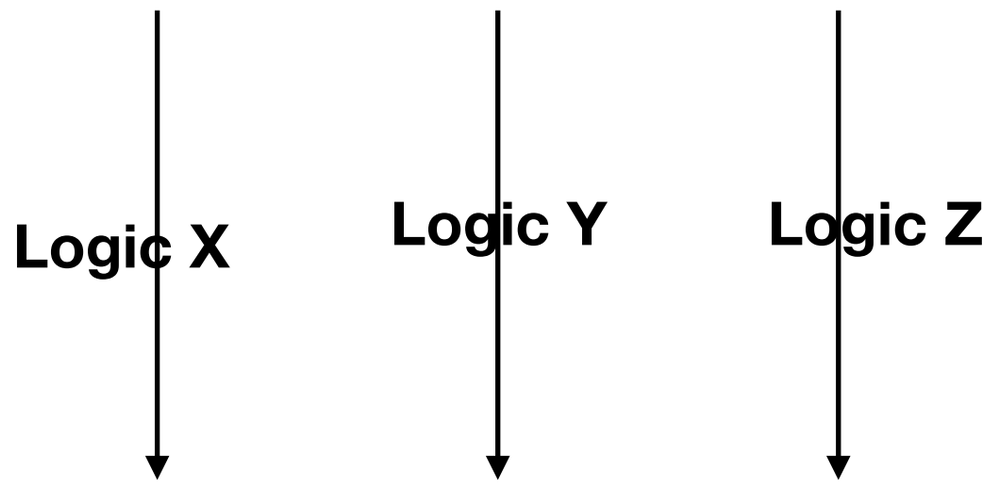


Task Parallelism

Logic Z

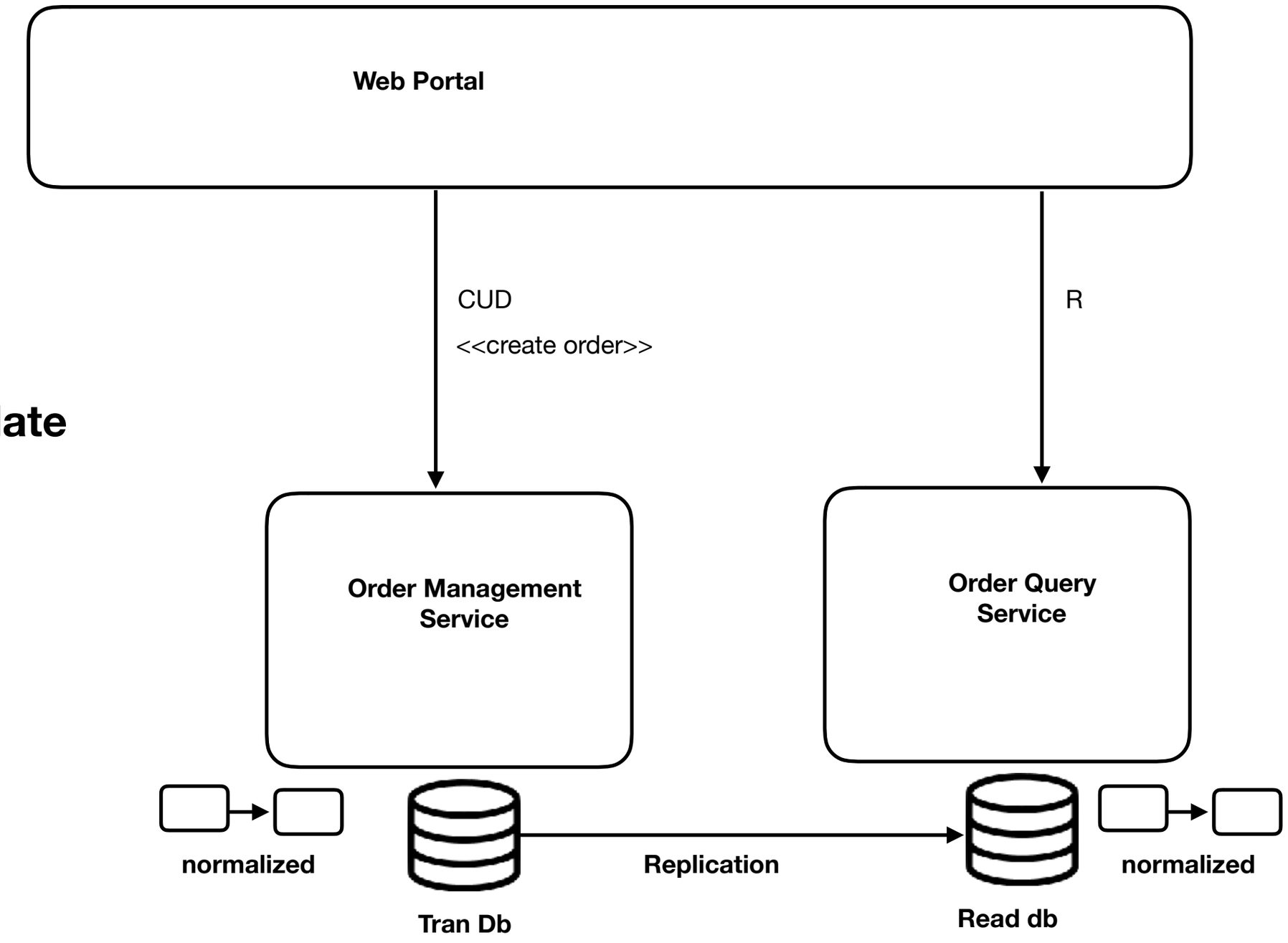
Logic X

Logic Y



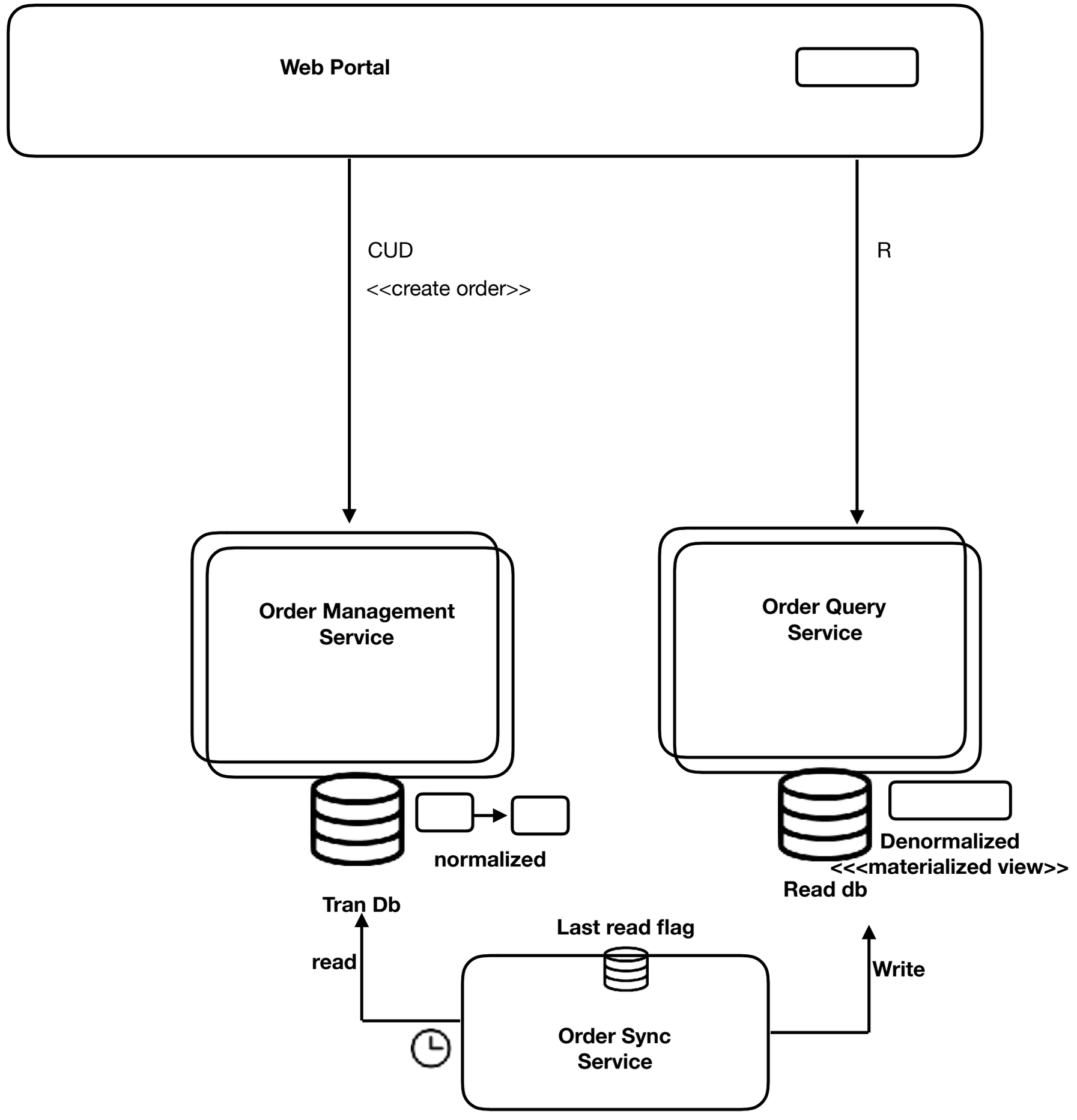
#0 db replication

@ performance of update

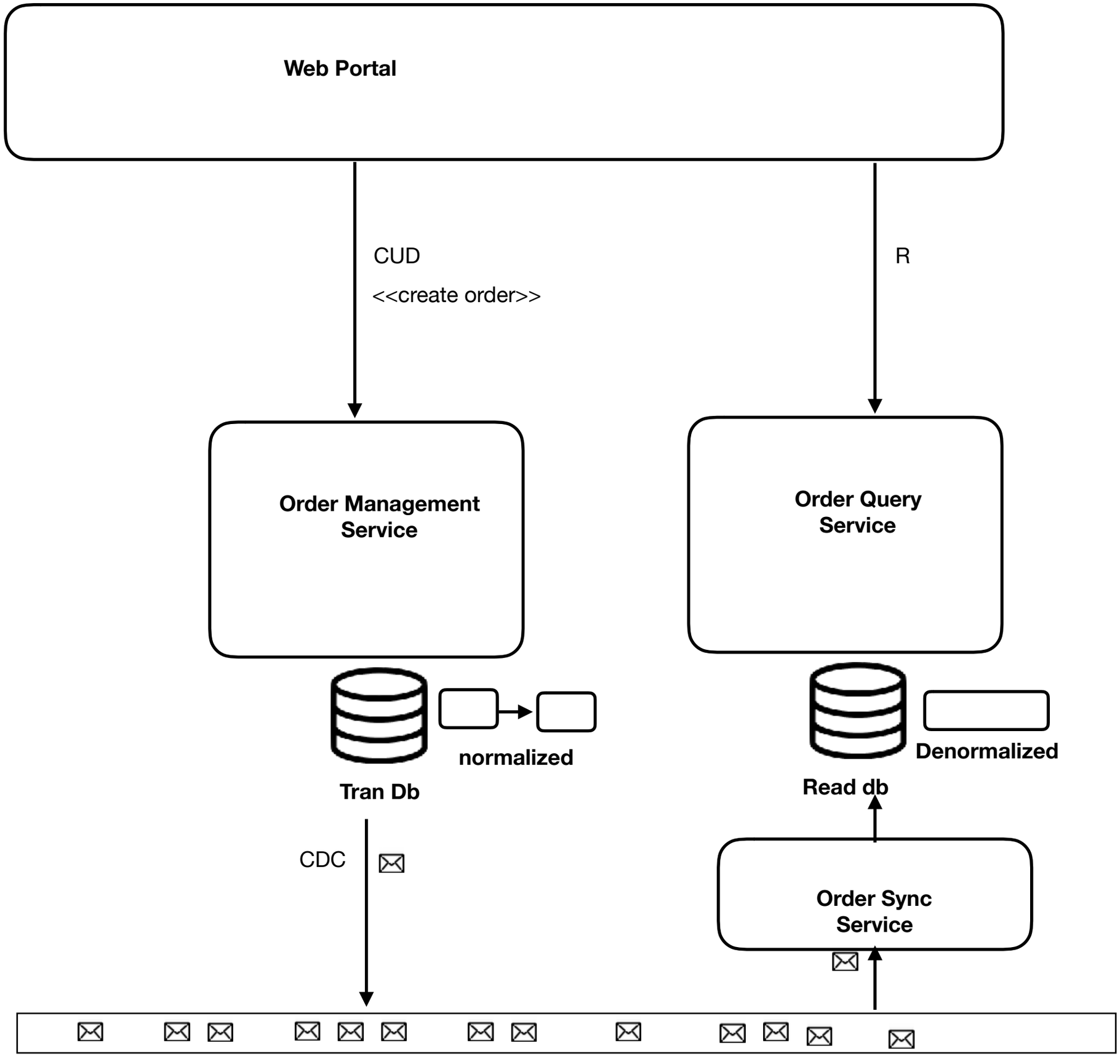


3rd normal form vs denormalized	
No duplicate	Lots of duplicate
Write performance	Write needs multiple updates
More joins	No joins
slow read	read performance

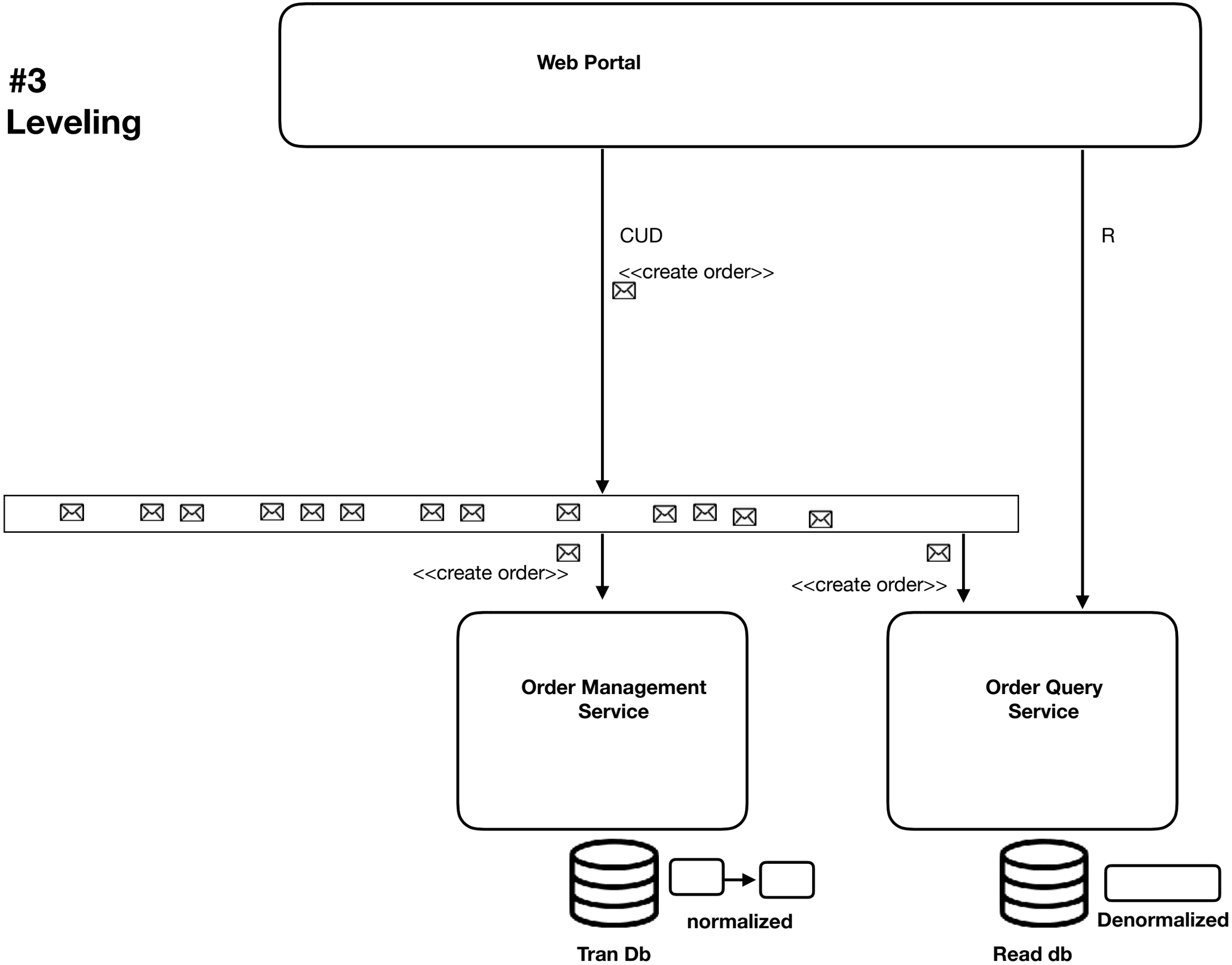
#1 polling


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

Principles

- YAGNI
-

Reference

Manage Business

Run Business

Report

Report Server

Web Portal

<<rest>>

API gateway

Grpc

Grpc

Notification Service

Listing Search Service

Booking API

PBI/ tableu

Stream Analytics

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

OverBooked

Listing Created

Booking Query Service

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