Active Recall :

1. Introduction to HLD

2. Servers
   - Latency
   - Scaling
   - Throughput
   - RR, Port
   - Cloud Computing
   - Distributed Systems
   - Terminologies : Spin up/down, Cron job

3. CAP Theorem
   - Definition : at most 2 goals
   - Consistency : different nodes seeing same data
   - Availability : System always respond
   - Partition Tolerance

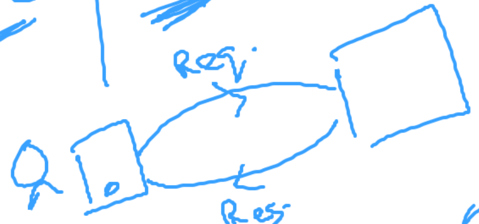
- CP = Consistency + Partition Tolerance → Banks
- AP = Availability + Partition Tolerance → social media
- CA = Consistency + Availability → Todo app
- CAP

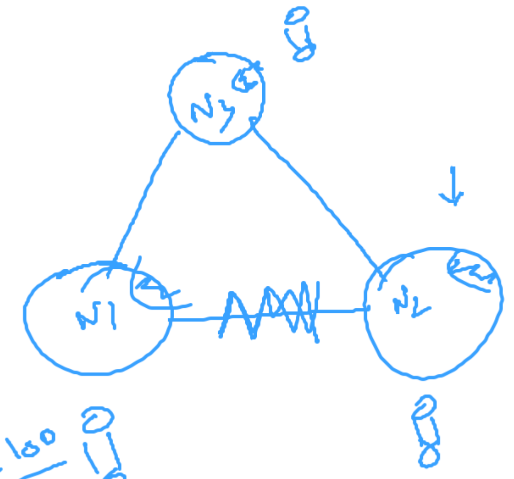
goals

units

L
sec

Th
mbps



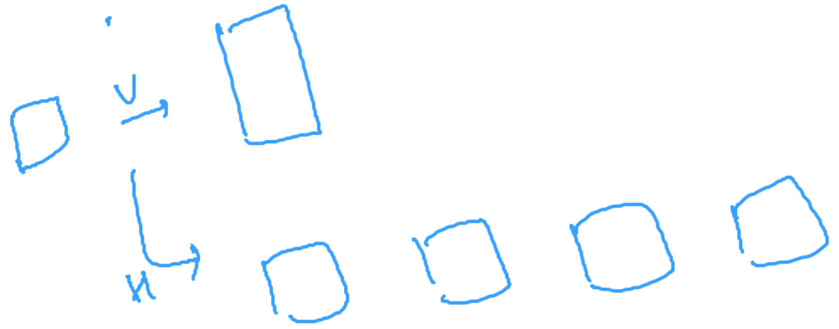


₹100
90
80

C | A

→ Monolith
→ Microservices

Scaling



~~A~~uto-scaling.

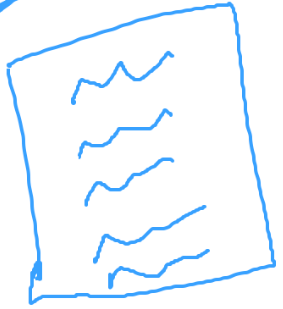
~~u~~p-scaling

~~d~~own-scaling

System : 10k/sec $\longrightarrow$ 50k/sec.

1. Scaling.
2. Optimis^n^ : Db queries
3. Db: Indexing
4. Code: loosely Couple. $\rightarrow$ ▢ ▢▢
5. M. Broker $\rightarrow$ Kafka, RabbitMQ ✓

Monolith (Legacy) → PHP legacy system
→ Phasing out



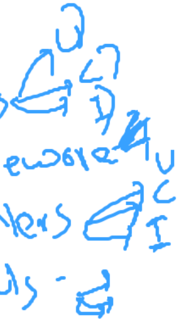
single + stone

Routes

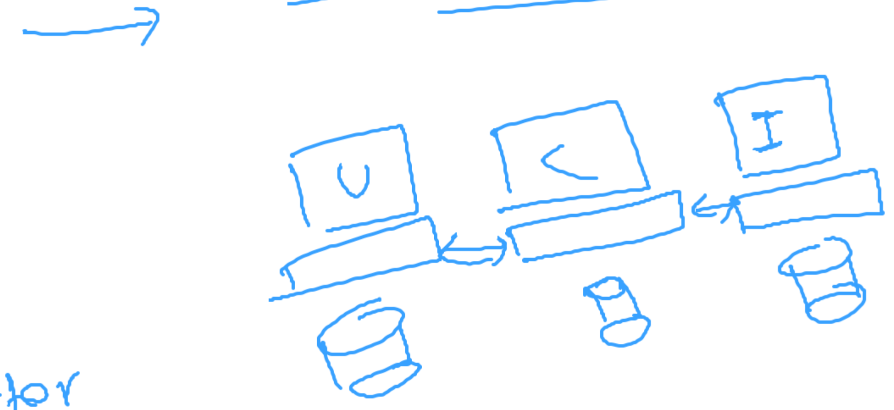
Frontend
Backend
DB

Server

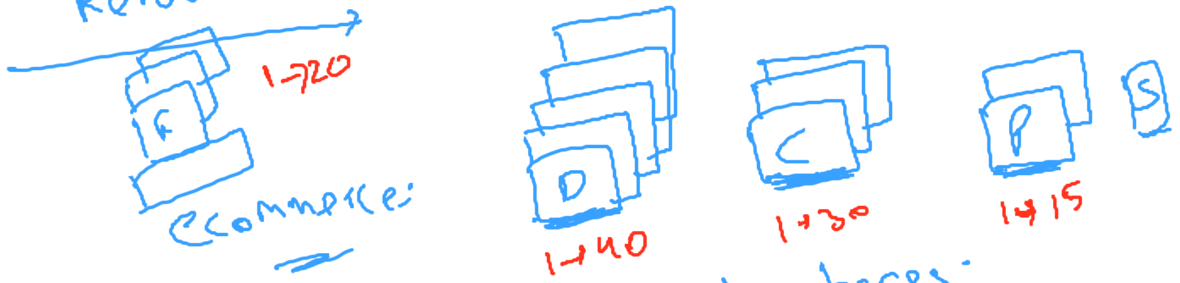
⇒ Project
→ Routes
→ middleware
→ controllers
→ models



Micro-services



Refactor



Ecommerce:

1L | day.

10K | day

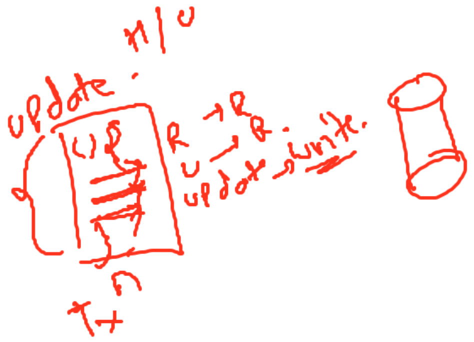
8K | day.

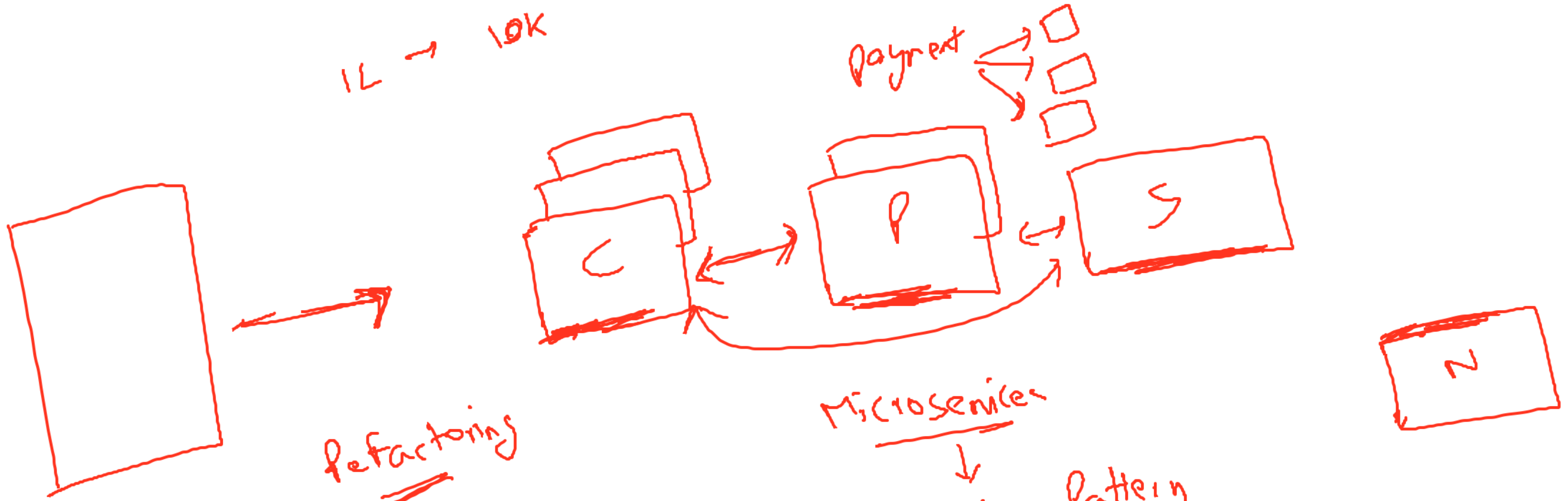
Advantages.

1. Scaling
2. Loosely Coupled
3. Avoiding single Point of Failure.

Recall

- Monolith
- Micro-services
  - Loosely Coupled. Justify.
  - example
- Single point of failure
- Up scaling / Downscaling
- Deployment
- Advantages of Micro-services : Easy Debugging, No SOF, Scaling
- Disadvantages of Micro-services : DB Transaction, m-S communication, Expensive





Refactoring

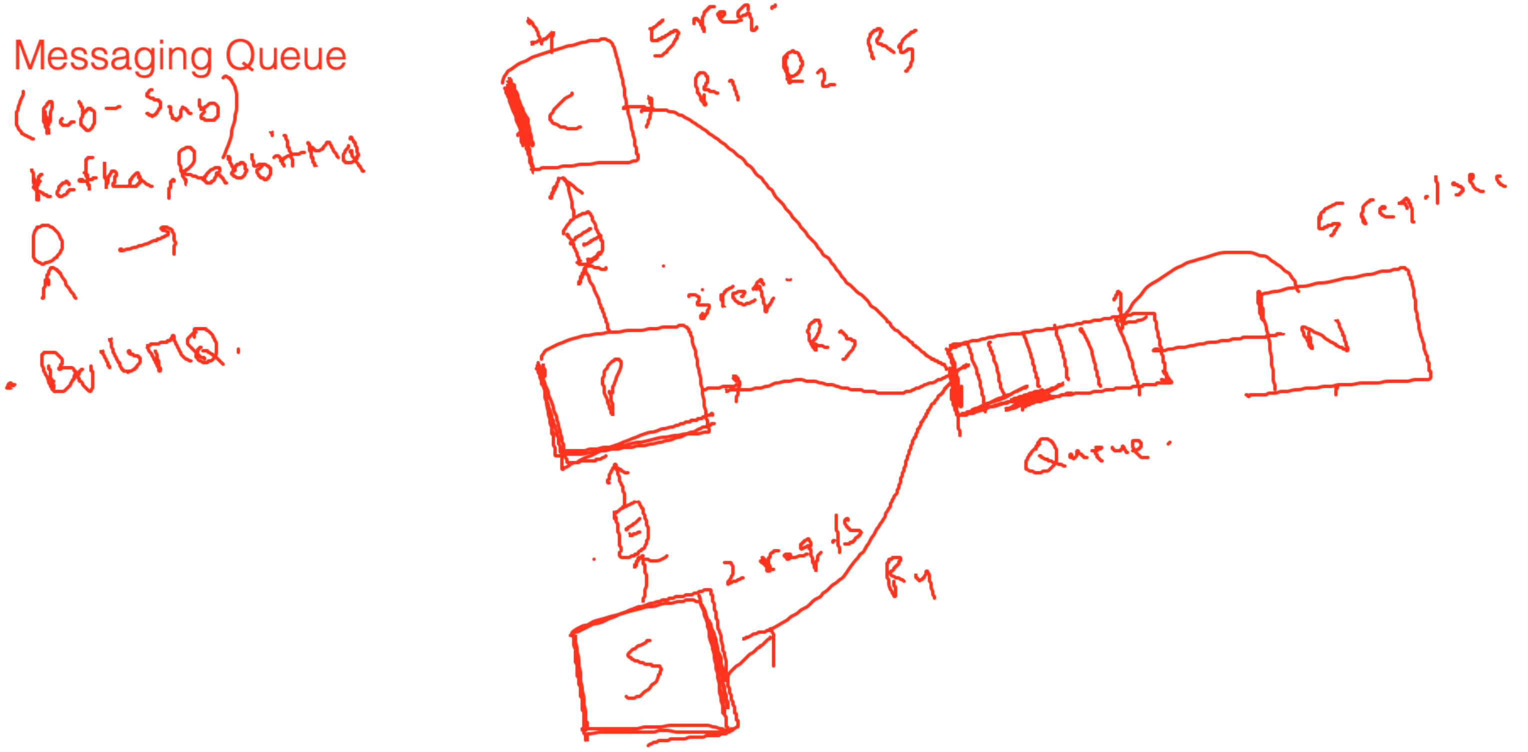
steps.

Microservices
↓
Design Pattern
eg: SAGA.

Communication b/w services.

Messaging Queue

- (Pub- Sub)
- Kafka, RabbitMQ
- BullMQ.



S 9
F 1

1-2/10

Pub → ?

Sub → ?

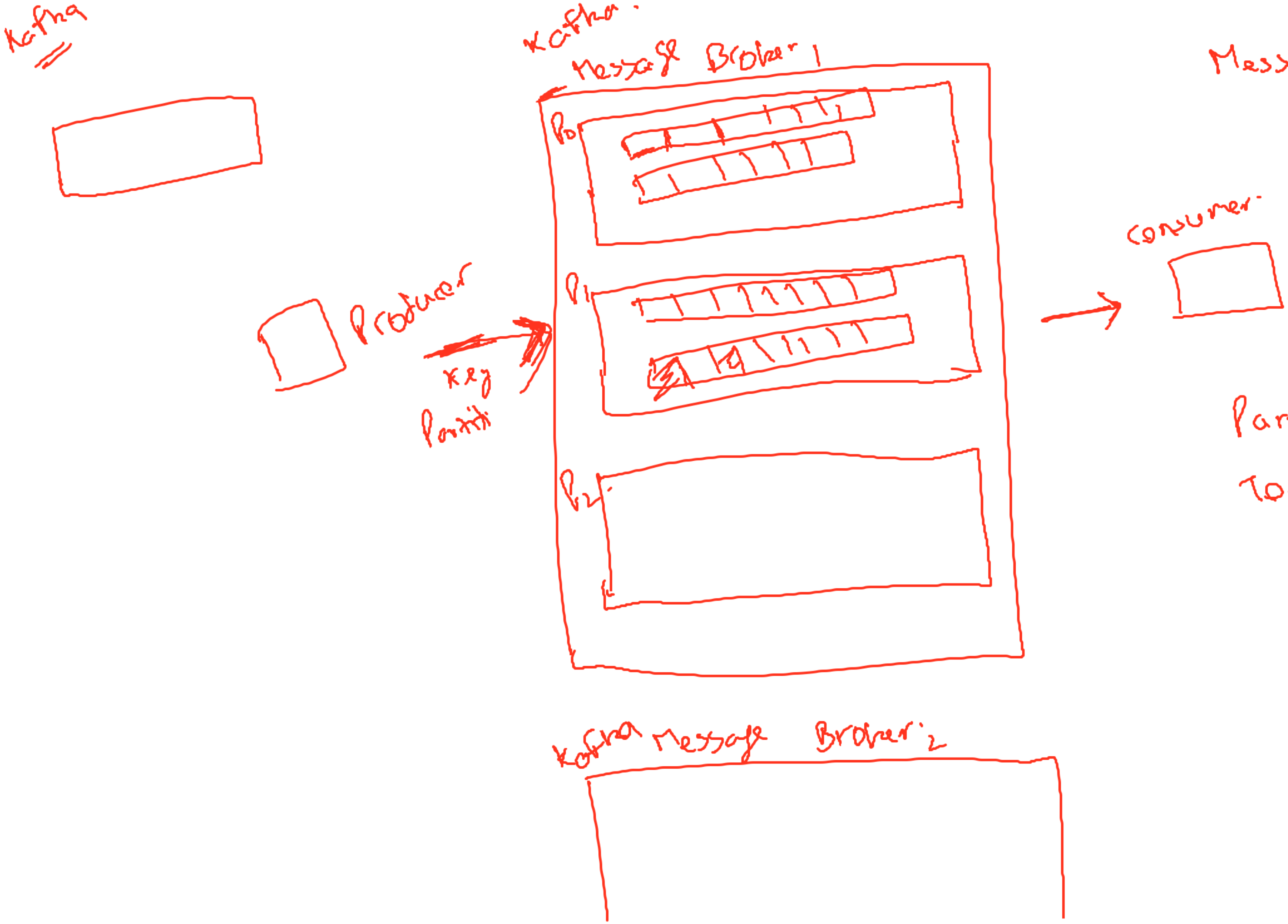
10k/D

Problem?

① Priority

② Retry functionality.

R-Attempt : 5 times.

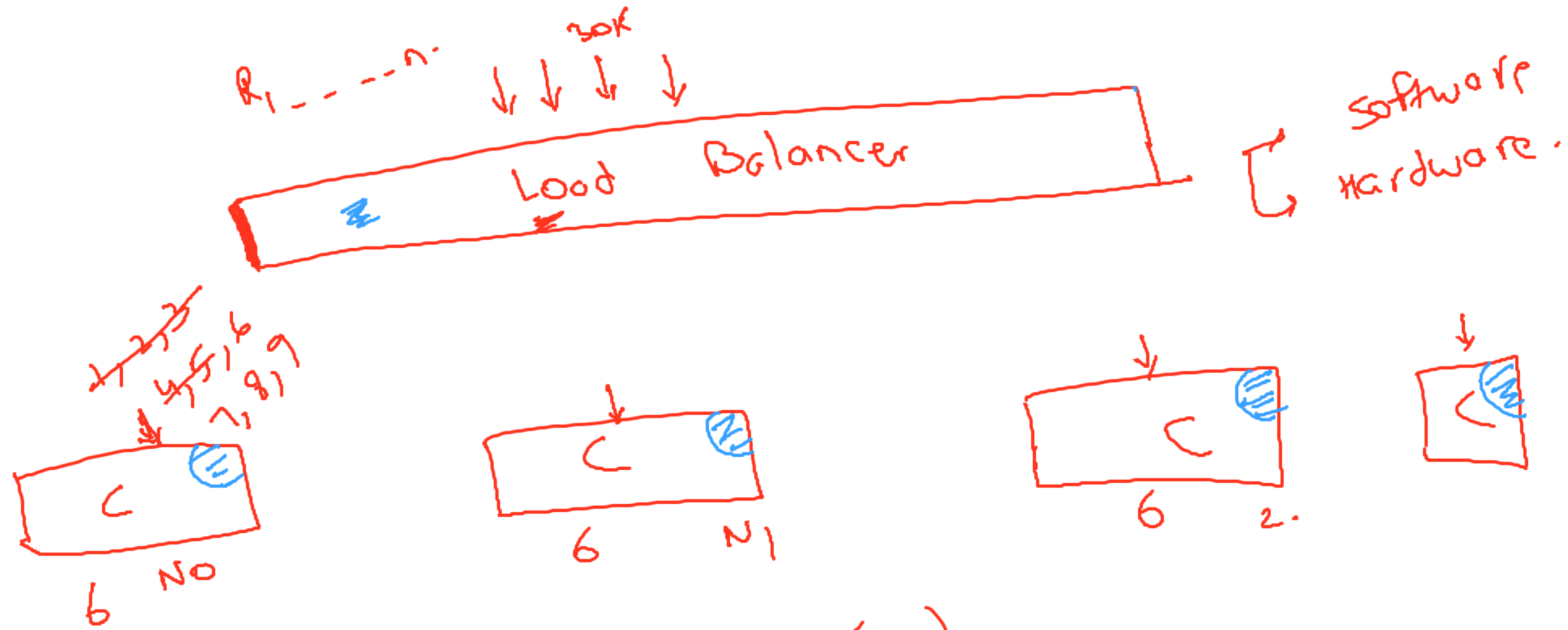


Message Brokers
↓
servers

Partition

Topics.

HLD



Normal Hashing → Hash function. (o/p)

$1 \% 3 \to 1$

$\% 4$

N0	3	6	9
N1	1	4	7
N2.	2	5	8

Scale↑↓ →

N0	4	8	
N1	1	5	9
N2	2	6	
N3	3	7	

Rebalancing: Large.

⇒ Consistent Hashing.

↳ Reduce the Rebalancing

Monolith vs Micro-services

- Refactoring
- System Designs
- Decompose / Steps to Refactor
- D, C, P, S
- Scaling of Micro-Services

Messaging Queue

- Communication b/w services
- Pub-Sub, Producer-Consumer
- Kafka, RabbitMQ, BullsMQ

LoadBalancer

- Normal Hashing
- Consistent Hashing