ML System Design-2

Start at 9:05

→ End-to-End Perspective

⇒ Design A video Streaming platform like Netflix

⇒. Ask requirements
- Provide your assumption

→ Functional Requirement

- → User | Team | Brand uploading
- → Stream | Interact with video
- → Personalised Recommendation

Non-Functional

- → High availability
- → Low Latency →
- → Scalability & Efficiency
- → Ease of use Ux

→ HLD → High Level Design

↳ Blueprint of services | Tech | models

LLD → Low Level design

→ • Specificity of services in HLD

→ Tip → ask for validation.

System Design →

1) Break down the problem → identify what services needs to be created

→ • upload servic.

• Streaming service → user can search → view → Like / dislike

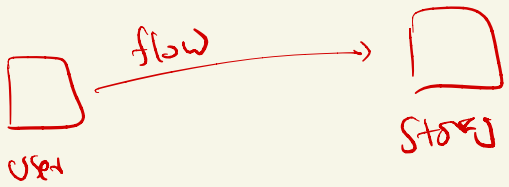
• Recommedation Service

2) Estimation / Assumptions →

Lo. How many videos would be uploaded

- Total users → 10M
- Total videos → 5000
- Beginning → Most users would be in US

3) Data → • where the data will come from
• what all data you will have access
• where will you store the data



Type of data →

① User data → Name, email, location, gender, subscription

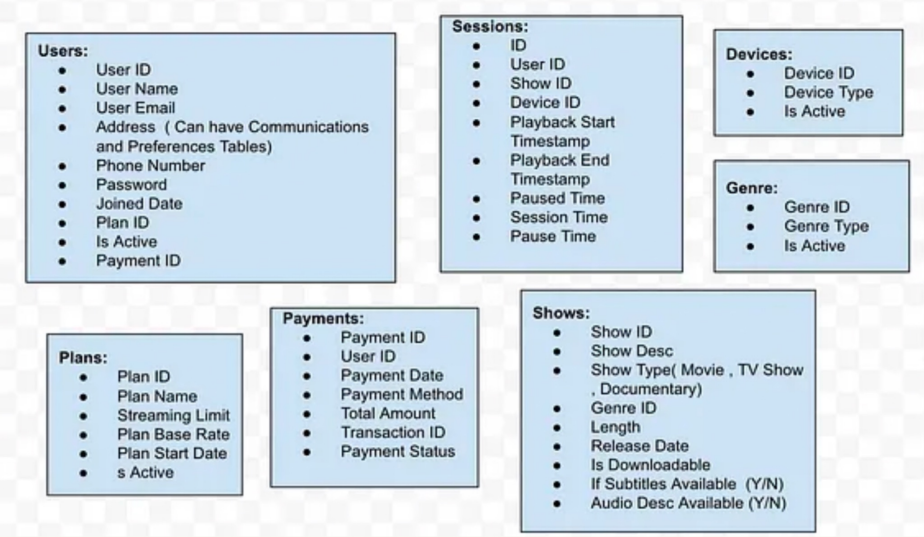
② user interaction → • views • Likes • Time spent
• clicks • Comment • Time of day

③ Video data (meta) → • genre, • Cast • release year
• avg rating, • url, • pics

④ Video →

⑤ Contextual → location, Device used,

⑥ External → • Social media interaction
• reviews



① How to the store data

② relationship b/w data

→ • One SQL database → • Postgresql

→ • one nosql db → • mongodb + S3

Netflix
- Postgres
- S3
- Cassandra

→ Distributed databases & Warehouses

→ Optimise the schema design

→ Caching mechanism → to reduce latency

→ • Aws RDS → Sql database
- Aws DynamoDB / Apache Cassandra → No sql datae
- Apache kafka / Aws kinesis → Streaming data

→ Data will grow very rapidly → Scale | Scaling

Scaling

Horizontal	Vertical Scaling
→ Increase the number of machine	→ Increasing the machine Specs
→ easier to upgrade	→ Harder to upgrade. Lo may have downtime
→ Harder to implement	→ easier to implement
→ Costlier	→ cheaper

→ Netflix → S3 → → Data lake → raw/unstructured
→ Data warehouse → structured data

S3 ATHENA → 10 CSVs
→ S3 [ ]

ATHENA can query underlying csv using sql

→ S3 is damn cheap →
↳ Glacier →

→ Most (99%) services → Count

⇒ Compute & Orchestration →

→ Cloud
- Scalable
- Secure
- Ease to use

vs

On-prem
- Sensitive data

→ Most Companies → have multiple Services

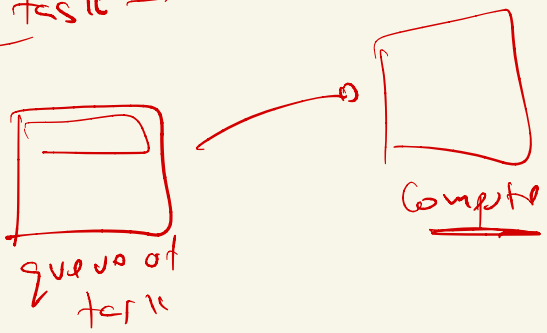
→ Loan Recommn → (Input user info) → Loan-Rec

→ Car price predictn

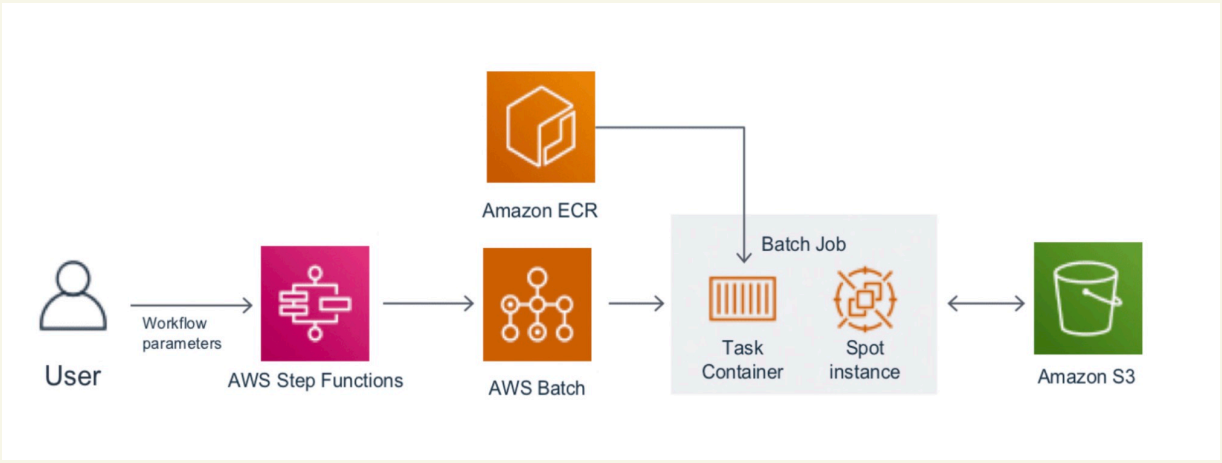
⇒ - few services → parallel
→ sequentially.

→ A lot of deep learning task →

→ Batch processe →



for optimum usage.



Stepfunctions → workflow automation service

↳ How many times to run a service

- what resources will it take.

Aws Batch → provide necessary compute to task.

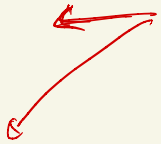
→ High availability → ensure that your service is up 100% time.

→ • keep properly scaling -

→ • Load balancing

→ app A

app B



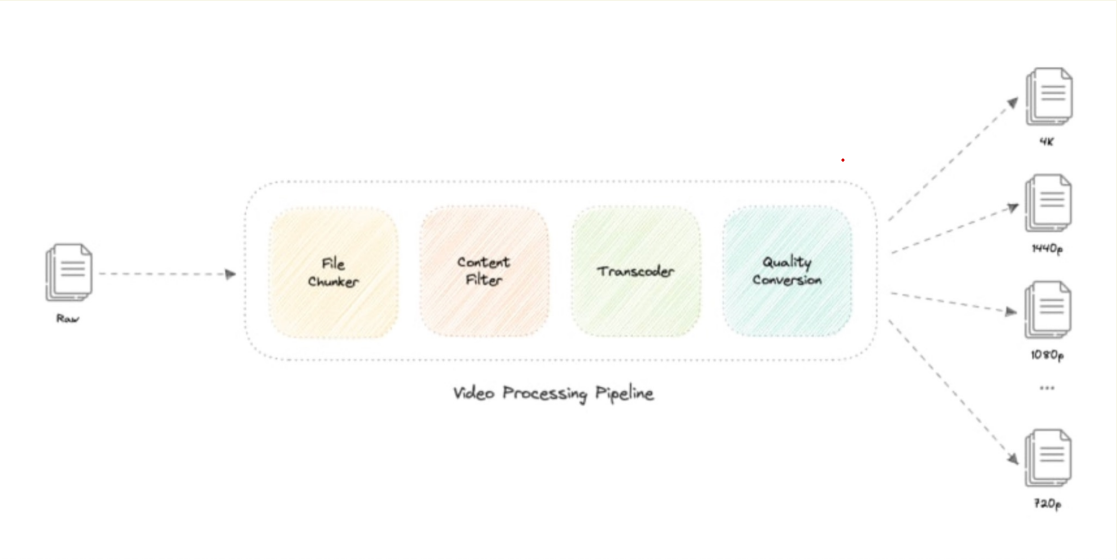
You have same service running in multiple places

→ Lower the pixel → lower the file size

Store the data in multiple resolution -

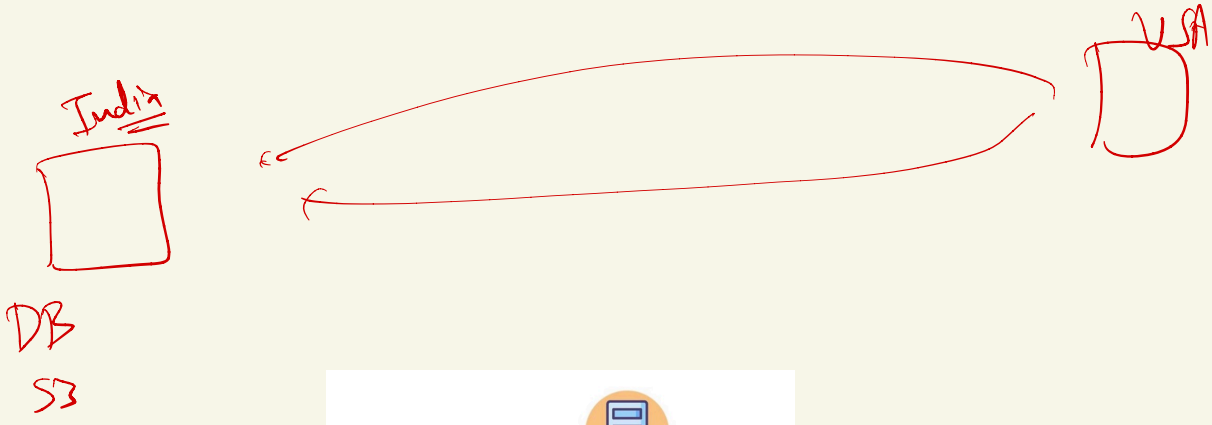
Good Big File → Compress → serve to user XXX

Big file → Compression → Store it → serve to user ✓



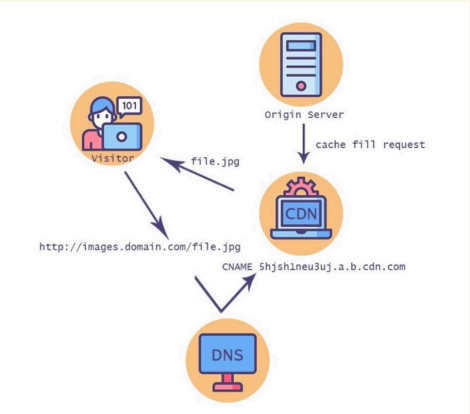
→ Speed things up → Low Latency

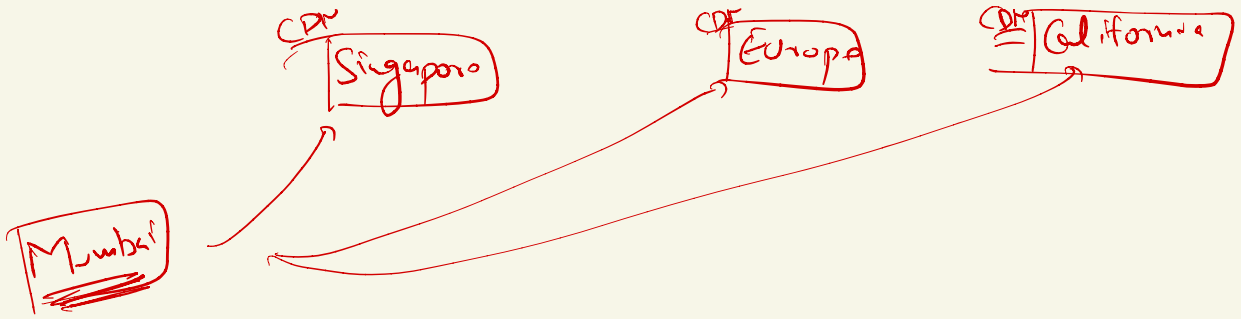
→ CDN → Content Delivery Network.



DB

S3





netflix.com/movies → DNS

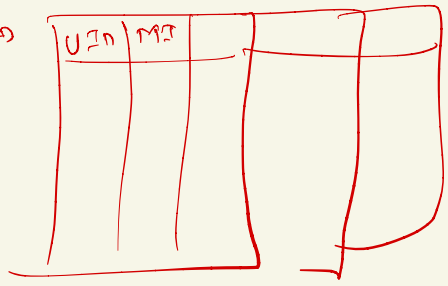
→ CDN → OCA → Low Latency

8 Mbps (Mumbai)

8 Mbps (Monetary)

→ User Interaction → NoSQL

	MOVIE-1	MOVIE-2	MOVIE-3	MOVIE-4
UID-1	LIKE - Yes TIMESTAMP 2:00:00 END No RECOMENDED Yes	LIKE - No TIMESTAMP 2:03:00 END Yes RECOMENDED No		
UID-2	LIKE Yes TIMESTAMP 1:58:21 END No RECOMENDED No	LIKE Yes TIMESTAMP 2:03:00 END Yes RECOMENDED Yes	LIKE No TIMESTAMP 1:20:00 END No RECOMENDED No	LIKE No TIMESTAMP 1:40:00 END Yes RECOMENDED Yes
UID-3	LIKE Yes TIMESTAMP 3:00:31 END Yes RECOMENDED Yes			

→ → 

→ Recommendation System →

→ Algo

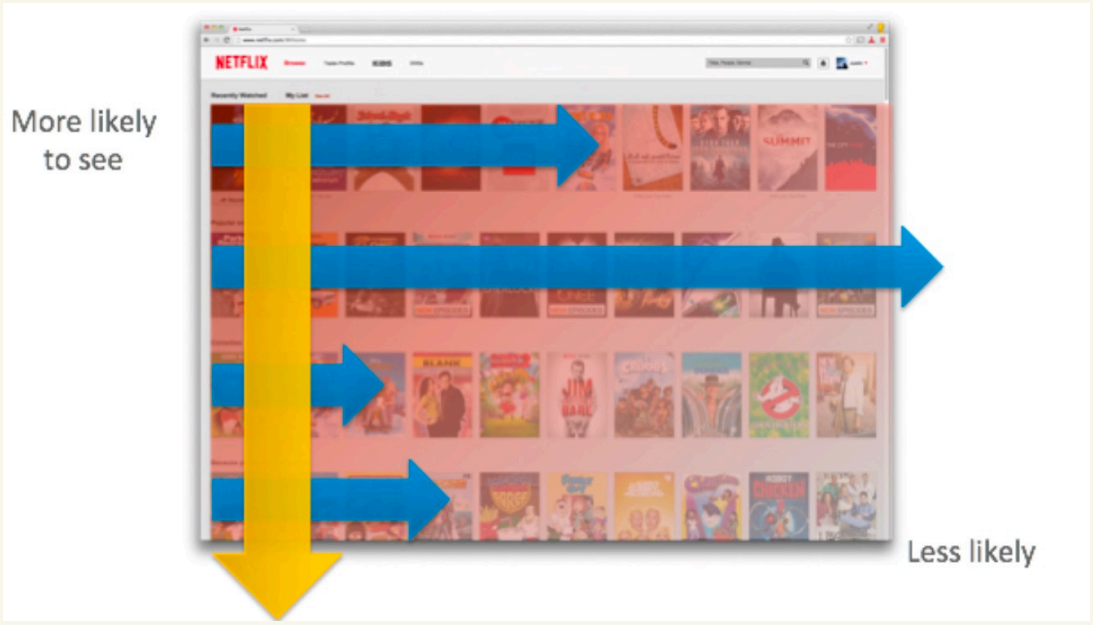
→ System Design →
↳ How many RS

→ Row level RS

→ 1 RS to define the sequence of Row

→ For every row you have are ranked

Content vs
→ Similarity → based recommending



→ 1) Identify less popular datapts to use in model

- User interaction
- ~~User~~ User similarity

- Time of day ←
- device ←
- avg time spent ←

→ 2) Identify the algo

→ we will do AB Testing →

↓

on good performance → we will deploy it