Design Netflix

## Clarifying Questions

1. We design the core product or a subsystem ?

2. Availability, latency, throughput for each part, prioritize them.

3. Users number or rows ? Global or national scale ?

4. upper Avg storage for all the (sub)system?

We go on Cloud storage in AWS EC3 (**NO EC2 - you stop/or end an instance its data are deleted**)or GCP for some 100nds of TBs ?

5. upper Avg data consumption of content per user (row)

6. design the (R)DBMS -> shards based on the pk: userid ?

7. the upper peak traffic of users, throughput.

8. distribution to avoid throughput bottleneck in one storage ?

use CDN for videos (static data) – static website content is added to proxy servers

that are globally distributed.

9. caching on between cdn and the rdbms ? for streamming to avoid latencies from DB l/O.

10. load balancer on the static content for the user API

11. input for the Recommendation engine? User metadata (logs) are the input to map reduce.

As input to HDFS we could have: {userid, event, video id}

-> (maybe parque compression), parquet files

**Map: takes tuples (key, value) per user.**

-> Aggregate logs on userId

-> Return intermediary key/value pairs index.

{"userId1": [("CLICK", "videoId1"), ("CLICK", "videoId1"), ..., ("PAUSE", "videoId2")]}

Intermediary k/v pairs will be shuffled appropriately and fed into our Reduce functions.

**Reduce**: we write function based on the target data the business wants to answer,

sorting and aggregating based on keys.   
Maybe if we want on score:

("userId1|videoId1", score)

- > output of HDFS.

Then ML engineers have the data for the recommendation engine.