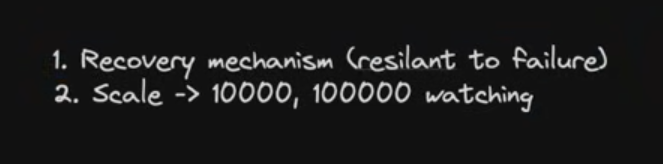
FUTURE UPDATES🡪



ADDING A RECOVERY MECHANISM🡪

Lets assume our web socket goes down while people being in a game

A screen shot of a computer

Description automatically generated

If ws server goes down , our game is lost , hence state is lost

We need to persist data in DB.

WE NEED STATELESS SERVER

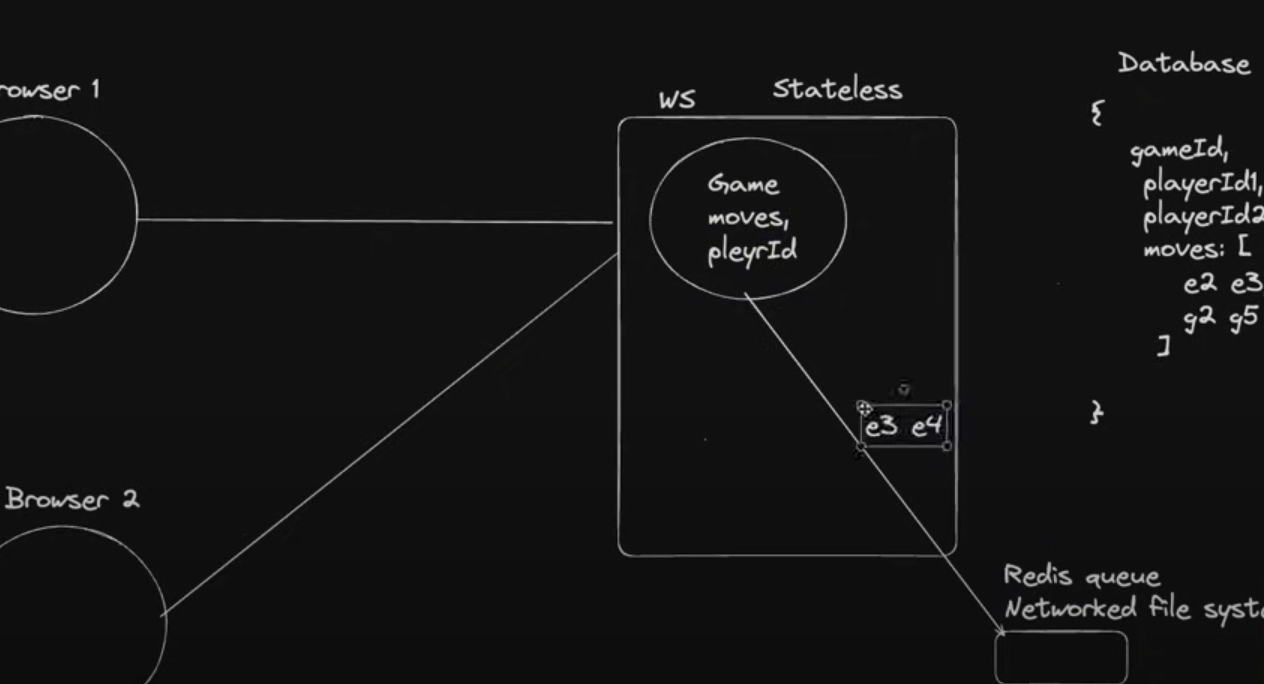
BUT THE THING IS IF WE DO NOT MAINTAIN A STATE IT BECOMES VERY EXPENSIVE TO FETCH FROM DB AND CHECK IF WE COULD MOVE , HENCE WE NEED TO MAINTAIN A STATE

A black background with white text

Description automatically generated

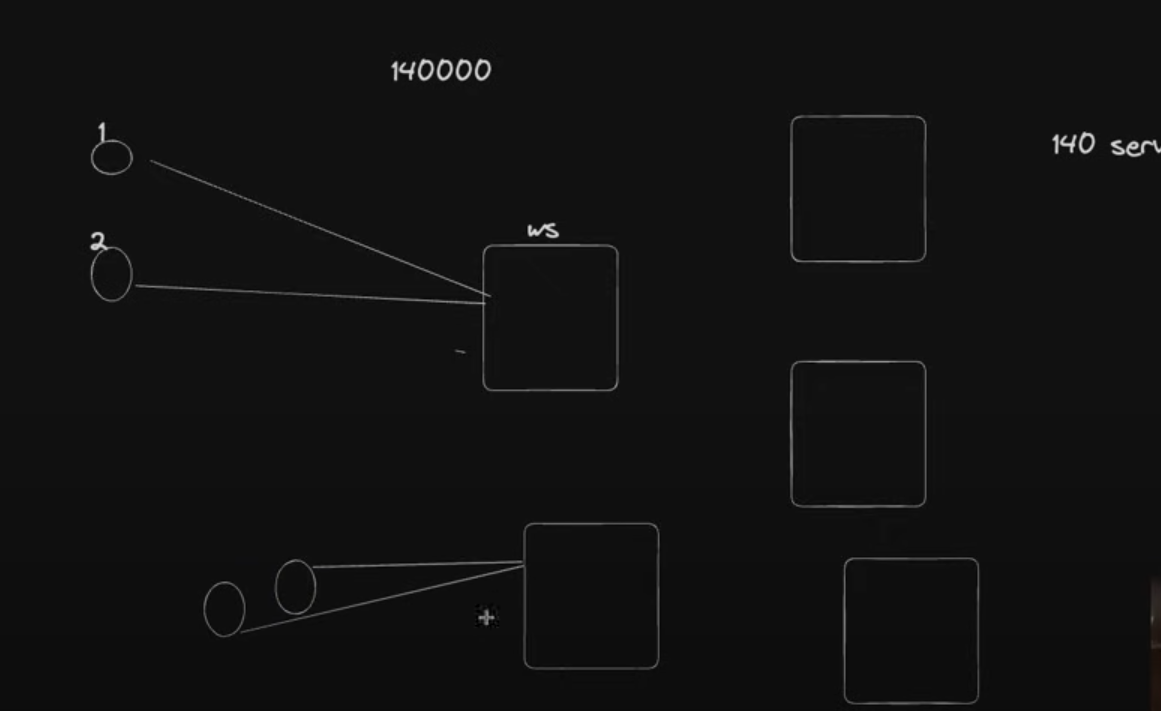
MAINTAINING THE STATE BOTH ON SERVER AND DB AND USING DB ONE WHEN SERVER CRASHES TO RESTORE THE SERVER STATE

BUT LATENCY WILL INCREASE



WHENEVER A MOVE HAPPENS WE COULD SEND IT TO REDIS QUEUE TO UPDATE THE DB ACCORDING TO THAT MOVE

HENCE LOW LATENCY

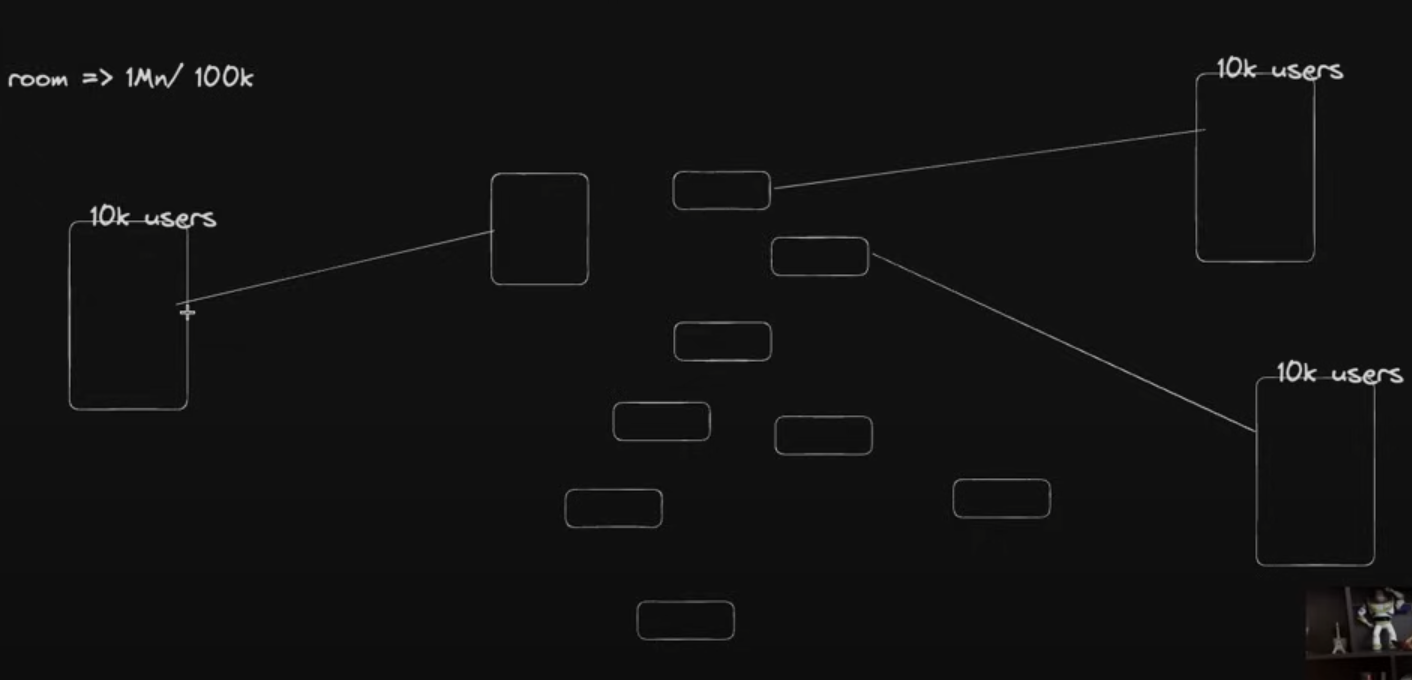
SCALING THE SYSTEM  


We can have 140 servers

But all we need to do is people playing with each other are connected to same ws server

We have implemented sharding

SYSTEM DESIGN APPROACH -2

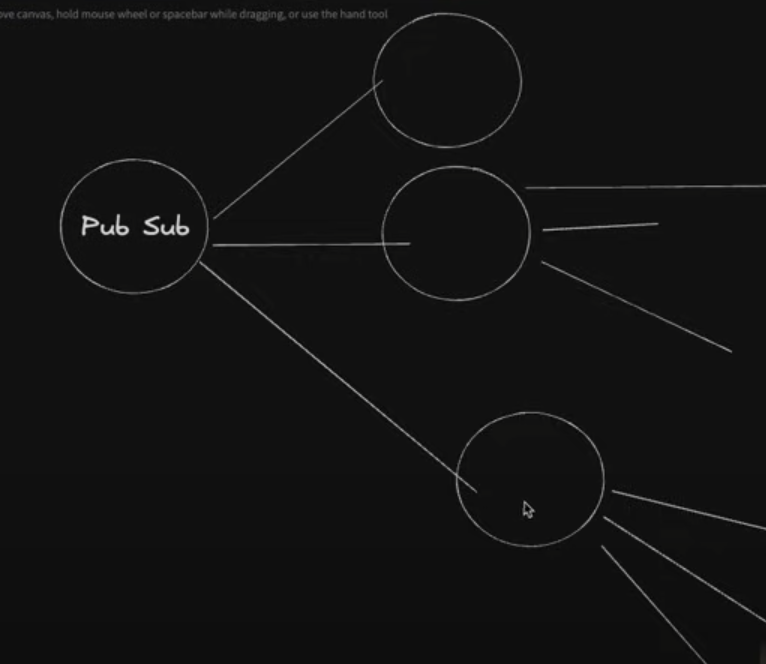


People in the same room , connected to different web servers

Different players connected to different web servers

🡪Whenever a move is played , it should be broadcasted to all rooms A screenshot of a computer

Description automatically generated

Pub/sub serves will publish it to all subs 

Million people connected to 100 servers , only 100 servers connected to a pub sub , hence no bottle neck