Rate limitter (RL)

- locks users request
  or
- allowes certain no. of requests in a certain time

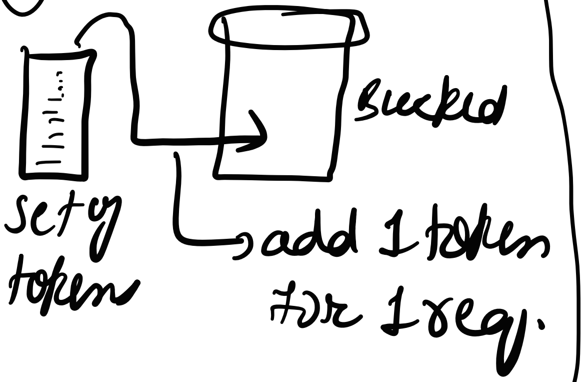
→ prevent DDOS
- Prevent cost
- Prevent/manage load of system

- ~~Server~~ Serverside RL

→ IP adders (most common)
↳ - always unique to
↳ - narrow down set of IPs which are bad one
↳ Send 429 status code to inform user that they are throttled
↳ Logging mechanism on our end to analyze the traffic

Technique to RL

1. Token bucket



spikes in traffic can let some users

prevent users to login

② Fixed window system

given a time window you have fix no. of ~~responses~~ req until that window no other req is entertained.

↳ downside → many req come towards end of window & start of window

↳ variation → sliding window
→ tweak window size as needed

Server side why

safe as org have control over it

components

server side

RL

API servers

Rule engine → algo & rules for RL

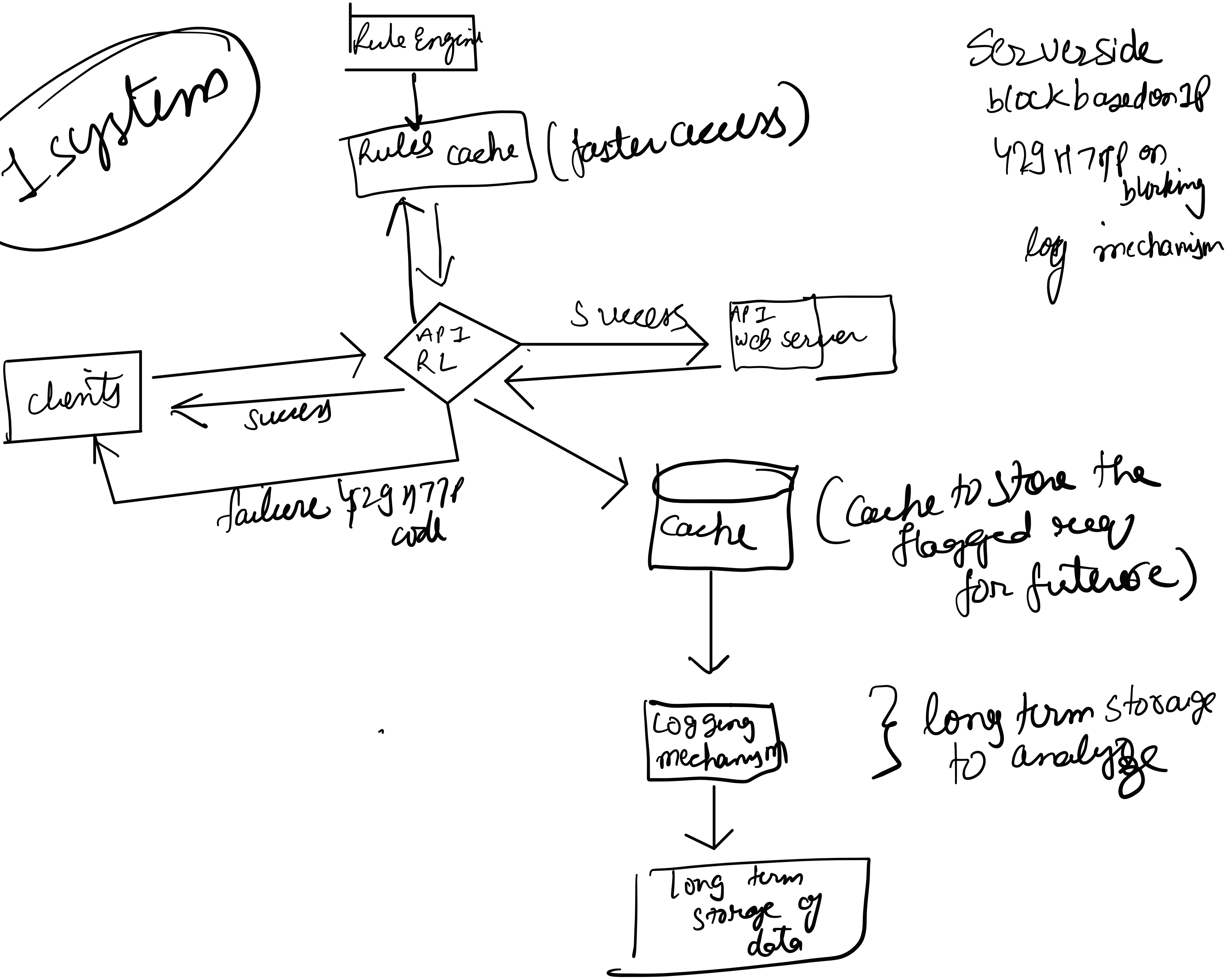
Cache high throughput

→ writes in cache as req comes & read from cache if it is already block

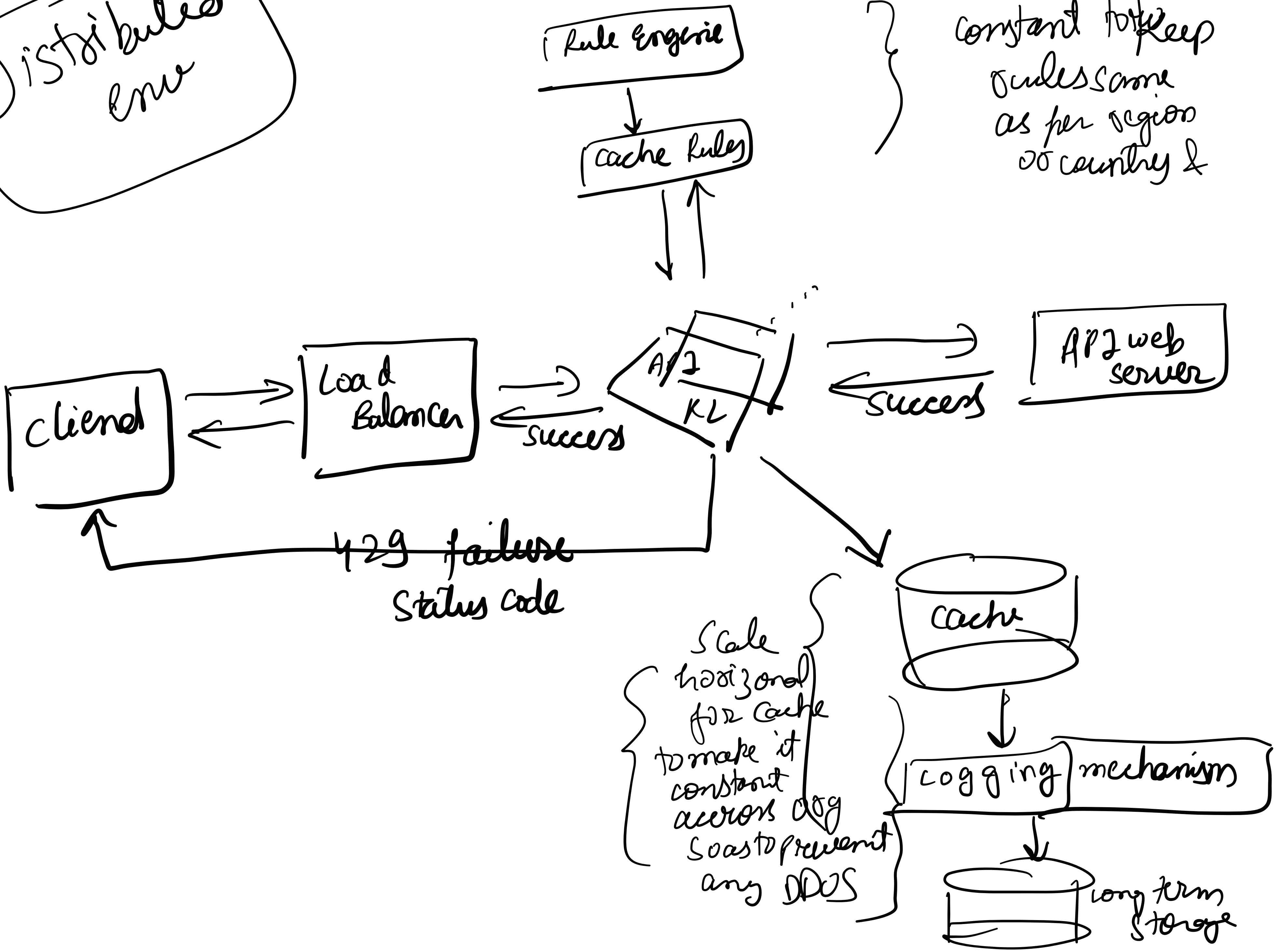
→ logging

1 systems

Serverside
block based on IP
429 HTTP on blocking
log mechanism



Distributed env



constant to keep rules same as per region or country &

Scale horizonal for cache to make it constant across org so as to prevent any DDOS

If you have different
caches then you will
Rate limit per datacenter
~~then~~ VS Rate limitting
across globl

diff config for cache can be used
like 1Read & 1W cache
or 1RW cache
across all datacenter

→ Rule cache can have multiple
instances as Rules
won't change over a
period of time

→ It is not good to keep different RL
based on geography as ppl can bypass RL based on VPN