Boxes / Towers

↳ barebone computers are called Servers: Servers is something that serves requests

* the server in turn takes this requests and sends back a response

* Instagram/FB/etc they have millions of servers

↳ when a person does any action ex) look-up a user

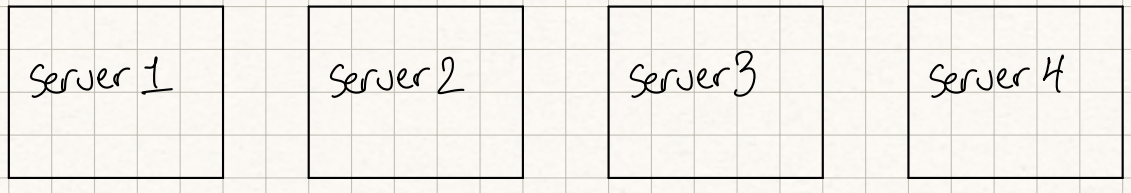
↳ that person is sending a request

* Social Networks have millions/billions of users

↳ these users are all making different types of requests

↳ they are making millions of requests

* the single server cannot handle all these requests, so the service slows down ⟶ so we acquire more servers



* however, how do we distribute requests among our servers?

↳ We use a load balancer to evenly distribute the number of requests each server gets

Requests have Request IDs

↳ these IDs are random
and we can say they're randomly generated from 0 to M-1

↳ We take this request ID (we'll refer to it as r_1)
we hash it $r_1 \rightarrow m_1$

$h(r_1) \rightarrow m_1$

m_1 can be mapped to a particular server

how? → $m_1 \% n$ (n being the # of servers)

↳ the result of this, we send the hashed request (m_1) to the respective server

ex) $h(r_1) \rightarrow m_1 \% 4 = 2$

↳ this request will go to server 2.

* So in general → the hash function should be uniformly random

↳ you can expect all of the servers to have uniform load

* if you have x requests

you will have $\frac{x}{n}$ load and

your load factor is $\frac{1}{n}$

* n being the number of servers

* What if we add more servers?

* If we add more servers it will disrupt the flow.

* requestIDs are rarely different and usually have user information

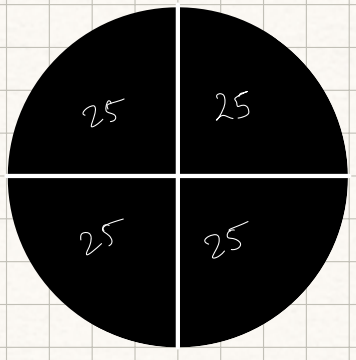
so if we have 4 servers

* a specific user making requests will usually be sent to the same server

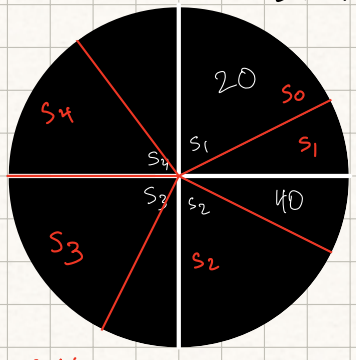
* being stored in the cache making their requests be ressolved faster

but if we increase the number of servers by too much, it disrupts the load for each server

And all the useful cache information you had gets dumped b/c the load each server had completely changed



*when we had 4 servers each had 25% of the workload



Server 1 lost 5 buckets (+5)
server 2 gained those buckets (+5)
(+10)
(+10)
(+15)
(+15)
(+20)
(+20)
100

* there was a swap of 100 requests per server

ex) request that would've gone to server 0 now go to server 1 etc

* we can add servers, but we need to minimize the overall change per server so instead of 100 swaps we could do 20? something smaller

Consistent Hashing