Day 05: local LB
global LB: latency; policies; maintenance

cluster LB: utilization

Options
1. round robin
2. random
3. track load & assign based on load
4. [Diagram: client routing to random LB, which flips a coin based on coin flip and picks a server]

[Diagram: DNS]
Stateful LB (key/value or cache)

failure = potential to lose state (need to login again 😞)

1. Store data in database
2. Put it all on the client/have it store like state
3. Create backup LB (replicas)
<table>
<thead>
<tr>
<th>DB</th>
<th>client</th>
<th>replicas</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. added latency for DB access</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. need a distributed system for the DB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. security</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. total end-to-end control of infra (client + LB + server)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. keeping the data up to date between the primary &amp; the backups</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. more servers</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. Computation cost
2. Control map from hash to servers

hash(c) \rightarrow \text{input is some}
\quad \downarrow \quad \text{output is some random}
\quad \text{for example, random string} = \text{hash (1-1-1-t)}

[0, 255]
(1) all LB have to agree on hash(1)
(2) all LB have to agree on the definition of Buckets
Consistent Hashing

key Space

client = key ↦

server = key

⇒ hash (server IP) mod N

hash (client IP) mod N

makes this random

0 to N-1

new servers

S1, S2, S3

clients

0 5 10 128 N-1