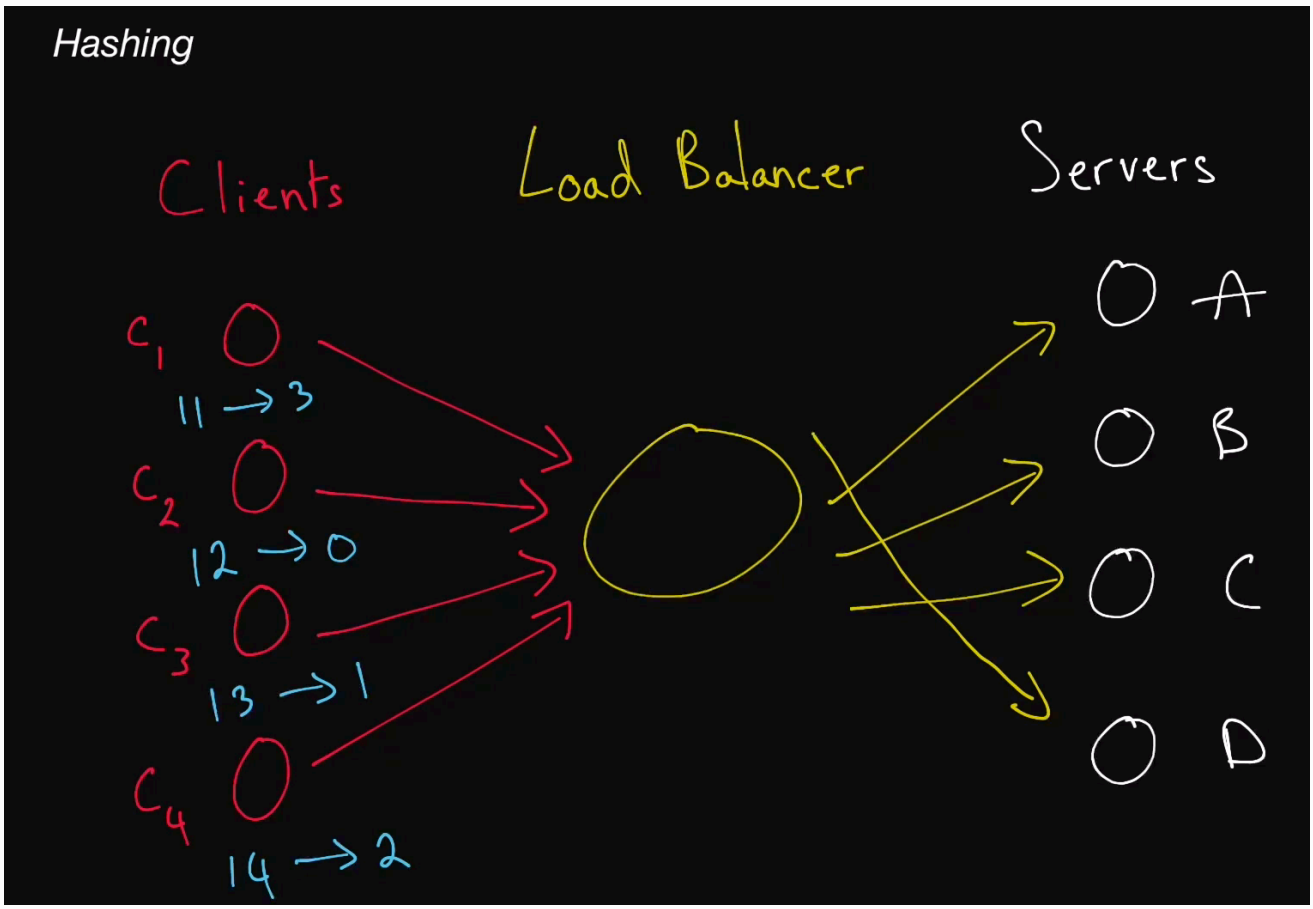
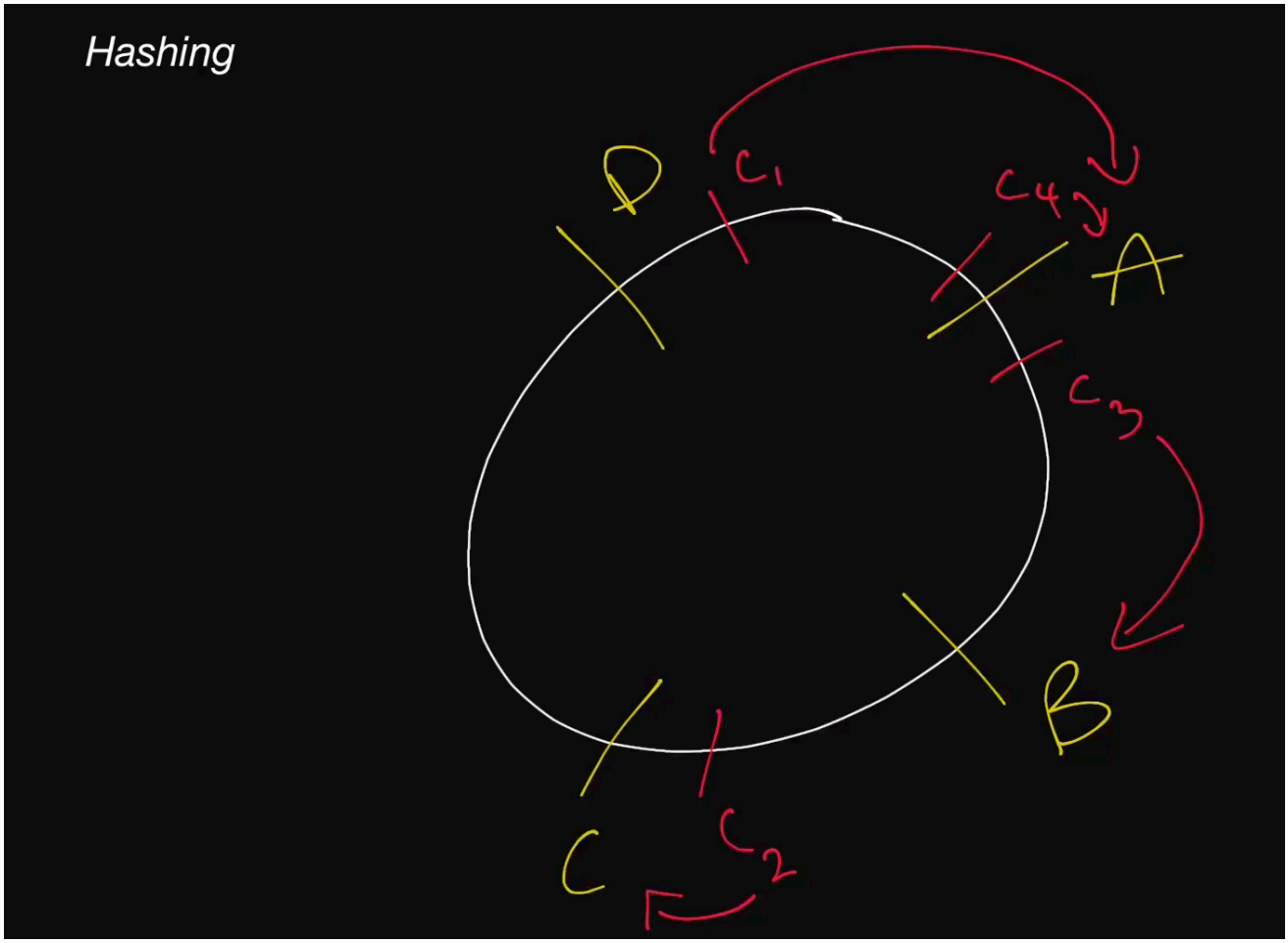

Hashing
Clients
Load Balancer
Servers
C1
11 → 3
C2
12 → 0
C3
13 → 1
C4
14 → 2
A
B
C
D


Hashing
D
C1
C4
A
C3
B
C
C2

hashing_example.js — hashing

hashing_example.js

hashing_utils.js

hashing_example.js > serverSet1

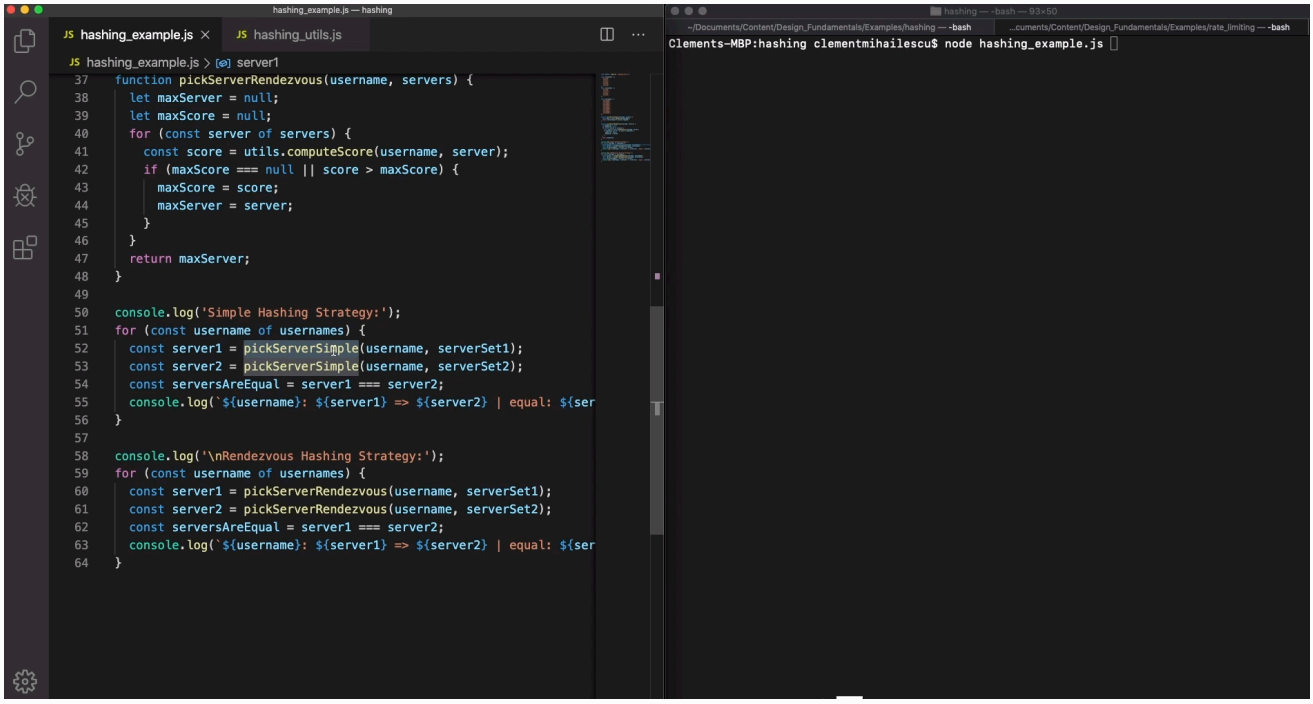
```
const utils = require('./hashing_utils');

const serverSet1 = [
  'server0',
  'server1',
  'server2',
  'server3',
  'server4',
  'server5',
];
const serverSet2 = [
  'server0',
  'server1',
  'server2',
  'server3',
  'server4',
];

const usernames = [
  'username0',
  'username1',
  'username2',
  'username3',
  'username4',
  'username5',
  'username6',
  'username7',
  'username8',
  'username9',
];

function pickServerSimple(username, servers) {
  const hash = utils.hashString(username);
  return servers[hash % servers.length];
}
```

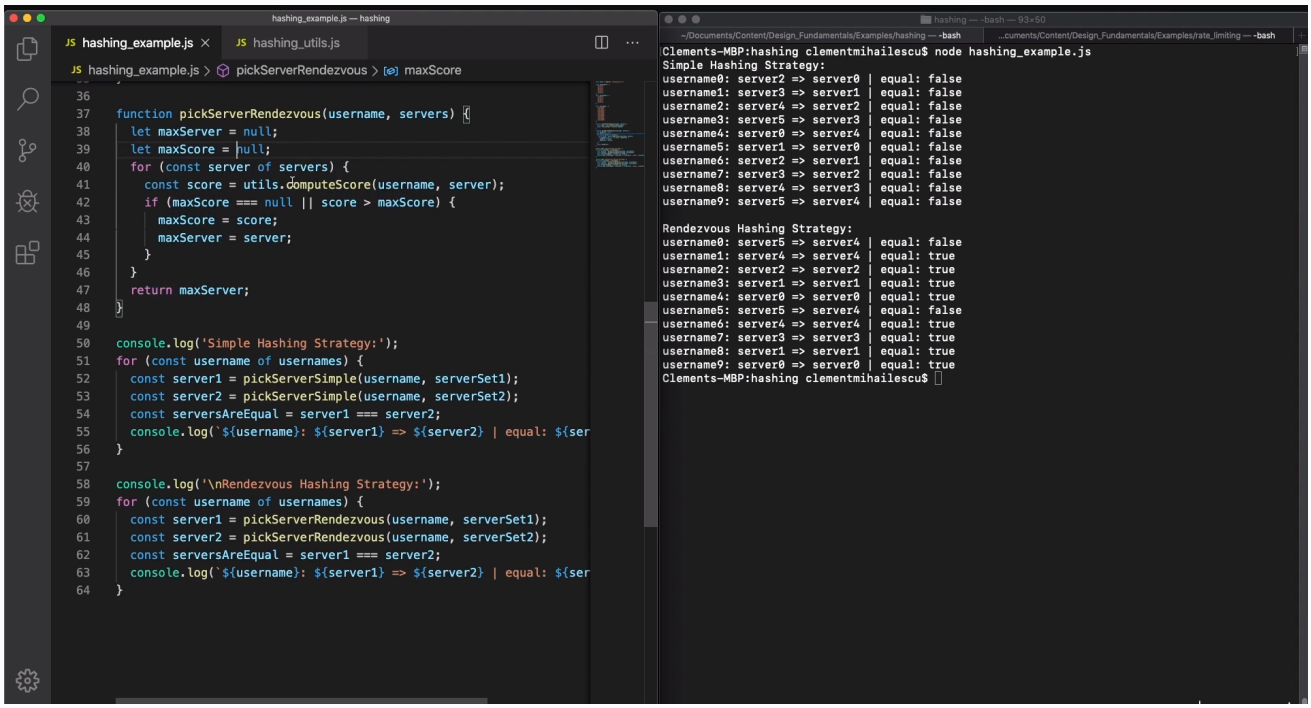
hashing — -bash — 93×50

~/Documents/Content/Design_Fundamentals/Examples/hashing — -bash

...cuments/Content/Design_Fundamentals/Examples/rate_limiting — -bash

```
Clements-MBP:hashing clementmihailescu$ node hashing_example.js
```


hashing_example.js — hashing
hashing_example.js
hashing_utils.js
hashing_example.js > server1
function pickServerRendezvous(username, servers) {
let maxServer = null;
let maxScore = null;
for (const server of servers) {
const score = utils.computeScore(username, server);
if (maxScore === null || score > maxScore) {
maxScore = score;
maxServer = server;
}
}
return maxServer;
}
console.log('Simple Hashing Strategy:');
for (const username of usernames) {
const server1 = pickServerSimple(username, serverSet1);
const server2 = pickServerSimple(username, serverSet2);
const serversAreEqual = server1 === server2;
console.log(`${username}: ${server1} => ${server2} | equal: ${ser
}
console.log('\nRendezvous Hashing Strategy:');
for (const username of usernames) {
const server1 = pickServerRendezvous(username, serverSet1);
const server2 = pickServerRendezvous(username, serverSet2);
const serversAreEqual = server1 === server2;
console.log(`${username}: ${server1} => ${server2} | equal: ${ser
}
hashing — -bash — 93×50
~/Documents/Content/Design_Fundamentals/Examples/hashing — -bash
...cuments/Content/Design_Fundamentals/Examples/rate_limiting — -bash
Clements-MBP:hashing clementmihailescu$ node hashing_example.js


hashing_example.js — hashing
hashing_example.js
hashing_utils.js
hashing_example.js > pickServerRendezvous > maxScore
function pickServerRendezvous(username, servers) {
let maxServer = null;
let maxScore = null;
for (const server of servers) {
const score = utils.computeScore(username, server);
if (maxScore === null || score > maxScore) {
maxScore = score;
maxServer = server;
}
}
return maxServer;
}
console.log('Simple Hashing Strategy:');
for (const username of usernames) {
const server1 = pickServerSimple(username, serverSet1);
const server2 = pickServerSimple(username, serverSet2);
const serversAreEqual = server1 === server2;
console.log(`${username}: ${server1} => ${server2} | equal: ${ser
}
console.log('\nRendezvous Hashing Strategy:');
for (const username of usernames) {
const server1 = pickServerRendezvous(username, serverSet1);
const server2 = pickServerRendezvous(username, serverSet2);
const serversAreEqual = server1 === server2;
console.log(`${username}: ${server1} => ${server2} | equal: ${ser
}
hashing — -bash — 93×50
~/Documents/Content/Design_Fundamentals/Examples/hashing — -bash
...cuments/Content/Design_Fundamentals/Examples/rate_limiting — -bash
Clements-MBP:hashing clementmihailescu$ node hashing_example.js
Simple Hashing Strategy:
username0: server2 => server0 | equal: false
username1: server3 => server1 | equal: false
username2: server4 => server2 | equal: false
username3: server5 => server3 | equal: false
username4: server0 => server4 | equal: false
username5: server1 => server0 | equal: false
username6: server2 => server1 | equal: false
username7: server3 => server2 | equal: false
username8: server4 => server3 | equal: false
username9: server5 => server4 | equal: false
Rendezvous Hashing Strategy:
username0: server5 => server4 | equal: false
username1: server4 => server4 | equal: true
username2: server2 => server2 | equal: true
username3: server1 => server1 | equal: true
username4: server0 => server0 | equal: true
username5: server5 => server4 | equal: false
username6: server4 => server4 | equal: true
username7: server3 => server3 | equal: true
username8: server1 => server1 | equal: true
username9: server0 => server0 | equal: true
Clements-MBP:hashing clementmihailescu$

2 Prerequisites

Hashing Function

A function that takes in a specific data type (such as a string or an identifier) and outputs a number. Different inputs *may* have the same output, but a good hashing function attempts to minimize those **hashing collisions** (which is equivalent to maximizing **uniformity**).

Load Balancer

A type of **reverse proxy** that distributes traffic across servers. Load balancers can be found in many parts of a system, from the DNS layer all the way to the database layer.

3 Key Terms

Consistent Hashing

A type of hashing that minimizes the number of keys that need to be remapped when a hash table gets resized. It's often used by load balancers to distribute traffic to servers; it minimizes the number of requests that get forwarded to different servers when new servers are added or when existing servers are brought down.

Rendezvous Hashing

A type of hashing also coined **highest random weight** hashing. Allows for minimal re-distribution of mappings when a server goes down.

SHA

Short for "Secure Hash Algorithms", the SHA is a collection of cryptographic hash functions used in the industry. These days, SHA-3 is a popular choice to use in a system.