Scaling Strategies

From Vertical to Horizontal Scaling

Learning Objectives

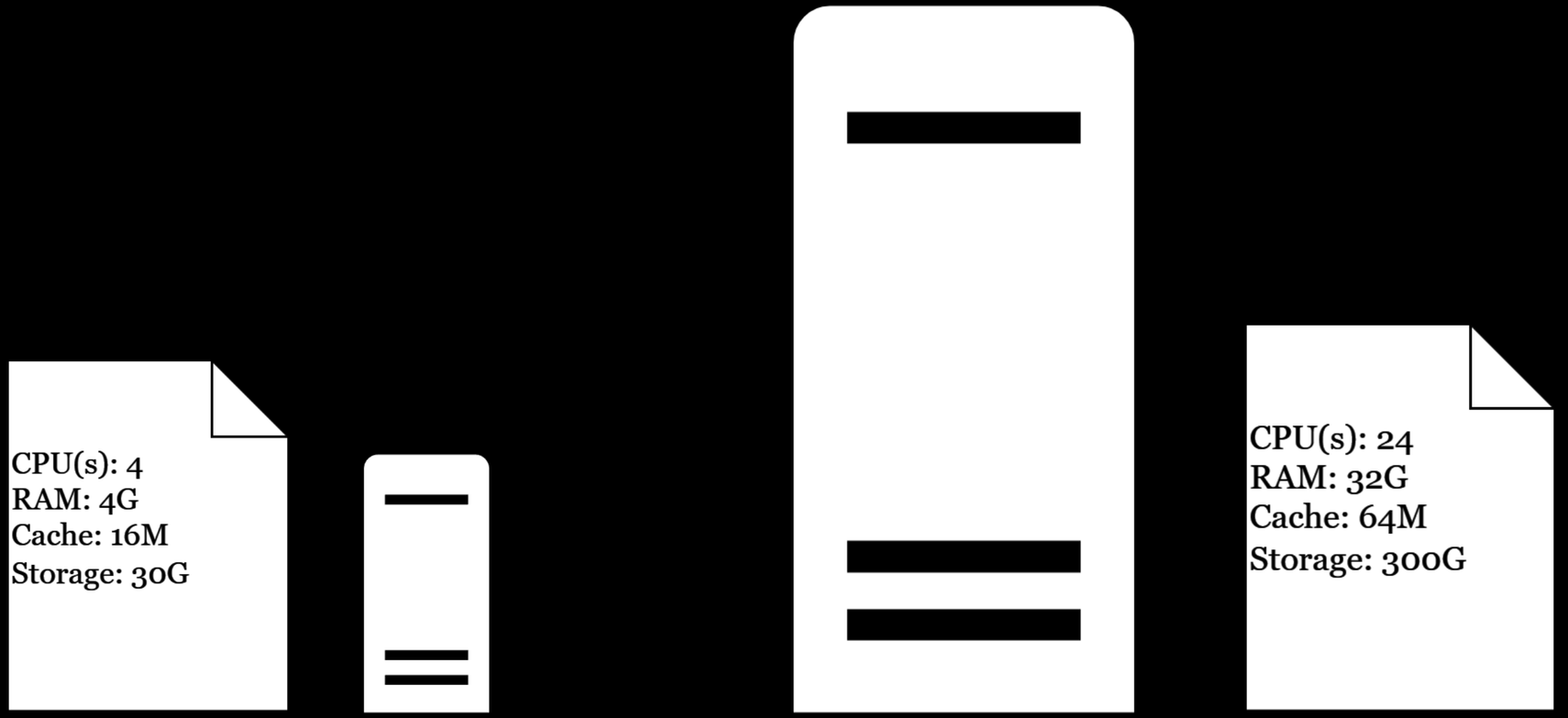
By the end of this session, you will have acquired the following information:

- Vertical Scaling (Scaling Up)
- Horizontal Scaling (Scaling Out)
- Consistent Hashing
- Load Balancing

Illustrative Example

- Alireza is a regular customer of an e-commerce website.
- He attempted to make a purchase during Black Friday.
- Adding a product to his basket took about 3 seconds.
- The website typically handles 10,000 transactions per month.
- On Black Friday, the system was overloaded with 5,000 transactions in a single day.

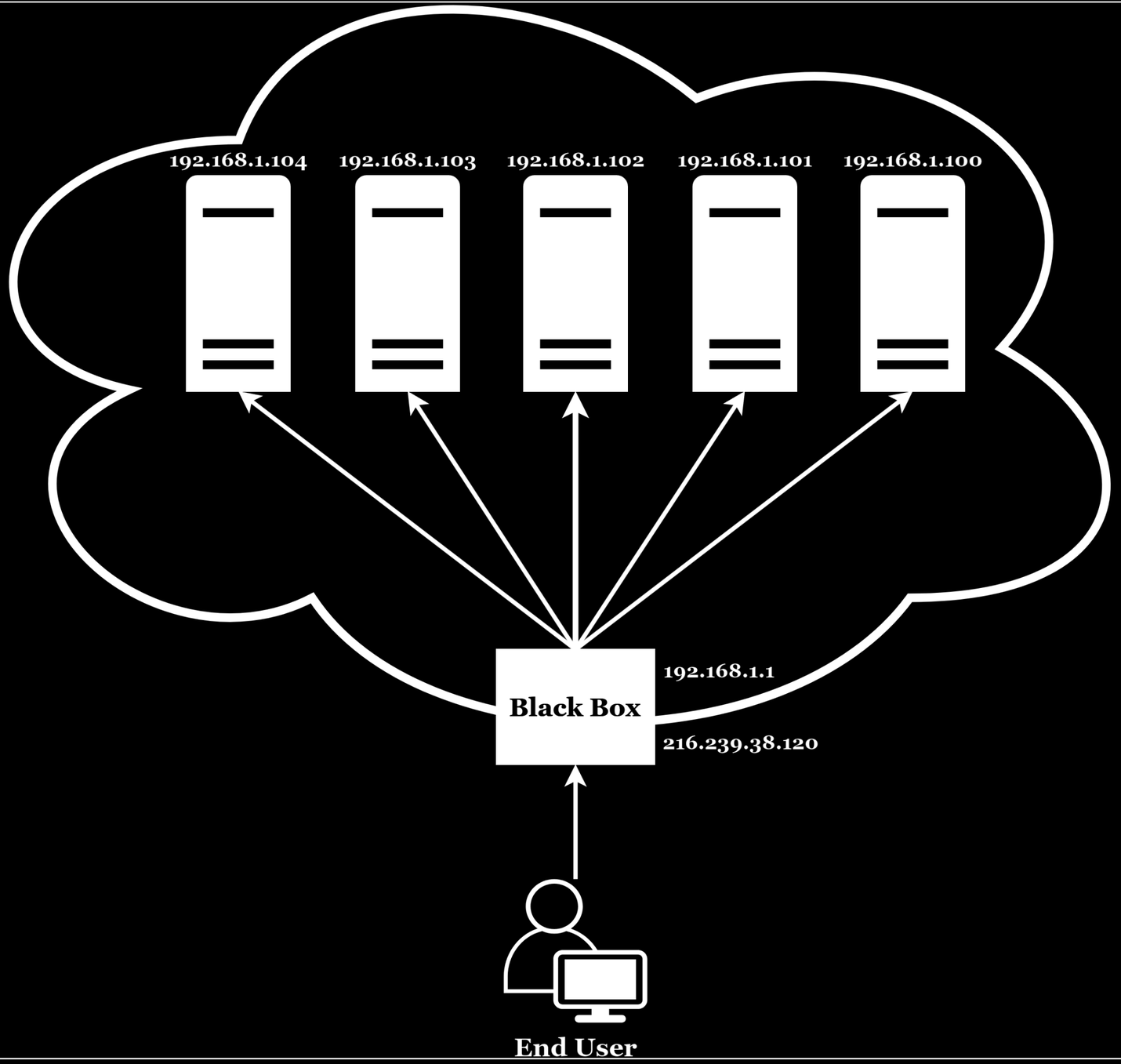
Vertical Scaling


CPU(s): 4
RAM: 4G
Cache: 16M
Storage: 30G
CPU(s): 24
RAM: 32G
Cache: 64M
Storage: 300G

Illustrative Example

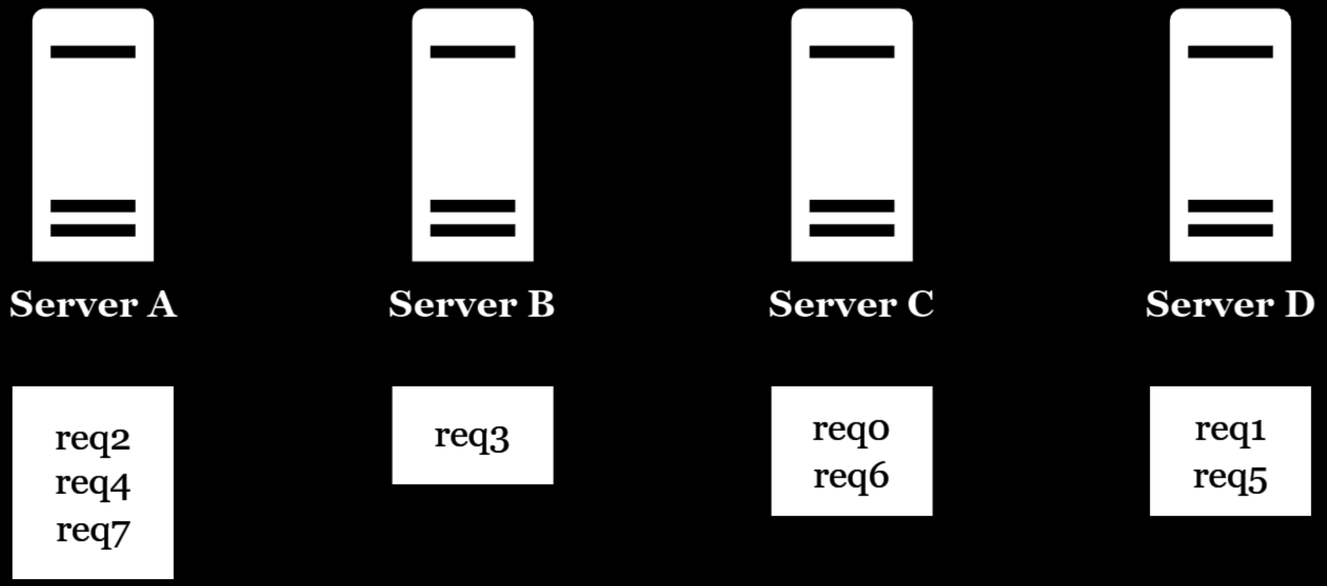
- Alireza is a regular customer of an e-commerce website.
- He attempted to make a purchase during Black Friday.
- Adding a product to his basket resulted in a Service Unavailable response.
- The website typically handles 10,000 transactions per month.
- On Black Friday, the system was overloaded with 5,000 transactions in a single day.

Horizontal Scaling

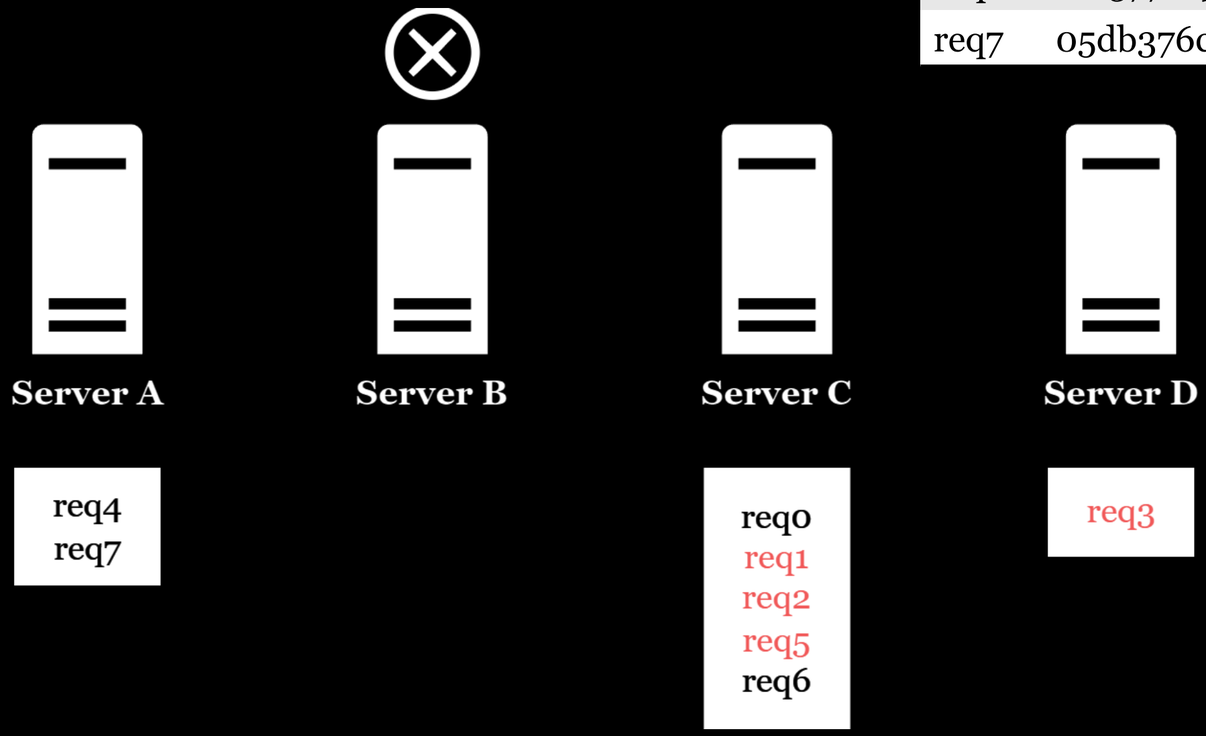

192.168.1.104
192.168.1.103
192.168.1.102
192.168.1.101
192.168.1.100
Black Box
192.168.1.1
216.239.38.120
End User

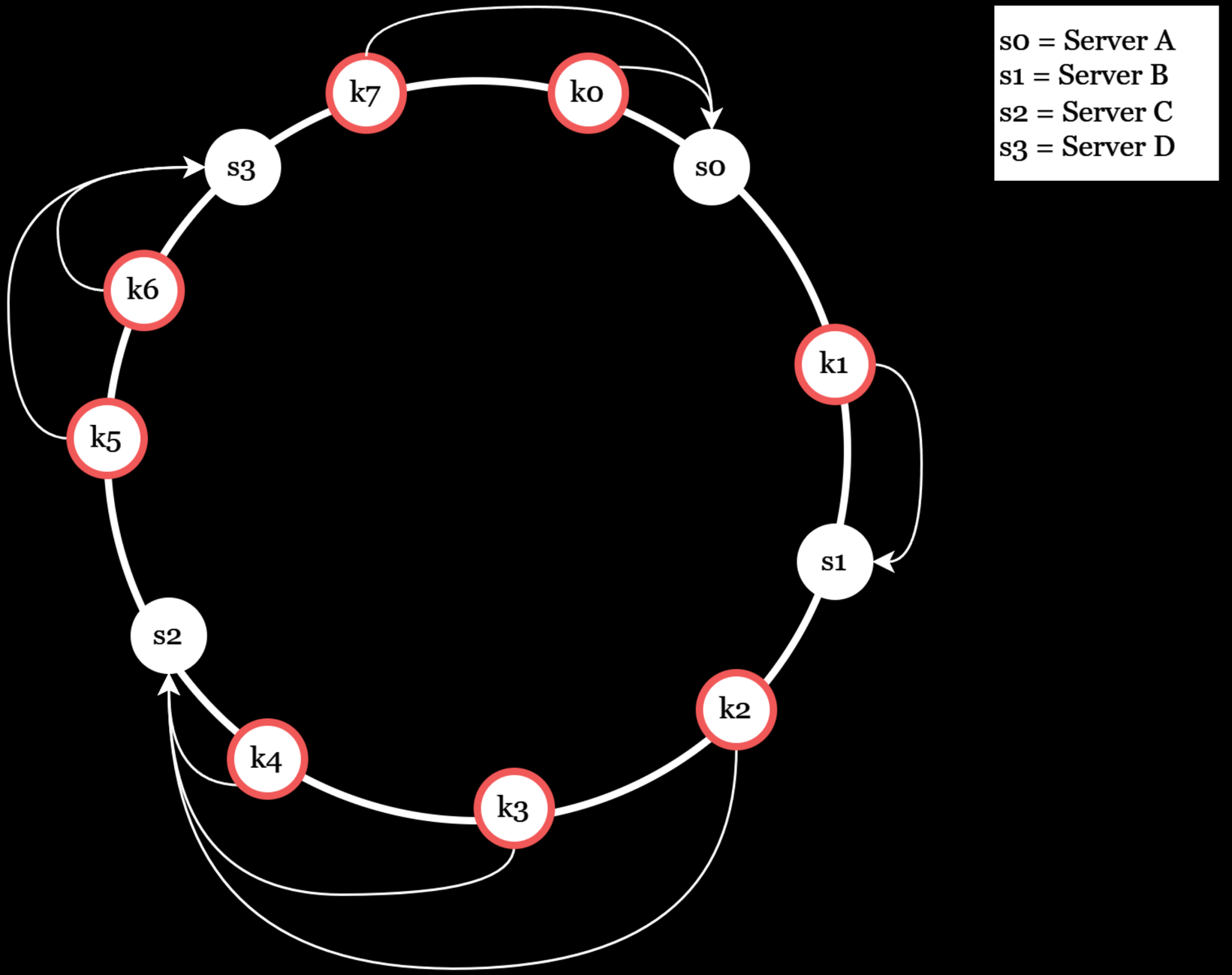
Consistent Hashing

Key	Hash	Hash % 4
req0	adb1ef332d1f6e99e809fb9b00a08efcad930e82	2
req1	1073ab6cda4b991cd29f9e83a307f34004ae9327	3
req2	87ba78e0f03afcef60657f342ec5567368fadd8c	0
req3	3b88ea816c78ec104041a75e78f32ec804eaac39	1
req4	c34bf5a9ecca6edc3128018b1dd235a0f7bdff20	0
req5	af065e03e22fe1f5f95b8ce9f4761c74da076857	3
req6	6df377ec91a0df5f054484fbfd0c13d7ed27d832	2
req7	05db376c6fa6453bd9c80b31d2c675977851be34	0

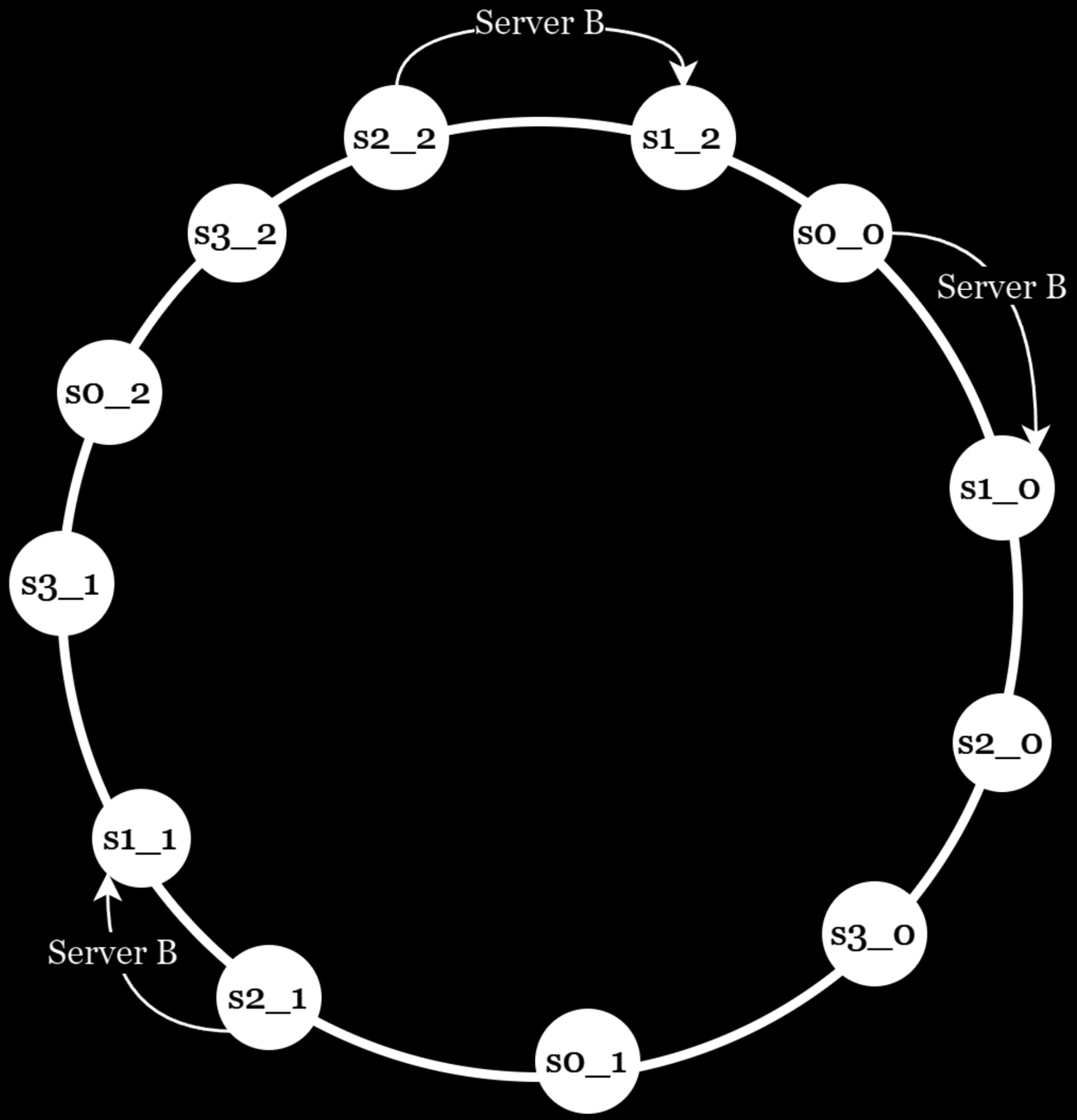


Key	Hash	Hash % 3
req0	adb1ef332d1f6e99e809fb9b00a08efcad930e82	1
req1	1073ab6cda4b991cd29f9e83a307f34004ae9327	1
req2	87ba78e0f03afcef60657f342ec5567368fadd8c	1
req3	3b88ea816c78ec104041a75e78f32ec804eaac39	2
req4	C34bf5a9ecca6edc3128018b1dd235a0f7bdff20	0
req5	Af065e03e22fe1f5f95b8ce9f4761c74da076857	1
req6	6df377ec91a0df5f054484fbfd0c13d7ed27d832	1
req7	05db376c6fa6453bd9c80b31d2c675977851be34	0





s0 = Server A
s1 = Server B
s2 = Server C
s3 = Server D



s0_0 = Server A
s0_1 = Server A
s0_2 = Server A
s1_0 = Server B
s1_1 = Server B
s1_2 = Server B
s2_0 = Server C
s2_1 = Server C
s2_2 = Server C
s3_0 = Server D
s3_1 = Server D
s3_2 = Server D

Complexity

Add a node	$O\left(\frac{K}{N} + \log N\right)$
Remove a node	$O\left(\frac{K}{N} + \log N\right)$
Add a key	$O(\log N)$
Remove a key	$O(\log N)$

Load Balancing

Load Balancing Benefits

- Optimizing resource utilization
- Maximum throughput
- Reducing latency
- Ensuring fault-tolerant configurations

```
http {
    upstream customersvc {
        server 192.168.1.101 weight=5;
        server 192.168.1.102;
        server 192.68.1.100 backup;
    }

    server {
        location /customer {
            proxy_pass http://customersvc;
        }
    }
}
```

A request is sent to the server with the least number of active connections, with server weights taken into consideration

```
upstream backend {
    least_conn;
    server 192.168.1.101;
    server 192.168.1.102;
}
```

The method guarantees that requests from the same address get to the same server unless it is not available.

```
upstream backend {
    ip_hash;
    server 192.168.1.101;
    server 192.168.1.102;
}
```

The server to which a request is sent is determined from a user-defined key which can be a text string, variable, or a combination.

```
upstream backend {
    hash $request_uri consistent;
    server 192.168.1.101;
    server 192.168.1.102;
}
```

Further Resources

- A Guide to Consistent Hashing