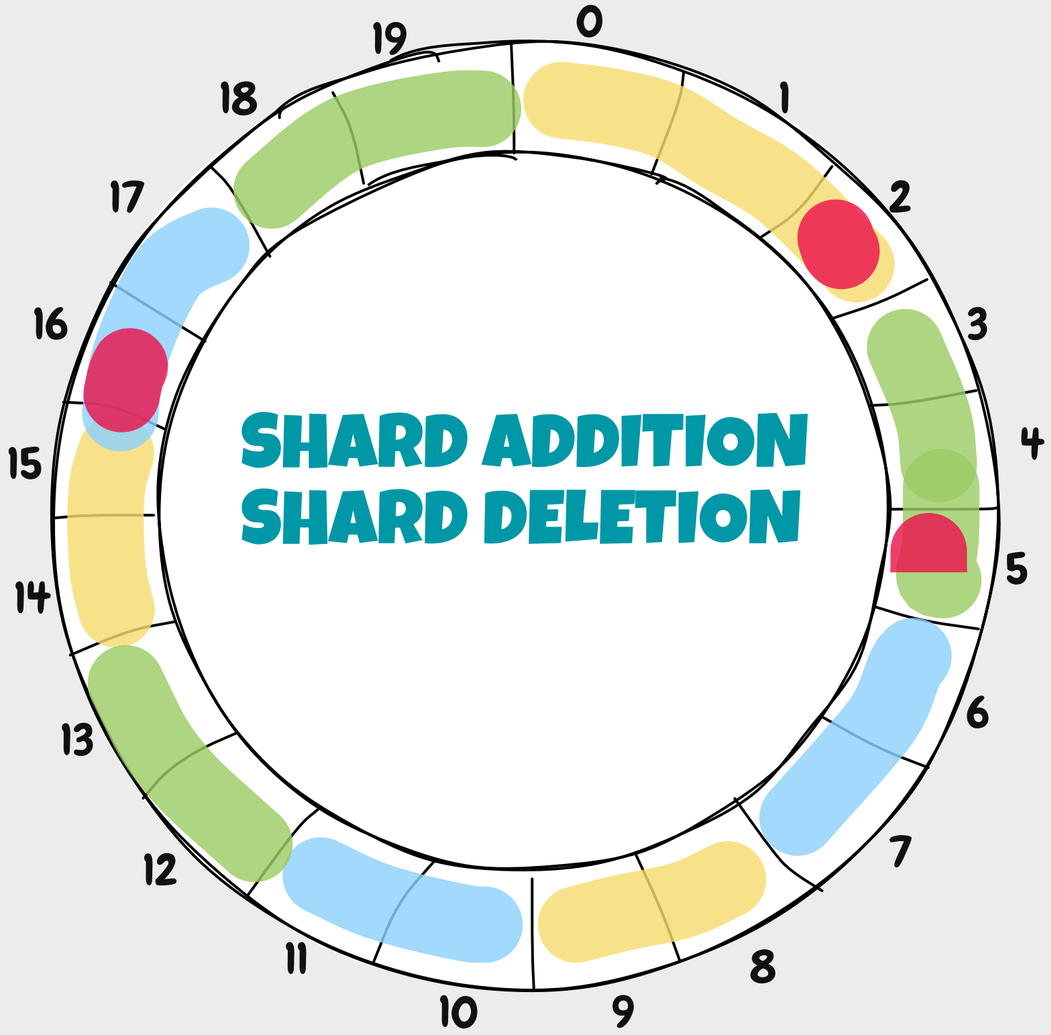
Consistent Hashing

The Complete Guide



REQUIREMENTS

You have multiple databases shard to store data.

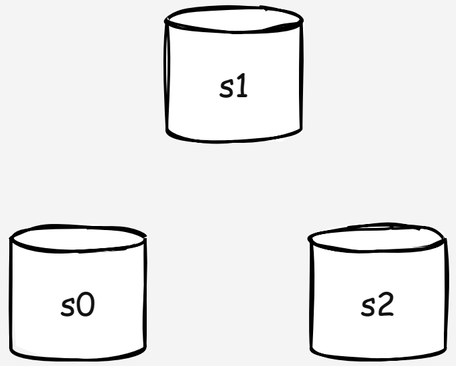
The amount of data can grow or shrink with time.

If amount of data grows, then we should be able to add extra shard and rabalance.

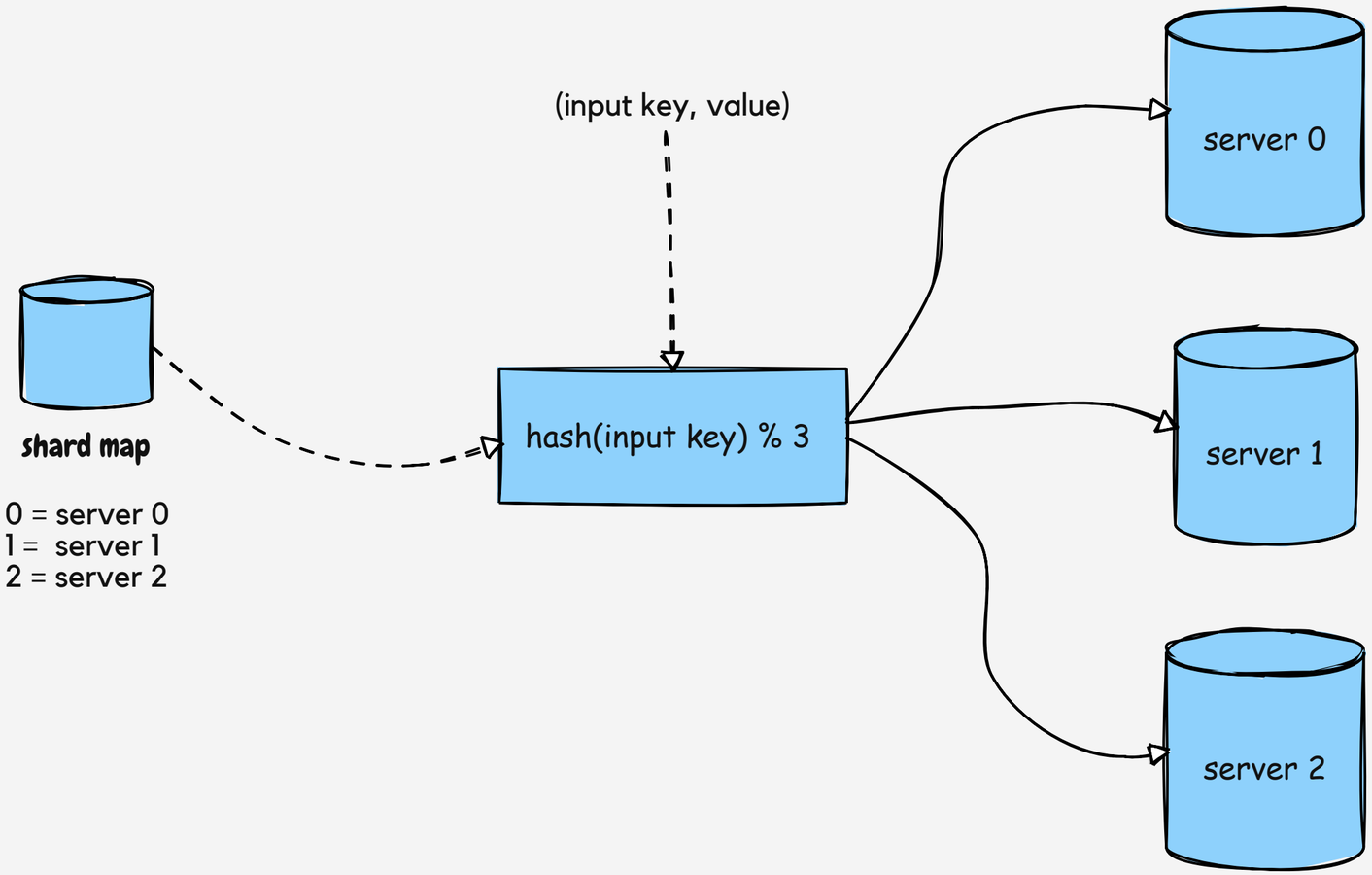
The new shard to migrate the data equally from all the shards.

If amount of data shrinks, then we should be able to rebalance & then remove one shard.

The shard to be deleted should migrate the data equally to all the shards.



SIMPLE MODULO



Change in number of servers requires redistribution of almost all keys

Key assignment + Rebalances in simple Modulo method

Say, Number of keys = 20

hash(1) = 1, hash(2) = 2 etc

16

keys need to be rebalanced /
moved from the previous server to a new server

Server ids with 3 servers

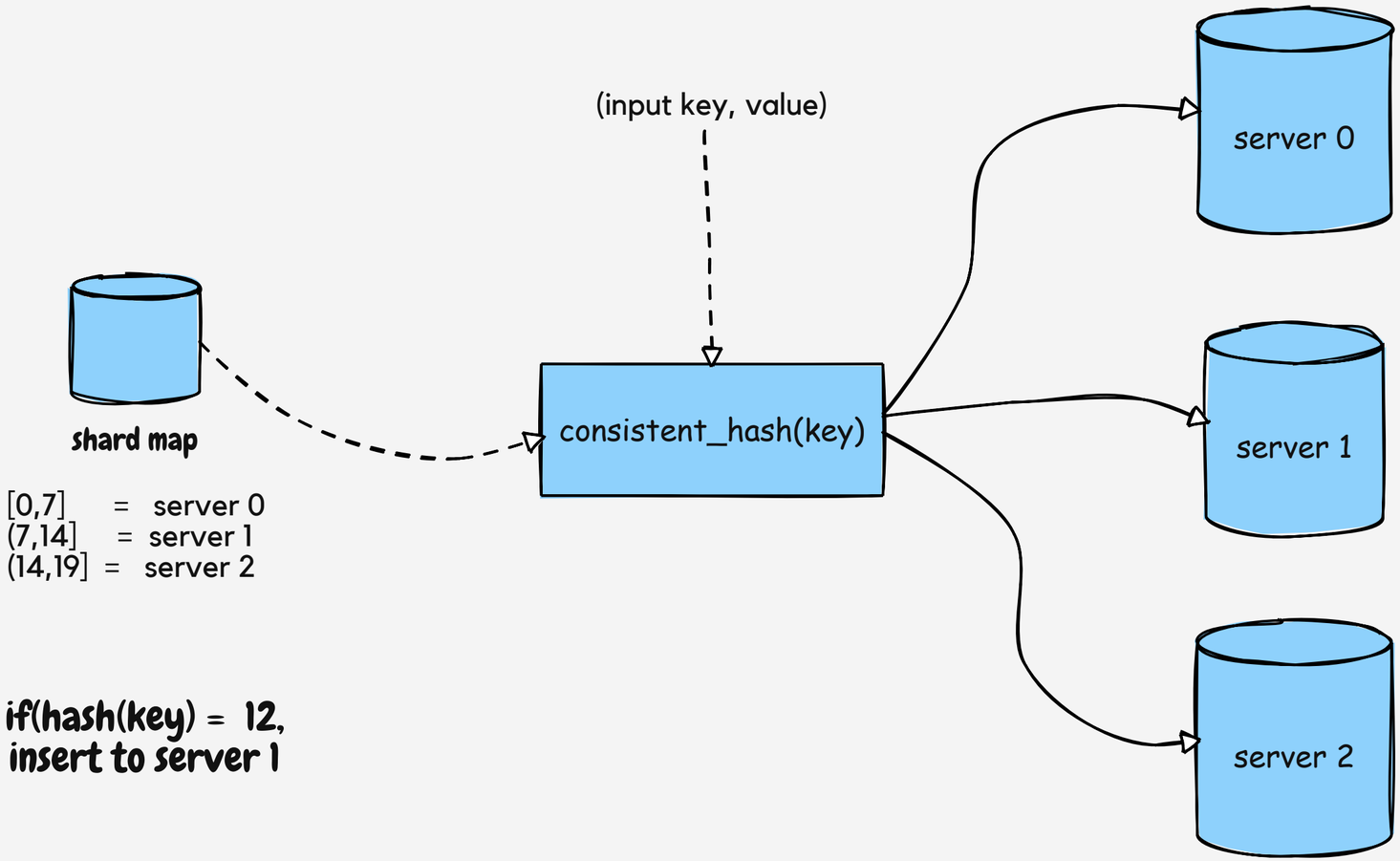
h(key0) = 0 => server = 0%3 = 0 => Server 0
h(key1) = 1 => server = 1%3 = 1 => Server 1
h(key2) = 2 => server = 2%3 = 2 => Server 2
h(key3) = 3 => server = 3%3 = 0 => Server 0
h(key4) = 4 => server = 4%3 = 1 => Server 1
h(key5) = 5 => server = 5%3 = 2 => Server 2
h(key6) = 6 => server = 6%3 = 0 => Server 0
h(key7) = 7 => server = 7%3 = 1 => Server 1
h(key8) = 8 => server = 8%3 = 2 => Server 2
h(key9) = 9 => server = 9%3 = 0 => Server 0
h(key10) = 10 => server = 10%3 = 2 => Server 1
h(key11) = 11 => server = 11%3 = 0 => Server 2
h(key12) = 12 => server = 12%3 = 1 => Server 0
h(key13) = 13 => server = 13%3 = 2 => Server 1
h(key14) = 14 => server = 14%3 = 0 => Server 2
h(key15) = 15 => server = 15%3 = 0 => Server 0
h(key16) = 16 => server = 16%3 = 1 => Server 1
h(key17) = 17 => server = 17%3 = 2 => Server 2
h(key18) = 18 => server = 18%3 = 0 => Server 0
h(key19) = 19 => server = 19%3 = 1 => Server 1

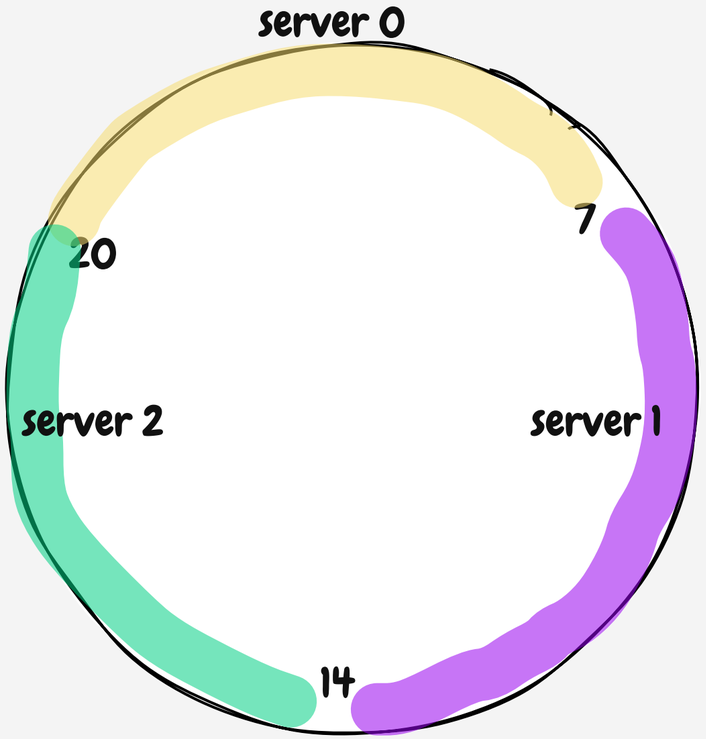
Server ids with 4 servers

h(key0) = 0 => server = 0%4 = 0 => Server 0
h(key1) = 1 => server = 1%4 = 1 => Server 1
h(key2) = 2 => server = 2%4 = 2 => Server 2
h(key3) = 3 => server = 3%4 = 0 => Server 3
h(key4) = 4 => server = 4%4 = 1 => Server 0
h(key5) = 5 => server = 5%4 = 2 => Server 1
h(key6) = 6 => server = 6%4 = 0 => Server 2
h(key7) = 7 => server = 7%4 = 1 => Server 3
h(key8) = 8 => server = 8%4 = 2 => Server 0
h(key9) = 9 => server = 9%4 = 0 => Server 1
h(key10) = 10 => server = 10%4 = 2 => Server 2
h(key11) = 11 => server = 11%4 = 0 => Server 3
h(key12) = 12 => server = 12%4 = 1 => Server 0
h(key13) = 13 => server = 13%4 = 2 => Server 1
h(key14) = 14 => server = 14%4 = 0 => Server 2
h(key15) = 15 => server = 15%4 = 0 => Server 3
h(key16) = 16 => server = 16%4 = 1 => Server 0
h(key17) = 17 => server = 17%4 = 2 => Server 1
h(key18) = 18 => server = 18%4 = 0 => Server 2
h(key19) = 19 => server = 19%4 = 1 => Server 3

hash(key)	Server id before	Server id after
0	0	0
1	1	1
2	2	2
3	0	3
4	1	0
5	2	1
6	0	2
7	1	3
8	2	0
9	0	1
10	1	2
11	2	3
12	0	0
13	1	1
14	2	2
15	0	3
16	1	0
17	2	1
18	0	2
19	1	3

SIMPLE CONSISTENT HASHING





SHARD ADDITION
REBALANCES IN SIMPLE CONSISTENT HASHING

INNITIAL SHARD MAPPING

[0,7] = server 0
(7,14] = server 1
(14,19] = server 2

Server ids with 3 servers

h(key0) = 0 => Server 0
h(key1) = 1 => Server 0
h(key2) = 2 => Server 0
h(key3) = 3 => Server 0
h(key4) = 4 => Server 0
h(key5) = 5 => Server 0
h(key6) = 6 => Server 0
h(key7) = 7 => Server 0
h(key8) = 8 => Server 1
h(key9) = 9 => Server 1
h(key10) = 10 => Server 1
h(key11) = 11 => Server 1
h(key12) = 12 => Server 1
h(key13) = 13 => Server 1
h(key14) = 14 => Server 1
h(key15) = 15 => Server 2
h(key16) = 16 => Server 2
h(key17) = 17 => Server 2
h(key18) = 18 => Server 2
h(key19) = 19 => Server 2

NEW SHARD MAPPING

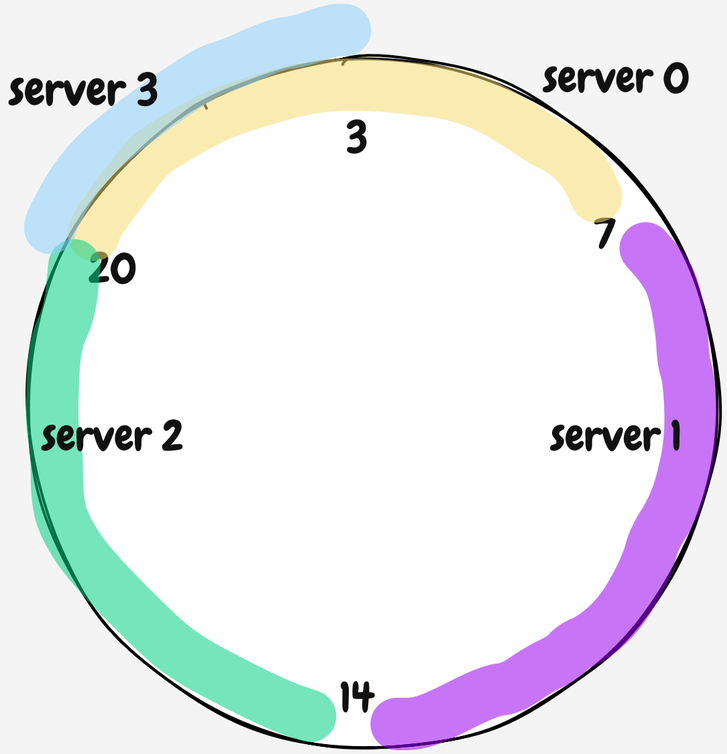
[0,3] = server 3
(3, 7] = server 0
(7,14] = server 1
(14,19] = server 2

Server ids with 4 servers

h(key0) = 0 => Server 3
h(key1) = 1 => Server 3
h(key2) = 2 => Server 3
h(key3) = 3 => Server 3
h(key4) = 4 => Server 0
h(key5) = 5 => Server 0
h(key6) = 6 => Server 0
h(key7) = 7 => Server 0
h(key8) = 8 => Server 1
h(key9) = 9 => Server 1
h(key10) = 10 => Server 1
h(key11) = 11 => Server 1
h(key12) = 12 => Server 1
h(key13) = 13 => Server 1
h(key14) = 14 => Server 1
h(key15) = 15 => Server 2
h(key16) = 16 => Server 2
h(key17) = 17 => Server 2
h(key18) = 18 => Server 2
h(key19) = 19 => Server 2

4

keys need to be rebalanced /
moved from the previous server to a new server



SHARD DELETION
REBALANCES IN SIMPLE CONSISTENT HASHING

INNITIAL SHARD MAPPING

[0,7] = server 0
(7,14] = server 1
(14,19] = server 2

Server ids with 3 servers

h(key0) = 0 => Server 0
h(key1) = 1 => Server 0
h(key2) = 2 => Server 0
h(key3) = 3 => Server 0
h(key4) = 4 => Server 0
h(key5) = 5 => Server 0
h(key6) = 6 => Server 0
h(key7) = 7 => Server 0
h(key8) = 8 => Server 1
h(key9) = 9 => Server 1
h(key10) = 10 => Server 1
h(key11) = 11 => Server 1
h(key12) = 12 => Server 1
h(key13) = 13 => Server 1
h(key14) = 14 => Server 1
h(key15) = 15 => Server 2
h(key16) = 16 => Server 2
h(key17) = 17 => Server 2
h(key18) = 18 => Server 2
h(key19) = 19 => Server 2

NEW SHARD MAPPING

(3, 14] = server 0
(14,19] = server 2

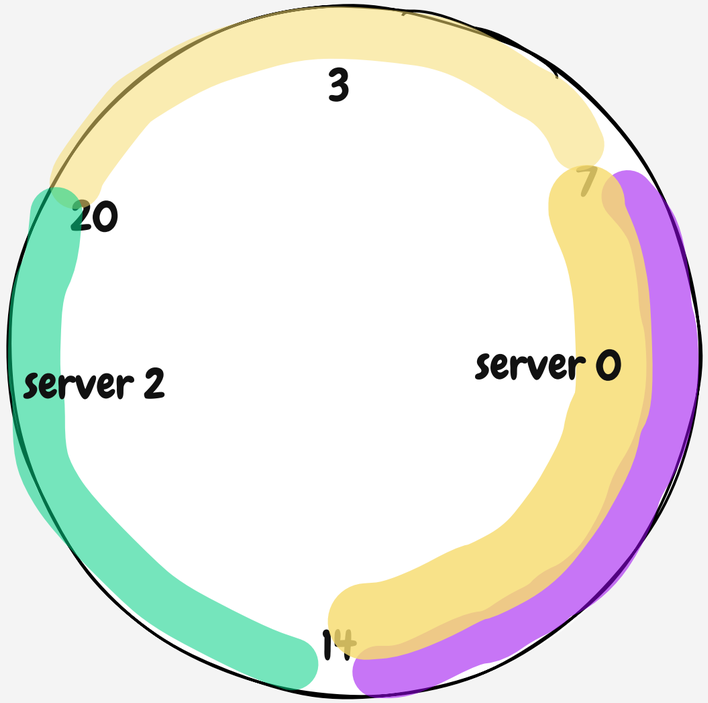
Server ids with 4 servers

h(key0) = 0 => Server 0
h(key1) = 1 => Server 0
h(key2) = 2 => Server 0
h(key3) = 3 => Server 0
h(key4) = 4 => Server 0
h(key5) = 5 => Server 0
h(key6) = 6 => Server 0
h(key7) = 7 => Server 0
h(key8) = 8 => Server 0
h(key9) = 9 => Server 0
h(key10) = 10 => Server 0
h(key11) = 11 => Server 0
h(key12) = 12 => Server 0
h(key13) = 13 => Server 0
h(key14) = 14 => Server 0
h(key15) = 15 => Server 2
h(key16) = 16 => Server 2
h(key17) = 17 => Server 2
h(key18) = 18 => Server 2
h(key19) = 19 => Server 2

7

keys need to be rebalanced /
moved from the previous server to a new server

We still first need to rebalance.
We cannot connect just delete a shard. That scenario of deleting a shard is data loss. which is handled using replicas.



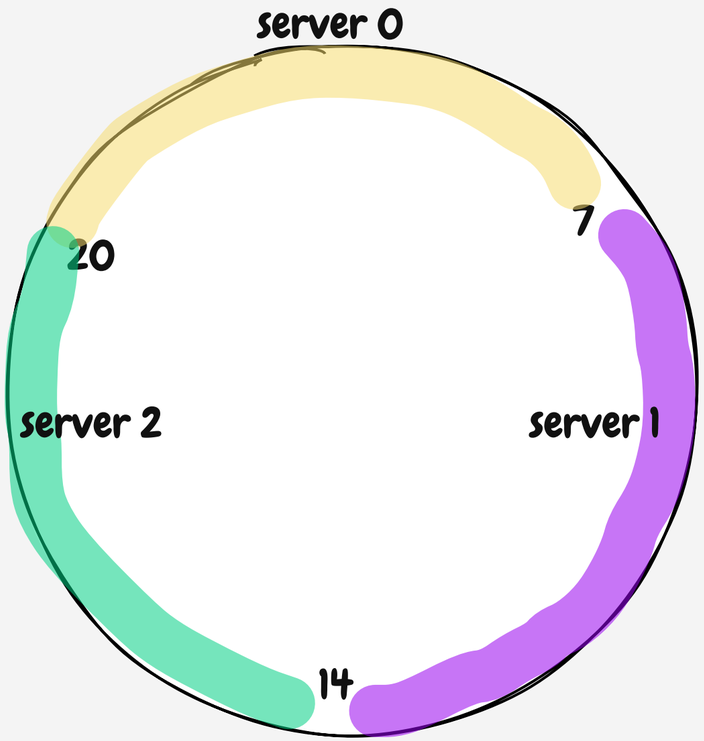
PROBLEM WITH SIMPLE CONSISTENT HASHING

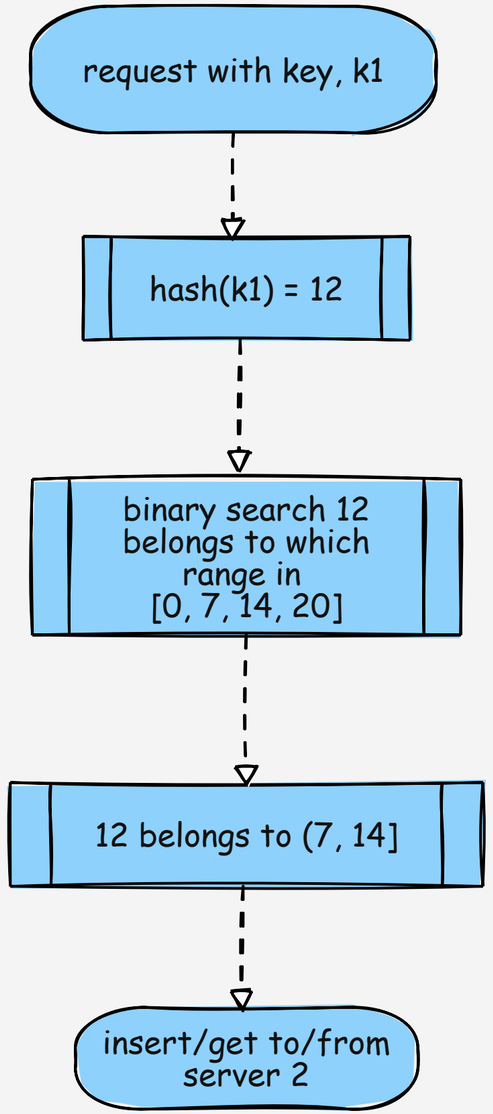
Server load is not equally distributed

Addition scenario:
If, there are 3 existing servers and you add one more server and 90 keys need to be rebalanced from existing servers, then 30 keys from each server needs to me migrated.

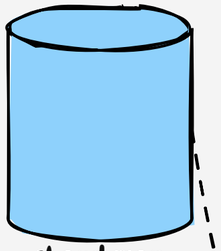
Deletion scenario:
If, there are 3 existing servers and you delete a server.
If the server to be deleted has 100 keys.
Then, the remaining 2 server should migrate 50 keys to each servers.

CONSISTENT HASHING GET AND PUT OPERATION ALGORITHM





Consistent Hashing with Virtual Nodes



shard map

[0,2] = server 0A
(2,5] = server 1A
(5,7] = server 2A
(7,9] = server 0B
(9,11] = server 2B
(11,13] = server 1B
(13,15] = server 0C
(15,17] = server 2C
(17,19] = server 1C

virtual shard to actual shard map

server 0A => server 0
server 0B
server 0C

server 1A => server 1
server 1B
server 1C

server 2A => server 2
server 2B
server 2C

(input key, value)

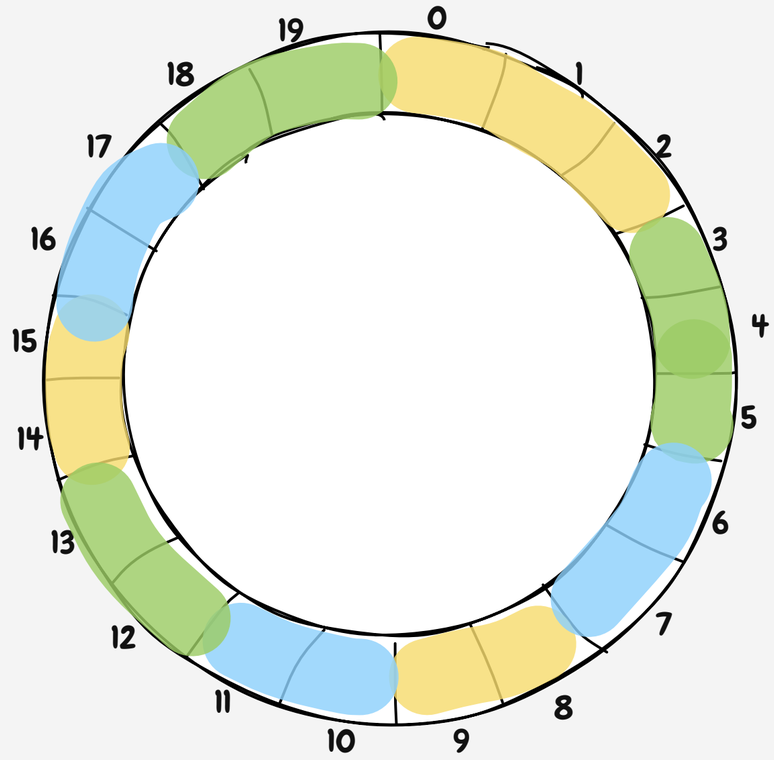
consistent_hash(key)

if(hash(key) = 12, insert to server 2

server 0

server 1

server 2



server 0

server 1

server 2

SHARD ADDITION
CONSISTENT HASHING WITH VIRTUAL NODE

OLD

shard map

[0,2] = server 0A
(2,5] = server 1A
(5,7] = server 2A
(7,9] = server 0B
(9,11] = server 2B
(11,13] = server 1B
(13,15] = server 0C
(15,17] = server 2C
(17,19] = server 1C

virtual shard to actual shard map

server 0A => server 0
server 0B
server 0C

server 1A => server 1
server 1B
server 1C

server 2A => server 2
server 2B
server 2C

NEW

shard map

[0,1] = server 0A
(1,2] = server3A
(2,4] = server 1A
(4,5] = server 3B
(5,7] = server 2A
(7,9] = server 0B
(9,11] = server 2B
(11,13] = server 1B
(13,15] = server 0C
(15, 16] = server 3C
(16, 17] = server 2C
(17, 19] = server 1C

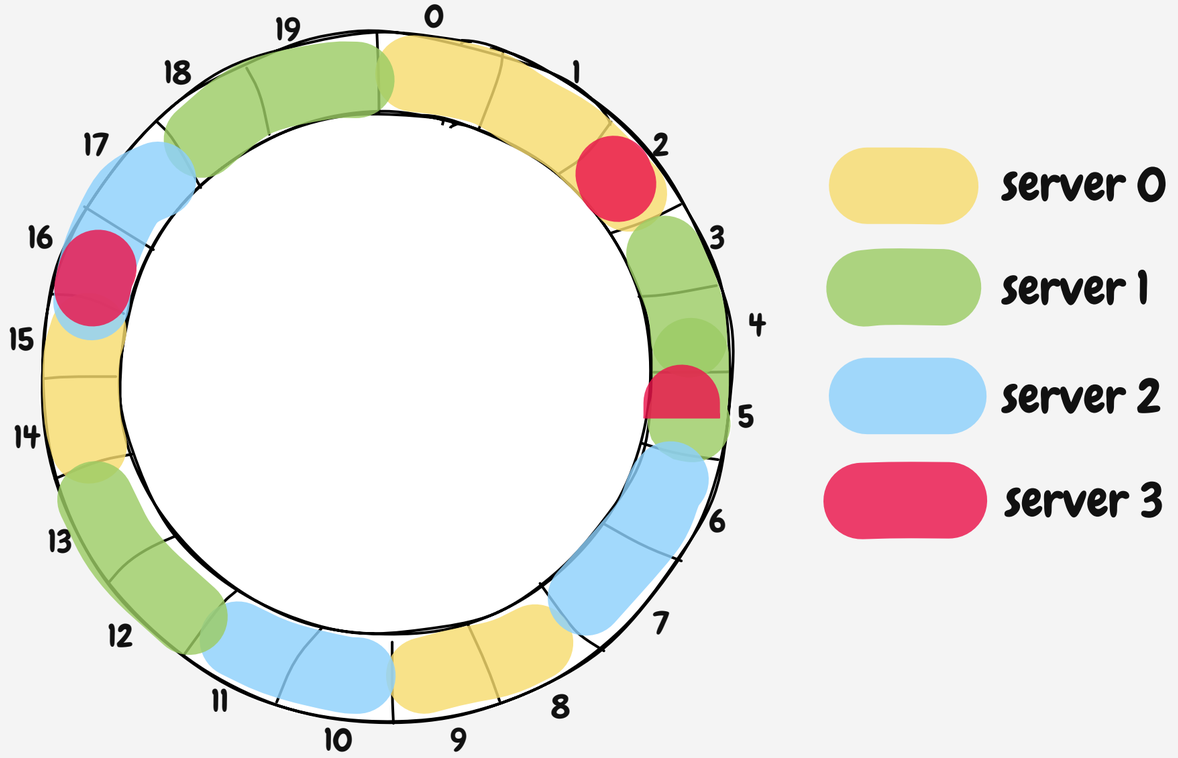
virtual shard to actual shard map

server 0A => server 0
server 0B
server 0C

server 1A => server 1
server 1B
server 1C

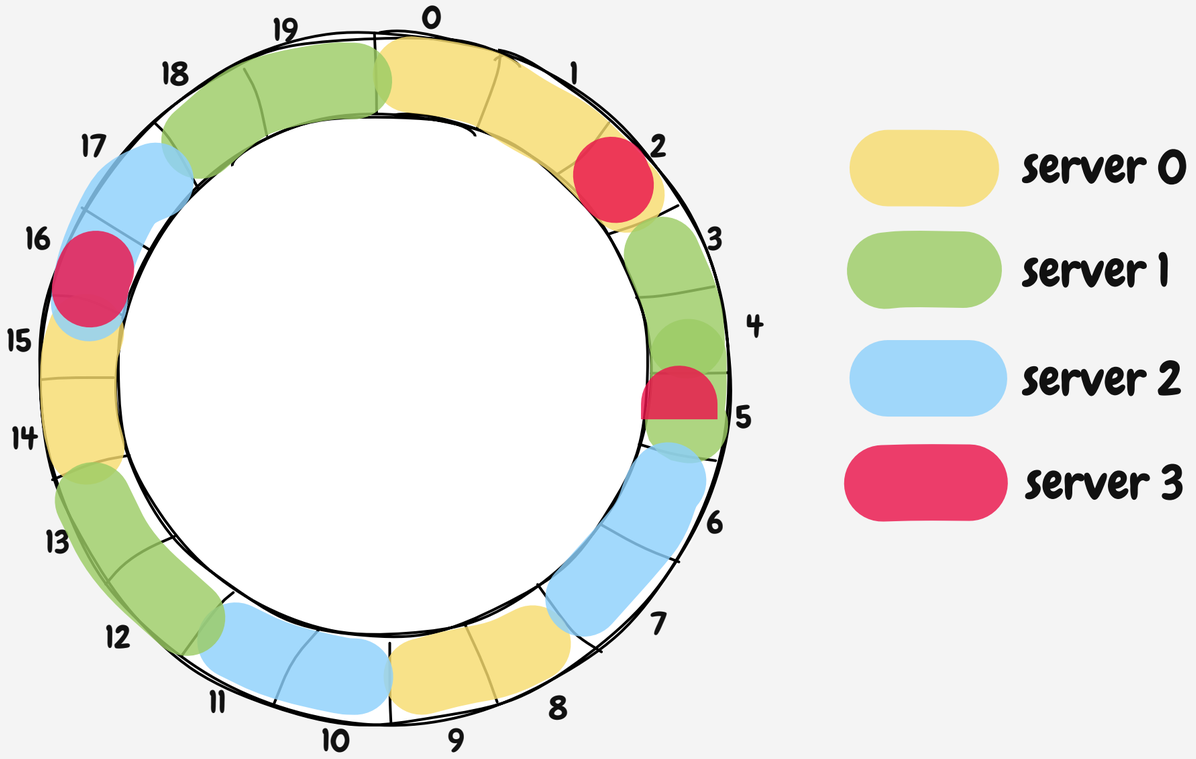
server 2A => server 2
server 2B
server 2C

server 3A => server 3
server 3B
server 3C



3 movements needed from each servers server 0, server 1 and server 2 to move the data to server 3

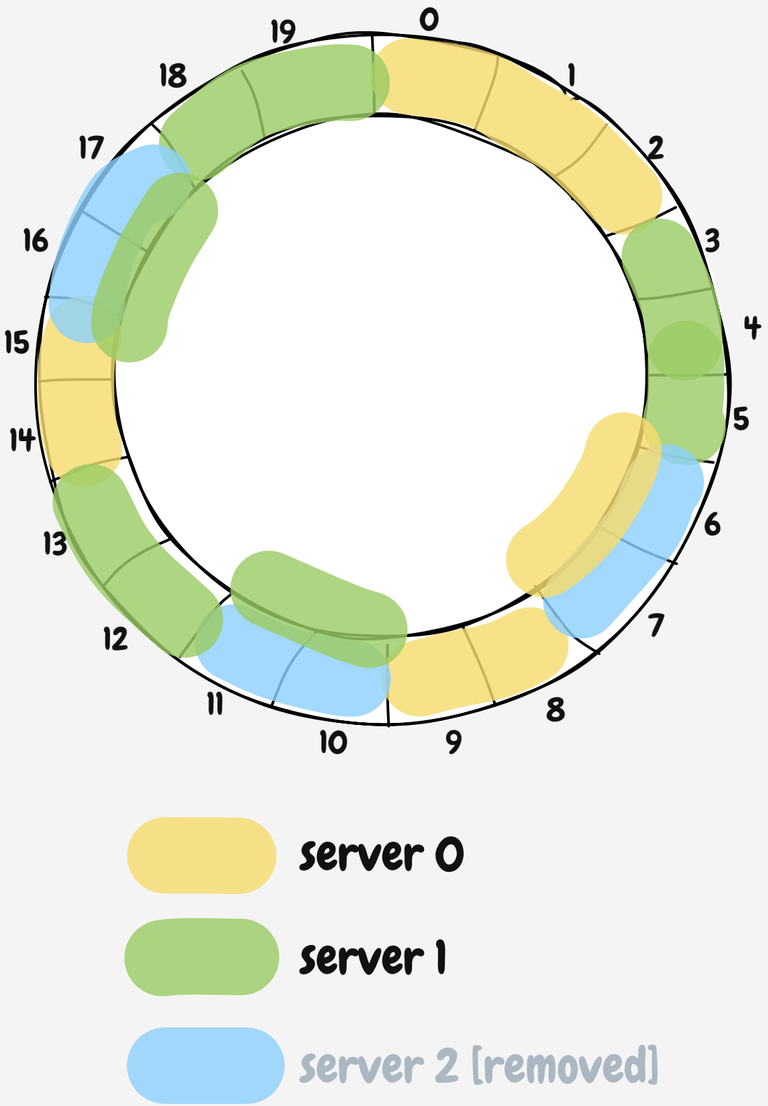
hash(key)	Server id before	Server id after
0	0	0
1	0	0
2	0	3
3	1	1
4	1	1
5	1	3
6	2	2
7	2	2
8	0	0
9	0	0
10	2	2
11	2	2
12	1	1
13	1	1
14	0	0
15	0	0
16	2	3
17	2	2
18	1	1
19	1	1



Number of movements =3
number of movements from server 0 =1
number of movements from server 1 =1
number of movements from server 2 =1

Addition of new server takes up load EQUALLY from all servers

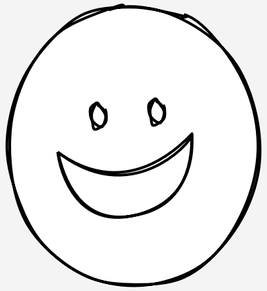
SHARD DELETION
CONSISTENT HASHING WITH VIRTUAL NODE



SERVER 3 TO BE REMOVED AND DATA TO BE REBALANCED

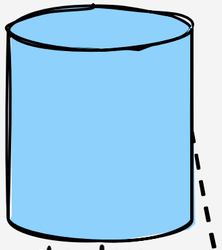
Number of movement = 6 [because the node to be removed contains 6 keys

Number of keys moved to server 0 = 2
Number of keys moved to server 1 = 4



REMOVAL OF SHARDS RE-DISTRIBUTES THE DATA EQUALLY TO REMAINING SHARDS

SUMMARY



shard map

[0,2] = server 0A
(2,5] = server 1A
(5,7] = server 2A
(7,9] = server 0B
(9,11] = server 2B
(11,13] = server 1B
(13,15] = server 0C
(15,17] = server 2C
(17,19] = server 1C

virtual shard to actual shard map

server 0A => server 0
server 0B
server 0C

server 1A => server 1
server 1B
server 1C

server 2A => server 2
server 2B
server 2C

(input key, value)

consistent_hash(key)

if(hash(key) = 12, insert to server 2

server 0

server 1

server 2