Consistent Hashing Ring
and
OpenStack Swift Ring

PROBLEMS WITH REHASHING

- The distribution changes whenever we change the number of servers.
- Sometimes, the keys won't be found because they won't yet be present at their new location.
- With the more number of redistributions, the more number of misses, the more number of memory fetches, placing an additional load on the node and thus decreasing the performance.

BRAC UNIVERSITY
Inspiring Excellence

What next?

- How to solve rehashing problem
- Consistent Hashing

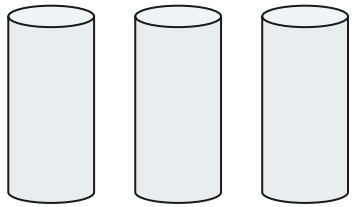
BRAC UNIVERSITY
Inspiring Excellence

How to solve rehashing problem

In the distributed system, there are lots of data and and lots of metadata. When we do rehashing it kept alive until the servers are alive. That means, at least when a server need a maintenance break, and shouted down or rebooted, hashing will be changed and we will lost where, what will be and what's hashing is which one. Like wisely, if needs a new server or a server crashed or got missing or a new server to add, every time there will be need of a new hashing, as most hashes modulo N will change.

BRAC UNIVERSITY
Inspiring Excellence

Example of rehashing problem

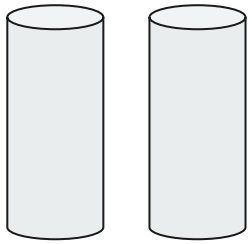


A B C

Location Of Data
Shamiul -> A
Sakib -> B
Sakif -> C

Key	Hash	Hash mod 3
Shamiul	77788899	0
Sakib	55566770	1
Sakif	99887700	2

BRAC UNIVERSITY
Inspiring Excellence



A B

Key	Hash	Hash mod 3
Shamiul	77788899	0
Sakib	55566770	1
Sakif	99887700	0

Location Of Data after losing a server
Shamiul -> A
Sakib -> B
Sakif -> A

BRAC UNIVERSITY
Inspiring Excellence

Solution :

As Rehashing, directly dependent on the number of server in that system. If it is possible to remove this dependency, our problem is solved. If we able to do this, at the time of added a server or removing server, the number of keys generates will be minimized. That one such brilliant schema is called Consistent Hashing.

BRAC UNIVERSITY
Inspiring Excellence

Consistent Hashing

Consistent hashing, was first come to public by Karger et al. at MIT in an academic paper from 1997(according to Wikipedia).

Consistent Hashing operates independently of the server or objects in a distributed system' hash table by assigning a place or position on hash ring or an abstract circle, which allows servers and objects to scale any time without affecting the whole system.

BRAC UNIVERSITY
Inspiring Excellence

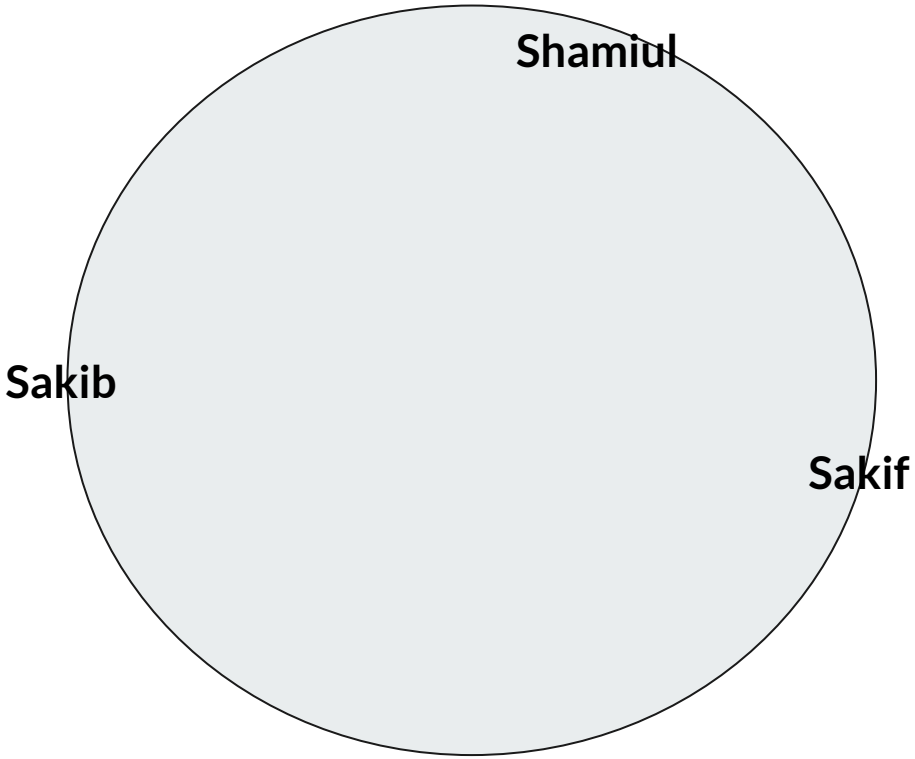
How Consistent Hashing Work

As, we know, Consistent hashing set a position for server or object. if we mapped the hash output range in a edge of a circle, the minimum possible corresponding angle will be zero, and the possible maximum value will be in a big integer, and corresponding angle of 360 degrees, and all other values will take position in between them. For this, if we take a key and compute its hash, we can find out about it's position on the circle's edge.

Let's take a value, 7 for the maximum value, see the mapping

BRAC UNIVERSITY
Inspiring Excellence

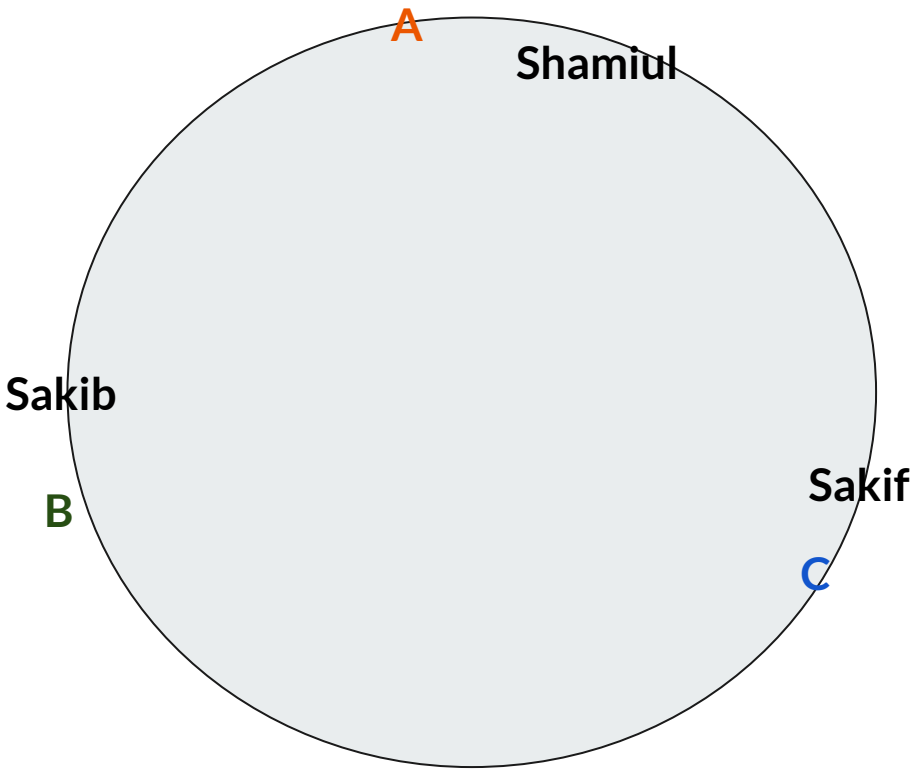
How Consistent Hashing Work



Key	Hash	Angle
Shamiul	889977	75
Sakib	778899	180
Sakif	665544	352

BRAC UNIVERSITY
Inspiring Excellence

How Consistent Hashing Work



We placed the servers on the edge of the circle, by pseudo-randomly assigning them angles too. This should be done in a repeatable way . But a convenient way to do this thing is hashing the server name, or IP address, or some ID, and for any other key with the angle.

BRAC UNIVERSITY
Inspiring Excellence

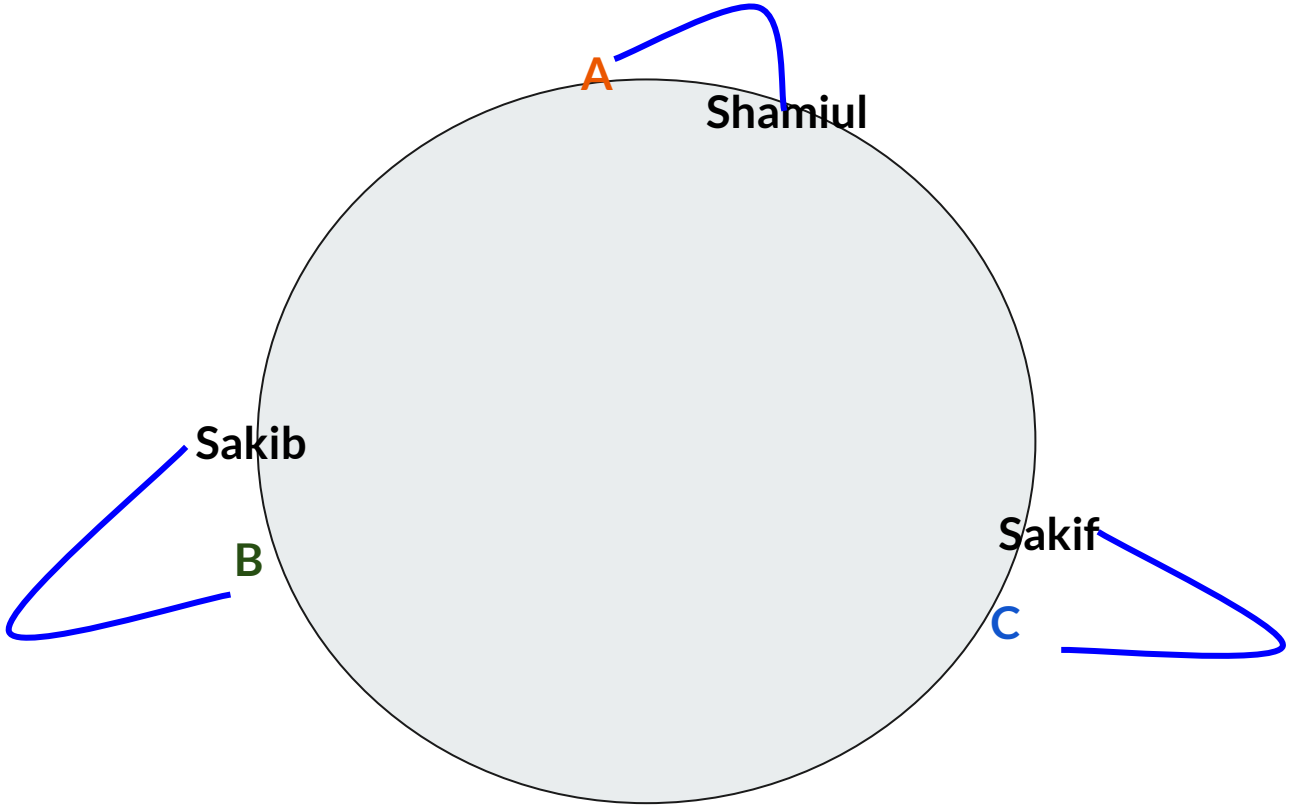
How Consistent Hashing Work

Key	Hash	Angle
Shamiul	889977	75
Sakib	778899	180
Sakif	665544	352
A	554890	95
B	689012	190
C	998812	285

As, we have the keys for the both objects and servers on the same circle, we can set rules to the existings, as well as for the new, likewise: set each object key to belongs in the nearest server, in a clockwise direction or in a counterclockwise.

BRAC UNIVERSITY
Inspiring Excellence

How Consistent Hashing Work



Key	Hash	Angle
Shamiul	889977	75
A	554890	95
Sakib	778899	180
B	689012	190
Sakif	665544	352
C	998812	285

BRAC UNIVERSITY
Inspiring Excellence

How Consistent Hashing Work

Key	Hash	Angle	Label	Server
Shamiul	889977	75	“A”	A
Sakib	778899	180	“B”	B
Sakif	665544	352	“C”	C

we will keep a sorted list of server values (which could be angles or numbers in any real interval), and walk this list (or use a binary search) to find the first server with a value greater than, or equal to, that of the desired key. If no such value is found, we need to wrap around, taking the first one from the list.

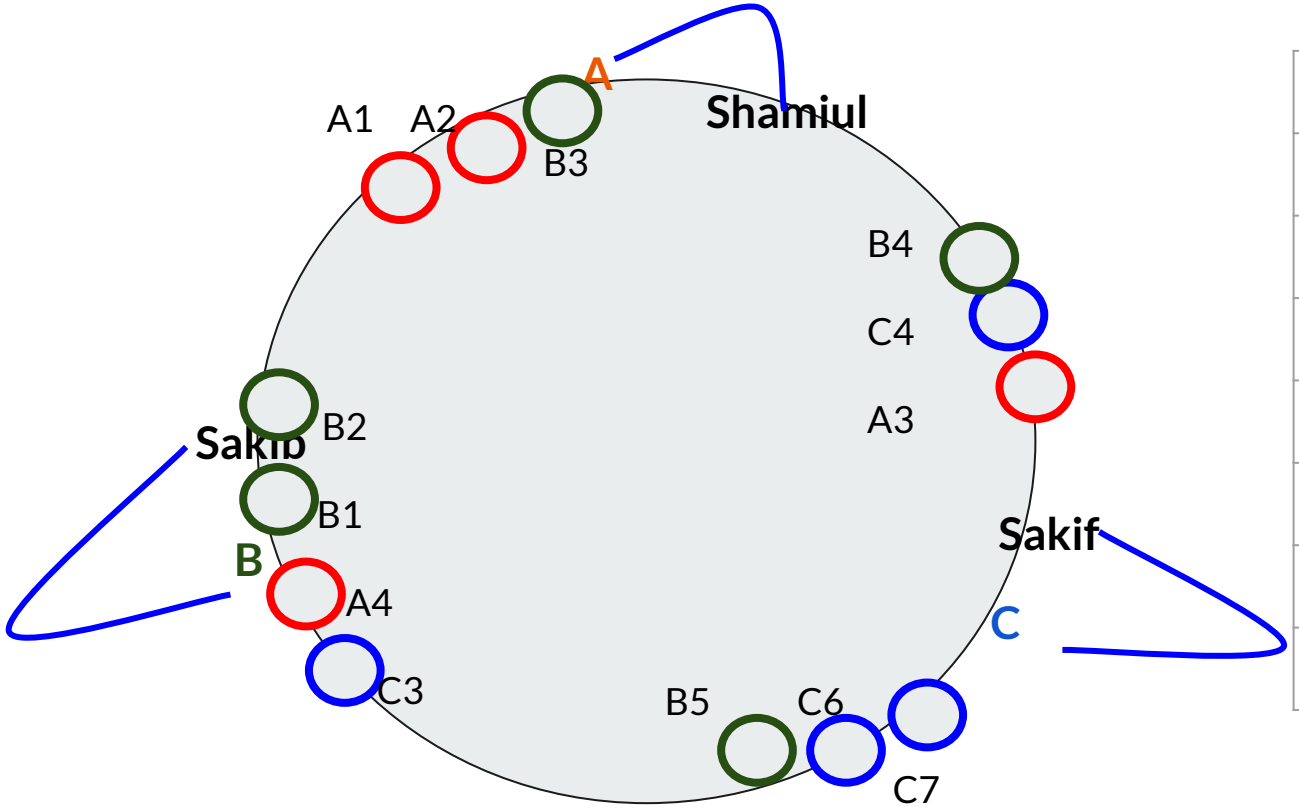
BRAC UNIVERSITY
Inspiring Excellence

How Consistent Hashing Work

For making sure object keys are distributed evenly in all servers, we can use weight rules and rather then using many labels for each server, we can use scattered or distribute the servers along the circle. In example, we have servers like “A”, “B”, “C”, instead of having this labels, we can use “A0”.... “A9”, “B0”... “B9” etc. As we said using the weight rules, which will help to balance the server’s load balancing. For example if server A is twice as powerful as the others, then it can be assigned twice as many labels, and as a result, it would end up holding twice as many objects and reduce pressure from the other less powerful servers

BRAC UNIVERSITY
Inspiring Excellence

How Consistent Hashing Work



Key	Hash	Angle
B3	46554	94
A2	36328	100
A1	25525	105
B2	65620	175
B1	89324	185
A4	87464	200
C3	35265	215

Key	Hash	Angle
B4	53244	25
C4	35420	19
A3	54836	14
C7	98536	285
C6	63540	280
B5	25755	275

BRAC UNIVERSITY
Inspiring Excellence

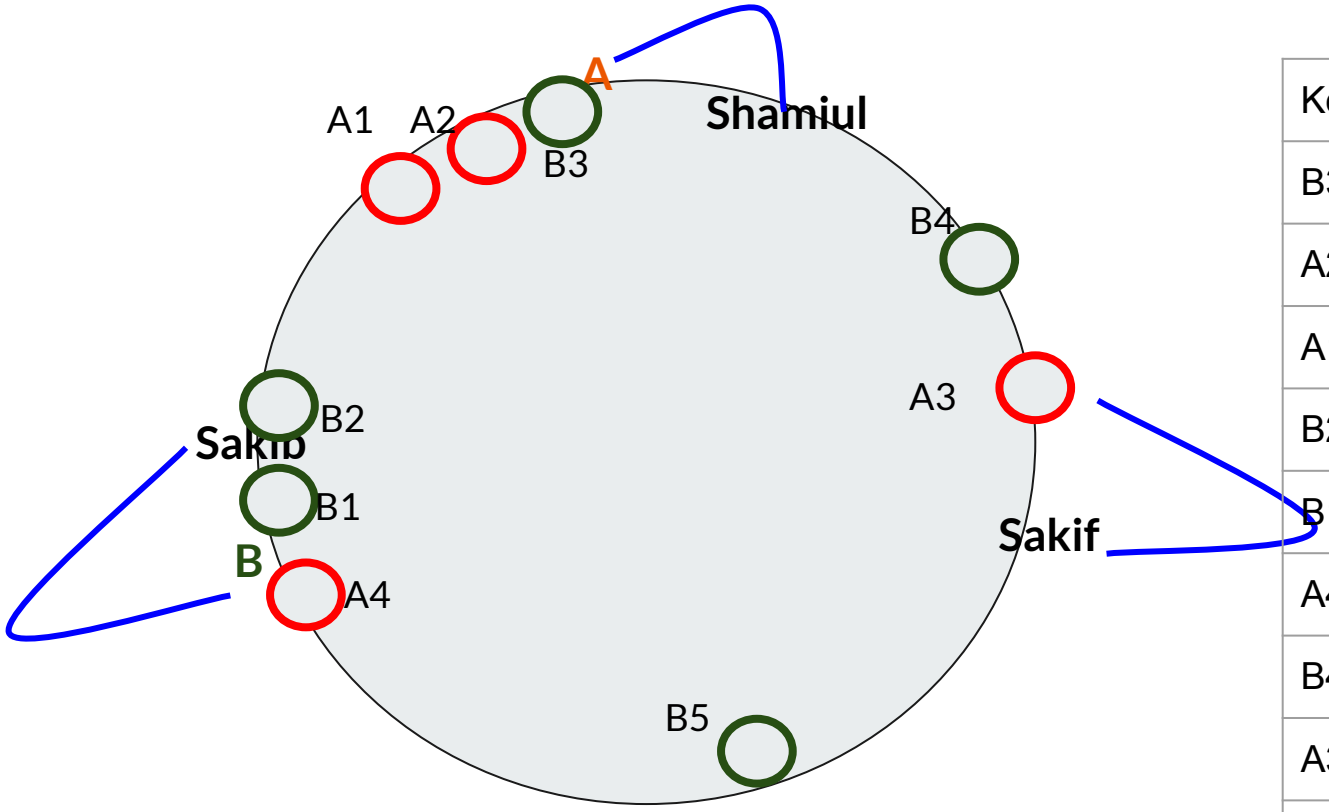
How Consistent Hashing Work

Key	Hash	Angle	Label	Server
Shamiul	889977	75	“A2”	A
Sakib	778899	180	“B1”	B
Sakif	665544	352	“C7”	C

Rather than having liner schema, in circle schema we can add remove server and hash them easily. If we remove a server, we need to remove labels as well as from the circle, and this is hash the objects randomly and label, reassign them in to other servers. And these does not even affect the existing object keys, leave all other keys untouched.

BRAC UNIVERSITY
Inspiring Excellence

How Consistent Hashing Work

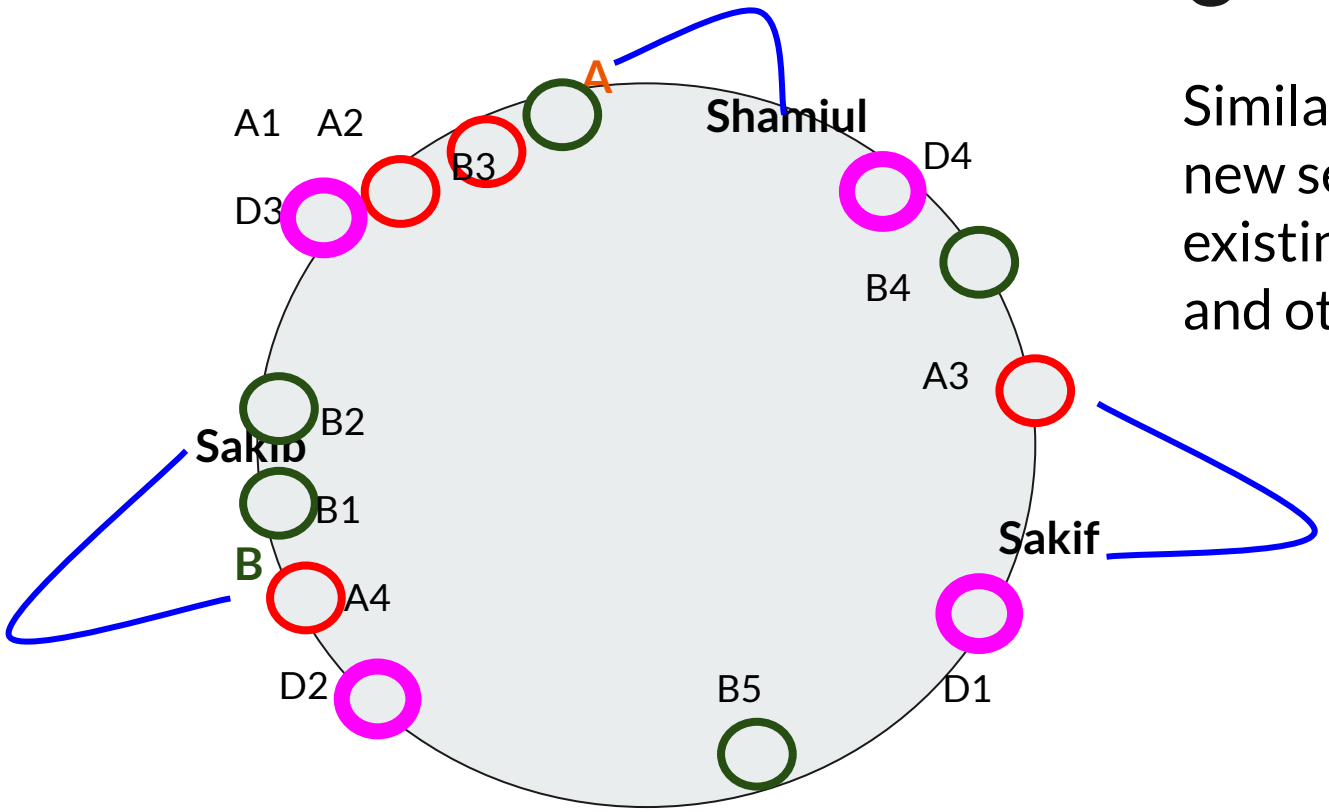


Key	Hash	Angle
B3	46554	94
A2	36328	100
A1	25525	105
B2	65620	175
B1	89324	185
A4	87464	200
B4	53244	25
A3	54836	14
B5	25755	275

Key	Hash	Angle	Label	Server
Shamiul	889977	75	“A2”	A
Sakib	778899	180	“B1”	B
Sakif	665544	352	“A3”	A

BRAC UNIVERSITY
Inspiring Excellence

How Consistent Hashing Work



Similar ways, If we add a server, we will label new server and distribute and one -third of the existing keys will be reassigned to new server and others will kept same.

BRAC UNIVERSITY
Inspiring Excellence

How Consistent Hashing Work

Key	Hash	Angle
B3	46554	94
A2	36328	100
A1	25525	105
B2	65620	175
B1	89324	185
A4	87464	200
B4	53244	25
A3	54836	14
B5	25755	275

Key	Hash	Angle
D1	56782	108
D2	79585	205
D3	89152	285
D4	78556	30

Key	Hash	Angle	Label	Server
Shamiul	889977	75	“A2”	A
Sakib	778899	180	“B1”	B
Sakif	785566	352	“D1”	D

BRAC UNIVERSITY
Inspiring Excellence

How Consistent Hashing Work

This is how, consistent hashing works.

In simply, only k/N keys need to be remapped when k is the number of keys and N is the number of servers and more specifically, the maximum of the initial and final number of servers.

BRAC UNIVERSITY
Inspiring Excellence

How partitions are placed in Openstack swift

- If there are any unbalanced discs with uneven number of partitions in the swift. When we will run a rebalance, the swift applies a mechanism that helps to make discs with equal numbers.
- Swift make sure to make data available even while moving the data

For example, While moving those partitions, if there are 3 replicas of a data. The ring builder will locks two replicas on primary nodes for a certain time, when the third one is on the move.

BRAC UNIVERSITY
Inspiring Excellence

How partitions are placed in swift(con.)

- The algorithm swift uses it keeps replicas as far apart as possible. So that if any accident happen it doesn't affect the other servers

Example:

If there are 3 replicas and 3 zones. Each replica will go to a different zone.

If there are 3 replicas and 2 zones. 1 replica will go to one zone and Other two replicas will go to another zone but in different discs within the zone.

BRAC UNIVERSITY
Inspiring Excellence

How partitions are placed in swift (Cont.)

Example (cont.)

If there are 3 replicas and 1 zones with 3 or more discs. Each replica will go to different discs of the same zone.

If there are 3 replicas and 1 zones with 2 discs. 1 replica will go to one disc and another 2 replica will go to the another disc. But that other discs will only keep 1 replica.

<In older version of swift it was must to have 3 separate zones for 3 replicas but with new version it is not necessary.>

BRAC UNIVERSITY
Inspiring Excellence

How partitions are placed in swift (Cont.)

Hand off notes

Swift supports handoff node to tackle the crisis of disc failure.

Swift use active replication means one node asks other nodes if they have a copy of the other replications within a short span of time. If It find out one disc is not working properly it makes a copy from the remaining replica and passes it to another working disc. When the fail disc again start working it passes the copy to the desired disc.

BRAC UNIVERSITY
Inspiring Excellence

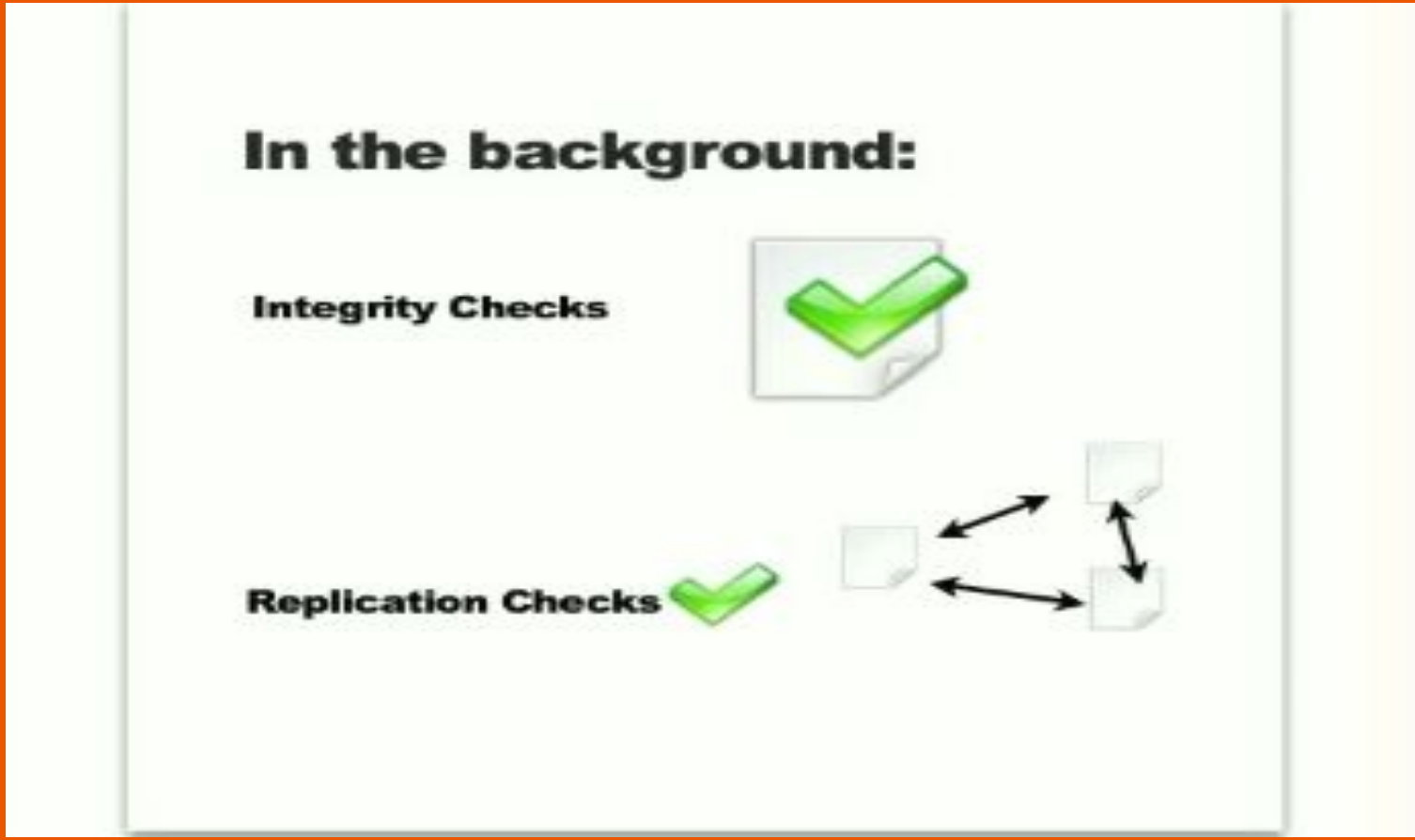
Active Replication

In the background swift runs an active consistency check that reads the uploaded object of a disc and makes sure their checksum matches. If swift finds any checksum of an object that doesn't match it discards the object and places it in a trash directory.

It also runs an active replication check consistently and if it finds any desired disc is missing any object , it creates a copy from the remaining replicas and places it on the desired disc.

BRAC UNIVERSITY
Inspiring Excellence

Active Replication



BRAC UNIVERSITY
Inspiring Excellence

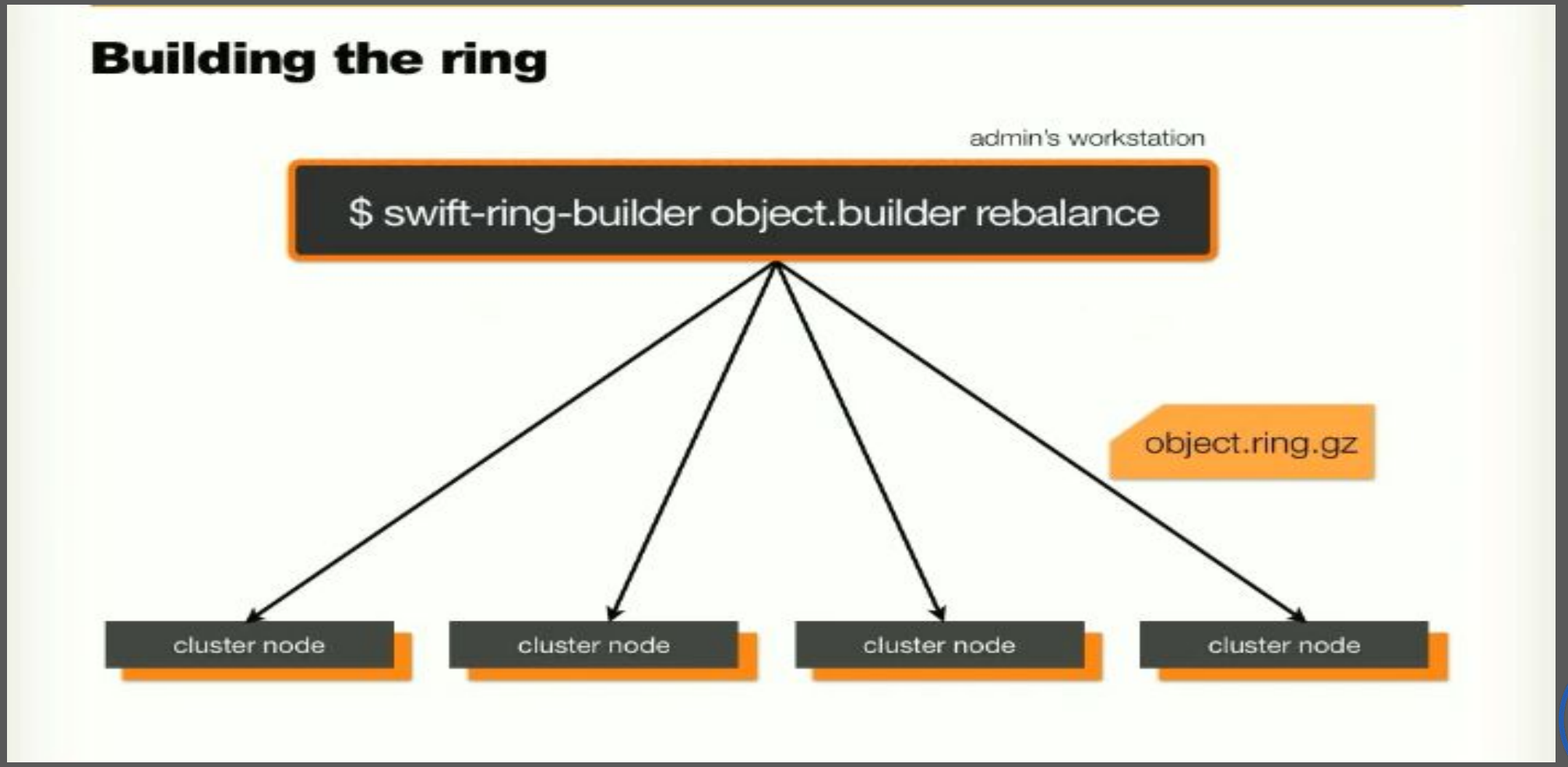
Ring Builder

Ring Builder is a configure that is run offline by system Admin. This configure file handle many operations. Such as:

- ➔ Add a device
- ➔ Remove a device
- ➔ Rebalance

BRAC UNIVERSITY
Inspiring Excellence

Ring Builder



BRAC UNIVERSITY
Inspiring Excellence

References

OpenStack Swift

A Guide to Consistent Hashing

Hashing in Distributed Systems

Distributed Database Things to Know: Consistent Hashing

BRAC UNIVERSITY
Inspiring Excellence

References

https://www.toptal.com/big-data/consistent-hashing

http://courses.cse.tamu.edu/caverlee/csce438/readings/consistent-hashing.pdf

https://www.researchgate.net/figure/Virtual-Rehashing-of-RQALSH-for-c-2_fig1_319867705

https://www.ably.io/blog/implementing-efficient-consistent-hashing

BRAC UNIVERSITY
Inspiring Excellence

Reference:

https://medium.com/system-design-blog/consistent-hashing-b9134c8a9062

https://en.wikipedia.org/wiki/Distributed_hash_table

https://www.educative.io/edpresso/what-is-a-distributed-hash-table

https://hazelcast.com/glossary/distributed-hash-table/

https://medium.com/system-design-blog/consistent-hashing-b9134c8a9062

https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_1232

https://www.toptal.com/big-data/consistent-hashing

BRAC UNIVERSITY

Inspiring Excellence

Acknowledge to

Md Shamiul Islam, Md Nazmur Sakib, Md Sadiqul Islam Sakif

BRAC UNIVERSITY

Inspiring Excellence