Let’s roll by 09:05 !!!

System Design 2

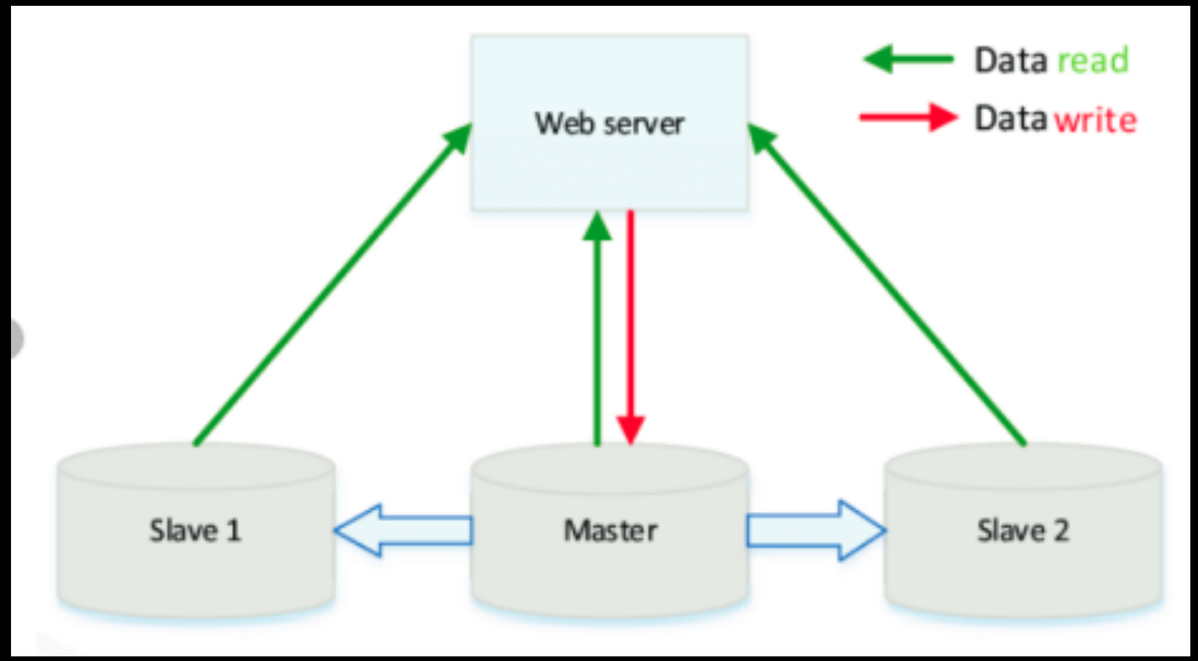
Gopi Krishnan R

Topics to be Discussed

- System Design 1 Recap
- Master Slave Architecture
- Latency
- CDN
- Shuffling and Sharding
- Consistent Hashing
- Proxy Server
- Caching
- MVP - TinyURL

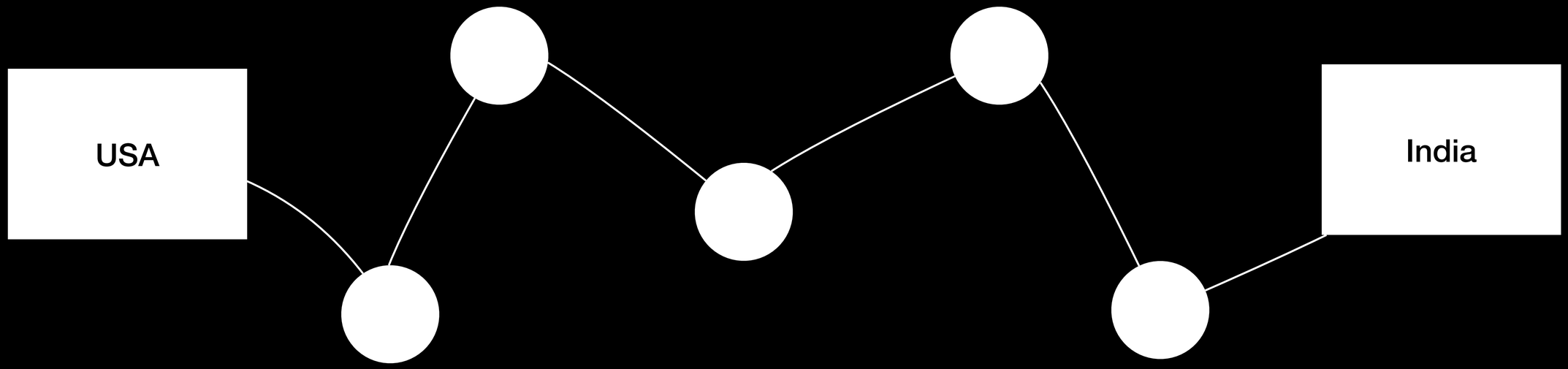
Master Slave Arch

- Component that acts and distributes network or application traffic across a number of servers to improve responsiveness and availability of applications, websites or databases.



Latency

In computer networking, latency is an expression of how much time it takes for a data packet to travel from one designated point to another. Ideally, latency will be as close to zero as possible. Network latency can be measured by determining the round-trip time (RTT) for a packet of data to travel to a destination and back again



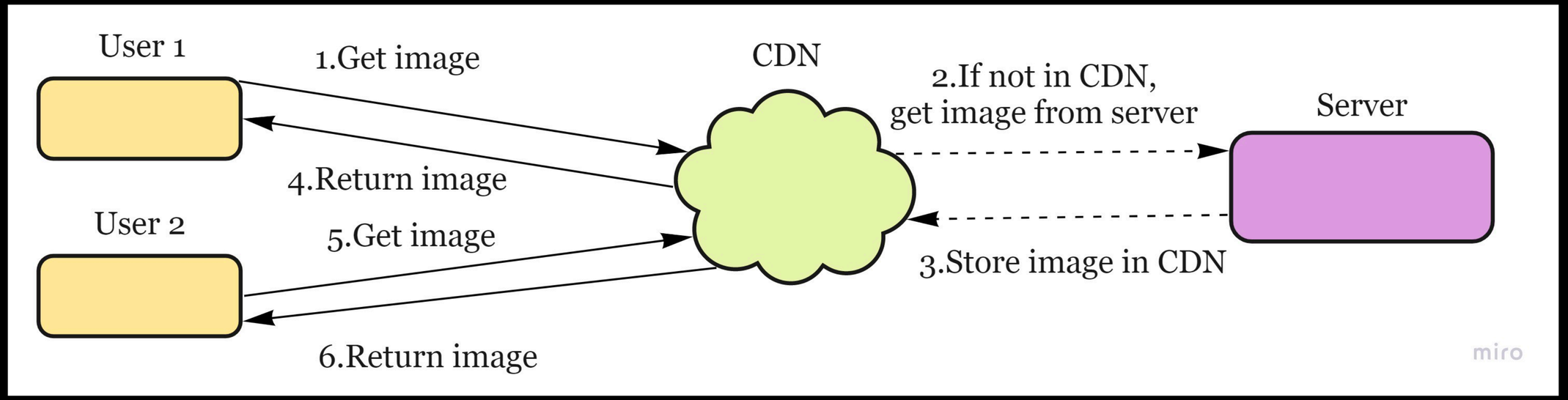
Measured in milliseconds, network latency is the time it takes a site visitor to connect to your webserver, their request to be processed, and the server to begin sending data. Several factors impact latency, including:

- Server performance – There is a correlation between server performance metrics—including server speed, the hardware used (e.g., HDD/SDD drives), and available RAM—and your site latency.

- Round-trips – A round-trip is the journey taken by an object request (e.g., HTML files, stylesheets, and script files) to your web server and back to the user. Round-trip time (RTT) is primarily affected by the distance between webserver and user, as well as the number of intermediate points through which a connection travels.

Content Delivery Network

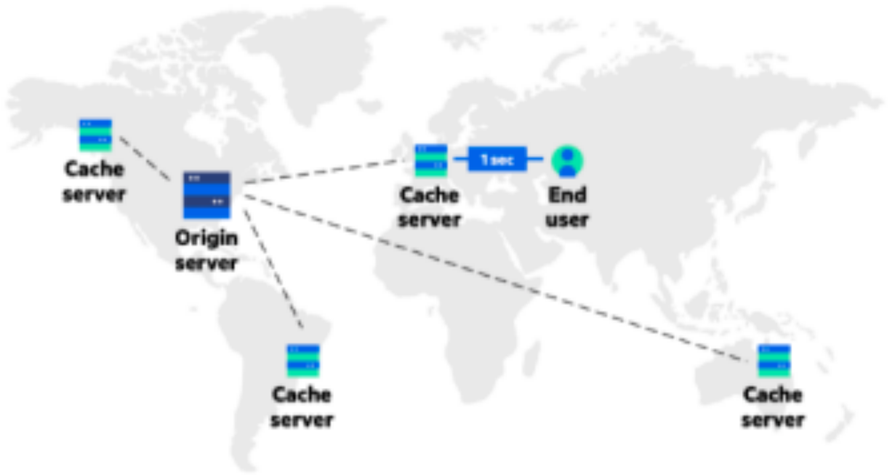
- System of distributed servers (network) that deliver webpages and other web content to a user based on the geographic locations of the user, the origin of the webpage and a content delivery server



Source: https://medium.com/geekculture/cracking-the-system-design-interview-theory-basics-c57f5326181b

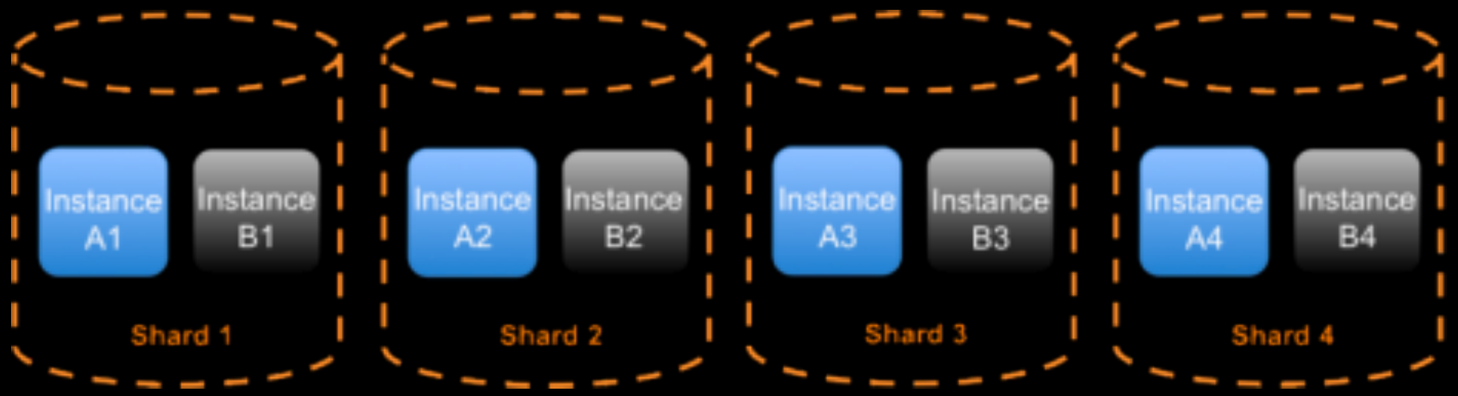
Without CDN

With CDN



Sharding and Shuffling

Sharding is a technique traditionally used with data storage and indexing systems. Instead of spreading traffic from all customers across every instance, we can divide the instances into shards.



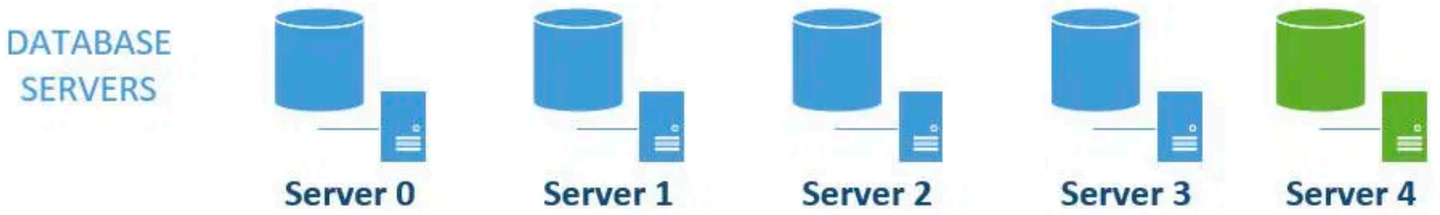
With sharding, we are able to reduce customer impact in direct proportion to the number of instances we have. Even if we had 100 shards, 1% of customers would still experience impact in the event of a problem. One sensible solution for this is to build monitoring systems that can detect these problems and once detected re-shard the problem requests to their own fully isolated shard.

With Shuffle Sharding, we can do better. The basic idea of Shuffle Sharding is to generate shards as we might deal hands from a deck of cards. Take the eight instances example. Previously we divided it into four shards of two instances. With Shuffle Sharding the shards contain two random instances, and the shards, just like our hands of cards, may have some overlap.

hashValue = hashFunction(key)
serverIndex = hashValue % numberOfServers

KEYS RESIDING ON SERVER				
	Key 0 Key 4	Key 1 Key 5	Key 2 Key 6	Key 3 Key 7

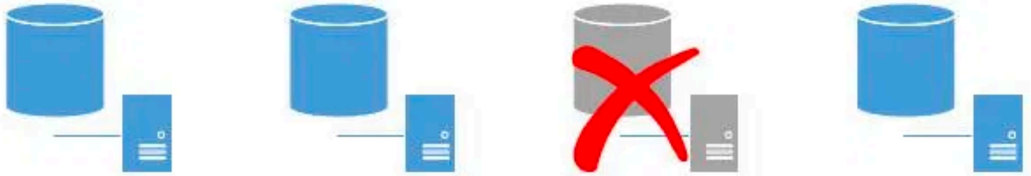
hashValue = hashFunction(key)
serverIndex = hashValue % numberOfServers



KEYS RESIDING ON SERVER					
	Key 0	Key 1	Key 2	Key 3	Key 4
	Key 5	Key 6	Key 7		

hashValue = hashFunction(key)
serverIndex = hashValue % numberOfServers

DATABASE SERVERS



Server 0 | Server 1 | Server 2 | Server 3

KEYS RESIDING ON SERVER

Server 0	Server 1	Server 2	Server 3
Key 0	Key 1		Key 2
Key 3	Key 4		Key 5
Key 6	Key 7		

Consistent Hashing

Consistent hashing is one of the techniques used to bake in scalability into the storage architecture of your system from grounds up.

In a distributed system, consistent hashing helps in solving the following scenarios:

- To provide elastic scaling (a term used to describe dynamic adding/removing of servers based on usage load) for cache servers.
- Scale out a set of storage nodes like NoSQL databases.

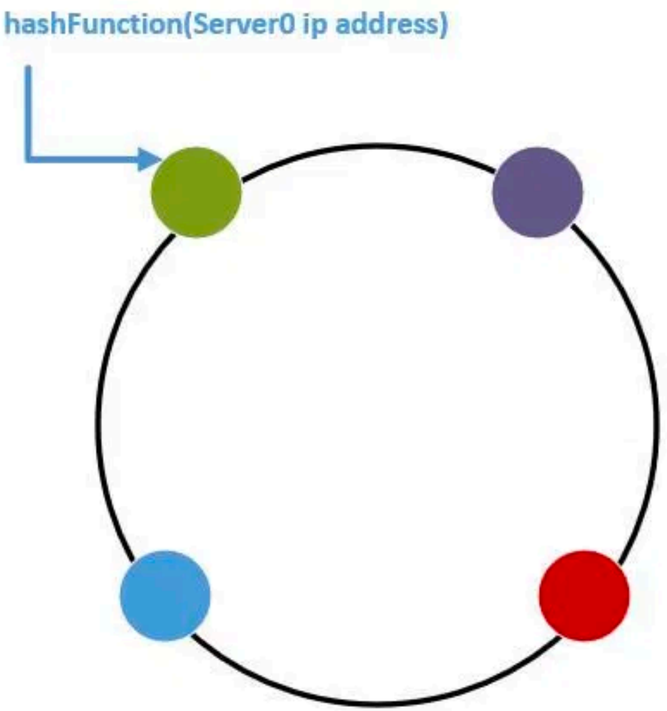
.

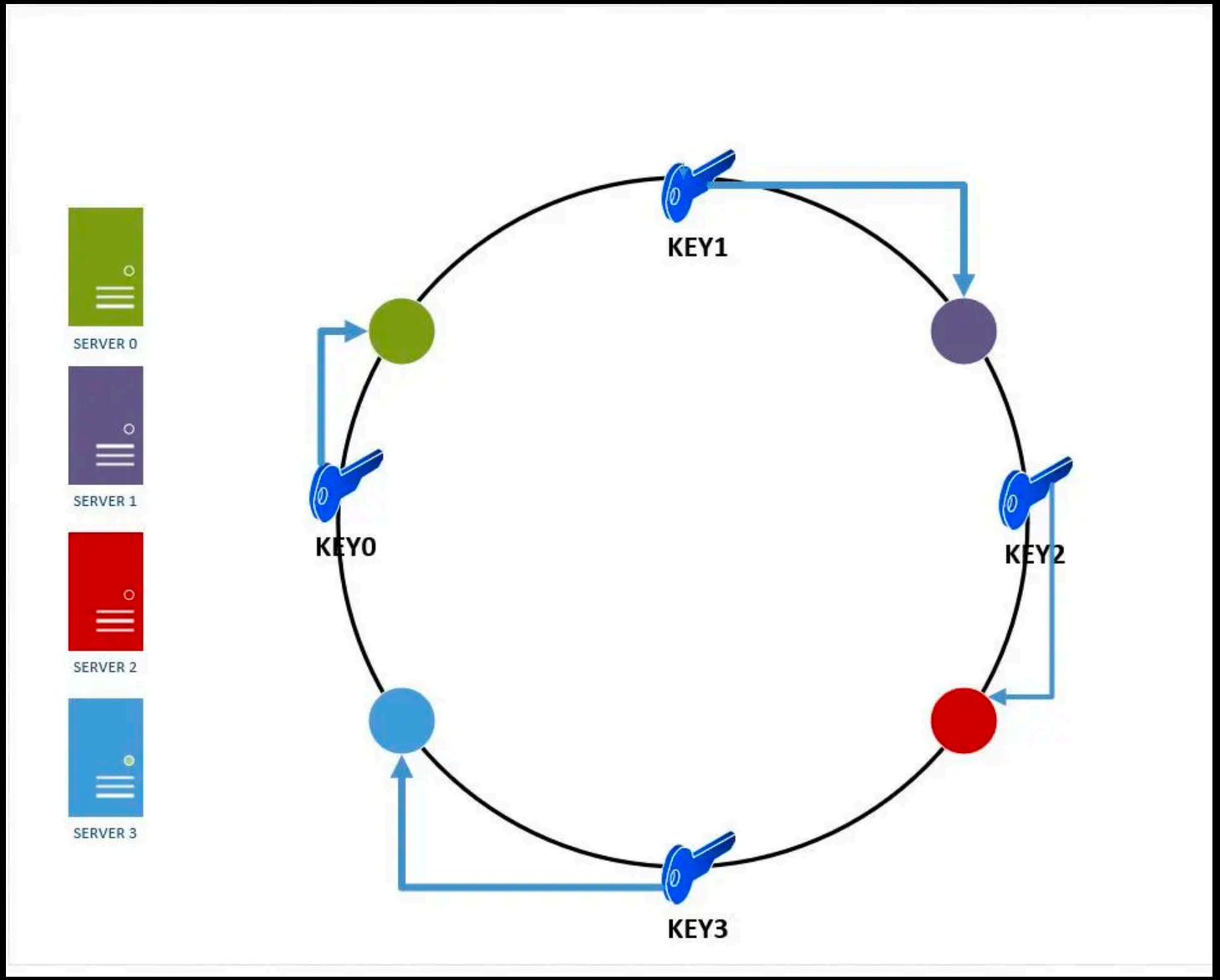

SERVER 0

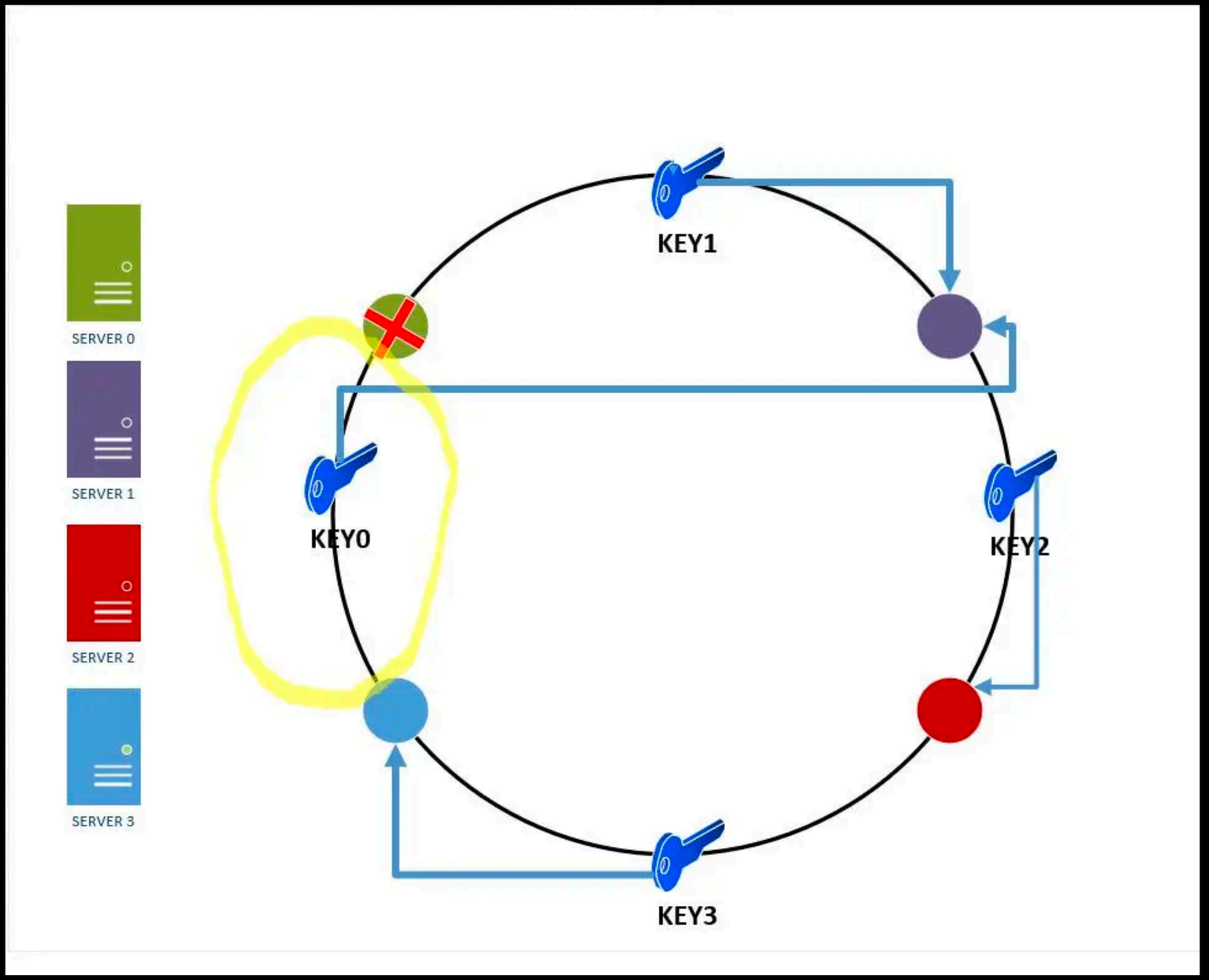

SERVER 1


SERVER 2


SERVER 3

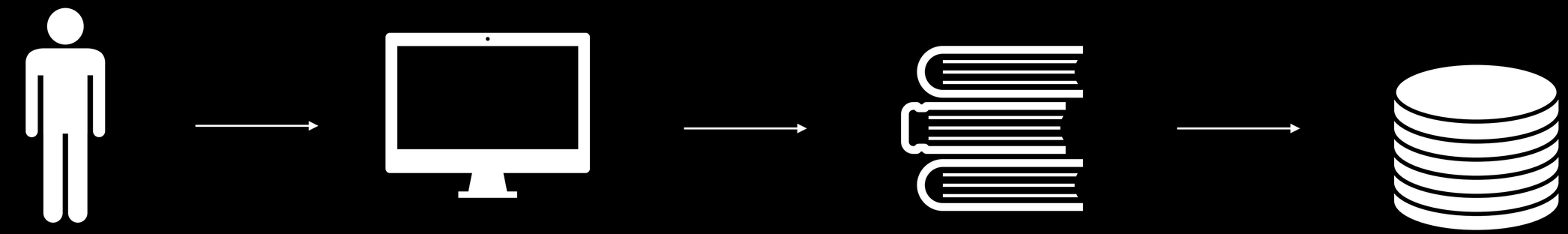

hashFunction(Server0 ip address)


SERVER 0
SERVER 1
SERVER 2
SERVER 3
KEY1
KEY0
KEY2
KEY3


SERVER 0
SERVER 1
SERVER 2
SERVER 3
KEY1
KEY0
KEY2
KEY3

Caching

Temporary storage area that stores the result of expensive responses or frequently accessed data in memory so that subsequent requests are served more quickly.



Client Front End Back End Database

Proxy Server

- Piece of software or hardware that acts as an intermediary for requests from clients seeking resources from other servers.