Disha Shetty
B00965377

CS 552 - Cloud Computing

Ans 1.] 1) (1.) NOSQL stands for "Not Only SQL" meaning that it consists a range of non-relational database management systems that can use different types of data models.

(2.) NO SQL databases are a class of non-relational data storage systems for e.g. Big Table and Dynamo.

(3.) They usually do not require a fixed table schema and in addition to that if any case we need to accquire relation between tables, they do not use the concept of joins.

2.) (1.) CAP Theorem is a fundamental theorem that says as follows:

For a distributed system it is impossible to simultaneously provide or guarantee to be consistent, available, and partition tolerant.

(2.) * Consistency means all copies have same value.

* Availability : it says that every request received by a non-failing node the system must result in a response.

* partition tolerance - A network if partitioned into multiple sub-systems (due to separate optimization and/or failures)

(3.) So basically, just in case of network partition, a distributed system cannot give a guarantee to be providing consistency or availability but both simultaneously.

Ans 3.) (1.) A financial trading application can be taken here as an example. This application allows the traders to buy and sell stocks ~~and~~ in real time which demands high availability. ~~Now due~~

(2.) Now due to this there might be a chance of of heavy traffic or server downtime or any failure of service which could lead to drastic financial loss for the traders and their clients.

(3.) so to avoid it, this application relies on load balancing, failover mechanisms and other hardware software systems. So during failure, it can redirect the traffic to other available resourse without any disruption by scaling.

Ans 4.) (1.) Lets take an example of health care system which stores the patient records, their medical histories and the test results.

(2.) So, these kind of crucial data should be consistent and with high accuracy. ~~If any~~ failure to do so for eg. if any data is inconsistent or incorrect data could lead to incorrect diagnoses which could be fatal.

(3.) High consistency also ensures that users accessing the data at the same time should be the latest one. This is can only be achieved by implementing strict transactional mechanisms i.e either if a change has to be done it should be committed else any failure should be rolled back. (It should strictly adhere ACID properties so that the database remains in consistent even in unexpected events.

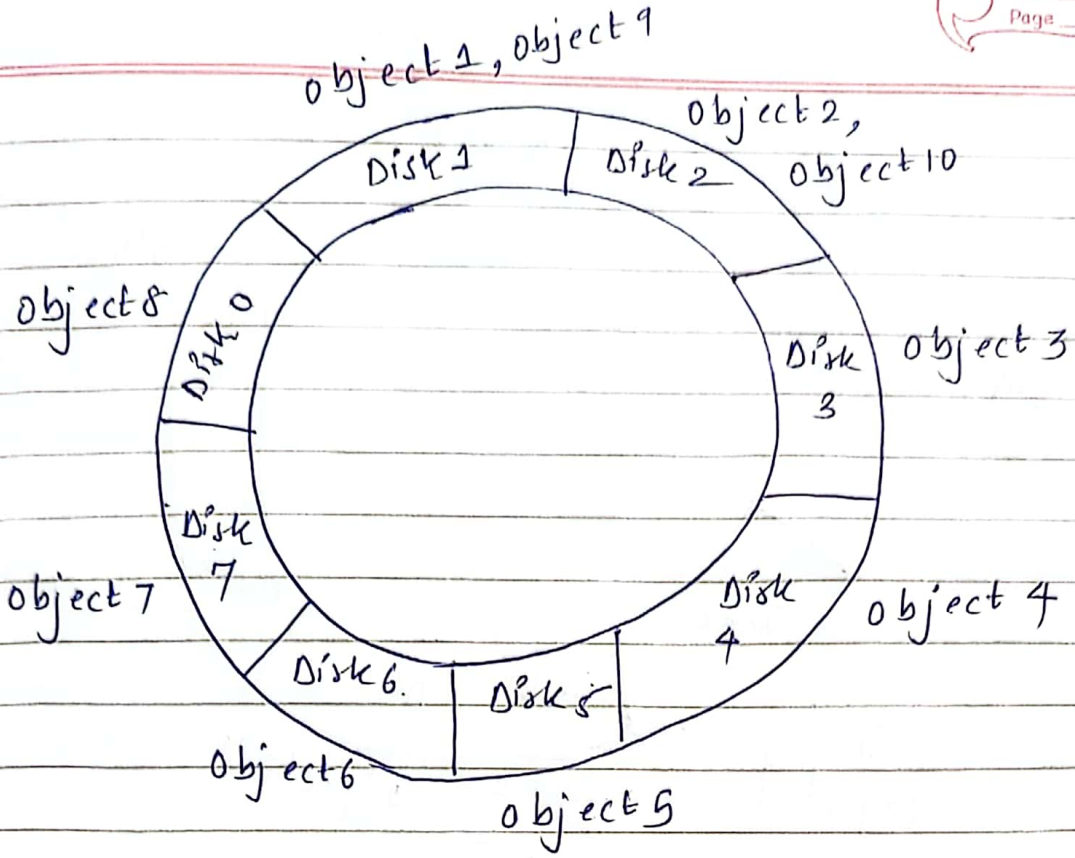
Ans 5.) ① The most fundamental reason for NoSQL is most preferred is that it is ~~usa~~ usually particularly useful for applications that needs to handle very large, semi-structured data sets such as log analysis, social networking feeds and such which requires horizontal scalability and in addition to that to have flexible data modeling.

② for smaller data sets or the ones which have very less query uplations relational (SQL) database is ~~are~~ the better and most opted one. This usually lies true for organizational database which have a very well-defined structure unlike NoSQL (which needs no structure) & relationships between data elements. ↳ uses flexible model.

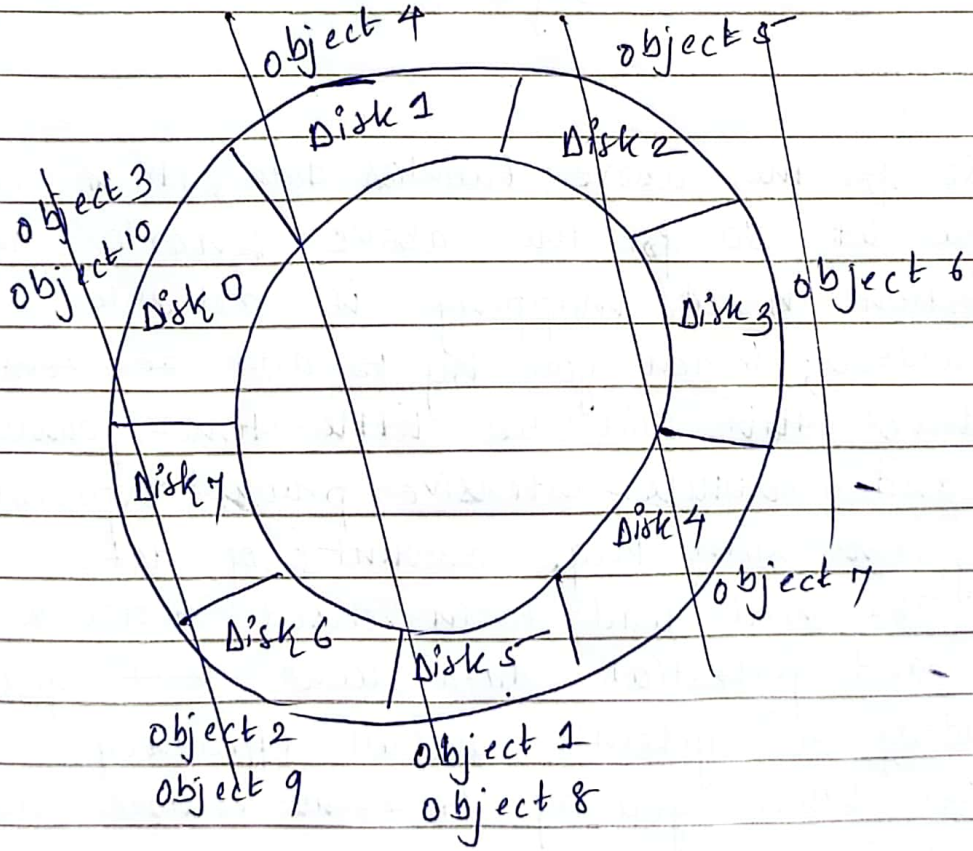
③ Sine NoSQL uses flexible model, it can also accomodate changes to the data model which makes easier to scale up by adding more nodes in database cluster without interruption or compromising ~~the~~ its performance.

Q2.)

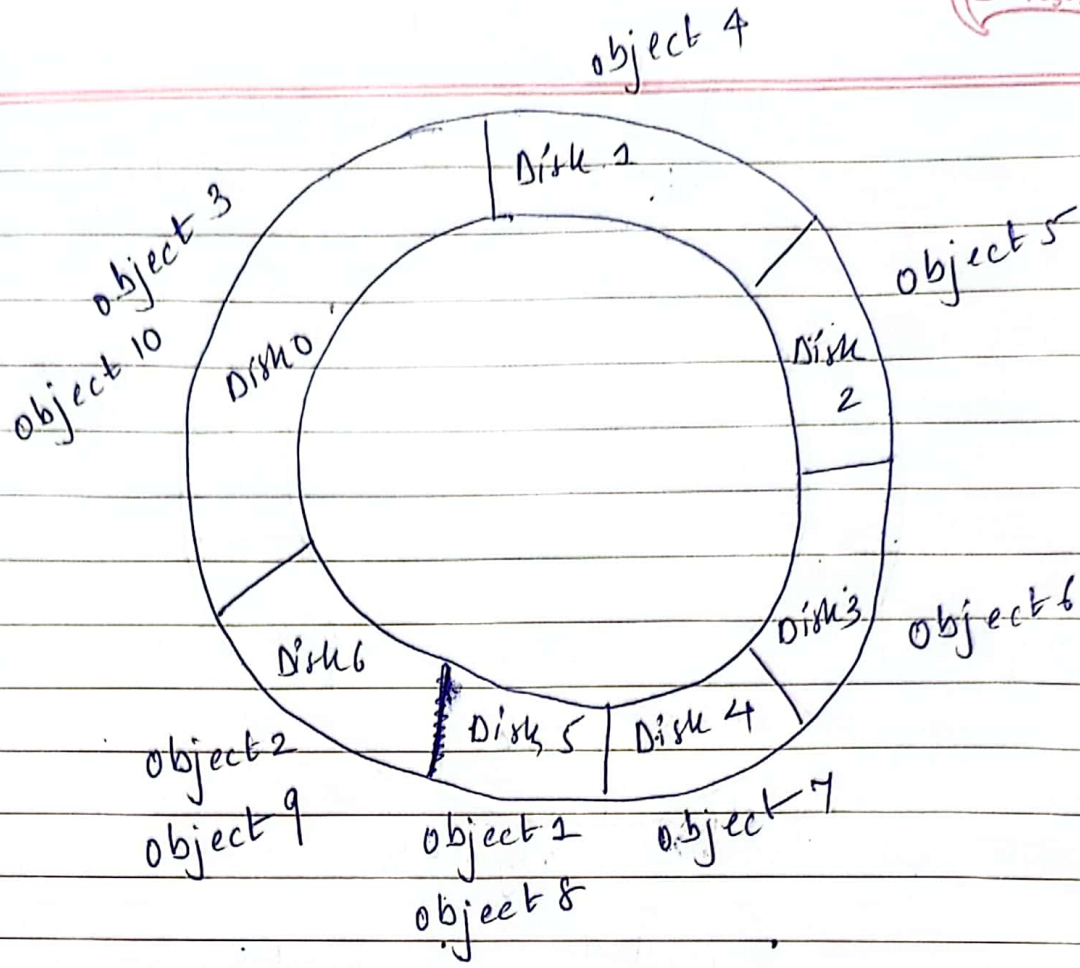
1.)


object 1, object 9
object 2, object 10
Disk 1
Disk 2
object 8
Disk 0
Disk 3
object 3
Disk 7
object 7
Disk 4
object 4
Disk 6
Disk 5
object 6
object 5

Ans 2.)


object 4
object 5
Disk 1
Disk 2
object 3
object 10
Disk 0
Disk 3
object 6
Disk 7
Disk 4
object 7
Disk 6
Disk 5
object 2
object 9
object 1
object 8

Ans 2.]



Ans 3.] ① So for the above tabular data, it is smaller data set. So for the above senario hash function based mapping is reliable.

② However, when you try to add ~~or remove~~ drives this will be little time-consuming and resource-intensive process, especially for systems with large amounts of data.

③ This will lead to increased network traffic and potential data skew ~~leat~~ further leading to system's overall efficiency.

④ ~~For~~ When you try to ~~remo~~ remove the drives, there is a high chance that the hash values of all objects will stay the same, but we need to recompute the mapping value for all objects and then again remap them to the different drives.

Ans 4.] (1) As mentioned in the previous answer, hash function based mapping ~~gives us~~ is not reliable when we need to add or remove drives.

(2) To overcome that, we can opt for consistent hashing algorithm.

(3) In consistent hashing, rather than assigning single object to a disk, ~~we can assi assign a dr no. num number of drives to get~~ each drive will get a new range of hash values it is going to store so each object which gets mapped to the ~~di~~ disk will remain the same and any objects whose hash value is within range of its current drive will remain.

(4) So, ~~for the~~ when we need to add a device, we can give the hash value 3.

~~Object~~	Disk 0	object 1	object 2	~~object 3~~
	Disk 1	object 3	object 4	
	Disk 2	object 5	object 6	
	Disk 3	object 7	object 8	
	Disk 4	object 9	object 10	
	Disk 5	EMP	EMP	} empty.
	Disk 6	EMP	EMP	
	Disk 7	EMP	EMP	

(5) So here, ~~to~~ since disk 5, disk 6 and disk 7 are empty we are free to add objects in it.

(6) Also if we need to delete any object, the disk gets empty, which does not make much a difference.

Ans 5.] ① The major problem that arises ~~for~~ when a ~~there is a large numb~~ the ~~drive~~ object number is large. So each drive has a large range of hash values.

② Due to this, there is a high probability that multiple objects may map to one single drive and ~~also~~ a also another probability that ~~other drive will ha~~ some Objects may map ~~of~~ to another drive which will be way less than the previous one.

③ This in turn leads to imbalance issue i.e it is imbalance distribution and multiple objects mapping to the same device.

④ One of the solution for this issue is "weighted consistent hashing" and it addresses the heterogeneity of underlying devices by assigning different weights to them on the basis of capacity or performance.

⑤ These markers serve to split the large hash range into smaller chunks, which helps to evenly distribute objects into drives and improve load balancing.

Ans 6.] (1.) Considering the device heterogeneity for distributing objects, one solution for this would be "weighted consistent hashing".
(2.) Basically we assign different weights to them based on their capacity or performance. In this way, ~~more objects will~~ the device with higher weigh will receive more virtual nodes and thus a larger portion of the hash range.
(3.) This will help evenly distribute objects into drives and improve load balancing.

Ans 3.]

1.) (1.) Data Replication can bring complexity to data reads due to the need to choose which replica to read from, but it also offers benefits such as improved read performance, higher availability and better fault tolerance.

(2.) By distributing the data across multiple replicas, data replication can help ensure that the system can continue to provide reliable and fast access to data, even in the face of failures or network issues.

(3.) Elaborating the above point, when data is replicated across multiple nodes, if one replica becomes unavailable due to node failure, network issues or maintenance other replicas can still serve read requests, ensuring data accessibility. This can improve data availability and reduce the risk of data loss or downtime.

(4.) Data replication allows for better resilience against data loss, as the chances of all replicas being lost simultaneously are very low. In any scenario if one replica fails or becomes corrupted, the system has the alternate option to use other replicas to recover the data, ensuring that the data is available at all time and reliable.

(5.) So by maintaining multiple copies of data across different nodes, data replication provides fault tolerance and enables data recovery in case of node failures or data corruption.

Ans.

2.) ① To ensure strong data consistency across replicas in a distributed system, its important to implement a consensus protocol.

② Consensus protocols are designed to ensure that all replicas agree on the same state, even in the presence of failures and network partitions.

③ These protocols use a leader election process and a series of message exchanges to reach agreement on the order of operations and ensure that a majority of replicas must agree on the updated value before it is considered committed.

④ So, to ensure strong data consistency across replicas in a distributed system one must implement a consensus protocol. It is important to carefully design the protocol to balance the trade-offs between latency, scalability and fault tolerance and ensure that the protocol can handle different failure scenarios and network conditions.

Ans 3.) ① The major drawback of the above update protocol that ensures strong data consistency is that there is a high chance of increase latency and reduce the write performance. There has to be co-ordination in system regarding the updates across replicas and wait for a majority to respond.

Solution:

(2.) One way to address this drawback is use eventual consistency model. In this replicas are allowed to be temporarily inconsistent and updates are propogated asynchronously without the need for co-ordination and waiting for a majority.

Ans 4.) Data Replication:

Pros:

1.] Simple implementation: Data replication is relatively easy to implement compared to other techniques like erasure coding, which requires more complex algorithms.

2.] improve in Read performance: Read requests can be distributed across replicas, reducing the load and improving overall read performance. This will only help where read performance is critical.

3.] Better fault tolerance and availability: It provides redundancy by storing multiple copies of data across different nodes, ensuring that the system can continue to function even if one or more nodes fail.

Cons:

1.) Bandwidth for replication: Data replication requires data to be transferred between different nodes, which consumes more network bandwid

2.] High storage overhead: This requires multiple copies of data to be stored across different nodes, which can result in higher storage requirements.

Erasure coding

pros:

1.] fault tolerance with lower storage and bandwidth requirements: This provides fault tolerance in a distributed system with lower storage and bandwidth requirements compared to data replication.

2.] Reduced storage overhead compared to replication: This reduces storage overhead required to provide fault tolerance in distributed system. Encoding data into multiple fragments, where only a subset of those fragments are needed to recover the original data.

cons:

1.] ~~More~~ complex implementation: This requires additional algorithms and mechanisms to encode and decode data fragments and manage those fragments across multiple nodes.

2.] Slower read and ~~xea~~ write performance due to encoding and decoding: This results in slower read and write performance compared to data replication.

3.) Computational overhead (increase): This leads to higher CPU and memory usage on nodes, which can impact overall system performance.

A senario wherein → large-scale storage system such as CDNs or big data platform where factors such as storage capacity and cost are taken into consideration as the most. So in this case erasure coding can provide the required fault tolerance and durability with significantly lower storage

making it a more cost-effective solution. For eg. erasure coding can be used to protect against data loss while minimizing storage costs.

Ans 4.]

1.) ① Once the new flow arrives at an Open flow switch, the switch does not know how to process the packet since there are no rules initially. This forwards the packet header to SDN controller.

② SDN controller receives the packet header and then uses Alice's algorithm to determine how the packet should be forwarded based on the network state, topology & packet header information.

③ Later on it sends a flow rule back to the openflow switch indicating how the switch should handle the packets.

④ Moving on the switch after receives the flow rule installs it and processes the current packet and subsequent packets of the same flow following on the rule.

⑤ ~~As and as the flow~~ The controller continues to make decisions as and as the flows arrive and send flow rules to switches and ~~then it updates the flow~~ simultaneously updates their flow tables.

Ans 2.) ~~In an SDI~~

(1.) The communication between the SDN controller and Openflow switches can cost more with respect to latency and bandwidth.

(2.) This cost is required for short flows that contain only a few packets which will suffer as the overhead of requesting and installing flow rules can be relatively high compared to the total amount of data transmitted in the flow.

(3.) We can follow the below strategies:

* • proactive flow rule- It This strategy can be used to reduce the overhead of communication between the SDN controller and openflow switches for short flows. Instead of waiting for the first packet of a flow to arrive, the controller can push flow rules to switches proactively based on anticipated traffic patterns. Switches will already have the necessary rules in place when a flow arrives, eliminating the need for the initial communication.

* Flow aggregation: It is particularly useful for networks that handle a high volume of traffic. By reducing the number of rules required, the communication overhead between the controller & switches is minimized, which can help to reduce latency and improve overall network performance.

Ans 3.) (1.) To ensure that virtual machines (VMs) belonging to different tenants on a single physical host are isolated, Open vSwitch (OVS) can use VLAN tagging or VXLAN tunneling.

(2.) These techniques enable VMs from different tenants to use the same physical network infrastructure while keeping the traffic seperated.

(3.) The L2 agent is responsible for creating and managing virtual networks specific to each tenant and is involved in this process.

Ans 4.) (1.) Here, L2 agents plays a critical role in enabaling communication between VMs within the same tenant network.

(2.) The agent creates and manages the virtual network infrastructure that connects VMs of the same tenant.

(3.) In this case, VM1 and VM2 both belong to Tenant 1, so the L2 agent will create a virtual network for Tenant 1 and connect VM1 and VM 2 to this network, allowing them to communicate with each other. ~~This ensures that~~

(4.) This ensures that the traffic between VM1 and VM 2 remain isolated from other tenant's traffic while sharing the same physical network infrastructure.

Ans 5.) To enable communications between VMs of different tenants, following steps need to be ~~taken care~~ implemented:

(1.) An external network needs to be created which acts as a gateway for inter-tenant communication. Can be physical or virtual network.

(2.) A router needs to be created which connects Tenant 1 and Tenant 2's virtual networks to the external network. The L3 agent is responsible for creating and managing the routes.

(3.) The L2 agent needs to be configured to allow inter-tenant traffic as L2 agent only allows traffic within the same tenant network.

Ans 6.) To implement the specified firewall rules using OVS flow table entries for VM1 ~~a~~ with IP 192.168.1.1. Here's what we can do: (Assumptions)

(1.) Create a flow table entry to allow ingress traffic on port 80 (HTTP) and on port 22 (SSH) for VM1. (Define a flow rule)

(2.) ~~VM1 is connected to an~~ OVS bridge name 'br-int' is connected to VM1.

(3.) port 'vm1-port' on the OVS bridge where VM1's virtual network interface card (vNIC).

(4.) Focus on TCP traffic for the specified ports. & in addition to that consider both ingress & egress traffic for VM1.

1.) Define flow rule to allow incoming traffic on port 80.

ovs-ofctl add-flow <bridge-name> "table=0, priority=100, ip, nw_dst=192.168.1.1, tcp, tp_dst=80, actions=normal".

2.) Define flow rule to allow incoming traffic on port 22.

ovs-ofctl add-flow <bridge-name> "table=0, priority=100, ip, nw-dst=192.168.1.1, tcp_dst=22, actions=normal".

3.) Block all other ingress traffic to VM1 with IP 192.168.1.1

4.) Defining the ~~rule~~ flow rule to drop all incoming traffic.

ovs-ofctl add-flow <bridge-name> "table=0, priority=50, ip, nw_dst=192.168.1.1, actions=drop".