CS580K : MIDTERM 2

Name : Shubham Patwa

1. (15 points) Q1: Scalable Databases – NoSQL

(1) What does NoSQL stand for? (3 points)

It stands for Not only SQL. It is a class of non-relational data base systems.

(2) What is the CAP Theorem? (3 points)

CAP Theorem has 3 characteristics . It says that it can provide only two of them simultaneously and it is not possible to provide all the characteristics.

1. Consistency : All copies have same value
2. Availability: Every request received by a non-failing node the system must result in a response
3. Partition tolerance : Even after network is partitioned into multiple sub systems , it still works correctly.

(3) Please list one scenario demanding high availability? (3 points)

Scenario : Web applications example : Client requesting something after it must get back a meaningful expected response. The website shouldn't be down and must give back a expected response.

(4) Please list one scenario demanding high consistency? (3 points)

A consistency model determines rules for visibility and order updates.
Scenario : A transaction has taken place with a bank account and the same account has been requested another transaction-> The account balance must be updated accordingly and immediately as expected. If a error occurs, it might affect the next transaction.

(5) What do you think are the fundamental reasons that NoSQL has better scalability than Relational (SQL) databases (3 points)?

Even if tried with relational databases for scaling, it does not scale well for writes , even with adding a cache layer

1. NoSQL does not require a fix schema as SQL/relational databases require.

2. NoSQL has CAP theorem i.e , it relaxes on one or more ACID properties.

These are the fundamental reasons.

2.

(**1) Suppose we have 8 devices as shown in Figure 1, how do you distribute the above 10 objects based on their mapping values? Please draw it in Figure 1. (4 points)**

Disk 0 : Object 8

Disk 1 : Object 1 , Object 9

Disk 2: Object 2, Object 10

Disk 3: Object 3
Disk 4: Object 4

Disk 5: Object 5

Disk 6:Object 6

Disk 7:Object 7

(2) Now we remove one device, and only have 7 devices left as shown in Figure 2. How will the 10 objects be re-distributed? Please draw it in Figure 2 as well. (4 points

Disk 0 : Object 3, Object 10

Disk 1 : Object 4

Disk 2: Object 5

Disk 3: Object 6
Disk 4: Object 7

Disk 5: Object 1, Object 8

Disk 6:Object 2, Object 9

(3) What are the main problem(s) from this (above) hash function based mapping, when we need to delete (or add) devices? (4 points)

As we can see, when we use hash based function mapping, the hash values of all objects remain the same , but the mapping value to disk devices have all been re computed. Thus, this is expensive operation. Solution for this is the 'consistent hashing algorithm'

(4) Consistent hashing algorithm is used to solve the above problem(s) caused by simple hash functions for object distribution/mapping among available devices. Please come up with a "consistent hashing" solution for distributing the above 10 objects among 8 devices, and demonstrate how it solves the problem(s) caused by the simple hash functions. For example, when we need to delete (or add) one device, what would happen for your "consistent hashing" solution? (5 points)

We create the range of hash value for each drive : (Hypothetical Values)

Disk Range of Hash Value :

Disk 0 : 0000 – 10005

Disk 1: 10006 – 20011

Disk 2:….

..

..

So on..

Now, For distributing the above 10 objects, based on already calculated hash values of objects, we just put the objects in the disks according to their hash value as we put objects in a bucket.

Since, the values of all the objects are in serial order,

Disk 0 contains: object 1, Object 2, Object 3, Object 4 , Object 5

Disk 1 contains : object 6, Object 7,Object 8,Object 9, Object 10

Disk 2 to disk 7 will be empty for now.

Now all object have been assigned a disk.

Now, when we remove a object, suppose let's say object 1, each disk will get a new range of hash values it is going to store. However, object's hash value will still remain the same.

But here, number of object that have to be moved due to this removal is very less.

This is the advantage of consistent hashing algorithm

(5) Please state any issues with your "consistent hashing" solution, and briefly describe the possible solutions. If you think there are no issues, please also say so. (5 points)

As we can see, disk drives from disk 2 to disk 7 are empty since the objects hash values were in serial order. Since the consistent hashing algorithm uses range of hash values of objects to store in disk drives, multiple objects have gotten mapped to a just two drives instead of 8. This will create a imbalance issue.

This is the drawback of consistent hashing algorithm

Solutions:

We can use multiple markers in consistent hashing algorithm .

Here instead of having one big hash range for each drive, multiple markers serve to split those large hash range into smaller chunks.

(6) "Consistent hashing" does not consider the heterogeneity of the underlying devices. For example, some devices are more powerful (e.g., with more space), while some devices are not. Any solutions to consider the device heterogeneity for distributing objects? (5 points)

Here too, multiple markers can be used. They help to evenly distribute the objects into drives with the help of load balancing. Load balancing can be of help to consider the heterogeneity of underlying devices.

Please Turn Over

3. (16 points) Q3: Object Storage – Reliability Data Replication is a straightforward-yet-powerful technique in large-scale, distributed systems such as object storage. For example, if one node containing one data replica fails, we can always get the data back from other replicas. However, it brings complexity for managing these replicas.

(1) Do you see any complexity for data reads? Actually, are there any benefits for reads brought by data replication? (4 points)

Well, no there will be no complexity for data read except for choosing which replica to read data from.

Benefits:

The replicas can be on various locations. One of the benefit of data read from a replica can be that it can read data from its closest replica. This will save time and cost.

(2) For each update to the data, definitely we have to update all its replicas (say 3 replicas). To ensure strong data consistent — clients always see the same value of the data from different replicas — how do you design the update protocol? (4 points)

We first forward updates to all other replicas

Once the replicas receive the updates, they send the client an acknowledgement.

Once the final commit has been done to the disk from the replicas , they send a final commit.

This makes it strongly consistent and safe , but slow .

(**3) What is the major drawback of the (above update protocol that ensures strong data consistency? Do you have some ideas to address this (hint: we may not need data consistency right away. Instead, we can make data consistent eventually, after a while.)? (4 points)**

Major drawback:

As I stated previously, though the consistency is strong and safe, it is slow.

Solution : Eventual consistency

We do not need to send ack right away. When no updates occur for a long time, eventually all updates, will go through the system , and all replicas will be consistent and then send a final ack/commit.

(4) Under some scenarios, we can replace “data replication” with “erasure coding” to achieve the same goals such as data durability (i.e., data is always correctly stored) and fault tolerance. Please list the pros and cons of these two techniques. Can you come up with one representative scenario that we prefer “erasure coding” instead of “data replication”? (4 points)

Replication is where there are full copies of stored objects

With erasure code, we do not store full copies, thus it utilizes less space. Here the file of k blocks is encoded into n blocks with parity.

Erasure Code:

Pros:

1. Uses Less Space
2. Allows for failure of two or more elements

Cons:

1. Parity calculation is CPU intensive
2. Increased latency

This is viceversa for Replication.

Scenario where we prefer erasure coding :

When we do not have much space, but cpu intensive operations are fine, then we can use erasure coding, for example: Object based storage in cloud.

Please turn Over

4. (27 points) Q4: Software Defined Network (1) Alice implements a new network algorithm running on the controller side as shown in Figure 4. Suppose there are no rules in all OpenFlow switches, initially. Please describe the main steps how Alice's algorithm works in such an OpenFlow based SDN network setup. (5 points)

Once the n/w receives the 1st packet , the openFlow switch will talk to the controller asking for rules. Basically it will encapsulate the received packet within a message and send that message to the controller. Once controller receives the packet , it will look up the rules & n/w services for the flow.

The controller then finds that there are no rules yet and finds that Alice's code will handle all the newly arrived packets. So the controller will insert all the rules programmed by Alice's code to all the switches involved in the flow. The controller will then send a message back to the openFlow switch that the rules have been installed and it can now forward it by following all the rules.

Thus, the openFlow switch will forward the packet to the next one and so on, using all the rules.

(2) Communication between the SDN controller and OpenFlow switches is expensive. Which types of flows, short flows (i.e., containing fewer packets) or long flows (i.e., containing more packets), suffer more from such communication cost, and why? What can we do to reduce this overhead. (5 points)

The short flows will suffer from such communication cost.

Consider only one packet is in transmission. Now, when the packet has to travel, new rules have to be made, and other things have to be taken care of for a single packet. Delay will be more. Cost will be more. This is all just for a single packet.

However, for more packets, a little cost is fine since many packets will be in transmission and the rules will be same for all the packets.

(3) To ensure isolation among VMs (e.g., VM1 and VM1') belonging to different tenants within a single physical host, what isolation technique should OVS use? Which network service of OpenStack (L2 or L3 agent) will be involved? Note that, L2 agent is for switching services, while L3 agent is for routing services. (4 points)

A simple way is to use VLAN since they use the same physical host.

It should use L2 agent.

(4) How to ensure that VM1 can communicate with VM2 (both belonging to Tenant 1)? Which network services of OpenStack (L2 or L3 agent) will be involved? (4 points)

Here, the communication will take place over the router.

We can ensure the communication by defining gateways in advance. However, it will affect performance.

Consider VM2 will go to Netgateway 2 which will inturn go to Netgateway1 . Netgateway 1 will communicate with VM 1 and in reverse path will take place to VM2.

Here, L3 agent will be involved since the netgateway communication will take place over software routers.

(5) What functions/services do we need to enable Tenant 1 and Tenant 2 to talk to each other? Which network services of OpenStack (L2 or L3 agent) will be involved? (4 points)

We need to enable network virtualization by both the OVSes and controller and as well as net gateways. Here too, L3 agent will be involved since they can pass over net gateways .

By enabling network virtualization, we can also enable tunneling. With the help of OVS and controller , Tenant 1 and tenant 2 will be able to talk to each other using net gateways / tunneling.

(6) As we can see from Google Cloud Platform (and also from OpenStack), Firewall rules are associated with VMs (e.g., blocking most of the ports). We can realize such rules within OVS. Please realize the following Firewall rules using ovs flow table entries: Blocking all ports except for port 80 and 22, for VM1 with IP being 192.168.1.1. Please state all your assumptions. (5 points)

We know that :

SSH/HTTP usually use: port 22

TCP/UDP usually use : port 80

Now assuming that they use these ports as default, it means that , if we are blocking all other ports , we are essentially isolating that VM from other VMs .

5. (15 points) Q5: Put All Together Now we have seen many techniques which are used in industry. Suppose you are building Facebook, or Amazon, or Google-like web services. What techniques do you plan to leverage for what kind of purposes? You can specify what type of website you are going to build, what types of services you want to provide, and what kinds of techniques you want to apply (and why you want to use these techniques). How do you address issues like dynamic workloads from clients, heterogeneous hardware resources, possible datacenter outrage, reducing IT cost, etc. Please justify your choices. (Hint: you don't need to list all the services. Instead, you can focus on one particular service and think about if you are going to deploy it, what technique do you need?)

Suppose I am building Amazon like website .

It has multiple services such as amazon aws, amazon e commerce, amazon prime, etc.

I will be focusing on amazon prime music.

I will be building only the amazon prime music website.

For website building, -> I will use serverless computing . Since, it will handle dynamic client workloads and autocaling automatically,

For database for music, I will NoSQL database such as amazon dynamo DB. It wil integrate and will also be much better to scale horizontally.

I can also use lambda functions (example : for getting input from a user and sending a song his way)

For hosting the website, I will use aws ec2 instance.

Now, I don't need to worry about possible datacenter outage, since it will be taken care of aws.

IT cost will be reduced since autoscaling is possible horizontally as it will use only as much as it needs.