SYSTEM DESIGN CHEATSHEET - ARYAN

What type of database to choose?

1)SQL db:

- Relational, when we need a table and have to perform joins
- When ACID properties are required (for example, bank transfers would require consistency and isolation, so bank management systems would use RDBMS)
- When a rigid schema is required. For example, when you want to make strict relationships between tables, or you wanna maintain unique, not_null constraints.

2)NoSQL db:

-Document-based, key-value pairs, Graph database
-You choose NoSQLDatabase when flexible schema is required.
For e.g.e-commerce like Amazon can have Products in varied schema. Electronic products can have capacity (lt), power-rating (3- star) whereas Clothing products can have size (S,M,L,XL), material (cotton).
-You choose NoSQL when horizontal scaling is required.They scale very well.

****A general pattern for interviews or designing any system is to keep the dimension schemas(that do not change for years and years) under NoSQL structure and incremental data(which arrives everyday and needs to be updated in the db) under SQL structures.**

What is scaling?

-adjusting the server capacity as the amount of data grows
Types:

1)Vertical scaling:

Database can be scaled verticaily by adding more power (CPU, RAM or SSD) to a single machine,so that database can handle more load. Which is why it can't add power beyond a point. SQL dbs are often scaled like this

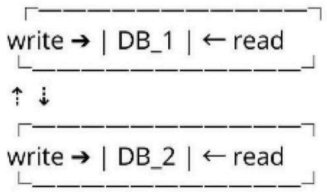
2)Horizontal scaling:

Adding more computers and balancing loads with load balancer. (aka sharding). It is easier on NoSQL dbs. Database sharding means partitioning of data in multiple databases. Incoming data can be saved in databases based on some key. for e.g. user_id in case of saving user profile.

Availability: Availability is measured based on replication factor.Replication factor of 4 means a database is having 3 additional copies which keep themselves in-sync.

Active-Active:

Data read and write happen in any database replica. (Eventual read and write Consistency, High Availability, High Performance)



Eventual Read Consistency:-

1. record_A write request to DB_1
2,record_A read request from DB 2 immediately

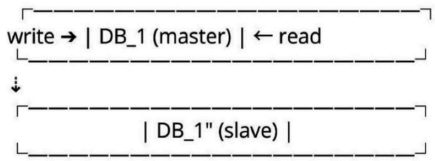
record_A is not be available in DB_2 if sync is yet to happen

Eventual Write Consistency:-

1. record_A write request to DB_1
2. record_A update request to DB_2 immediately
record_A is not be available in DB_2 if sync is yet to happen

Active-Passive (Master Slave):

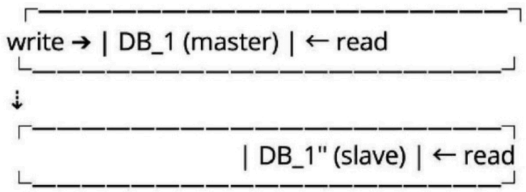
Data read and write happen only in master database. Data is synced with slave database asynchronously. When Master goes down,Slave started serving as master.
(High read and write Consistency, High Availabilîty, Low Performariceÿ



Active-Passive is generally used in financial sector where read and write consistency of financial transactions is very important.

Write-Read (Master-Slave

Data write happen in master database. Data read happen from any database.
(High write and eventual read Consistency, High Availability, Low write and High read Performance)



Write-Read is generally used in social media such as Facebook, LinkedIn, Twitter, Instagram where read/write ratio is very high and eventual read consistency is acceptable.

Geo-Sharding: Sharding based on location. Used by apps like uber and grab to store live location of drivers or by tinder to provide location based recommendations

How to do?

1. Divides the geo locations into cells (fore.g.1 cell= 100x100 miles square).
2. You get the live location (coordinates lat, long) of user and pass to S2 library, which returns the cell_id in which that location falls into.
3. Use this cell_id as a key to partition the data.

Consistent Hashing

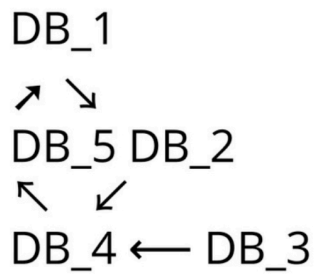
Consistent hashing is the algorithm used by -

1. Most of the load balancers to distribute the load across multiple services.
2. Most of the database horizontal partitioning to distribute the data across multiple databases.

The idea of consistent hashing is to place all the services (or databases) across the consistent hash ring and distribute the load based on some hashing algorithm. This gives us the flexibility to remove or add the service (or database) to the ring without disturbing the whole cluster with minimum or no data loss.

total database in cluster =5 [DB_1, DB_2, DB_3, DB_4, DB_5]

consistent hashing =5 databases makes a consistent hash ring



-If DB_2 goes down,Next database in the ring i.e. DB_3 is going to take additional load (future writes). If replication is not implemented then existing data of DB_2 will be lost but other database will still work.

-If DB_6 is added newlybetween DB_5 andDB_1 then it share the 50% load of the next database in the ring i.e.DB 1.

CACHING:

Keeping recently used data in some local memory space so that re access of that data is faster.
****READ ABOUT REDIS**

TEXT SEARCH CAPABILITIES:

Almost all the websites requires searching capabilities in one way or another. For e.g. Amazon for product search, Netflix for movies search, Youtube for video search, Social media websites for user profile search, and so on and so forth. Some of the popular text searchengine-
-Elastic Search(recommended)
-Apache Solr
-Apache Lucene
*Apache Lucene is the core framework written based on Map- Reduce and inverted-index algorithm.
*Elastic Search and Apache Solr has been written on top of Apache Lucene provide additional search capabilities such as fuzzy search, type ahead, search suggestions.
*You don't rely on these for storage. You generally save data in your primary database and write to Elastic Search or Apache Solr asynchronously.

What is Bloom Filter?

Bloom filter takes the constant time and space to answer Yes/No to this kind of questions:-
1. Does this username exist or already taken?
2. Does this item exist in the set?

Bloom filter gives:-
1.100% accurate result when it says "No”
2. Probabilistic result when it says "Yes”. Sometimes it might be false positive.

Bloom filter data-structure is represented as an array of bits like this:-

Index	0	1	2	3	4	5	6	7
8 Bit Array	0	0	0	0	0	0	0	0

Case1:
Check username “John” exist?
-hash1(john) = 100 => 100%8 =4
-hash2(john) = 54 => 54%8 =6
Since 4th and 6th bit of our array is not set, “John” doesn't exist for sure. Set the bits 4th and 6th bit now:-

Index	0	1	2	3	4	5	6	7
8 Bit Array	0	0	0	0	1	0	1	0

Case2:
Check username “Bill”exist?

hash1(Bill) = 81 => 81%8 =1

hash2(Bill) = 31 => 31%8 =7

Since 1st and 7th bit of our array is not set, “Bill” doesn't exist for sure. Set the1 st and 7th bit now:-

Index	0	1	2	3	4	5	6	7
8 Bit Array	0	1	0	0	1	0	1	1

Case3:
Check username “Garry” exist?

hash1(Garry) = 89 => 89%8 =1

hash2(Garry) = 90 => 90%8 =2

Since 1st bit is set but 2nd bit is not set, “Garry” doesn't exist for sure. Set the 2nd bit now:-

Index	0	1	2	3	4	5	6	7
8 Bit Array	0	1	1	0	1	0	1	1

Case4:
Check username "Jack” exist?

hash1(Jack)=89 => 89%8 =1

hash2(Jack) = 90 => 90%8 =2

Since both 1st and 2nd bit are set, “Jack" may or may not exist. Check the database.

Conclusion:

1. Number of bits takes very ĺess space and in practice numbers of bits in array can be huge providing more precise results. We have used only 8 bits in the example
2.Bloom fiiter provide 100% accurate result if user doesn't exist
3. Bloom fiiter provide estimated resuit if user exist. Estimated result is more precise if hash function is evenly distributed and more hash functions are used. We have used only 2 hash functions in the example

CASE STUDIES:

- https://lnkd.in/eJGpAEX6
- https://lnkd.in/eEAUhkp2
- https://lnkd.in/eXeXxnpm
- https://lnkd.in/eBeSW6Q3
- https://lnkd.in/enjENKRf
- https://lnkd.in/eUR7mUya
- https://lnkd.in/eVtTh6pY
- https://lnkd.in/eYpk9wat
- https://lnkd.in/eq_TGNHK
- https://lnkd.in/eNgMkQjN
- https://lnkd.in/etCt5KhG
- https://lnkd.in/eArpPubt
- https://lnkd.in/e6VkezVX
- https://lnkd.in/ewR5fNnN
- https://lnkd.in/e-WjtfdY
- https://lnkd.in/e_SpQRhm
- https://lnkd.in/e_kn-ekT
- https://lnkd.in/eT9HYZzd
- https://lnkd.in/edE3h42S
- https://lnkd.in/e_HxRFRZ
- https://lnkd.in/e7Vk62Ge
- https://lnkd.in/eVKeuVw8
- https://lnkd.in/e7m6VsJE
- https://lnkd.in/eGhdJRyH
- https://lnkd.in/e4-uXTJD
- https://lnkd.in/eTR5KYCW
- https://lnkd.in/eMd7_rXc
- https://lnkd.in/ee4Wz9ij
- https://lnkd.in/eTZcYpis
- https://lnkd.in/eVQKG2jn
- https://lnkd.in/eQrUTZdp
- https://lnkd.in/erS838tQ
- https://lnkd.in/eCZNfCJi
- https://lnkd.in/e5_JzgBt
- https://lnkd.in/exMTnp2i
- https://lnkd.in/e5GmQJHY
- https://lnkd.in/eXZkp-rS
- https://lnkd.in/e8m6-R2P
- https://lnkd.in/eG2R9Hd8
- https://lnkd.in/ejXs3B8E
- https://lnkd.in/ex6t4yjb

Follow @howaryanakhouri on insta :)