System Design (/cour ses/system-design)
/ Stor age Scalability (/cour ses/system-design/topics/stor age-scalability/)
/ Highly Available Database

Show Notes

Highly Available Database

Bookmark

Design a distr ibuted key value stor e which is highly available and is networ k par tition toler ant

Blog (https://blog.inter viewbit.com) | About Us (/pages/about_us/) | FAQ (/pages/faq/) | Contact Us (/pages/contact_us/) | Ter ms (/pages/ter ms/) | Pr ivacy Policy (/pages/pr ivacy/)

System Design Inter view Questions (/cour ses/system-design/) | Google Inter view Questions (/google-inter view-questions/) | Facebook Inter view Questions (/facebook-inter view-questions/) | Amazon Inter view Questions (/amazon-inter view-questions/) | Micr osoft Inter view Questions (/micr osoft-inter view-questions/) | Puzzles Questions (/puzzles/)

Features:

> *This is the first part of any system design interview, coming up with the features which the system should support. As an interviewee, you should try to list down all the features you can think of which our system should support. Try to spend around 2 minutes for this section in the interview. You can use the notes section alongside to remember what you wrote.*

Q: What is the amount of data that we need to stor e?
A: Let's assume a few 100 TB.

Q: Do we need to suppor t updates?
A: Yes.

Q: Can the size of the value for a key incr ease with updates?
A: Yes. In other wor ds, its possible a sequence of keys could co-exist on one ser ver pr eviously, but with time, they gr ew to a size wher e all of them don't fit on a single machine.

Q: Can a value be so big that it does not fit on a single machine?
A: No. Let's assume that there is an upper cap of 1GB to the size of the value.

Q: What would the estimated QPS be for this DB?
A: Let's assume around 100k.

- **Estimation:**

> *"This is usually the second part of a design interview, coming up with the estimated numbers of how scalable our system should be. Important parameters to remember for this section is the number of queries per second and the data which the system will be required to handle.*
> *Try to spend around 5 minutes for this section in the interview."*

Total storage size : 100 TB as estimated earlier
Total estimated QPS : Around 100k

Q: What is the minimum number of machines required to store the data?

A: Assuming a machine has 10TB of hard disk, we would need minimum of 100TB / 10 TB = 10 machines to store the said data. Do note that this is barebe minimum. The actual number might be higher if we decide to have replication or more machines incase we need more shards to lower the QPS load on every shard.

()

- **Design Goals:**

> “
>
> **Latency** - Is this problem very latency sensitive (Or in other words, Are requests with high latency and a failing request, equally bad?). For example, search typeahead suggestions are useless if they take more than a second.
> **Consistency** - Does this problem require tight consistency? Or is it okay if things are eventually consistent?
> **Availability** - Does this problem require 100% availability?
>
> *There could be more goals depending on the problem. It's possible that all parameters might be important, and some of them might conflict. In that case, you'd need to prioritize one over the other.* ”

Q: Is Latency a very important metric for us?

A: Since we want to be available all the time, we should try to have lower latency.

()

Q: Consistency vs Availability?

A: As the question states, we need good availability and partition tolerance. Going by the CAP theorem (Nicely explained at http://rohergt reiner.com/2014/08/cap-theorem-revisited/ (http://rohergt reiner.com/2014/08/cap-theorem-revisited/)), we would need to compromise with consistency if we have availability and partition tolerance.
We can however aim at having eventual consistency. As is the case with any storage system, data loss is not acceptable.

()

Deep Dive:

> *Lets dig deeper into every component one by one. Discussion for this section will take majority of the interview time(20-30 minutes).*

> Note : In questions like these, the interviewer is looking at how you approach designing a solution. So, saying that I'll use a NoSQL DB like Cassandra is not an ideal answer. It is okay to discuss the architecture of Cassandra for example with rationale around why some components were designed the way they were..

Q: Is shar ding r equir ed?

A: Lets look at our ear lier estimate about the data to be stor ed. 100TB of data can't be stor ed on a single machine.
Lets say that we somehow have a r eally beefy machine which can stor e that amount of data, that machine would have to handle all of the quer ies (All of the load) which could lead to a significant per for mance hit.

> Tip: You could argue that there can be multiple copies of the same machine, but this would not scale in the future. As my data grows, its possible that I might not find a big beefy enough machine to fit my data.

So, the best cour se of action would be to shar d the data and distr ibute the load amongst multiple machines.

()

Q: Should the data stor ed be nor malized?
(http://www.studytonight.com/dbms/database-nor malization.php
(http://www.studytonight.com/dbms/database-nor malization.php))

A: If the data is nor malized, then we need to join acr oss tables and acr oss r ows to fetch data. If the data is alr eady shar ded acr oss machines, any join acr oss machines is highly undesir able (High latency, Less indexing suppor t).
With stor ing denor malized infor mation however , we would be stor ing the same fields at mor e than one place. However , all infor mation r elated to a r ow (or a key) would be on the same machine. This would lead to lower latency.

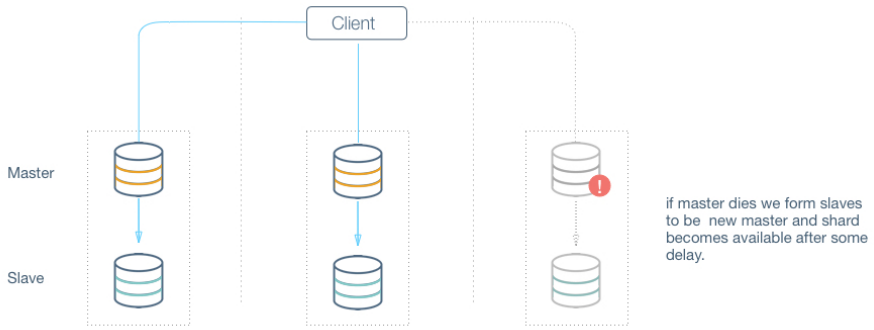
However , if the shar ding cr iter ia is not chosen pr oper ly, it could lead to consistency concer ns (After all, we ar e stor ing the same data at multiple places). However , for this case, we ar e mor e concer ned with availability and r eady to compr omise on consistency as long as things become eventually consistent.
In this case, it seems like having denor malized r ows makes shar ding easier for us and suits our use case better .

()

Q: How many machines per shar d ? How does a r ead / wr ite look in ever y shar d?

A: Going back to our design goals, low latency and high availability ar e our design goals.
Lets assume we have somehow shar ded the r ows into shar ds. Hence, lets fir st look at how the ar chitectur e might look at within a shar d.

Master Slave



One simple solution might be to have a master node in each shar d which has a slave node which r eads all new updates fr om master and keeps updating itself (The slave in this case might not have the same view as master and would lag a little bit). Clients can r ead fr om either the master or the slave depending on which r esponds ear lier (or being slightly mor e intelligent with the r eads to give mor e pr efer ence to the master , and fallback to slave after the r ead call to master). That could lead to inconsistent views on newer entr ies acr oss master and client, but would ensur e high r ead availability.
However , what happens to wr ites when the master goes down. The wr ites star t failing since only master was taking up the wr ites.

We can ar gue that we can have the slave become the new master in such a case. However , even that implies unavailability for the per iod of failover fr om master to the slave as new master .
Also, if the slave is lagging a bit, and then the master has a har dwar e failur e, we r un the r isk of losing data.

This means that we definitely need mor e than one machine taking the wr ite tr affic if we ar e to be available all the time.

Multi Master

Lets say we modify the pr evious design wher e both machines accept wr ite AND r ead tr affic. Lets name the machine m1 and m2.
If m1 accepts wr ite without depending on m2, then it is possible m1 is doing wr ite on a r ow state which is not the same as m2. That could lead to huge consistency concer ns and the DB might become for ever inconsistent. DBs might get oper ations out of or der and it can cause eventual inconsistency if the or der of oper ation matter s (double the column value and then incr ement it vs incr ement the column value and then double it).

Fr om above examples we see that any system with a*master* node will not be highly available, ther efor e we move to peer to peer systems.

()

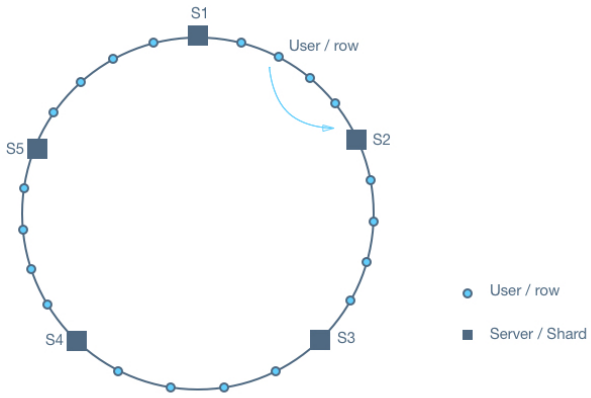
Q: Can a peer to peer system be highly available in case of a DB machine dying?

Hint: Single point of failur e!

A: We defne a peer to peer system wher e ever y node is eq ually pr ivileged and any two nodes can communicate. Yes, since we don't have a single point of failur e anymor e, ther efor e our system can theor etically be available even in pr esence of dying DB machines. Dynamo and Cassandr a ar e examples of examples of such systems, both of them lack the *master* node and ther efor e have no single point of failur e. Our highly available datastor e will be highly based on Dynamo and Cassandr a, as a r eader you don't need to know about them.

Q: How will we exactly shar d data for a peer to peer system?

A: Refer to https://www.inter viewbit.com/pr oblems/shar ding-a-database/ (https://www.inter viewbit.com/pr oblems/shar ding-a-database/) for a detailed answer .



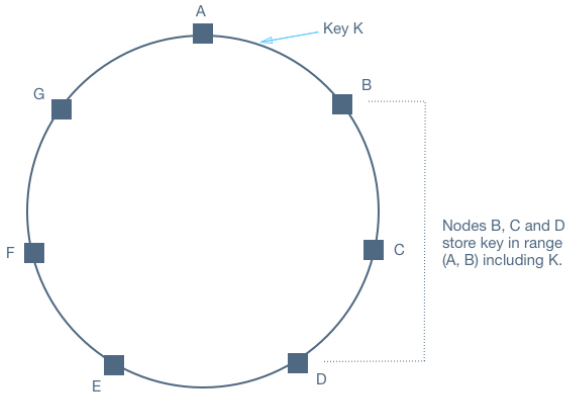
()

Q: How do we stor e r edundant data?

A: We will need to stor e duplicate data to pr event data loss in case of some of our DB machines getting cor r upted. To stor e the data we can use consistent hashing which assigns ever y data to a par ticular node on the r ing. Let's call as our r eplication factor (we will stor *P* copies of data). Now for a data D, we have to choose *P* nodes wher e we will stor e copies of D.
Now, how do we choose these P nodes? We will choose the *P* clockwise consecutive nodes star ting fr om the node wher e D is mapped by our hashing function.
An impor tant point to dicuss her e is that even though any data might be mapped to a par ticular vir tual node, the vir tual node is not the master node for this data for either r ead or r ight. A client can r eq uest to r ead or wr ite a data fr om any node they want. This is essential in cr eating a highly available system.

Got suggestions ? We would love to hear your feedback.

Loved InterviewBit ? Write us a testimonial. (http://www.quora.com/What-is-your-review-of-InterviewBit)



()

Q: How does a wr ite/r ead happen in our system?

A:

Write request:

A client can contact any node in the r ing with a put() r eq uest to wr ite data, this node acts as a coor dinating node for this r eq uest. Coor dinating node then for war ds the r eq uest to the mapping nodes for the data(hashing) and waits for acknowledgement fr om them. When it r eceives **W**(explained later) acknowledgements, it r etur ns a wr ite-accepted message to the client.

Read request:

To per for m a get() r eq uest, client can connect to any node in the r ing which becomes the coor dinating node for this r eq uest. The coor dinating node then asks all r eplica nodes for this data and r etur ns consolidated data to the client when **R** of them r eplies back.

Read and Write consistency:

W and **R** ar e called wr ite and r ead consistency number r espectively. To r ecap, **W** is the minimum number of nodes fr om which the coor dinating node should get an ack befor e making a wr ite successful and **R** is the minimum number of nodes fr om which the coor dinating node should get back r ead values to r etur n them back to the client.

R, W together for ms q uor um of the system. For a r ead to be consistent(r etur n the latest wr ite), we need to keep **W + R > P**.

Depending on the featur e r equir ement W and R can be adjusted, for example to have ver y fast wr ites we can keep **W = 1** and **R = P**. If our system is r ead heavy we can keep **R = 1** and **W = P**. If r ead and wr ite ar e equally distr ibuted, we can keep both R and W as (P+1)/2.

()

Q: What if a machine goes down?

A: Since no node is only r esponsible for a piece of data, it's going down won't r eally affect our wr ites/r eads. As long as **W** out of P nodes ar e available for some key, it can be updated(similar ily for r ead).
Note that in case of less than **W** nodes available to wr ite for some data, we can r elax our wr ite consistency(sacr ificing data consistency for availability).

()

Q: What kind of consistency can we pr ovide?

A: If we keep **W = P**, we can pr ovide str ong consistency but we won't be available for some wr ites even if one of our DB machine dies.
Ear lier we saw in master -master configur ation that in networ k par tition cases, our master s may diver ge to pr ovide availability. In our cur r ent system, essentially all of our nodes ar e master and the point that they will diver ge should be taken for gr anted and we should build our system consider ing it.
Ther efor e we should build for the case wher e **W** is less than P, hence our wr ites will be pr opagated i.e. some machines will have an updated view of data and some will not, ther efor e they ar e not consistent. The best we can guar entee her e is eventual consistency, that is in due time, all the changes will be applied to ever y ser ver and in the end they will all have a consistent view of the data.

To achieve eventual consistency, we need to be able to r esolve differ ences between data on two ser ver s. Ther e ar e a couple of detect and r esolve data conflicts that may ar ise.
Fir st, if data(key, value) we stor e is such that value is just a single column, we can use a simple cr iter ia of LWW(last wr ite wins) to r esolve conflicts. So if two ser ver s have differ ent view of a key, in the r esolve step we can update the ser ver with the stale with the new data and ther efor e become consistent.

The other way is to stor e augmented data for each r ow indicating all the coor dinating nodes for the r ow till now. Now, to detect and under stand conflict we can compar e the augmented data. If one is a subset of the other (all the wr ites seen by one of the r ow has been seen by the other r ow) we can safely ignor e the one with smaller augmented data. Other wise, we have a conflit for our r ow and need application level logic to r esolve it. This way is usually r eq uir ed when our value if composed of mor e than one independent column.

()

You have now mastered this problem!

Discussion

Post a comment (https://discuss.interviewbit.com/session/sso?return_path=https://discus

H

henry_henry almost 2 year s ago

How exactly do peer-to-peer systems work? Doesn't there still have to be a high-l 2

Reply (https://discuss.inter viewbit.com/session/sso?r etur n_path=https://discuss.inter viewbit.com/t/how-exactly-do-peer -to-peer -systems-wor k-doesnt-ther e-still-have-to-be-a-high-l)

S

Got suggestions ? We would love to hear your feedback.

Loved InterviewBit ? Write us a testimonial. (http://www.quora.com/What-is-your-review-of-InterviewBit)