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

☐ Show Notes

**Highly Consistent Database**

Bookmark

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

**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?

**Answer:** 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.* "

❷ ◀   Total storage size : 100 TB as estimated earlier
Total estimated QPS : Around 10M

**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 bare 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.

💬 ()

> **"**
>
> **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:** No, but it would be good to have a lower latency.

💬 ()

**Q:** Consistency vs Availability?

**A:** As the question states, we need tight consistency and partitioning. Going by the CAP theorem ( Nicely explained at http://robertgreiner.com/2014/08/cap-theorem-revisited/ (http://robertgreiner.com/2014/08/cap-theorem-revisited/)), we would need to compromise with availability if we have tight consistency and partitioning. 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).* **"**

💡 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)

> 66    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 HBase is not an ideal answer. It is okay to discuss the architecture of HBase for example with rationale around why some components were designed the way they were. 99

**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.

Let's 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 q uer ies ( All of the load ) which could lead to a signifcant per for mance hit.

> 66    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. 99

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)

**Q:** Can I shar d the data so that all the data r equir ed for  answer ing my most fr equent q uer ies live on a single machine?

💬 ()

**A:** Most applications ar e built to stor e data for  a user  ( consider  messaging for  example. Ever y user  has his / her  own mailbox ). As such, if you shar d based on ever y user  as a r ow, its okay to stor e data in a denor malized fashion so that you won't have to q uer y infor mation acr oss user s. In this case, lets say we go with stor ing data in a denor malized fashion.

**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 machine, 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 ).

🗨 ()

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

**Q:** Can we keep just one copy of data?

🗨 ()

**A:** Since ther e is only one copy of the data, r eading it should be consistent. As long as ther e ar e enough shar d to ensur e a r easonable load on each shar d, latency should be acceptable as well. Reads and wr ites would wor k exactly how they wor k with a single DB just that ther e would be a r ow -> shar d -> machine IP ( Given a r ow, tell me the shar d it belongs to and then given the shar d, give me the machine I should be q uer ying / wr iting to ) r esolution layer in between.
Ther e is just one tiny pr oblem with this model. What if the machine in the shar d goes down? Our shar d will be unavailable ( which is fine as gover ned by the CAP theor em ). However , what if the machine dies and its har d disk becomes cor r upt. We suddenly r un into the r isk of losing the data which is not acceptable. Imagine losing all your messages because your shar d went down and the har d disk got cor r upted. That means we definitely need mor e than one copy of data being wr itten with us.
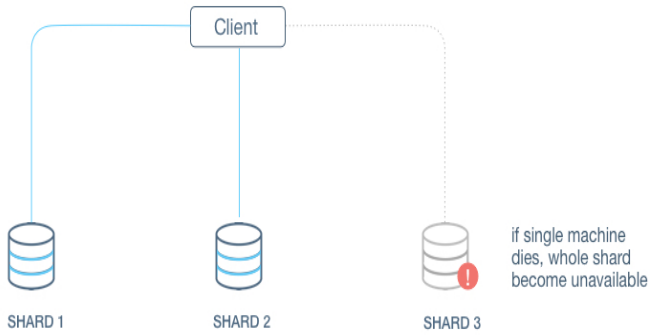
**Q:** What pr oblem may ar ise if we keep multiple copies of data?

     ()

**A:** Let's say we keep 3 copies of data ( The pr obability of all 3 machines dying and having a cor r upted disk is negligible ). Now, the issue is how do we maintain all of the copies in absolute sync ( Consistency, r emember ? ). One naive way would be that a wr ite would not succeed unless its wr itten to all 3 copies / machines. That would make our  wr ite latency go up significantly apar t fr om making wr ites ver y unr eliable ( My wr ite fails if it fails on any of the machines ). Let's see if we can make this a bit better .
If we have to allow wr ites succeeding even if the wr ite has been wr itten on a major ity of machines (2 out of 3, let's say), to maintain consistency, its impor tant that ther e is a master  machine which keeps tr ack of this infor mation. This master  machine can tr ack which machines have a par ticular  block in each shar d. This means that ever y r ead will go thr ough this master  machine, figur e out the machines with the block and q uer y fr om the r eq uir ed block. The machines which do not have the block can check with this master  machine to see which block ar e not pr esent on it, and catch up to r eplicate the block on it.

However , now if this master  machine dies, our  whole shar d is unavailable till this machine comes back up. If this machine has a cor r upted har d disk, then the unavailability becomes indefinite ( Note that we do not loose data in this case, as total data is the union of data pr esent on 3 nodes ). This is not an

**Q:** What if the master keeping tr ack of wher e the blocks ar e stor ed dies?

() 

**Anwer:** To over come this pr oblem we keep a standby master which in the failover pr ocess becomes the acting master and keeps unavilability to minimum. Now, to keep the standby master upto date we can have a shar ed networ k file system. When any namespace modification is per for med by the Active master , it dur ably logs a r ecor d of the modification to an edit log file stor ed in the shar ed dir ector y. The Standby node constantly watches this dir ector y for edits, and when edits occur , the Standby node applies them to its own namespace. In the event of a failover , the Standby will ensur e that it has r ead all of the edits fr om the shar ed stor age befor e pr omoting itself to the Active state. This ensur es that the namespace state is fully synchr onized befor e a failover occur s.
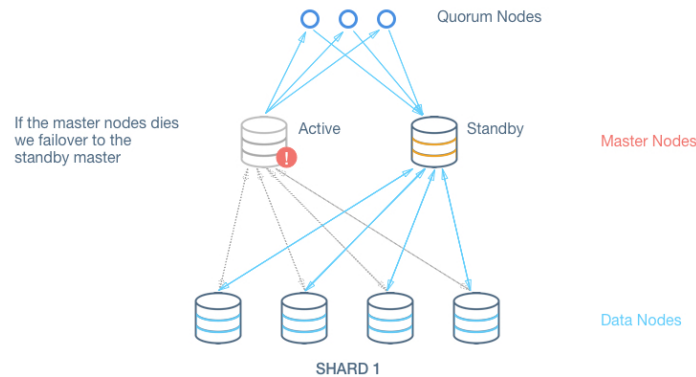


**A:** Going back to our design goals, latency and consistency ar e our design goals.
A simple way to r esolve this is to make sur e we only have one machine per shar d. Reads and wr ites would wor k exactly how they wor k with a single DB. However , if the machine holding the only copy dies and its har d disk becomes cor r upt, we suddenly r un into the r isk of losing the data which is not acceptable. That means we definitely need mor e than one copy of data being wr itten with us. Lets say that number is 3. Now, the issue is how do we maintain all of the copies in absolute sync ( Consistency, r emember ?).
One naive way would be that a wr ite would not succeed unless its wr itten to all 3 copies / machines. That would make our wr ite latency go up significantly apar t

fr om making wr ites ver y unr eliable ( My wr ite fails if it fails on any of the machines ).

If we have to allow wr ites succeeding when the wr ite has been wr itten on a major ity of machines ( 2 out of 3, lets say ), to maintain consistency, its impor tant that ther e is a master machine which keeps tr ack of this infor mation. This master machine can tr ack which machines have a par ticular block in each shar d. However , now if this master machine dies, our whole shar d is unavailable till this machine comes back up. If this machine has a cor r upted har d disk, then the unavailability becomes indefinite.

Ther e ar e couple of ways to keep unavailability to minimum using a standby master keeping tr ack of master node data thr ough a shar ed file system(Explained in detail in the last hint).

🗨 ()

✅ ◀     🏆 **You have now mastered this problem!**

# Discussion

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

**P**

piscado        🕐 over 1 year ago

**What are the cons if we use last modified time(solution given in highly available**    3

Reply (https://discuss.inter viewbit.com/session/sso?r etur n_path=https://discuss.inter viewbit.com/t/what-ar e-the-cons-if-we-use-last-modified-time-solution-given-in-highly-available)

💡    **Got suggestions ? We would love to hear your**

**K**

       **feedback.**

🗨    **Loved InterviewBit ? Write us a testimonial.**

**(http://www.quora.com/What-is-your-review-of-InterviewBit)**