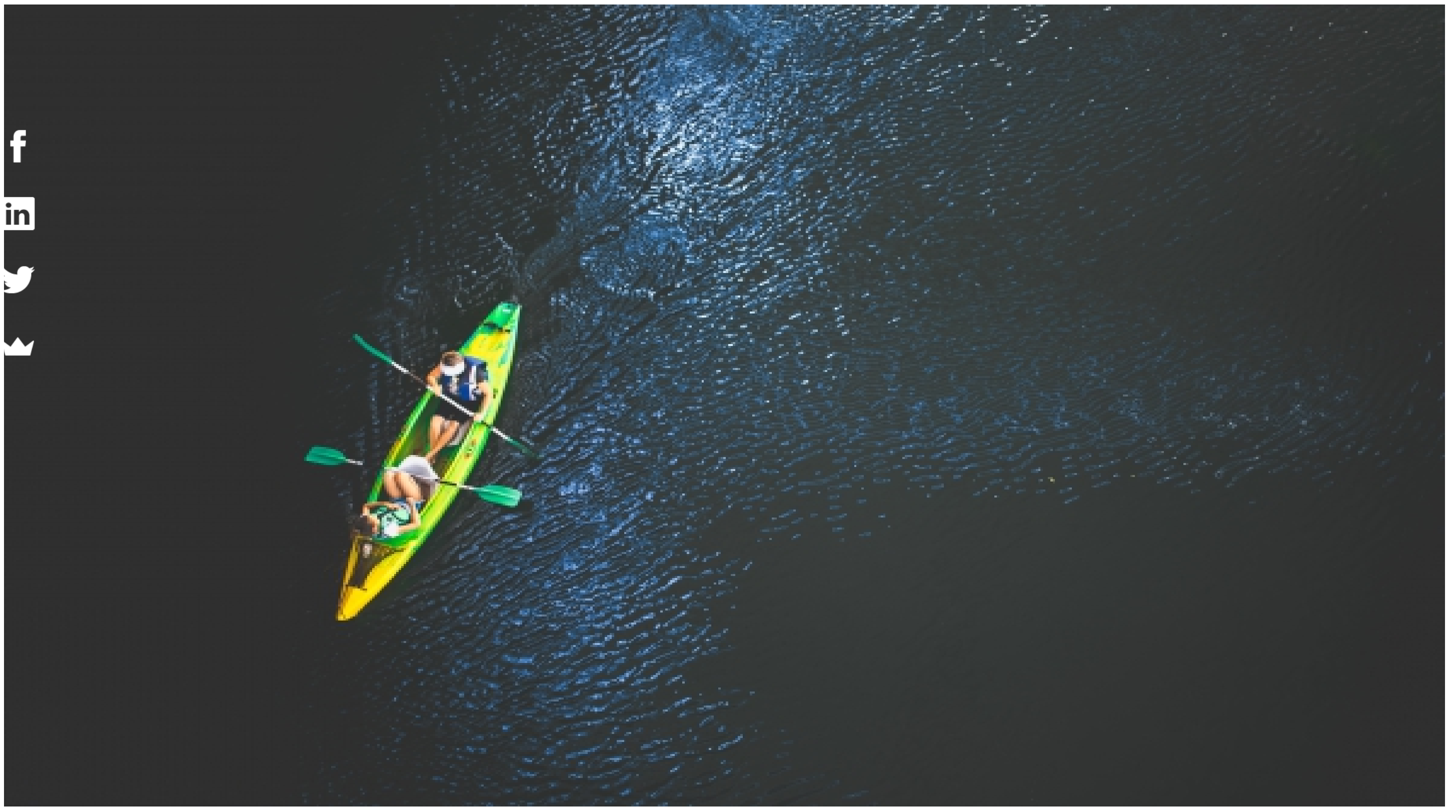

Design a Key-Value Store (Part II)

This is the second post of Design a Key-Value Store series posts. If you haven't read the first post (http://blog.gainlo.co/index.php/2016/06/14/design-a-key-value-store-part-i/), please go check it.

In our previous post, we mostly focus on the basic concepts of key-value store, especially the single machine scenario. When it comes to scaling issues, we need to distribute all the data into multiple machines by some rules and a coordinator machine can direct clients to the machine with requested resource.

There are many things you need to consider when designing the distributed system. When splitting data into multiple machines, it's important to balance the traffic. That's why it's better to make sure that keys are distributed randomly.

In this post, we will continue our discussion about distributed key-value storage system. We're going to cover topics like system availability, consistency and so on.

System availability

To evaluate a distributed system, one key metric is system availability. For instance, suppose one of our machines crashes for some reason (maybe hardware issue or program bugs), how does this affect our key-value storage system?

Apparently, if someone requests resources from this machine, we won't be able to return the correct response. You may not consider this issue when building a side project. However, if you are serving millions of users with tons of servers, this happens quite often and you can't afford to manually restart the server every time. This is why availability is essential in every distributed system nowadays. So how would you address this issue?

Of course you can write more robust code with test cases. However, your program will always have bugs. In addition, hardware issues are even harder to protect. The most common solution is replica (https://en.wikipedia.org/wiki/Replica). By setting machines with duplicate resources, we can significantly reduce the system downtime. If a single machine has 10% of chance to crash every month, then with a single backup machine, we reduce the probability to 1% when both are down.

Replica VS sharding

At first glance, replica is quite similar to sharding. So what's the relation of these two? And how would you choose between replica and sharding when designing a distributed key-value store?

First of all, we need to be clear about the purpose of these two techniques. Sharding is basically used to splitting data to multiple machines since a single machine cannot store too much data. Replica is a way to protect the system from downtime. With that in mind, if a single machine can't store all the data, replica won't help.

Consistency

By introducing replicas, we can make the system more robust. However, one issue is about consistency. Let's say for machine A1, we have replica A2. How do you make sure that A1 and A2 have the same data? For instance, when inserting a new entry, we need to update both machines. But it's possible that the write operation fails in one of them. So over time, A1 and A2 might have quite a lot inconsistent data, which is a big problem.

There are a couple of solutions here. First approach is to keep a local copy in coordinator. Whenever updating a resource, the coordinator will keep the copy of updated version. So in case the update fails, the coordinator is able to re-do the operation.

Another approach is commit log. If you have been using Git, the concept of commit log should be quite familiar to you. Basically, for each node machine, it'll keep the commit log for each operation, which is like the history of all updates. So when we want to update an entry in machine A, it will first store this request in commit log. And then a separate program will process all the commit logs in order (in a queue). Whenever an operation fails, we can easily recover as we can lookup the commit log.

The last approach I'd like to introduce is to resolve conflict in read. Suppose when the requested resource locates in A1, A2 and A3, the coordinator can ask from all three machines. If by any chance the data is different, the system can resolve the conflict on the fly.

It's worth to note that all of these approaches are not mutually exclusive. You can definitely use multiple ones based on the application.

Read throughput

I'd also like to briefly mention read throughput in this post. Usually, key-value storage system should be able to support a large amount of read requests. So what approaches will you use to improve read throughput?

To improve read throughput, the common approach is always taking advantage of memory. If the data is stored in disk inside each node machine, we can move part of them in memory. A more general idea is to use cache. Since the post – design a cache system (http://blog.gainlo.co/index.php/2016/05/17/design-a-cache-system/) has an in-depth analysis of this topic, I won't talk about it too much here.

Summary

Don't take the analysis here as something like standard answers. Instead, those common solutions should give you inspirations to help you come up with different ideas.

There's no solution that works for every system and you should always adjust your approach based on particular scenarios.

I'd like to learn more (http://www.gainlo.co/?utm_source=blog&utm_medium=footer http%3A//blog.gainlo.co/index.php/2016

(http://www.facebook.com/sharer.php?u=http://blog.gainlo.co/index.php/2016/06/21/design-key-value-store-part-ii/) 17 (http://twitter.com/share?url=http://blog.gainlo.co/index.php/2016/06/21/design-key-value-store-part-ii/&text=Design+a+Key-Value+Store+%28Part+II%29+) (http://www.linkedin.com/shareArticle?mini=true&url=http://blog.gainlo.co/index.php/2016/06/21/design-key-value-store-part-ii/) 17 (http://reddit.com/submit?url=http://blog.gainlo.co/index.php/2016/06/21/design-key-value-store-part-ii/&title=Design a Key-Value Store (Part II)) 0

Subscribe

We'll email you when there are new posts here.

Enter your email address here...

Subscribe

Related Posts:

1. **Design a Key-Value Store (Part I) (http://blog.gainlo.co/index.php/2016/06/14/design-a-key-value-store-part-i/)**
2. **Design eCommerce Website (Part II) (http://blog.gainlo.co/index.php/2016/08/28/design-ecommerce-website-part-ii/)**
3. **Design a Cache System (http://blog.gainlo.co/index.php/2016/05/17/design-a-cache-system/)**
4. **Random ID Generator (http://blog.gainlo.co/index.php/2016/06/07/random-id-generator/)**

5 thoughts on "Design a Key-Value Store (Part II)"

AKSHAY KULKARNI
June 27, 2016 at 4:25 am (http://blog.gainlo.co/index.php/2016/06/21/design-key-value-store-part-ii/#comment-1275)

This is regarding Consistency. Suppose a resource at a machine is updated ? How are the replicas of that resources updated ? What protocol do we follow ? Coordinator based approach just keep track that resource has been updated but there would be some limit on the amount of data which coordinator maintain. So how and when does actual propagation of update takes place.
In commit log approach, how does the background process know what is the updated value of the resource ? Do we maintain timestamp also ? If yes, we also know that in distributed environment timestamp of different are not exactly synced.

Reply (http://blog.gainlo.co/index.php/2016/06/21/design-key-value-store-part-ii/?replytocom=1275#respond)

KEVIN
July 29, 2016 at 1:44 am (http://blog.gainlo.co/index.php/2016/06/21/design-key-value-store-part-ii/#comment-1623)

Coordinator keeps only updates, not the entire data set. Some sort of request/response protocol can be used. That is, once the update request had been processed by the replicas, they can acknowledge with a response and the coordinator can go on with the rest of the updates.

Commit log is implemented like a queue. So all the updated values are already dequeued. You wouldn't be able to see them in the commit log right?

Reply (http://blog.gainlo.co/index.php/2016/06/21/design-key-value-store-part-ii/?replytocom=1623#respond)

KEVIN
July 29, 2016 at 1:44 am (http://blog.gainlo.co/index.php/2016/06/21/design-key-value-store-part-ii/#comment-1624)

"If a single machine has 10% of chance to crash every month, then with a single backup machine, we reduce the probability to 1% when both are down."

Not sure how 1% came about.

Reply (http://blog.gainlo.co/index.php/2016/06/21/design-key-value-store-part-ii/?replytocom=1624#respond)

JAKE
August 1, 2016 at 9:48 pm (http://blog.gainlo.co/index.php/2016/06/21/design-key-value-store-part-ii/#comment-1699)

The probability of one machine to crash is 10% and two machines to crash simultaneously should be 10% * 10% = 1%.

Reply (http://blog.gainlo.co/index.php/2016/06/21/design-key-value-store-part-ii/?replytocom=1699#respond)

FRANKIE (HTTP://EIBFGYFMMUQ.COM)
October 29, 2016 at 3:43 pm (http://blog.gainlo.co/index.php/2016/06/21/design-key-value-store-part-ii/#comment-3166)

Holy shtzini, this is so cool thank you.

Reply (http://blog.gainlo.co/index.php/2016/06/21/design-key-value-store-part-ii/?replytocom=3166#respond)

LEAVE A REPLY

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment

Design a Key-Value Store (Part I) (http://blog.gainlo.co/index.php/2016/06/14/design-a-key-value-store-part-i/)

Build a Web Crawler (http://blog.gainlo.co/index.php/2016/06/29/build-web-crawler/)

CATEGORIES

Coding Interview Questions (http://blog.gainlo.co/index.php/category/coding-interview-questions/)

Common Pitfalls (http://blog.gainlo.co/index.php/category/common-pitfalls/)

Common Questions (http://blog.gainlo.co/index.php/category/common-questions/)

Facebook Interview Questions (http://blog.gainlo.co/index.php/category/facebook-interview-questions/)

Mock Interview Thoughts (http://blog.gainlo.co/index.php/category/mock-interview-thoughts/)

Others (http://blog.gainlo.co/index.php/category/others/)

System Design Interview Questions (http://blog.gainlo.co/index.php/category/system-design-interview-questions/)

The Complete Guide to Google Interview Preparation (http://blog.gainlo.co/index.php/category/google-interview-preparation/)

Uber Interview Questions (http://blog.gainlo.co/index.php/category/uber-interview-questions/)

Gainlo - Mock Interview With Professionals

Visit Gainlo (http://www.gainlo.co/?utm_source=blog&utm_medium=site-footer http%3A//blog.gainlo.co/index.php