Explaining Apache ZooKeeper

Asked 14 years, 3 months ago Modified 3 months ago

Viewed 131k times

418

I am trying to understand ZooKeeper, how it works and what it does. Is there any application which is comparable to ZooKeeper?

If you know, then how would you describe ZooKeeper to a layman?

I have tried apache wiki, zookeeper sourceforge...but I am still not able to relate to it.

I just read thru http://zookeeper.sourceforge.net/index.sf.shtml, so aren't there more services like this? Is it as simple as just replicating a server service?

apache-zookeeper distributed-computing

Share

Improve this question

Follow

edited Jul 18, 2018 at 20:47

codeforester

42.8k ● 19 ● 118 ● 153

asked Sep 7, 2010 at 21:43

topgun_ivard

8,534 ● 10 ● 39 ● 46

7 Similar to but not the exact answer you are looking for: stackoverflow.com/questions/1479442/real-world-use-of-zookeeper – zengr Sep 7, 2010 at 22:29

3 cloudera.com/blog/2009/05/… – zengr Oct 11, 2010 at 1:26

You can read this paper ZooKeeper: Wait-free coordination for Internet-scale systems Written by two Yahoo! engineers – yaphet Sep 7, 2013 at 16:51

Here is a tech talk that is an introduction to Apache ZooKeeper by Camille Fournier who is the CTO of RentTheRunway. I hope it is helpful. – Genadinik Nov 20, 2013 at 19:31

@Luca Geretti...Acoording to me, Zookeper provides set of apis so that we can make use of it to coordinate the distributed application. correct me if I am wrong. – user3797438 Aug 12, 2014 at 7:22

8 Answers

Sorted by: Highest score (default)

468

In a nutshell, ZooKeeper helps you build distributed applications.

How it works

You may describe ZooKeeper as a replicated synchronization service with eventual consistency. It is robust, since the persisted data is distributed between multiple nodes (this set of nodes is called an "ensemble") and one client connects to any of them (i.e., a specific "server"), migrating if one node fails; as long as a strict

majority of nodes are working, the ensemble of ZooKeeper nodes is alive. In particular, a master node is dynamically chosen by consensus within the ensemble; if the master node fails, the role of master migrates to another node.

How writes are handled

The master is the authority for writes: in this way writes can be guaranteed to be persisted in-order, i.e., writes are **linear**. Each time a client writes to the ensemble, a majority of nodes persist the information: these nodes include the server for the client, and obviously the master. This means that each write makes the server up-to-date with the master. It also means, however, that you cannot have concurrent writes.

The guarantee of linear writes is the reason for the fact that ZooKeeper does not perform well for write-dominant workloads. In particular, it should not be used for interchange of large data, such as media. As long as your communication involves shared data, ZooKeeper helps you. When data could be written concurrently, ZooKeeper actually gets in the way, because it imposes a strict ordering of operations even if not strictly necessary from the perspective of the writers. Its ideal use is for coordination, where messages are exchanged between the clients.

How reads are handled

This is where ZooKeeper excels: reads are concurrent since they are served by the specific server that the client connects to. However, this is also the reason for the eventual consistency: the "view" of a client may be outdated, since the master updates the corresponding server with a bounded but undefined delay.

In detail

The replicated database of ZooKeeper comprises a tree of *znodes*, which are entities roughly representing file system nodes (think of them as directories). Each znode may be enriched by a byte array, which stores data. Also, each znode may have other znodes under it, practically forming an internal directory system.

Sequential znodes

Interestingly, the name of a znode can be *sequential*, meaning that the name the client provides when creating the znode is only a prefix: the full name is also given by a sequential number chosen by the ensemble. This is useful, for example, for synchronization purposes: if multiple clients want to get a lock on a resource, they can each concurrently create a sequential znode on a location: whoever gets the lowest number is entitled to the lock.

Ephemeral znodes

Also, a znode may be *ephemeral*: this means that it is destroyed as soon as the client that created it disconnects. This is mainly useful in order to know when a client fails, which may be relevant when the client itself has responsibilities that should be taken by a new client. Taking the example of the lock, as soon as the client having the lock disconnects, the other clients can check whether they are entitled to the lock.

Watches

The example related to client disconnection may be problematic if we needed to periodically poll the state of znodes. Fortunately, ZooKeeper offers an event system where a *watch* can be set on a znode. These watches may be set to trigger an event if the znode is specifically changed or removed or new children are created under it. This is clearly useful in combination with the sequential and ephemeral options for znodes.

Where and how to use it

A canonical example of Zookeeper usage is distributed-memory computation, where some data is shared between client nodes and must be accessed/updated in a very careful way to account for synchronization.

ZooKeeper offers the library to construct your synchronization primitives, while the ability to run a distributed server avoids the single-point-of-failure issue

you have when using a centralized (broker-like) message repository.

ZooKeeper is feature-light, meaning that mechanisms such as leader election, locks, barriers, etc. are not already present, but can be written above the ZooKeeper primitives. If the C/Java API is too unwieldy for your purposes, you should rely on libraries built on ZooKeeper such as cages and especially curator.

Where to read more

Official documentation apart, which is pretty good, I suggest to read Hadoop: The Definitive Guide which has a chapter explaining essentially what ZooKeeper does, followed by an example of a configuration service.

Share Improve this answer Follow

edited Aug 28 at 15:25

answered Jan 14, 2012 at 18:19

Luca Geretti
10.3k ● 7 ● 50 ● 60

2 I'm not sure I understand the communication scheme you are suggesting, but you can use ZooKeeper to "publish" information from a producer and have several consumers read it. If on the other hand there exists only one instance of each kind of server, there is little benefit in using ZK.
– Luca Geretti Jun 27, 2014 at 9:59

85 IMO this fails to explain what ZooKeeper is to a layperson. When would I need ZooKeeper? What would I write to it? What problem does it solve? Is it a key-value store? A search engine? A distributed lock? Why would I pick ZooKeeper over e.g. Redis or a file or JIRA or post-it notes? You clearly know a lot about ZooKeeper - but can you explain it less technically? – Dan Passaro Jan 10, 2017 at 21:14

1 As Zookeeper has linear writes, that does not stop me to use Asynchronous APIs to create nodes and take the response in a callback ? Though internally it may not allow concurrent writes , or am I missing something ? – jdk2588 Jun 15, 2017 at 14:26

1 "Each time a client writes to the ensemble, a majority of nodes persist the information: these nodes include the server for the client, and obviously the master" => could you please point me to a doc. or something where this is explained? I'm wondering whether it is possible that a state change was successfully made excluding the server to which the client is connected (in which case, the client can experience the strange behavior of not being able to read its own write for a moment) – senseiwu Feb 22, 2018 at 9:21

5 Completely and utterly antithetical to the question asked. If it was a clock, he would be looking for "time keeping device" not a description of the mainspring, wheel train, escapement and their interaction based on the period of oscillation, moment of inertia and the impact of artificial sapphire crystals. – Rick O'Shea Oct 4, 2019 at 4:51

13

What problem does it solve?

Let's imagine we have a million files in a file store and the file count keeps increasing every minute of the day. Our task is to first process and then delete these files. One of

the approach we can think of is to write a script that does this task and run multiple instances parallelly on multiple servers. We can even increase or decrease the server count based on the demand. This is basically a distributed compute/data processing application.

Here, how can we ensure that the same file is not picked and processed by multiple servers at the same time? To solve this problem, all the servers should share the information b/w them regarding which file is currently being processed.

This is where we can use something like ZooKeeper. When the first server wants to read a file, it can write to the zookeeper the file name its going to process. Now the rest of the servers can look up ZooKeeper and know that this file is already picked up by the first server.

Above is a crude example and needs few other guard rails in place but I hope it gives an idea on what zookeeper is. ZK is basically a data store which can be accessed using the ZK API's. But it **should NOT** be used as a database. Only a small amount of data should be stored(usually in KB's). The upper limit is 1MB per znode. ZK is specifically built so that the distributed applications can communicate among each other.

Applications of ZK

Out of the box can be used for

- storing configuration: to store configuration that is accessed across your distributed application.
- naming service: store information such as service name and IP address mapping in a central place, which enables users and applications to communicate across the network.
- group membership: all the applications running on distributed servers can connect to ZK and send heartbeats. If any one server/application goes down then ZK can alert other servers/applications regarding this event.

Other features have to be built on top of the ZooKeeper API.

- locks and queues - useful for distributed synchronization.
- two phase commits - useful when we have to commit/rollback across servers.
- leader election - your distributed applications can use ZK to hold leader elections for automatic failovers.
- shared counter

Below is the page that explains how these features can be implemented
https://zookeeper.apache.org/doc/current/recipes.html

ZooKeeper can have many more applications. The features have to be built on top of ZK API's based on the requirements of your distributed system.

NOTE: ZK should not be used to store large amounts of data. Its not a cache/database. **Use it to exchange small piece of information that your distributed applications need to start, operate and failover.**

How data is stored?

Data is stored in a hierarchical tree data structure. Each node in the tree is called **znode**. Max size of a znode is 1MB. **znodes can have data and other children znodes.** Think of a znode like a folder on your computer where the folder can have files with data but also the folder itself can have data just like a file.

Why use ZK instead of our own custom service?

- Atomicity and Durability
- Zookeeper itself is distributed and Fault tolerant. The architecture involves one leader node and multiple follower nodes. In case a ZK follower node goes down, it will automatically failover. The client sessions are replicated hence ZK can automatically move clients to a different node. If the Leader node goes down then a new leader is elected using the ZK consensus algorithm.
- Reads are very fast since its served from in-memory store.
- Writes are written in the sequence in which it arrived. Hence maintains ordering.

- Watches will send out notification to the client who set the watch on some data. This reduces the need to poll ZK. Note that watches are one time triggers and if you get a watch event and you want to get notified of future changes, you must set another watch.
- Persistent and ephemeral znodes are available. Both are stored on ZK disks. Persistent here means that the data will be persisted once the client who created it disconnects. Ephemeral means the data will be removed automatically when the client disconnects. Ephemeral znodes are not allowed to have children.
- There is also persistent sequential and ephemeral sequential znodes. Here the names of the znodes can have a suffix sequence number. similar to DB auto increment ID's, these sequence number keeps increasing and managed by ZK. This can be useful to implement queues, locks etc.

Is there any application which is comparable to Zookeeper?

etcd - https://etcd.io/docs/v3.3/learning/why/#zookeeper

edited Dec 27, 2021 at 13:54

answered Dec 26, 2021 at 13:02

ns15
8,584 ● 1 ● 66 ● 66

12

Zookeeper is one of the best open source server and service that helps to reliably coordinates distributed processes. Zookeeper is a CP system (Refer CAP Theorem) that provides Consistency and Partition tolerance. Replication of Zookeeper state across all the nodes makes it an eventually consistent distributed service.

Moreover, any newly elected leader will update its followers with missing proposals or with a snapshot of the state, if the followers have many proposals missing.

Zookeeper also provides an API that is very easy to use. This blog post, Zookeeper Java API examples, has some examples if you are looking for examples.

So where do we use this? If your distributed service needs a centralized, reliable and consistent configuration management, locks, queues etc, you will find Zookeeper a reliable choice.

Share Improve this answer

Follow

edited Jul 27, 2018 at 20:49

codeforester

42.8k ● 19 ● 118 ● 153

answered Jan 30, 2016 at 21:55

Binu George

1,070 ● 10 ● 14

6 "Zookeeper is a CP system (Refer CAP Theorem) that provides Consistency and Partition tolerance", I think that

Zookeeper have master and followers, when the master down, then one of the follower would be elected as the Leader, so Zookeeper should provide the AP, however the C is eventually consistently. – YuFeng Shen Dec 18, 2017 at 6:04

6 In terms of CAP theorem, "C" actually means linearizability. ZooKeeper in fact provides "sequential consistency" and it means updates from clients will be applied in the order that they were recieved.. This is weaker than linearizability but is still very strong, much stronger than "eventual consistency". Zookeeper is not A and this is because If leader cannot be elected (no quorum) then zookeeper will fail requests. This is why it is not highly available. – Binu George Jan 3, 2019 at 19:50

Do you think that Apache Zookeeper can be used for executing the consensus as an external system as it is explained in the following question? stackoverflow.com/q/70088996/5029509 – Questioner Nov 30, 2021 at 15:24

9

I understand the ZooKeeper in general but had problems with the terms "quorum" and "split brain" so maybe I can share my findings with you (I consider myself also a layman).

Let's say we have a ZooKeeper cluster of 5 servers. One of the servers will become the leader and the others will become followers.

- These 5 servers form a quorum. Quorum simply means "these servers can vote upon who should be the leader".

- So the voting is based on majority. Majority simply means "more than half" so more than half of the number of servers must agree for a specific server to become the leader.
- So there is this bad thing that may happen called "split brain". A split brain is simply this, as far as I understand: The cluster of 5 servers splits into two parts, or let's call it "server teams", with maybe one part of 2 and the other of 3 servers. This is really a bad situation as if both "server teams" must execute a specific order how would you decide wich team should be preferred? They might have received different information from the clients. So it is really important to know what "server team" is still relevant and which one can/should be ignored.
- Majority is also the reason you should use an odd number of servers. If you have 4 servers and a split brain where 2 servers seperate then both "server teams" could say "hey, we want to decide who is the leader!" but how should you decide which 2 servers you should choose? With 5 servers it's simple: The server team with 3 servers has the majority and is allowed to select the new leader.
- Even if you just have 3 servers and one of them fails the other 2 still form the majority and can agree that one of them will become the new leader.

I realize once you think about it some time and understand the terms it's not so complicated anymore. I

hope this also helps anyone in understanding these terms.

Share Improve this answer

Follow

answered Aug 11, 2017 at 9:01

Invest

635 ● 8 ● 9

Do you think that Apache Zookeeper can be used for executing the consensus as an external system as it is explained in the following question? stackoverflow.com/q/70088996/5029509 – Questioner Nov 30, 2021 at 15:25

3

Zookeeper is a centralized open-source server for maintaining and managing configuration information, naming conventions and synchronization for distributed cluster environment. Zookeeper helps the distributed systems to reduce their management complexity by providing low latency and high availability. Zookeeper was initially a sub-project for Hadoop but now it's a top level independent project of Apache Software Foundation.

More Information

Share Improve this answer

Follow

answered Aug 20, 2015 at 11:34

neel4soft

537 ● 1 ● 4 ● 12

4 What makes you say that zookeeper is centralized? Zookeeper can and should be run distributed. – Benjamin Hammer Nørgaard Feb 19, 2017 at 19:31

Do you think that Apache Zookeeper can be used for executing the consensus as an external system as it is explained in the following question? stackoverflow.com/q/70088996/5029509 – Questioner Nov 30, 2021 at 15:25

2

I would suggest the following resources:

1. The paper: https://pdos.csail.mit.edu/6.824/papers/zookeeper.pdf
2. The lecture offered by MIT 6.824 from 36:00: https://youtu.be/pbmyrNjzdDk?t=2198

I would suggest watching the video, read the paper, and then watch the video again. It would be easier to understand if you know Raft beforehand.

Share Improve this answer Follow

answered Mar 25, 2020 at 16:49

yyFred
903 ● 12 ● 14

2

My approach to understand zookeeper was, to play around with the CLI client. as described in Getting Started Guide and Command line interface

From this I learned that zookeeper's surface looks very similar to a filesystem and clients can create and delete objects and read or write data.

Example CLI commands

```
create /myfirstnode mydata
ls /
get /myfirstnode
delete /myfirstnode
```

Try yourself

How to spin up a zookeper environment within minutes on docker for windows, linux or mac:

One time set up:

```
docker network create dn
```

Run server in a terminal window:

```
docker run --network dn --name zook -d zookeeper
docker logs -f zookeeper
```

Run client in a second terminal window:

```
docker run -it --rm --network dn zookeeper
zkCli.sh -server zook
```

See also [documentation of image on dockerhub]

Share Improve this answer

Follow

answered Dec 27, 2020 at 21:00

Michael S

880 ● 1 ● 11 ● 21

Do you think that Apache Zookeeper can be used for executing the consensus as an external system as it is explained in the following question? stackoverflow.com/q/70088996/5029509 – Questioner Nov 30, 2021 at 15:25

0

Apache ZooKeeper is an open source technology for coordinating and managing configuration in distributed applications. It simplifies tasks like maintaining configuration details, enabling distributed synchronization, and managing naming registries.

It's aptly named - think about how a zookeeper goes around and takes care of all the animals, maintains their pens, feeds them, etc.

Apache ZooKeeper can be used with Apache projects like Apache Pinot or Apache Flink. Apache Kafka also uses ZooKeeper for managing brokers, topics, and partition info. Since Apache ZooKeeper open source, you can also pair it with any technology/project of your choosing, not just Apache Foundation projects.

Share Improve this answer Follow

answered Oct 15, 2021 at 17:07

Greg Brown
76 ● 1 ● 2

Do you think that Apache Zookeeper can be used for executing the consensus as an external system as it is explained in the following question?

stackoverflow.com/q/70088996/5029509 – Questioner Nov 30, 2021 at 15:25

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.