qnimate.com

Overview Of Redis Architecture

Narayan Prusty

8-10 minutes

Redis is a in-memory, key-value data store. Redis is the most popular key-value data store. Redis is used by all big IT brands in this world. Amazon Elastic Cache supports Redis which makes redis a very powerful and must know key-value data store. In this post I will provide you a brief introduction to redis architecture.

What Is In-Memory, Key-Value Store

Key-Value store is a storage system where data is stored in form of key and value pairs. When we say in-memory key-value store, by that we mean that the key-value pairs are stored in primary memory(RAM). So we can say that Redis stored data in RAM in form of key-value pairs.

In Redis, key has to be a string but value can be a string, list, set, sorted set or hash.

These are some example of Redis key-values pairs

name="narayan"
profession=["web", "mobile"]

Here name and profession are keys. And we have their respective values on right.

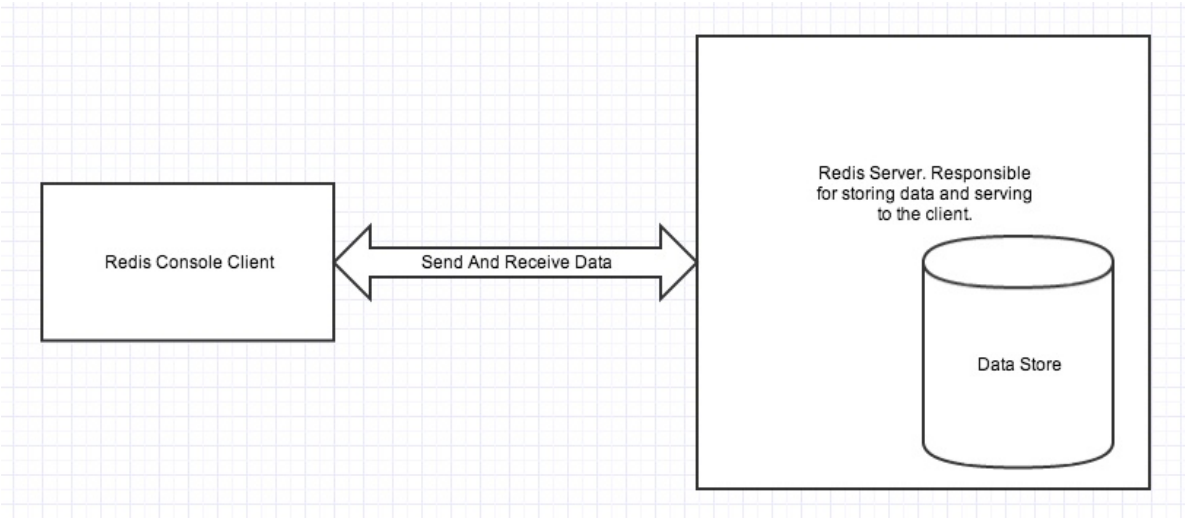
Advantage And Disadvantage of Redis over DBMS

Database Management systems store everything in second storage which makes read and write operations very slow. But Redis stores everything in primary memory which is very fast in read and write of data.

Primary memory is limited(much lesser size and expensive than secondary) therefore Redis cannot store large files or binary data. It can only store those small textual information which needs to be accessed, modified and inserted at a very fast rate. If we try to write more data than available memory then we will receive errors.

Redis Single Instance Architecture

Redis architecture contains two main processes: Redis client and Redis Server.



Redis client and server can be in the same computer or in two different computers.

Redis server is responsible for storing data in memory. It handles all kinds of management and forms the major part of architecture. Redis client can be Redis console client or any other programming

language's Redis API.

As we saw that Redis stores everything in primary memory. Primary memory is volatile and therefore we will loose all stored data once we restart our Redis server or computer. Therefore we need a way for datastore persistance.

Redis Persistance

There are three different ways to make Redis persistance: RDB, AOF and SAVE command.

RDB Mechanism

RDB makes a copy of all the data in memory and stores them in secondary storage(permanent storage). This happens in a specified interval. So there is chance that you will loose data that are set after RDB's last snapshot.

AOF

AOF logs all the write operations received by the server. Therefore everything is persistance. The problem with using AOF is that it writes to disk for every operation and it is a expensive task and also size of AOF file is large than RDB file.

SAVE Command

You can force redis server to create a RDB snapshot anytime using the redis console client SAVE command.

You can use AOF and RDB together to get best persistance result.

For more information of Redis persistance refer this official documentation.

Backup And Recovery Of Redis DataStore

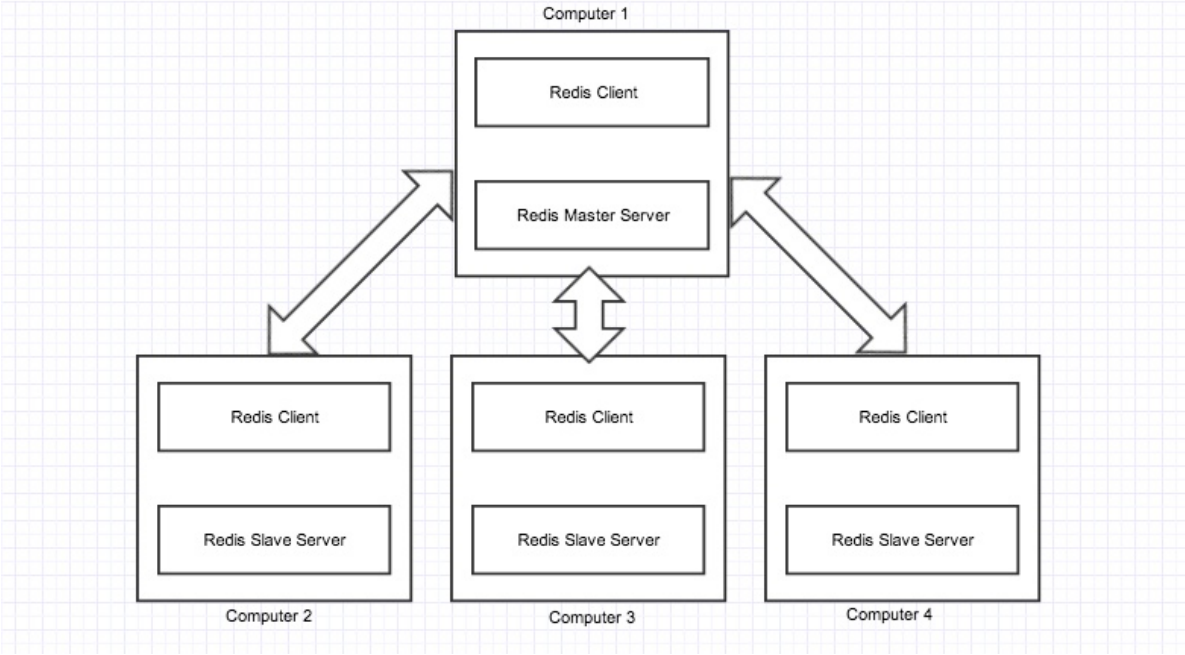
Redis does not provide any mechanism for datastore backup and recovery. Therefore if there is any hard disk crash or any other kind of disaster then all data will be lost. You need to use some third party server backup and recovery softwares to workaround it.

If you are using Redis in a replicated environment then there is no need for backup.

Redis Replication

Replication is a technique involving many computers to enable fault-tolerance and data accessibility. In a replication environment many computers share the same data with each other so that even if few computers go down, all the data will be available.

This image shows a basic Redis replication



Master and slaves are redis servers configured as such.

All the slaves contain exactly same data as master. There can be as many as slaves per master server. When a new slave is inserted to the environment, the master automatically syncs all

data to the slave.

All the queries are redirected to master server, master server then executes the operations. When a write operation occurs, master replicates the newly written data to all slaves. When a large number sort or read operation are made, master distributes them to the slaves so that a large number of read and sort operations can be executed at a time.

If a slave fails, then also the environment continues working. when the slave again starts working, the master sends updated data to the slave.

If there is a crash in master server and it looses all data then you should convert a slave to master instead of bringing a new computer as a master. If we make a new computer as master then all data in the environemt will be lost because new master will have no data and will makes the slaves also to have zero data(new master does resync). If master fails but data is persistent(disk not crashed) then starting up the same master server again will bring up the whole environment to running mode.

Replication helped us from disk failures and other kinds of hardware failures. It also helped to execute multiple read/sort queries at a time.

For more detailed information on redis replication read this official documentation.

Persistance In Redis Replication

We saw how persistance can tackle unexpected failures and keep our backend strong. But one way whole data can be lost is if the whole replicated environment goes down due to power failure.

This happens because all data is stored in primary memory. So we need persistance here also.

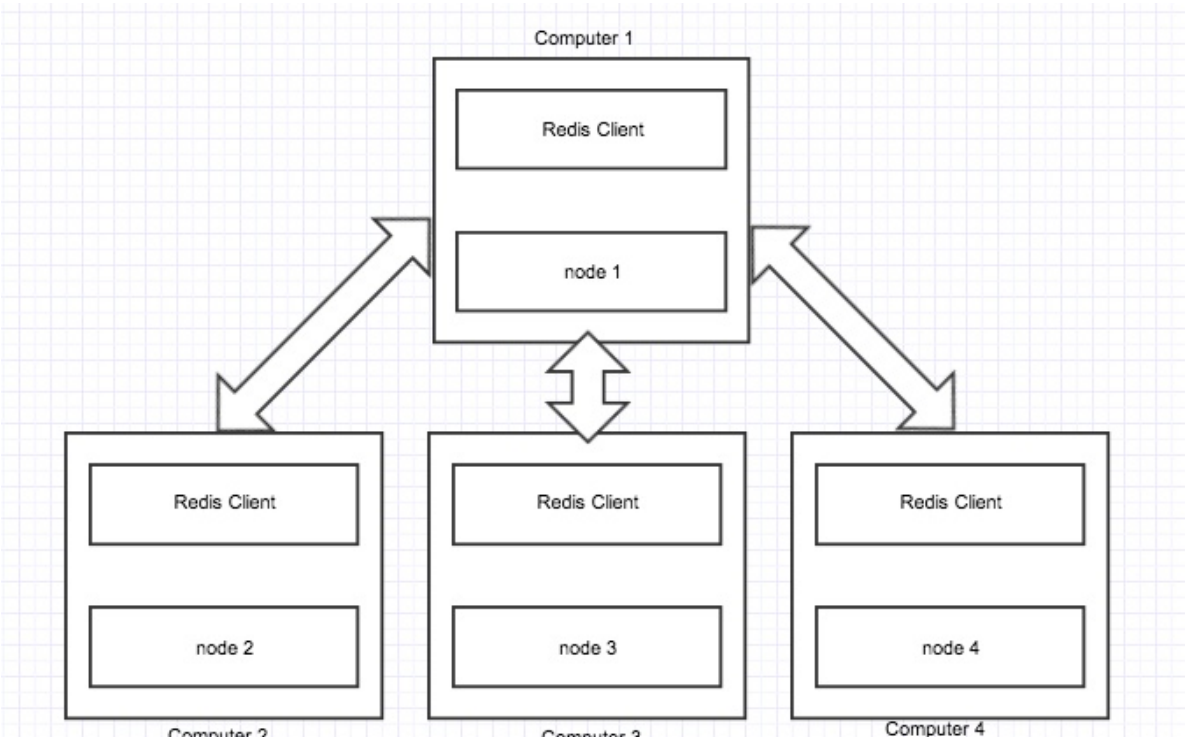
We can configure master or any one slave to store data in secondary storage using any method(AOF and RDB). Now when the whole environment is again started, make the persistent server as the master server.

Using persistance and replication together all our data is completely safe and protected from unexpected failures.

Clustering In Redis

Clustering is a technique by which data can be sharded(divided) into many computers. The main advantage is that more data can be stored in a cluster because its a combination of computers.

Suppose we have one redis server with 64GB of memory i.e., we can have only 64GB of data. Now if we have 10 clustered computers with each 64GB of RAM then we can store 640GB of data.



In the above image we can see that data is sharded into four nodes. Each node is a redis server configured as a cluster node.

If one node fails then the whole cluster stops working.

More detailed information on redis cluster read this official documentation.

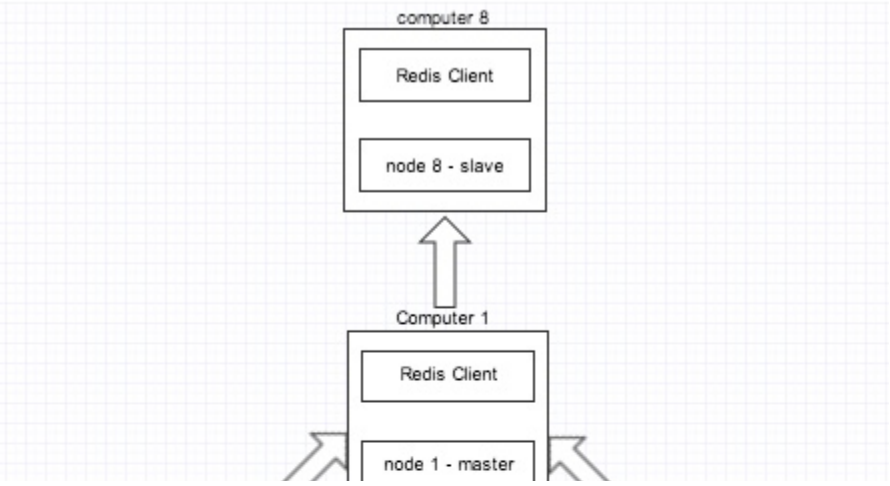
Persistance In Cluster

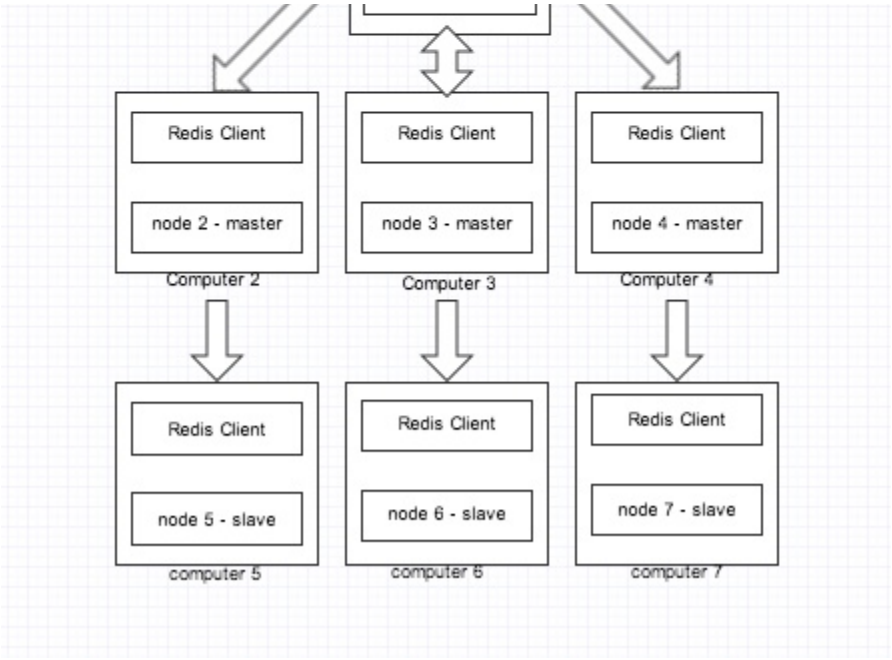
Data is stored in primary memory of nodes. We need to make the data of each node persistance. We can do that using the previously mentioned methods(AOF and RDF). Just configure every node for persistance storage.

Clustering And Replication Together

Suppose due to disk crash, one of our node goes down then the whole cluster stops working and never resumes. There is no way we can recover back the node as the data is completely lost.

To avoid this situation we can take a manual backup of each node regularly. But thats a tough and improper task. Therefore we can rely on replication to solve this problem.





Here we convert each node server to a master server. And we keep a slave for every master. So if any node(master) fails, the cluster will start using the slave to keep the cluster operating.

Redis Client

If you are first time getting yourself into redis then these links will be very helpful to install redis and learn redis client.

1. Try Redis: This is a awesome online Redis console client which will help you to learn how to use Redis console client.
2. Redis Quick Start: This article will help you to install redis and get started with it.
3. FAQ's: You can see the frequently asked questions about redis on this link.

Conclusion

Before writing this article I was trying to find articles on redis architecture. But I didn't find any. Therefore I wrote a article about it. I tried to make it very simple so that anyone who is completely

new to redis and DBMS can understand it. I hope I made everything clear. Leave comments for questions on it. Thanks for reading.

May 27, 2014