BEST PRACTICES IN CHOOSING STORAGE TYPE FOR CASSANDRA DATABASE

A. KOZLOV
Software Engineer,
Altoros Development

Altoros Development, Республика Беларусь
E-mail: akozlov.dev@gmail.com

What is Apache Cassandra?

Apache Cassandra is a distributed NoSQL database management system that was developed to work with huge amounts of data. The main advantages of this database are good horizontal scalability, fast inserts, and high availability. With its networking topology, Cassandra is a decentralized system. It means that there is no master node in a Cassandra cluster and all nodes are "the same." This allows for making reads and writes from/to each node, so Cassandra has almost linear scalability.

Data replication and consistency in Cassandra

Apache Cassandra has a masterless architecture. Each node in a cluster can handle read or write requests. Such a node is called a coordinator; it is responsible for data consistency during these operations. The coordinator sends write requests to all the

replicas where data should be stored during the *write* operation. When the coordinator receives information that the operation is completed on a certain number of nodes, it will return the message about the success of the operation to the client. The number of nodes that should answer the coordinator about a valid operation is defined by the consistency level.

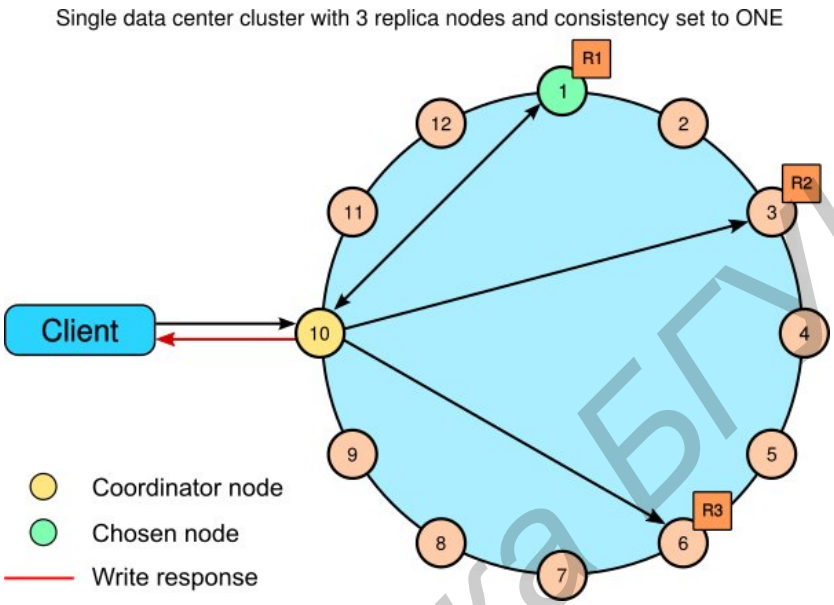

Fig. 1. Write operation in C* cluster with replication factor 3

The coordinator node will send read requests to all the replicas defined by the consistency level during the *read* operation. The client will receive the most up-to-date information. If any of the nodes contains data with an older version, this data will be rewritten to the latest state. This way, Cassandra is responsible for data consistency in a distributed file system. We can change the balance between consistency and operation performance by increasing or reducing the consistency level.

Cassandra storage types

This article shows how to use Cassandra with different storage types. First, we could divide cassandra storage configurations on 2 groups: configurations with network block storage and configurations with local disks. Generally using network storage devices is not recommended for Cassandra but sometimes it could be optimal way for cassandra deployment and for deployment of the whole system including application nodes and other datastores. The second most important thing is where Cassandra is deployed. It could be IaaS provider's environment like AWS or GCE or it could be your own hardware. If we talking about company's own hardware, it is also important to notice if there is a virtualization layer in the system. Performance of Cassandra cluster based on dedicated hardware is better than cluster on virtual nodes. But virtual environment is more flexible.

Configurations with Network block storage

Configuration with network block storage could be divided on 2 types:

– Hardware - Storage Area Network (SAN)
– Software block storage. (As the example, Ceph for OpenStack)

SAN is a network which provides access to consolidated block level storage. Usually SAN is a physical device that allows to allocate part of the block storage and mount it to the node through the network. Virtual block storage (as the example, Ceph for openStack) allows to combine multiple physical storage node into one block storage cluster. This is more cheap solution than external physical SAN.

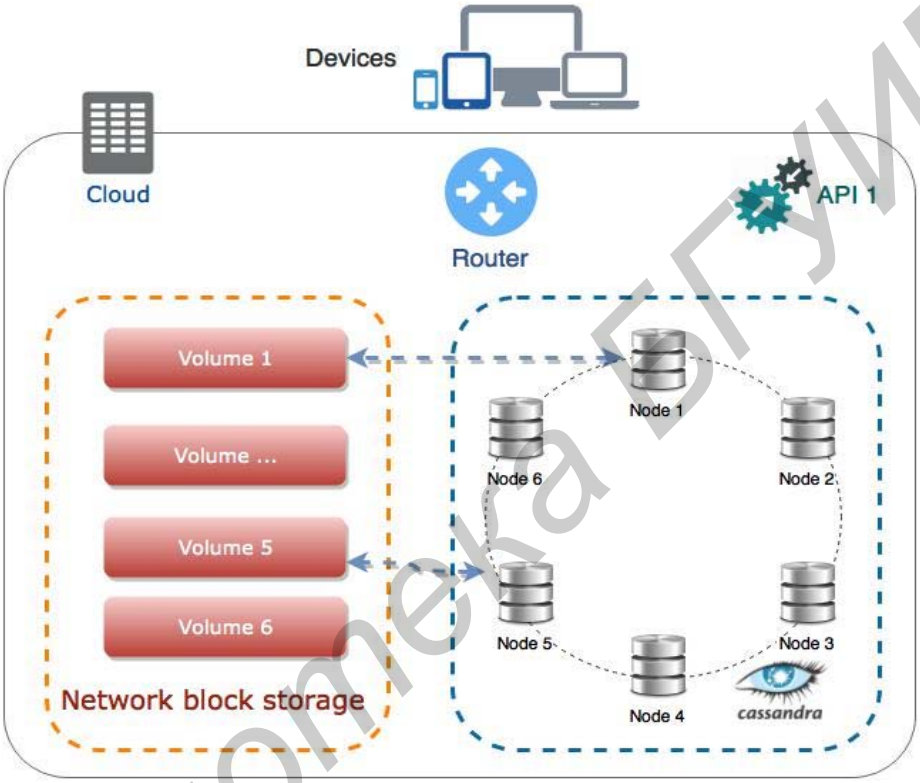

Fig. 2. Cassandra Schema with Network block storage

Cassandra schema with block storage is presented on figure 2. Every Cassandra Node communicate with its volume in block storage. This volume is mounted as disk partition to Cassandra node. All communications between Cassandra node and volume are handled by internal network.

Network block storage on your own hardware

Positive aspect of using block storage on your own hardware is security. You don't depend on cloud provider. You don't have remote data storage and you can manage this system by yourself. Block storage could be used with all the components of the system, for example by Cassandra, MariaDB and ElasticSearch. There is now need to manage configuration of block storage for each deployment. One data storage could work with all these components.

According to Datastax documentation, main negative aspect for using network storage is that it could become a single point of failure because of network problems. But I think that this situation is most actual for cases when network storage and

Cassandra nodes are in separate data centers. Another drawback is data size overhead. When using Cassandra with block storage, there are 2 different levels of data processing: Cassandra cluster level and data storage level. We need to apply data failover on both these levels. Cassandra requires 3 time replication and Storage needs 2 -time replication. Finally data would be replicated 6 time. This point would be discussed later in Storage Area Network and Software block storage sections.

Storage Area Network (SAN)

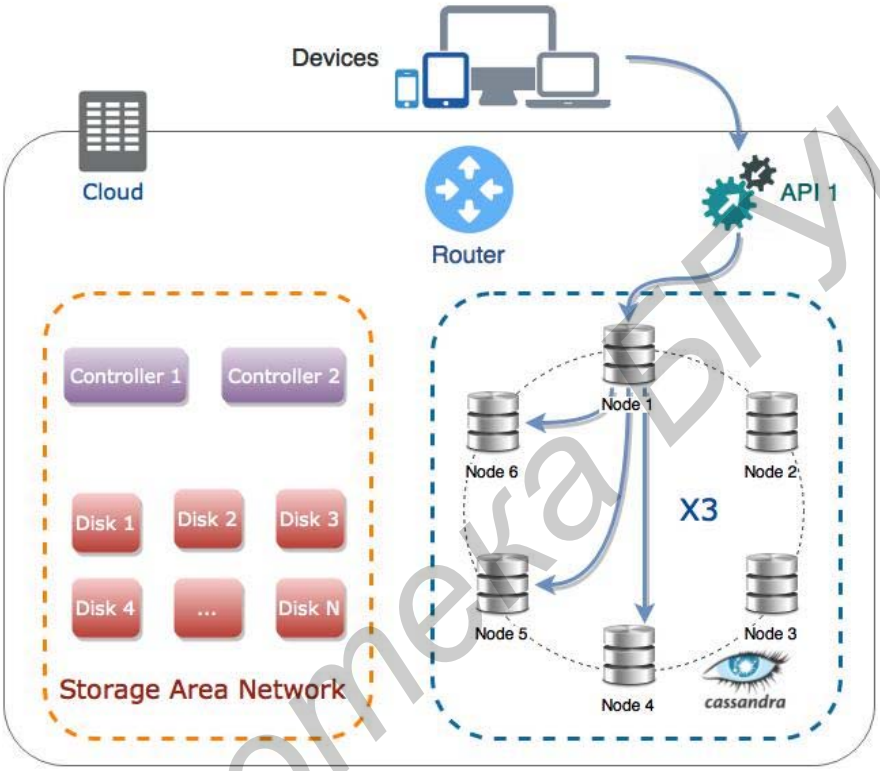

Fig. 3. Storage Area Network

Storage Area Network (SAN) is hardware block storage. It is represented on Figure 3. It is a bunch of disks connected together. Every disk is splitted on storage blocks. And these block are connected into one disk space. When new virtual partition is created, it receives blocks from several disks. It helps to divide input/output load on several physical disks.

External services could communicate with these disks by controllers. Controllers are also responsible for data replication. When cassandra node needs to write or read data from partition, it sends request to SAN controller. Then controller redirects this request to one or several disks.

Pros and cons of hardware SAN:

Positive:

– Security - it could be placed in local DC

– Universality - could be used by all components of the system (like ES, MariaDB and etc.)

– Data persistence. Data is replicated on storage level. If one disk is failed, Cassandra would work with health disk and wouldn't know about disk failure.

– Flexible - easy to change failed disk

– Performance. Performance of such disks is only 10-20% slower than Cassandra performance with local disks.

– Possibility to make cross-region data replication on storage level. It allows to set up asynchronous replication, because Cassandra allows to replicate data only synchronously.

Negative:

– Price. Hardware SAN costs too much

Software block storage

Software block storage is more cheap than Hardware. It is usually used with virtualization on the top you your hardware. If you have some physical nodes and you ran OpenStack on the top of these nodes, disks of the physical nodes could be united into one block storage space - CEPH. It is similar to SAN on figure 3. The difference is that Disk volumes and controllers are connected to each other through local Network. That's why Network could be bottleneck for this configuration. Communicating through internal network also reduces I/O performance. CEPH (Software block storage) architecture is presented on figure 4.

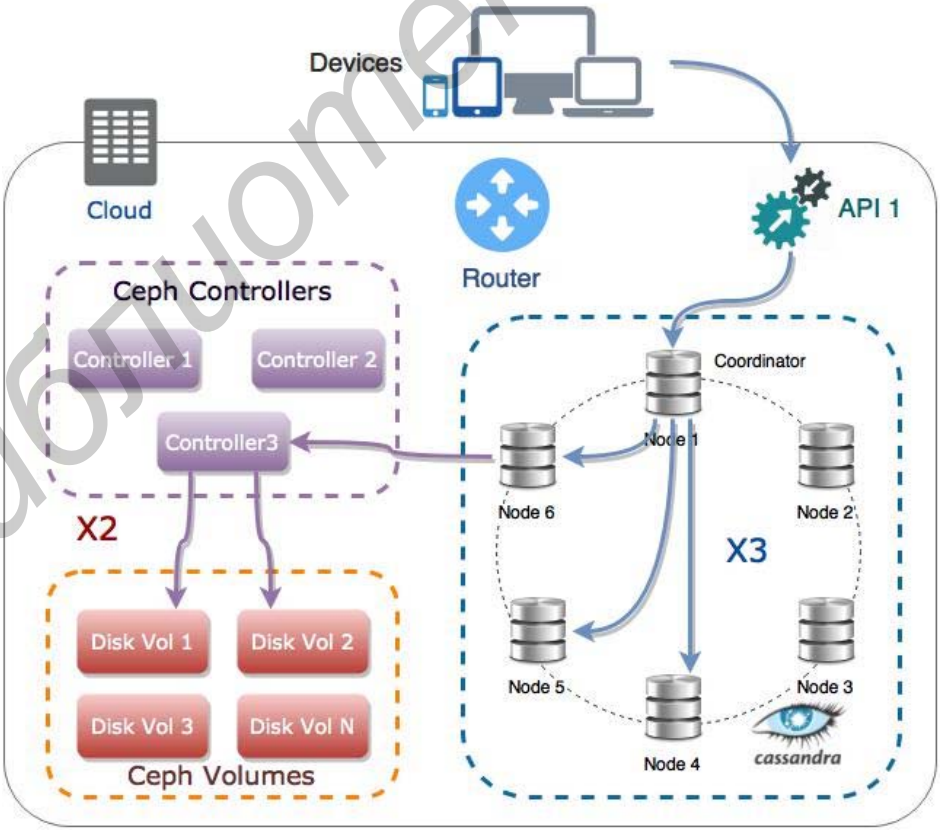

Fig. 4. Architecture of Ceph

Pros and cons of ceph

Positive:

– Security and universality - it is default Storage with OpenStack so data stores in local DC

– Data persistence. Data is replicated on storage level. If one disk is failed, Cassandra would work with health disk and wouldn't know about disk failure.

– Allows to split I/O load across multiple physical disks

–

Negative:

– Difficult to change failed disk. It requires to configure new volumes.

– Performance is low because of network overhead

– Data size overhead is is similar to SAN. 6 time replication is required

Deployment in IaaS provider's environment

Using block storage in IaaS provider's environment is similar to using Hardware SAN. Every Cassandra node receives Block Storage volume that works as disk partition. But, opposite to SAN on your own hardware, there is no need to manage these disks by yourself. This data would be replicated on the storage level so if one of the disks failed, system wouldn't work about it. It would work with disk replicas. As example, we can use Amazon AWS infrastructure. Block storage is called EBS (Elastic Block Storage). Pros and cons of cloud provider's block storage

Positive:

– Manageability. There is no need to support storage. If one or several disks failed, they would be replaced automatically.

– Durability - data is also replicated on storage level

– Backups. IaaS providers usually have mechanisms for data backuping.

Negative:

– Security. Data is stored in remote Data Centers.

– Cost. Powerful configuration could costs a lot.

– Network latencies. But this comment is applicable for previous versions of Cassandra and AWS configurations. Now EBS uses fast SSD disks. Also there are EC2 instance types like m4 and c4 that are EBS optimized. It means that network communications with EBS would have minimal latencies and overhead.

Using EBS storage for Cassandra in AWS is recommended production configuration.

Cassandra configurations with local disks

Local discs usually shows the best performance because there is now additional network communication between Cassandra node and disk volume. Also it is the cheapest and easiest way to configure Cassandra. But such configuration is not so flexible as block storage. Also process of replacing crashed disks could be more complex.

When we talk about local disks, it could be discs of 2 types, according to

deployment type:

- Local physical discs for deployment on hardware without virtualization

Using ephemeral disks usually means that storage would be physically attached to the host computer of the node. Ephemeral disks usually shows low latencies and good data throughput because there is no network overhead in such schema. But the negative moment with ephemeral disks is that there is now way how to change disk if it crashed. You have to replace all the virtual node with ephemeral disk in such case.

Magnetic and SSD disks

Ephemeral disks could be divided on 2 types according to hard drive type:

Magnetic discs are slower than SSD disks. These discs have lower throughput and lower IOPS value than SSD. These discs can't be used for system that handle a huge amount of small read and write transactions because such load type produces a lot of random reads and writes on disc. This is a weak point for magnetic discs. But rotational discs have acceptable sequential read and write performance so they could be used in Cassandra configurations for collecting data, when there are now much read and update operations.

If you use SSD disks, performance of one disk would be enough for one node for many cases. But if you use rotational discs, it is recommended to use at least 2 different physical discs: one for commit log and one for Cassandra storage.

Local discs connection configurations

When using local disks, there are several ways how to connect disks to Cassandra node:

Single disk

Single disk configuration means that every cassandra node would work with one physically attached disk. It's necessary to remember that cassandra uses disks for 2 cases: storing commit log and storing data. These operations require high disk IOPS throughput. So it is recommended to use SSD disks because rotational disk can't show enough performance. If you decided to use rotational disks it it recommended to use at least 2 disks: one for commit log and one for data. Sometimes IOPS performance of one data disk is also not enough for Cassandra. In this case it is recommended to use several physical data disks. In the next paragraphs we will discuss the ways to connect data disks: RAID0, RAID1 and JBOD

Sometimes Cassandra could be deployed in remote datacenters or in Cloud provider's IaaS infrastructure. When using such deployment it is hard to change failed disk immediately. It is a good practice to use on the nodes several 'passive disks'. It means that these disks would be mounted on Cassandra nodes but they wouldn't be

used be Cassandra. When one of the active disks failed it would be replaced by passive disk by Cassandra configuration. Failed disks could be replaced on new disk periodically (once in several weeks for example).

RAID0

RAID0 was recommended in previous versions of Cassandra. This type of disk connection allowed to increase storage performance. Such disk connections was used when SSD disks were too expensive. The negative moment of RAID0 is that after disk failure all the disk raid would be failed. It is necessary to replace dead disk in raid and restore all the raid data. If cassandra node contains a lot of data (several terabytes), restore process will demand too much time.

RAID0 should be used for cases when high disk IO performance required and there is no possibility to use SSD disks. Don't use RAID0 when there is no way how to easily replace crashed disk.

In current Cassandra versions JBOD mode shows good performance and it is more preferable than RAID0

RAID1

RAID1 was recommended for previous Cassandra versions in amazon and other IaaS deployments. For example, when using amazon, there are 2 disk types: ephemeral disks and persistent block storage (EBS). Using ephemeral disks means that these desks are physically connected to host node machines. Persistent storage is connected to node through network. Now amazon shows good network throughput between virtual node and persistent storage so EBS is recommended. But in past network throughput wasn't enough so using local ephemeral disk was recommended.

Negative moment in ephemeral disk that it is not a persistent storage. Disk would be removed after node terminated. Also there is no possibility to change disk if is crashed. You can only replace all the virtual node. That's why amazon recommended to use RAID1 with ephemeral disk. Even if one of the disks is failed, second disk in RAID1 will allow to continue processing data. And node with a dead disk can be replaced when system is not loaded, for example, at night.

Cassandra JBOD

Cassandra JBOD is alternative to RAID0. Cassandra has its own algorithm how work with JBOD. For example, there is a token range that could be handled by Cassandra node. When using several disk in JBOD mode, than cassandra process divide its token range into several small ranges. And every disk process its token range independently from other disks. This allows to work faster and separate load across several disks. Such configuration is a bit slower than RAID0, but it is more flexible. If one of the disk failed, there is no need to restore all the RAID. You can just change crashed disk and load data from replicas. At this time another Cassandra JBOD disks would continue to process data. Cassandra JBOD now is preferable disk connection.

Conclusion

If you use Cassandra in cloud providers IaaS environment like Amazon AWS, using cloud block storage like EBS would be optimal solution. This is persistent storage with replication on storage level. You don't have to manage disk storage and changed

failed disks. At the same time now cloud infrastructure can provide good throughput between virtual node and block storage so networking isn't a bottleneck at current time.

When using Cassandra deployment on your own hardware, SAN block storage shows good performance and good manageability. But it is very expensive solution and it is not profitable because of storage replication overhead. So using Cassandra local disks in JBOD mode is preferable. They show good performance and also it is easy to manage such deployment.