Seaweedfs Distributed Storage Part 3: Features.

Ali Hussein Safar · Follow
5 min read · Sep 8

Seaweedfs supports a wide range of interesting features:

1. **Adding volume server on the fly:** To increase cluster storage, a new volume server can be added effortlessly and instantaneously can be used to store new files. Also, adding more volume servers increases read and write speed.
2. **Master Failover:** The master servers are coordinated by Raft protocol, to elect a leader. The leader takes over all the work of managing the volumes. All other master servers just simply forward requests to the leader. If the leader dies, another leader will be elected then all the volume servers will send their heartbeat together with their volumes' metadata to the new leader. The new leader will take full responsibility. During the transition, there could be moments when the new leader has partial information about all volume servers. This just means those yet-to-heartbeat volume servers will not be writable temporarily.
3. **Garbage collection:** A deleted file's disk space won't be immediately recovered if your system has recently had a lot of deletions. Volume disk utilization is monitored by a background vacuum job. The vacuum job will make the volume read-only, create a new volume with only existing files.
4. **Replication:** In Seaweedfs replication is donated by the defaultReplication=ZYX parameter which is used when setting up the master servers. The three digits "ZYX" are used to define how replication is done in the cluster. Therefore, Z represents data center level replication, Y represents rack level replication, and X represents volume server level replication as shown in the following:
   XYZ=000 then no replication, just store one copy in the cluster.
   XYZ=001 then replicate once on the same rack.
   XYZ=010 then replicate once on a different rack in the same data center.
   XYZ=100 then replicate once on a different data center.
   XYZ=200 replicate twice on two other different data centers..
   XYZ=110 then replicate once on a different rack in the same datacenter.
   Note: The maximum allowed value for XYZ is 999.
5. **Erasure Coding:** Erasure coding is a method used to protect data against loss or corruption. It involves creating redundant pieces of data, known as parity from the original data. This redundancy allows for the recovery of lost or corrupted data by using the redundant pieces along with the remaining healthy data. In Seaweedfs, the erasure coding operation is donated by RS(K,N) formula where "K" is the number of shards that the volume will be split into and "N" is the number of parity shards. For instance, the default implemented EC in seaweedfs is RS(10,4). Therefore, a 30 GB data volume will be encoded into 14 EC shards, each shard is of size 3 GB (30 /10). Seaweedfs will try to store the shards in the volume so that it will provide the following:
   - If the number of servers is less than 4, EC can protect against hard drive failures.
   - If the number of servers is equal to 4, EC can protect against server failures.
   - If the number of racks is greater than 4, EC can protect against rack failures.
6. **Store file with a TTL:** Seaweedfs supports file expiration by defining TTL value after writing data to the cluster and this will make it considerable in case of content caching. However, after writing, the file content will be returned as usual if read before the TTL expiry. But if read after the TTL expiry, the file will be reported as missing, and then returns file is not found.
7. **Data Encryption:** Seaweedfs can encrypt data on the volume servers with generated keys stored in the filer store. Furthermore, every file is encrypted with a different key. When reading an encrypted file the filer service will fetch the file from the volume and then decrypt and deliver the file to the client.
8. **Cloud Drive:** This feature can provide the ability to mount an S3 bucket to the Seaweedfs file system (using filer service) and access the remote files through SeaweedFS. Effectively, SeaweedFS caches the files from the cloud. According to the cache size, all S3 bucket files can be cached with eviction when the cache is full. With the write-back caching mechanism all writes are done to the cache and the cache will upload or update files in the S3 bucket asynchronously. Since uploading to Amazon S3 is free, users can only pay for the storage they use. Therefore, this will lead to have the benefit of extremely fast access to the local SeaweedFS cluster and Near-Real-Time Backup to Amazon S3 with zero-cost upload network traffic.
9. **Storage Classification:** Since stored data has three types: hot, warm, and cold, it would be cost-efficient to place data accordingly.SeaweedFS supports storage classification, where you can place data to customizable disk types, and provides ways to move data to different tiers.
   => NVME => SSD => HDD => Cloud
   => Critical=> Hot =>Warm => Cold
   Seaweedfs implements this feature using volume server tagging for instance, when a new volume server is created, a –disk argument can be passed to set the disk tag as shown in the following command

```
weed volume -disk=ssd -port=8080 -dir=/dir/
```

Then client can use fuse mount to only mount the volumes with a tag of SSD or HDD. Therefore, critical services can use the SSD volumes, and fewer critical can use the HDD volumes.

9. **Tiered Storage with Cloud Tier:** This feature will allow seaweedFS to move full-volume files to the cloud storage provider because cloud storage is an ideal place to store warm data. Its storage is scalable, and the cost is usually low compared to on-premises storage servers. Usually, uploading to the cloud is free which makes it ideal for cold data.

10. **Images Resizing:** Seaweedfs provides the ability to scale images to different resolutions as shown in the following requests to the volume server

```
curl http://VolumeServerIP:8080/3/01637037d6.jpg?height=200&width=200
curl http://VolumeServerIP:8080/3/01637037d6.jpg?height=200&width=200&mode=fit
curl http://VolumeServerIP:8080/3/01637037d6.jpg?height=200&width=200&mode=fill
```

Conclusion

SeaweedFS is a distributed file system that is designed to be scalable, and easy to use. It is based on the idea of storing files in chunks that are distributed across a cluster of servers. This makes it very scalable, as more servers can be added to the cluster to increase capacity. SeaweedFS is also fault-tolerant, as it can continue to operate even if some of the servers in the cluster fail.

Part 1: Introduction.
Part 2: Reading and Writing Files' Process.

Resources

https://github.com/seaweedfs/seaweedfs/wiki.
https://www.usenix.org/legacy/event/osdi10/tech/full_papers/Beaver.pdf.

Thank you for reading my article on SeaweedFS. I hope you find it informative and helpful. If you enjoy the article and would like to support my work, follow me or you can buy me a coffee at https://www.buymeacoffee.com/ahsifer. Your support is greatly appreciated

Storage · Distributed Storage · Linux · System Administration · Object Storage