Cloud Data Base

Report: Project ECS Shutdown

Ebin Madan
Technicall University Munich
Munich, Germany
ebin.madan@tum.de

Salma Kefi
Technicall University Munich
Munich, Germany
kefisalma1907@gmail.com

1 INTRODUCTION

In the practical course of Cloud Data Base, a Cloud Data Base as a distributed and replicated storage service was built. This storage can accept multiple clients and store different key-value pairs based on consistent hashing to their respective server in the service. The data is stored persistently to the disk and intermittent in a data cache for quick accessibility. If the whole system consists of more than three servers, The key-value pairs are replicated on two additional servers. The client must be connected to one of the servers with the original data to do write or update operations. For reading data the client must be either connected to the original server with the respective data or to one of the replication servers. If one server shuts down - gracefully or ungracefully - another server takes the data that was stored in the crushed server and replaces the range of the previous one. The system is managed by an external configuration service (ECS), which is responsible for assigning ranges to the server and initializing transfers. To detect failure of Key value pair servers (KV servers) and initiate the recovery, the ECS pings the KV servers constantly.

2 PROBLEM STATEMENT

As noted in the introduction the storage system is managed by the ECS. If one KV Server (server with key-value pairs) fails or is shut down, the ECS manages the transfer of the data and the replacement of the data range with an existing KV server of the system. Therefore, an unexpected failure of the KV servers is handled and the system can proceed working. However, there still is a single point of failure, when the ECS crashes. So far, the ECS is the head of the system and if the ECS goes down, the KV servers that are connected to it will also fail. To solve this problem an addition to the current system was implemented to avoid the single point of failure of a graceful or ungraceful shutdown of the ECS.

2.1 Approach

To solve this problem many designing approaches have been taken, to react quickly and efficiently to an ECS crash. The implementation solves both problems of graceful and ungraceful shutdown (e.g., server crash) the same way: The most efficient and least noticeable approach from the outside to handle an ECS crash is to replace it with a new ECS. But the system can not open a new server on its own without knowing about the usable ports or hardware resources. This implementation avoids this problem by selecting one of the already existing KV servers to use its framework and hardware to transform into the new ECS. After one KV server is chosen and transformed to the new ECS, all the other KV servers connect to the new ECS, and the system is again stable and ready to proceed. The data of the transformed KV server is transferred to the KV server with the responsible hash range before it is deleted on the ECS.

2.2 Design

Firstly, the KV servers need to detect the shutdown of the ECS (figure 2). Therefore, we can use the heart-beating mechanism, which pings the ECS constantly like the mechanism used by the ECS to detect KV server failure. After the failure is detected, all KV servers go into the initialization mode. This means, that no client requests are further processed until the ECS is replaced and the connections are reset. Next, the KV servers need to promote one of the existing KV servers as the new ECS (figure 3). This implementation has this already specified in the code: The KV server with the range including the hash zero (i.e., 0x00000000000000000000000000000000) will be transformed to the new ECS. Every server checks with the meta data, whether it is responsible for the key with the hash zero and as a result one of the KV servers will be transformed and the other servers will remain as KV servers. In the following the next steps of the transformation of a KV server into an ECS and the process of the other KV servers will be explained separately:

2.3 New ECS server

Once the KV server that will be transformed into an ECS is found, the server starts the process of transformation: Firstly, the KV server needs to be closed completely. Therefore, all the connections to clients are closed and no more requests or connections are accepted by the server socket. Afterwards the running processes and threads by the echo logic are stopped. If replication is active, the replicas of this server's data is deleted before the connection to the replication servers are also closed (this is necessary for later steps). With the echo logic and its parallel processes completely closed, the server now can run the same code of the ECS with the same host and port as before, so that the remaining KV servers know where to connect to. The idea is that a manual started ECS and transformed ECS are not distinguishable for the KV servers. But there are some minor changes, because there is still data on the disk of the new ECS from its old duty as KV server. The ECS will remain that data until the first KV server reconnects or connects (figure 4). With this implementation, the data of the old KV server will never be lost, even if there is no other KV server in the system. After the first KV server connects and is added into the storage system, the ECS sends its data to this server and proceeds just as a traditional ECS. The idea to send the data to the first KV server that connects to the new ECS has several benefits. Firstly, as already mentioned, if the new ECS was the only KV server in the storage system before, the data will be stored with the ECS until a new KV server is added to the service. Another advantage is, that the new ECS does not need to calculate the hash and wait for the server with the specific hash to transfer it to. If there is replication, the ECS also does not need to transfer the data to the belonging replication servers. With this implementation the first KV server receives the data, when it is responsible for all the data in the hash circle. When the next server connects to the ECS, the data will be divided and distributed in the standard process of adding a new KV server to the service (figure 5). With this mechanism, the old data of the ECS will always be transferred to the KV server with the responsible hash range until all KV servers are added to the system again. The replication will also automatically be initiated after three KV servers are in the system without further implementation (figure 6).

2.4 KV Servers

If the KV Server is not responsible for the range which includes the hash of zero -this means the server will not be transformed in to an ECS- , it restarts the whole process of the echo logic without closing its connections to the clients. The data will be kept on the disk, but the replicas will be deleted in that process, because the replicas will be transferred to each other in the process of connecting to the ECS. The main data must remain on the disk, so that no data vanishes, and this can happen, because every server will keep its range in the hash circle except from one KV Server that will expand its range and stores the old data of the transformed KV server. In the process of resetting the echo logic, the KV Server will rebuild the connection to the ECS. The server gets the host and port for the new ECS from the meta data by checking which server is responsible for the hash range with the hash zero. From here on the process is identical to the standard process of connecting to the ECS and starting the KV server again. Eventually the KV server needs to store the data of the new ECS / old KV server, but this happens like every incoming transfer.

After the ECS is revived, all KV servers will connect to it at the same time. To handle the addition of all KV servers without interfering each other's transfers a queue is implemented. This queue handles all requests of adding one KV server after another to the system. Additionally, a new method in the API of KV servers was added: With the method "`ecs_address`" clients can receive information about the current ECS. It returns the host address and port, when it is successful. If a new KV server is added to the storage system manually after the replacement of the old ECS, the socket information of the current ECS can be obtained with this method and set as the bootstrap.

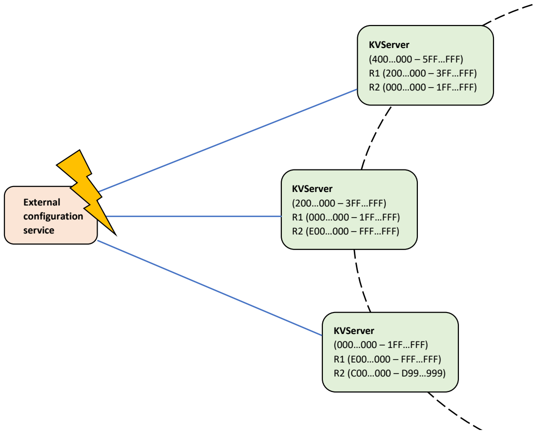

Figure 1: ECS shuts down.

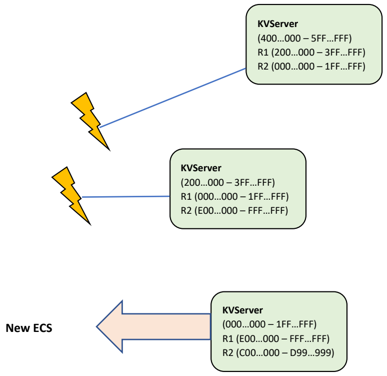

Figure 2: Shut down of ECS is detected by KV servers and the new ECS is defined.

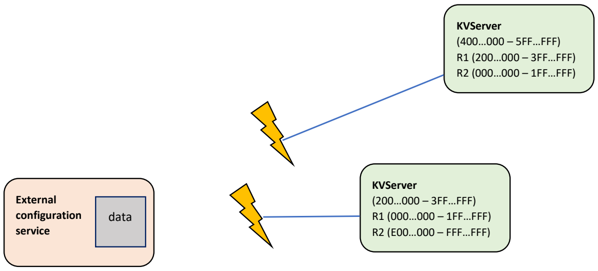

Figure 3: The specified KV server is transformed into the new ECS.

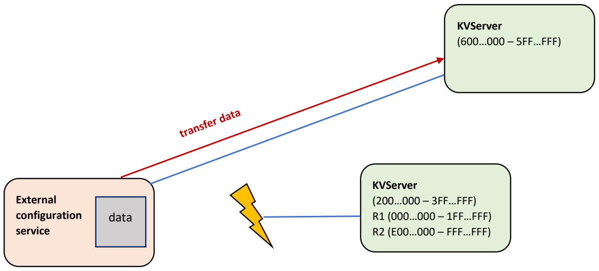
KVServer
(600...000 – 5FF...FFF)
transfer data
External configuration service
data
KVServer
(200...000 – 3FF...FFF)
R1 (000...000 – 1FF...FFF)
R2 (E00...000 – FFF...FFF)

Figure 4: First KV server is added to the system. The data of the new ECS is transferred.

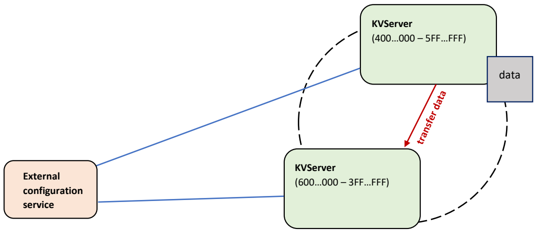

Figure 5: Second KV server is added to the system. The data of the new ECS is passed on.

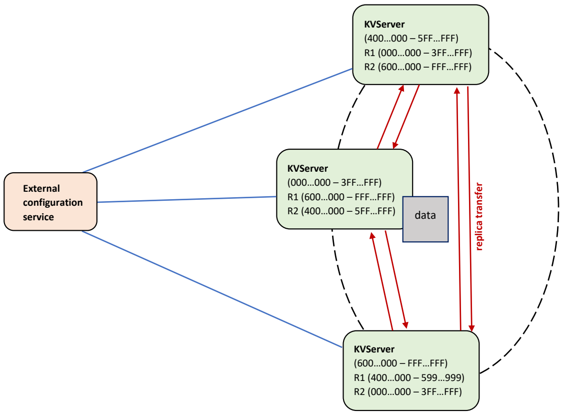
KVServer
(400...000 – 5FF...FFF)
R1 (000...000 – 3FF...FFF)
R2 (600...000 – FFF...FFF)
External configuration service
KVServer
(000...000 – 3FF...FFF)
R1 (600...000 – FFF...FFF)
R2 (400...000 – 5FF...FFF)
data
replica transfer
KVServer
(600...000 – FFF...FFF)
R1 (400...000 – 599...999)
R2 (000...000 – 3FF...FFF)

Figure 6: Third KV server is added to the system. The replication of the data is active.

3 EVALUATION

The evaluation for the new implementation was tested manually and with automatic tests. The tests are implemented in `/src/test/java/de/tum/i13/Project` and the corresponding data is stored in the folders named according to their test class. Everything is measured within one system. Hence, latency of the network or distance between the servers are not considered. Furthermore, the disk and processor of all generated servers are the same. This isolates the system and leads to better benchmarks of the implementation. In the following the results and the generated benchmarks are evaluated.

3.1 Functionality

The key functionalities are tested by the methods of the test class `/src/test/java/de/tum/i13/Project/Functionality.java`.

The method `testEcsFailure()` tests if after the ECS shutdown one of the KV servers transforms into the new ECS. Therefore, two KV servers are connected to the ECS and the ECS is removed. Afterwards the current ECS info is retrieved by the new method `ecs_address` to check the transformation.

In `testECSfailureWithMoreKVs()` more KV servers are generated to test, whether the core functionality works the same way and the new ECS can handle more than one KV server.

Now data is added to the KV servers: Therefore, the method called `testEcsFailureWithData()` tests the functionality of the new ECS to transfer its old data to a new KV server. For that, two KV servers are added to the ECS. A client is added to the KV server of the becoming ECS and stores key-value pairs. Then the ECS is shut down and the KV server transforms to the new ECS and transfers its old data to the other KV server. To check this, a client is again connected to the remaining KV server and confirms the storage of the data.

As already mentioned, the implementation of the ECS failure allows the new ECS to store its old data on the disk if there are no further KV stores in the system until a new KV server connects to the ECS manually. The method `testAddingKVtoNewECS()` tests this functionality by just adding one KV server to the storage system and storing data. Then the ECS is shut down again and the only KV server in the system is transformed to the new ECS. Because there is no other KV server in the system yet. The data remains on the disk of the new ECS. Now a new KV server is connected to the new ECS and the data is transferred immediately to this KV server. To verify this a client is connected to the new KV server and checks the data of the old KV server.

Further evaluation of the functionality was done manually. An error which is not avoidable in our implementation is when a KV server shuts down during the rebuild of the storage system. This will not cause the storage system to fail but the KV server will be removed without saving the data on its disk or reconstructing the data because the data is not replicated during storage rebuilding. Errors may also occur due to errors with the connection between sockets. This would result to a loss in data. But this case is very exceptional and will only happen if the connection or the run environment is unstable.

3.2 Performance

The performance is tested by the methods of `/src/test/java/de/tum/i13/Project/Performance.java`. The methods were used to vary the data size and number of KV servers. All ports and key-value pairs were generated randomly to receive unbiased results and the tests were performed several times to exclude outlined data.

First the scale of the implementation is tested. The diagram of figure 7 shows the duration it took to rebuild the storage system after shutting down the ECS on different number of KV servers. The data that is in the system consists of 1000 key-value pairs. The performance is increasing with the number of KV servers in the system. This is because every KV server must reconnect to the new ECS one after another and transfer all data and replicas again to the respective servers. The run time of the storage rebuilding is increasing almost linearly from 21ms with zero KV servers to 65116ms or 65s with 45 KV servers (the deviations in the plot lead back to internal system behaviours e.g. thread management). It must be considered that the system is also slowing down with the number of KV servers build on the same processor because of the number of Threads that are running in parallel. If the servers were running on their own processor and with their own storage, the performance would be better than in this benchmark.

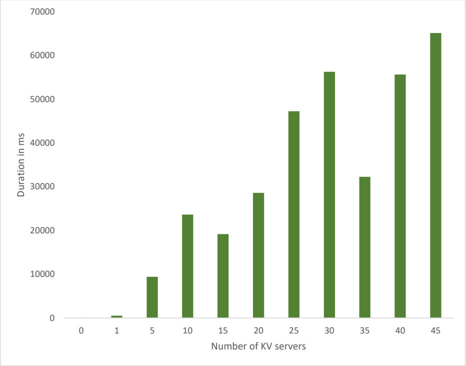

Figure 7: Time to rebuild the storage system after shutting down the ECS on different number of KV servers.

The next plot in figure 8 displays the duration until the first, second etc. KV server rejoins the storage system with data of 1000 key-value pairs. In general, the time is increasing for each KV server (blue chart). One point that is noticeable is that the addition of the first server takes some time after the ECS shuts down. The second server needs almost no time and the third one increases dramatically in time (orange chart). The reason for that are again the transfers. For the first server there is only the transfer of the old data from the ECS to the first server. This takes a little bit of time. For the second server there is often no transfer because the data of the second KV server is already on the disk (There could be the transfer of the old data of the ECS but this is not always the case). This means the addition of that takes negligible time. But for the third server replication is active. This means there are several replication transfers that are time consuming. From here on in the diagram there are no bigger inconsistency because the replication is active, and the number of transfers are always the same. But if we look at the durations (orange chart) in the long term, we see that the time needed to join the storage system is decreasing with the number of KV servers. This is the case because the hash ranges are getting smaller with the number of KV servers in the storage system. This leads to quicker look ups and faster transfers due to less data.

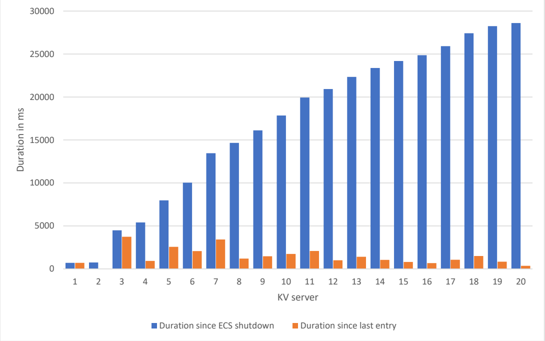

Figure 8: Time to for each KV server to enter the storage system after shutdown of ECS (blue) and the process without queue time (orange).

Now the performance of the addition on different size of data is measured. Therefore, figure 9 presents the duration of rebuilding the storage system on data from 0 to 1000 key-value pairs. There are no big inconsistencies, and the duration is increasing almost linearly. More data leads to longer transfers but the process remains the same for the 20 KV servers that were used for the evaluation.

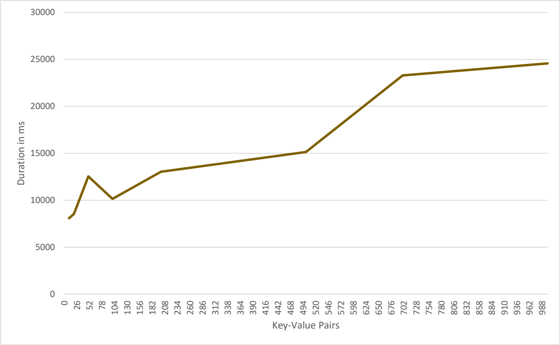

Figure 9: Time to rebuild the storage system with 20 KV servers after shutdown of ECS for different size of data.

Lastly, to compare the traditional building up of the storage system to the rebuilding of the storage system after ECS shutdown we have the benchmark in figure 10. The process is again demonstrated on storage system with several KV servers. The ECS always needs more time to build the storage than the transformed ECS to rebuild the storage. This difference is again caused by different number of transfers. For the rebuild there are only replica transfers as the original data is already on the disks. On the other hand, for the normal storage building the original data must be transferred according to the hash ranges as long to the replicas. It can be seen that this difference only occurs from three KV servers on because before that there is no replication active.

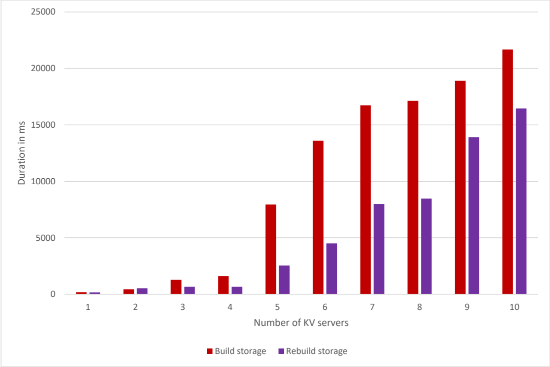

Figure 10: Time to build the storage system (red) and time to rebuild the storage system after ECS shutdown (purple).

4 CONCLUSION

The main contribution of the addition to the already existing storage system is to solve the single point of failure when the ECS shuts down. This is successfully achieved so that most of the cases regarding the failure or graceful shutdown of the ECS are covered in a very efficient way without influencing the existing functionality of the storage system.