Ring Based Leader Election Algorithm for Distributed Key-Value based Storage Servers

Beyza Altuntas
Faculty of Informatics
Technical University of Munich
beyza.altuntas@tum.de

Ibrahim Can Kaplan
Faculty of Informatics
Technical University of Munich
ibrahim.kaplan@tum.de

Halil Sahiner
Faculty of Informatics
Technical University of Munich
ge26kic@mytum.de

ABSTRACT
This paper introduces the concept of ring-based leader election algorithm[4] to the distributed key-value storage system to increase fault tolerance. In the system this algorithm is built upon, an External Configuration Server (ECS) controls the initialization of the key-value storage and detection of failures in the key-value storage. The algorithm tries to eliminate the single point of failure that comes with the central ECS by integrating the ECS into one of the key-value based storage server (KVS) and electing another KVS for the ECS in case of a failure in the previous one.

Author Keywords
distributed, scalable, fault-tolerant, leader election, key-value storage

INTRODUCTION
Starting from the 21th century, the internet became a widespread platform where millions of people connected with each other[7]. People use social media to connect with their friends, read the news, check their bank accounts, entertain themselves with numerous platforms. The increasing number of users have brought a lot of opportunities and also challenges to the internet industry.

Tremendous increase in the number of users made the old systems obsolete for sustaining that huge amount of data. The need for more storage spaces, handling millions of users' accesses, distributing storage servers all around the world to reach people pushed the existing boundaries in the industry. These challenges pressured the internet companies and the academia to find solutions for the increased number of user demand. To handle millions of people and devices' data in different parts of the world, distributed and scalable databases become an industry practice to solve these challenges[3].

Growing number of tech companies used the technological advancements from virtualization, storage technologies, microprocessors etc. to establish their data centers. In addition to these technological developments, they used and utilized commodity hardware to make this business model profitable[5]. The problem with the commodity hardware was their tendency to fail frequently. So another challenge has been added to the current problems. To be able to handle highly frequent failures. Monitoring techniques like heartbeats helped to detect failures, coordination algorithms like leader election helped to select new coordinators to manage other nodes. The usage of replication for recovering failed nodes/servers has become an industry standard[6]. It is achieved by replicating one server's data and storing it in a configurable number of redundant copies.

At the beginning of the project, a client-server architecture is developed as the first milestone. Client was capable of sending messages to the server, and the server was echoing the received message to the client back.

The foundational part of any cloud-based database is storage servers. A KVS with capability of using a configurable cache is developed as the second milestone. Clients were able to perform PUT, GET, DELETE operations on the storage server.

The ability of scaling out and in with the variation in the user demand created efficient usage of shared resources in cloud computing. An ECS added to the architecture of the system to handle the addition and removal of distributed nodes to the system in the third milestone.

Monitoring nodes and replacing them in case of a failure, replicating storage servers to a configurable number of redundant copies are the crucial abilities of a fault-tolerant, distributed database. Health-check of KVSs, replication and redistribution of data storage servers have become the additional responsibilities of the ECS Server in the fourth milestone.

PROBLEM STATEMENT
The developed architecture in fourth milestone to build a cloud database covers the fundamental aspects of a modern KVS: a distributed, scalable and replicated storage mechanism. The main missing part of this mechanism lies in one of the golden rules of distributed systems : no single point of failure. There should be no single node in the system which results in the failure of the entire system.

In the current architecture, ECS is the main bottleneck and the single point of failure in the system. Whenever the ECS fails,

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

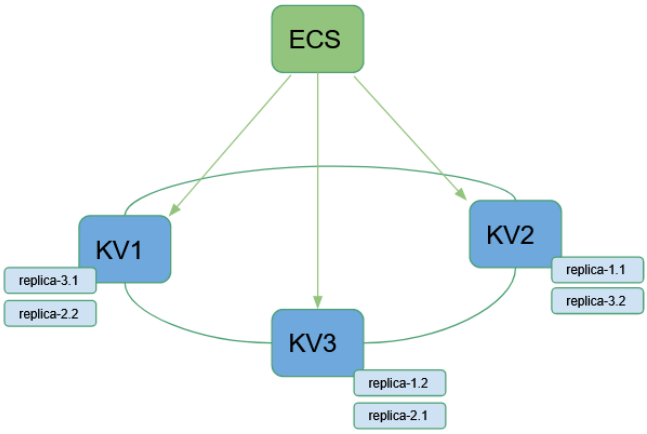

Figure 1. Architecture in Milestone 4

the system loses its ability to add and remove nodes, check the availability of KVSs, and replicate the data storage in redundant copies. It loses its distribution, scaling and replication abilities which are fundamental milestones to form a cloud database. It only sustains its KVS capabilities like serving to the clients with PUT, GET and DELETE operations. In case of a failure in any other node, the client may lose the connection to the entire system. If the failed node is not the connected one, the client can preserve its connection, but the connected node may not be able to redirect the client to other KVSs in case of update and delete operations for their corresponding keys.

The proposed solution for these problems in the current architecture is converting the existing system to a more peer-to-peer system. In case of a failure of ECS, any other node can be elected as the new ECS. They also preserve their KVS capabilities to serve the clients. This approach improves the robustness of the system and brings a fault-tolerant aspect to the database, which is also one of the crucial requirements of a modern cloud database.

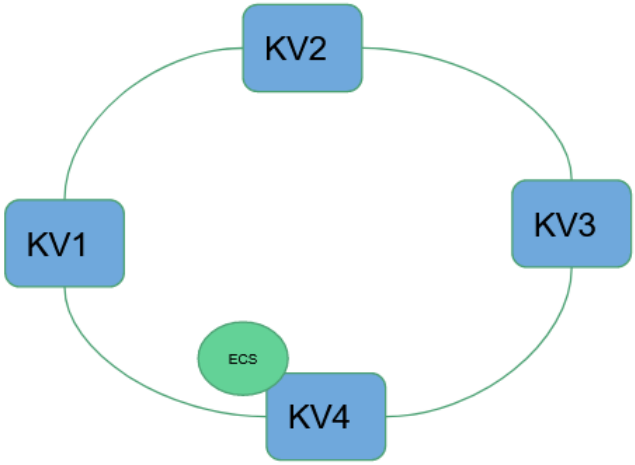

Figure 2. Architecture in proposed solution

APPROACH

In the final milestone, to implement a more peer-to-peer system, we decided to add leader election mechanism to our current system. In distributed computing, leader election is the process of designating a single process as the organizer of some task distributed among several nodes. As a result of the literature search, we found various leader election algorithms such as Chang-Robert Algorithm and Bully Algorithm[4]. Chang-Robert Algorithm is a ring-based coordinator election algorithm, employed in distributed computing. This algorithm assumes that each process has a Unique Identification (UID) and that the processes can arrange themselves in a unidirectional ring with a communication channel going from each process to the clockwise neighbor.

On the other hand, Bully Algorithm is a method for dynamically electing a coordinator or leader from a group of distributed computer processes where every process can send a message to every other process in the system[2]. Since both algorithms create a bus network of processes,we compared the total number of messages sent to complete the election. Required numbers of messages to elect a leader in the Chang-Robert Algorithm average number of message passes of order $(n \log n)$ whereas in the Bully algorithm it is n^2[4]. Values indicate that on average, ring based algorithm is more efficient than the bully algorithm. Therefore, we decided to implement the Chang-Robert Algorithm to perform leader election in our system.

ARCHITECTURE

To start a leader election, the system should detect the failure of the current ECS. We implemented a two-sided heartbeat mechanism to solve this problem. ECS sends a heartbeat message to every KVS in the ring periodically. Similarly, every KVS sends a heartbeat message to ECS. If a KVS can not get a reply in 700 ms from ECS, it triggers a leader election in the system. Our leader election mechanism consists of two different parts: leader election at the start and leader election when the existing ECS fails. To determine the ECS server at the start, we developed a system where a KVS searches for other servers in a specified IP range. If it gets a reply, it saves the location of the ECS, initiates the communication, and retrieves the key range. If it does not get a reply back from any of them, it declares itself as an ECS server. However, it has a drawback of a long cold start of the first server. Since there will be other KVS on the ring most of the time, we assume that this drawback could not affect our system too much. In the second part of the leader election, we calculated the MD5 hash value of the port and IP address of KVS to use it as unique identification. Implementation steps can be listed as follows:

1. Initially, each KVS in the ring is marked as a non-participant.
2. A KVS that notices a lack of leader by heartbeat mechanism starts an election. It creates an election message containing its hash value. It then sends this message clockwise to its successor.
3. Every time a KVS sends or forwards an election message, the KVS also marks itself as a participant.

4. When a KVS receives an election message it compares the hash value in the message with its own hash value.
    - If the hash value in the election message is larger, the KVS unconditionally forwards the election message in a clockwise direction.
    - If the hash value in the election message is smaller, and the KVS is not yet a participant, the KVS replaces the hash value in the message with its own hash value and sends the updated election message in a clockwise direction.
    - If the hash value in the election message is smaller, and the KVS is already a participant (i.e., the KVS has already sent out an election message with a hash value at least as large as its own hash value), the KVS discards the election message.
    - If the hash value in the incoming election message is the same as the hash value of the KVS, that KVS starts acting as the leader.

When a KVS starts acting as the leader, it begins the second stage of the algorithm as follows:

1. The leader KVS marks itself as a non-participant and sends an elected message to its neighbor announcing its election and hash value.
2. When a KVS receives an elected message, it marks itself as a non-participant, records the elected hash value, blocks any election-related messages, and forwards the elected message unchanged.
3. When the elected message reaches the newly elected leader, the leader discards that message and removes block in other KVS by broadcasting a message and the election is over.

For a successful leader election, we have some assumptions in our system as follows:

- The first ECS should be started beforehand to avoid multiple leaders when there are no KVS in the system.
- The range of KVS nodes in the application should reflect the possibility of server ids in the system since the ECS can be any node in the network.
- There shouldn't be a failure in the ring during the election since the communication of the election depends on successor relations. However, we covered the situation where the successor is an old ECS that is shut down without notice.

Since these assumptions do not affect our system in general, we are considering them as edge cases that can be implemented as future improvements in our system.

EVALUATION

The performance of the system is evaluated in four aspects which are the latency of adding a KVS to the ring, the latency of adding a KVS to the ring while communicating with clients, the latency of electing a new KVS to take the ECS role, and overall latency and throughput with different numbers of clients and KVSs.

The environment used for testing has AMD Ryzen 5600x [1] which supports 12 physical threads and 4x8 GB RAM with 3000 MHz. The clients and KVS applications are run in Docker containers to simulate a distributed environment that allows communication only through ports defined by the containers.

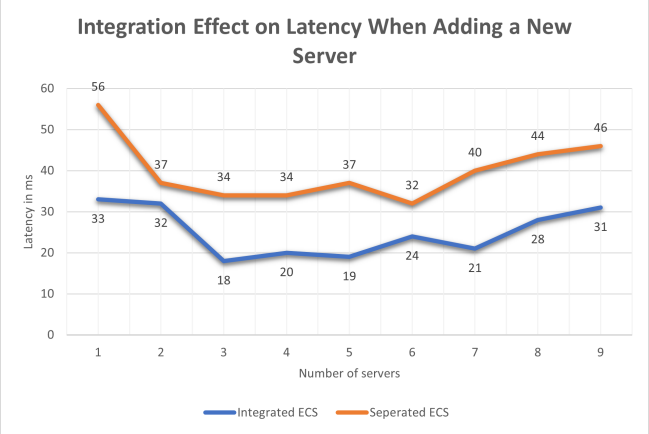

Figure 3. Integration Effect on Latency When Adding a New Server

Before integrating the ECS to the KVS node, adding a new KVS to the ring only burdens the ECS node. However, with the integration of ECS, the initialization process is competing with the processes which KVS created for clients and KVS to KVS communication for the resources. Hence, the tests for adding KVSs with client communication and without client communication are conducted. In Figure 3, the comparison between an integrated ECS and a separated ECS indicates that the decrease in the number of nodes in the machine with the more utilized node with integration decreases the latency in the overall communication latency. Additionally, with the increase in critical points in the communication threads, some optimizations are made on ECS and KVS communication handlers. Therefore, the latency decrease with the ECS integration was expected to some extent.

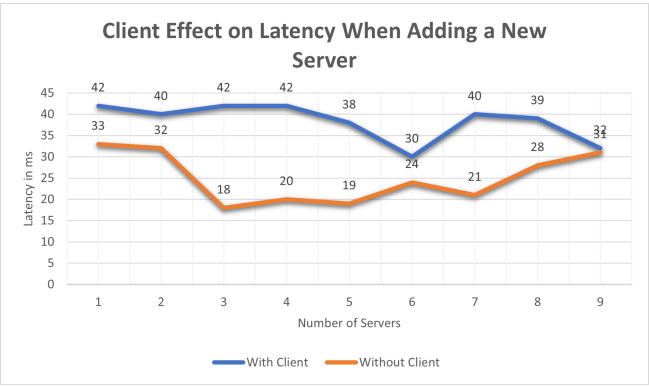

Figure 4. Client Effect on Latency When Adding a New Server

In Figure 4, an increase in the latency of adding a KVS to the ring is observed when a setup without any client connection

and with a client connection are compared. The test client was sending random KV pairs that caused different KVS connections during the test. Hence, a decrease in latency is observed due to the decrease in the workload of Client - KVS communication for the KVS with ECS. Nevertheless, the increase in KVSs created ECS - KVS communication overhead and increased the latency of the addition process. When these latencies are compared with the test without a client, the difference in the latency is 14 ms on average.

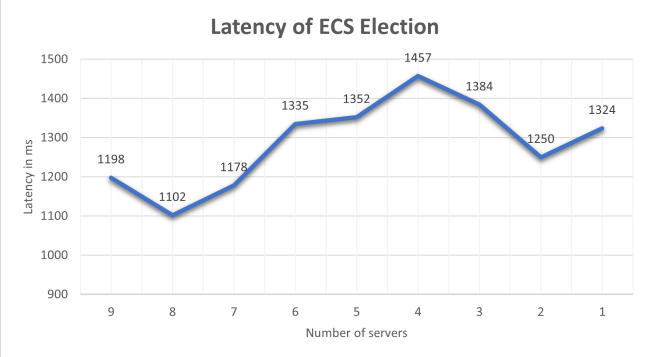

Figure 5. Latency of ECS Election

For benchmarking the latency of electing a KVS to take the ECS role, a setup without client communication to see the pure communication overhead of election and 700 ms heartbeat periods are created. In Figure 5, a trend in latency increases when the number of servers decreases is observed. Since the heartbeat messages from KVS nodes to the ECS are not simultaneous, increasing the number of servers enables the system to react faster to failures in ECS. The number of messages propagated in the ring to elect a server has little effect on the election latency when it is compared with the failure detection latency due to heartbeat intervals. From the experiments, the message propagation to elect a new leader takes around 50 to 100 ms depending on where the election starts.

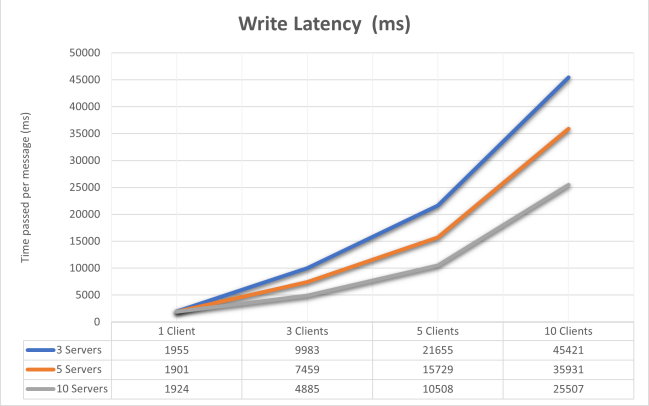

Figure 6. Write Latency

The write performance is evaluated for different numbers of clients and servers. The clients send 7885 KV pairs from a 16.5 MB KV pair pool that is the same for every client when there are multiple clients that send data to servers. Therefore, when we increase the number of clients the put requests get a "put update" reply. The file system for storing the KV pairs separates the keys and values in different files for fast searches. Furthermore, the updated pair is appended at the end of the key and value files to increase the speed of write operations. However, this causes an increase in file sizes and latency when the KV pairs are absent.

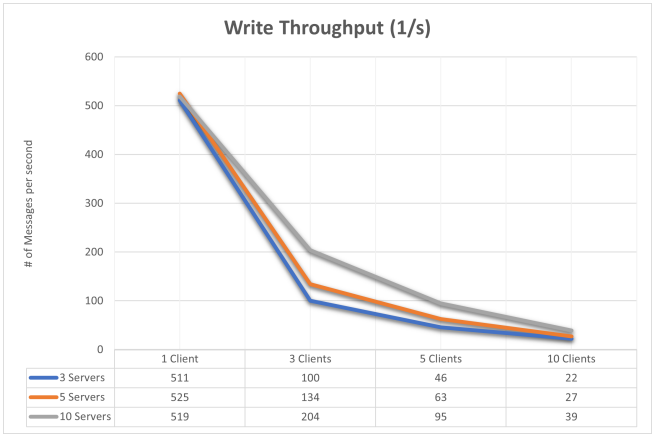

Figure 7. Write Throughput

In Figure 6 and Figure 7, the effect of the file system used in the system can be seen in the lower number of KVSs. If the number of KVSs increases, the KV pairs are distributed sparsely in between KVS which means faster key searches for first time KV pair writes from clients. However, this file system benefits dissolve when the number of clients increases. Since even though the KV pairs sent from clients are the same, the file sizes increase because the updates do not erase the old data but append the KV pair to the end of files. Hence, the first time KV pair writes faces a bigger latency with a high number of clients.

CONCLUSION

In conclusion, the ring based leader election algorithm introduced new advantages and disadvantages as an extension for the distributed key value storage that is built upon. The advantages fall under the topic of flexibility and robustness of the system. With the failure detection of ECS and electing a new ECS in case of failure, users of the distributed key value storage can be more flexible about the special hardware to run ECS and worry less about the risk of loosing all the data stored in the KVS in case of ECS failure. On the other hand, the disadvantageous characteristics of the system can be described as network overhead on existing nodes and the assumptions of the ring based leader election. With the double sided heartbeat for failure detection of ECS, KVSs communicate each other more frequently. It indicates that the clients communicating with the KVS that has the role of ECS can face slower responses when it is compared with separated ECS implementation. For the assumptions, the election process has some critical points that prevents any KVS in the existing ring to shutdown or initialization of new KVS to the system until the election finalizes. In the end, the increased robustness of the

system has some trade offs that cannot be eliminated; however, having a fall back mechanism that enables using fail-prone hardware ease the burden of maintaining a distributed system.

REFERENCES

[1] AMD 2022. AMD Ryzen™ 5 5600X. (20 August 2022). Retrieved August 20, 2022 from https://www.amd.com/en/product/10471.

[2] A. Arghavani, E. Ahmadi, and A. T. Haghighat. 2011. Improved bully election algorithm in distributed systems. In *ICIMU 2011 : Proceedings of the 5th international Conference on Information Technology Multimedia*. 1–6. DOI:http://dx.doi.org/10.1109/ICIMU.2011.6122724

[3] Sandeep Bhowmik. 2017. *Cloud Computing*. Cambridge University Press.

[4] Ernest Chang and Rosemary Roberts. 1979. An Improved Algorithm for Decentralized Extrema-Finding in Circular Configurations of Processes. *Commun. ACM* 22, 5 (may 1979), 281–283. DOI: http://dx.doi.org/10.1145/359104.359108

[5] H. Gilbert Miller and John Veiga. 2009. Cloud Computing: Will Commodity Services Benefit Users Long Term? *IT Professional* 11, 6 (2009), 57–59. DOI: http://dx.doi.org/10.1109/MITP.2009.117

[6] Ali Shakarami, Mostafa Ghobaei-Arani, Ali Shahidinejad, Mohammad Masdari, and Hamid Shakarami. 2021. Data Replication Schemes in Cloud Computing: A Survey. *Cluster Computing* 24, 3 (sep 2021), 2545–2579. DOI: http://dx.doi.org/10.1007/s10586-021-03283-7

[7] Statista Research Department 2022. Number of internet users worldwide from 2005 to 2021. (27 July 2022). Retrieved August 20, 2022, from https://www.statista.com/statistics/273018/number-of-internet-users-worldwide/.