A Novel Weight-based Leader Election Approach for Split Brain in Distributed System

Feng Jiang, Yongyang Cheng, Changkun Dong and Erdong Yu

China Telecom Cloud Computing Corporation, Beijing, China.
Email: {jiangfeng, chengyy2, dongck and yued}@chinatelecom.cn}

Abstract. The rapid development of High Availability (HA) system has attracted growing attention from both industry and academia. Due to the network and hardware failure, a complete system would split into two or more separate partitions, which begin to compete for shared resources, resulting in system chaos and data corruption. In this paper, we propose a novel weight-based leader election approach for split brain in a distributed system. The leader of separate partitions would be elected by embedding an arbitration program in the client. Unlike the traditional mode, the leader election in our proposed approach is lightweight and fully automatic, meaning that no additional hardware resources or manual intervention are required. The approach presented in this paper has been validated to be valid through qualitative and quantitative experiments.

1. Introduction

With the development of information technology, the business scenarios become more complex and the amount of data to be processed becomes larger. In order to improve the performance in scalability, availability and reliability, the distributed storage system uses multiple servers to share storage load. As we know, the distributed system has advantages of low cost, flexible expansion capability and unified resource pool management by nature [1]. Thus, it has gradually replaced the traditional storage and becomes a powerful way for processing massive data effectively. However, due to the network and hardware failure, a complete system would split into two or more separate partitions. A split brain condition is the result of the partition, where each side believes the other is dead, and then proceeds to take over shared resources as though the other side no longer owns any resources. In this case, if each side continues to respond to reads and writes while they are unable to communicate with each other, the whole distributed system would diverge and no longer be consistent. Thus, the key to solve the problem of split brain is to select a leader from given candidates based on some strategies. In general, there are several common approaches of leader election to avoid split brain. In work [2], an approach based on dedicated access is proposed, which enforces that the separate partitions could continue to communicate to select the leader by adding a heartbeat line. This is the easiest way to avoid split brain from the perspective of engineering practice. Work [3] and [4] use additional detection nodes, like database, IP address or USB, as additional detection nodes, to form a new voting group with original partitions to conduct the leader election. In this approach, only the partition that could normally connect to the additional detection node has the right to vote as a leader. In work [5] and [6], a logical concept "region leader" is proposed to avoid the stale read, which requires that all reading and writing requests from clients must go through the "region leader" before they are processed, which guarantees the consistency of data reading and writing at the cost of client's availability.

Although these mentioned approaches could reduce the probability of split brain occurring in some scenarios, challenges still remain to be solved. First of all, there is also the possibility of disconnection of the heartbeat line. In this case, the problem of split brain remains unsolved. Furthermore, the detection node-based approach works on the server layer, which not only leads to additional resource costs, but also requires manual completion. Finally, the region leader-based approach only could solve the split brain when a whole system is divided into three or more separate partitions, but no longer works when there are only two separate partitions.

The main contributions of our work contain the following.

- The arbitration process is lightweight. User only needs to install a program in the client instead of any additional hardware resources or manual interventions.
- We assign weights to voters according to their service levels. In this way, we could get the minimum weights to guarantee the whole system high available.
- We have improved the universality of the leader election approach to solve split brain so that it could keep working when there are only two separate partitions.

2. Our proposed Leader Election Approach

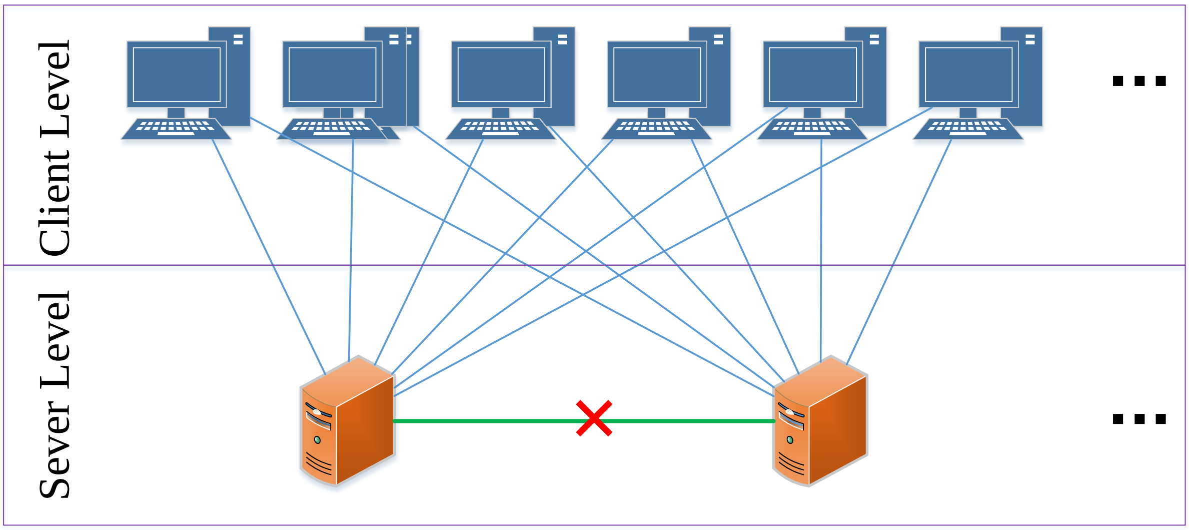

Figure 1. The scenario of split brain in a distributed system.

In order to solve the above mentioned problem, we propose a novel weight-based leader election approach for split brain in a distributed system. Figure 1 illustrates the scenario of split brain in a distributed system. In the traditional approaches, all voters have the same voting rights, meaning that all servers and additional detection nodes would immediately take part in the leader election when split brain occurs. However, only the existing server partitions could not elect a leader would the additional detection nodes be required for arbitration. Thus, in our proposed approach, voters are first assigned different roles according to their levels when split brain occurs. The server has three kinds of roles: speaker, senator and leader. On the contrary, the arbitration program in the client only has one role: senator. When a leader server could not receive the heartbeat signal in a given time threshold, the leader election begins and the role of servers becomes a speaker. In this case, speakers independently vote according to specific rules. Once a server receives votes from majority of speakers, its role becomes a leader and the election ends. To avoid an infinite loop, we set a threshold for the leader election. The role of servers becomes a senator and the arbitration program in the client is required if the threshold times out. Similarly, once a server receives votes from majority of senators, its role becomes a leader and the election ends. We also set a threshold for senators to start a new round of voting. Figure 2 illustrates the role change of voters in a leader election process.

In a large-scale distributed system, business process applications and databases are usually deployed across regions, which become quite sensitive to access delays. If the leader node randomly switches to a server in another area, it might lead to massive accesses to remote resources, greatly increasing response time. Thus, we assign different weights to corresponding voters, which could be flexibly modified according to resource deployment requirements. We design a two-stage process for leader election. The first stage is the unified lease time in the distributed system, while the second stage is decided by the weight, which means that the senator whose weight is larger would earlier conduct the self-election operation. To further quantitatively explain our proposed approach, we assume that there are m servers and n clients, where m>1. We define the variable Sweight$_i$ to represent the weight of server i and Cweight$_j$ to represent the weight of client j. Thus, the minimum weights to keep the system high available could be expressed as Equation (1).

$$Weightmin > \frac{1}{2}\left(\sum_{i=1}^{m} Sweighti + \sum_{j=1}^{n} Cweightj\right) \qquad (1)$$

In addition, there is a weight detection mechanism to ensure the priority of voter weights. The leader node broadcasts and detects all the voter nodes that could be connected. If the voter node with higher weight is found, the new node would actively initiate a role transfer to take over the leader role.

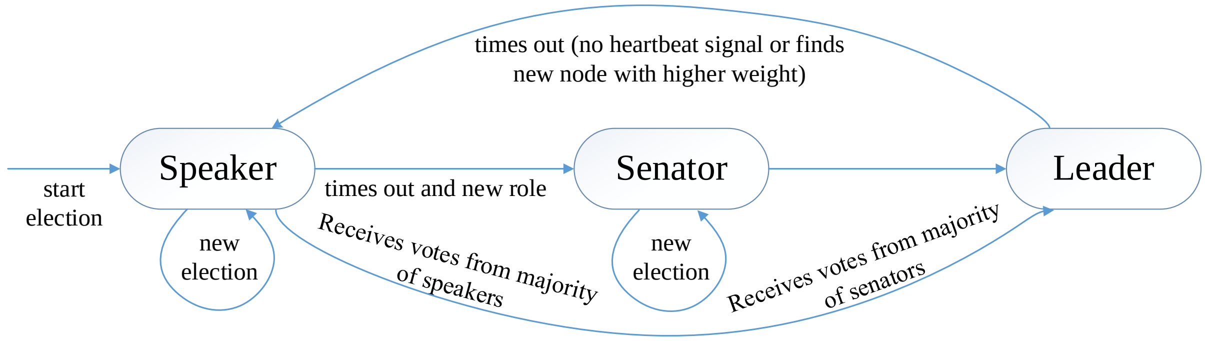
times out (no heartbeat signal or finds
new node with higher weight)
Speaker
times out and new role
Senator
Leader
start
election
new
election
Receives votes from majority
of speakers
new
election
Receives votes from majority
of senators

Figure 2. The role change of voters in a leader election process.

3. Performance Evaluation

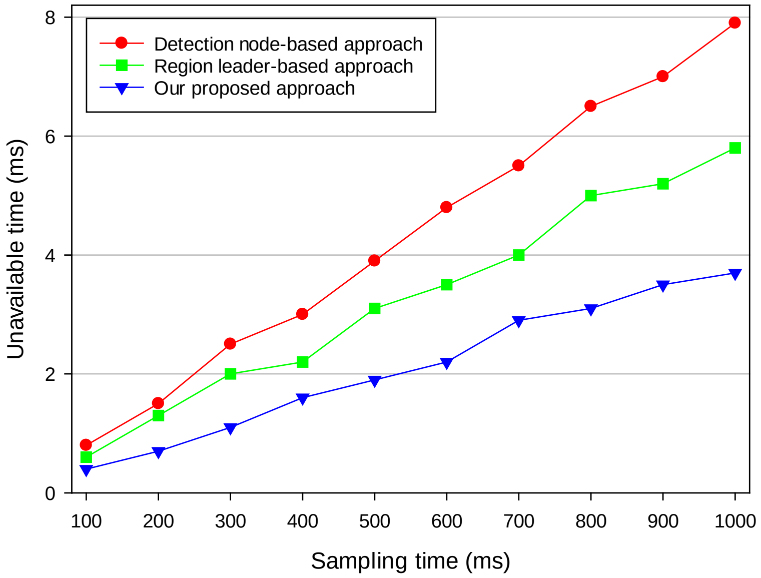

Figure 3. The comparison of unavailable time among different approaches.

To further quantitatively evaluate our proposed approach, we perform a series of quantitative experiments, which are performed on six PCs, each with 6GB of memory, 3.07 GHz CPUs. Furthermore, four PCs run as servers and two PCs run as clients. These servers form a distributed system and communicate with each other through heartbeat lines. In order to simulate the scenario of split brain, we randomly assign weights to PCs and cut off heartbeat lines among servers every 100 milliseconds. The sampling time is 1000 milliseconds and final results are illustrated as follows. Figure 3 illustrates that the unavailable time in our proposed approach is significantly lower than that in other two approaches. Figure 4 shows the average CPU utilization of arbitration program in the clients, which only takes about 2 percent of the whole CPU.

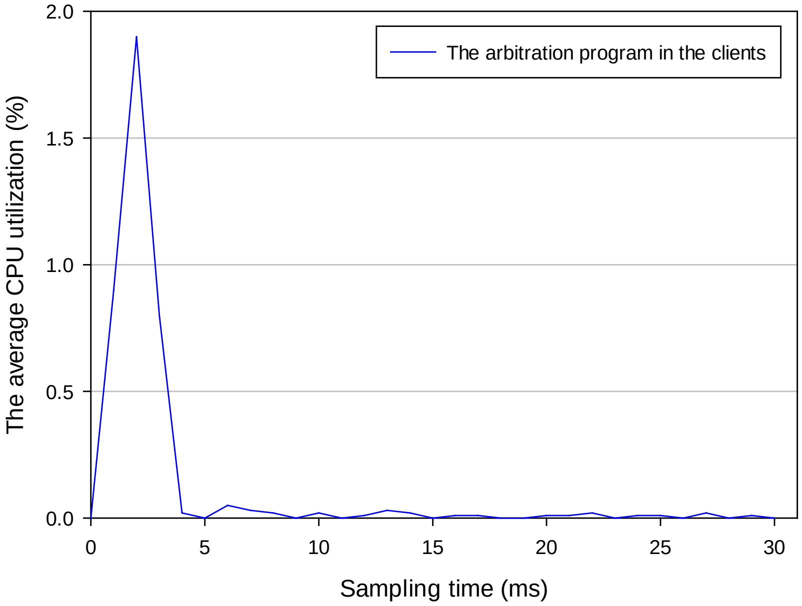

Figure 4. The CPU utilization in a leader election process.

4. Conclusions

In this paper, we present a novel weight-based leader election approach for split brain. To reach the goal, we initially install an arbitration program in the clients. Furthermore, we assign weights to voters and make them automatically perform the leader election. Finally, we validate our proposed approach to be valid through a series of quantitative experiments.

5. References

[1] C. A. Vissers, G. Scollo, M. Sinderen and E. Brinksma, "Specification styles in distributed systems design and verification", Theoretical Computer Science, vol. 89, pp. 179-206, 2017.

[2] M. Gerami, M. Xiao and J. Li, "Repair for Distributed Storage Systems with Packet Erasure Channels and Dedicated Nodes for Repair", IEEE Transactions on Communications, vol. 64, pp. 1367-1383, 2016.

[3] R. Zhang, P. Isola and A. A. Efros, "Split-Brain Autoencoders: Unsupervised Learning by Cross-Channel Prediction", In: IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 645-654, 2017.

[4] D. U. S. Rajkumar and R. Vayanaperumal, "A leader based intrusion detection system for preventing intruder in heterogeneous wireless sensor network", IEEE Bombay Section Symposium, 2015.

[5] C. E. Ren, Z. P. Shi and T. Du, "Distributed Observer-Based Leader-Following Consensus Control for Second-Order Stochastic Multi-Agent Systems", IEEE Access, vol. 6, pp. 20077-20084, 2018.
[6] A. Das, C. Zhou and A. Sen, "Region based fault-tolerant distributed file storage system design under budget constraint", in 6th International Workshop on Reliable Networks Design and Modeling, 2014.