Scholar Publishing Group
Distributed Processing System
https://doi.org/10.38007/DPS.2020.010207
ISSN 2790-0916 Vol. 1, Issue 2: 54-61

Enterprise Distributed System Based on Raft Algorithm

Garcia Clemente[*]

Univ Porto, Rua Campo Alegre Sn, P-4169007 Porto, Portugal

[*]*corresponding author*

Keywords: Raft Algorithm, Enterprise System, Distributed System, System Design

Abstract: With the development of the network from the initial simple form to the Internet era, especially when the development of several enterprises requires huge and complex data, in order to solve the problem of data consistency on different computing nodes, it is inseparable from the distributed Research on the key topic of data consistency in the field. The purpose of this paper is to study the analysis and design of enterprise distributed system based on Raft algorithm. The consistency and Raft algorithm of distributed storage system are introduced. In the network layer, the ROUTER-DEALER mode based on the ZMQ message queue technology is used to realize the message communication between each node of the distributed architecture. In the application layer, the idea based on the Raft consensus algorithm is used to realize the public business module. The allocation of the public business logic load balancing designates the public server, which ensures the consistency of the related interactive business logic data of the public server node, and can quickly switch and recover after the node stops working. In the data layer, the data cache function based on RedisCluster cluster technology is used to reduce the redundant operation of the database. The experimental results show that the fast data reading and writing ability of the enterprise distributed system is improved.

1. Introduction

In recent years, with the rapid development of network technology and the continuous growth of network data, people's vision for building a future intelligent life and the demand for high-availability and high-performance application services are increasing, making distributed systems occupy more and more in the development of social science and technology increasingly important position [1-2]. On the basis of the huge number of users, the number of user requests to be processed by these Internet applications per unit time and the amount of data generated during use are beyond the reach of any single-machine enterprise system. In order to solve this problem, distributed systems come into being. The distributed system splits the original single Internet application according to functions or services, and then deploys these split sub-functions or sub-services to multiple machines, so that many machines running the sub-services coordinate operation, Consistently provide services to the outside world, and the whole looks like a single system [3-4].

As the current mainstream consensus algorithm, Raft consensus algorithm can be used as the core algorithm of distributed architecture and private chain. The Raft algorithm cannot solve the problem of malicious node attacks. The improved algorithm of Tarhini A not only retains the efficient performance of the Raft algorithm, but also prevents attacks from malicious nodes, providing more options for future researchers to build private chains [4]. Pfaff M proposed a Raft blockchain consensus algorithm based on the credit model CRaft. In the stage of node credit evaluation, RBF-based support vector machine is used as anomaly detection method to build a node credit evaluation model. Then the Trusted Node List (TNL) mechanism is introduced to make the consensus stage in a trusted network environment. Finally, the ordinary nodes are synchronized to the consensus nodes to update the entire network blockchain. Experiments show that CRaft has better throughput and lower latency than the commonly used consortium chain consensus algorithm PBFT (Practical Byzantine Fault Tolerance) [5]. It is of practical significance to study the enterprise distributed system based on Raft algorithm [6-7].

Therefore, this paper studies the basic principles and usage scenarios of today's existing distributed consensus algorithms, and analyzes the advantages and disadvantages of the Raft algorithm. On the basis of the consistency of distributed storage system, an application layer design based on Raft consensus algorithm is proposed. A new solution is provided for solving the problem that system performance is limited by a single leader node in enterprise distributed system applications.

2. Research on Enterprise Distributed System Based on Raft Algorithm

2.1. Consistency of Enterprise Distributed Storage System

Enterprise distributed storage systems include multiple server nodes. Considering the existence of node failures and other problems, distributed storage systems usually use redundant storage to save multiple copies of a piece of data [8-9]. In this way, when a node fails, data can be read from the remaining replicas, ensuring data security. Due to the existence of multiple copies, how to ensure the consistency of data between copies has become a core issue in distributed systems. Consistency refers to the fact that multiple server nodes in a distributed system, under the protection of the consistency protocol, make their external states consistent. According to different consistency requirements, distributed system consistency can be divided into three categories: strong consistency, weak consistency and final consistency:

(1) Weak consistency: After a replica successfully writes a data A, the latest value of data A may or may not be read on other replicas. There is no guarantee how long after each replica the data will be consistent [10-11].

(2) Eventual consistency: After a replica successfully writes a data A, the latest value of data A may or may not be read on other replicas, but it is guaranteed to be eventually read after a certain time window. Eventual consistency can be seen as a special case of weak consistency [12].

(3) Strong consistency: After a piece of data A is successfully written, the latest value of data A can be read on other replicas immediately. Different distributed storage systems may adopt different consistency standards due to different application scenarios. Generally speaking, distributed storage systems will support strong consistency, but for performance considerations, you can choose to support eventual consistency [13].

2.2. Raft Algorithm

Raft algorithm is a distributed consensus algorithm. Raft algorithm is divided into multiple modules such as leader election, log replication, security, etc. In addition, there are additional modules such as snapshot and member change. Raft algorithm can also be said to be a strong leader. The so-called strong leader means that the cluster will introduce a leader node as the general control center of the cluster before providing normal services to the outside world. All large and small transactions in the cluster are determined and executed by the leader node, and the follower node just obeys Instructions for the leader node [14-15].

Assume that the cluster has received N customer requests within a period of time, there are K service nodes in the cluster system, the size of the confirmation message between nodes is Sack, the size of the request message for communication between nodes is Sreq, and the size of the message that the node responds to the client For Sclient, each node in the cluster has the same probability of receiving client requests. The message sent by the leader node in the Raft algorithm consists of the following parts (where the default node log is not missing, and the leader node does not need to perform a consistency check):

The size of the heartbeat request that the leader node is responsible for sending is:

$$(K-1)*S_{req} \tag{1}$$

The reply client request size is:

$$N*S_{\mathrm{client}} \tag{2}$$

The size of the log synchronization request sent by the leader node to other nodes in the cluster is:

$$N*(K-1)*S_{\mathrm{req}} \tag{3}$$

The leader node sends a log submission request to other nodes to confirm that the message size is:

$$N*(K-1)*S_{\mathrm{ack}} \tag{4}$$

3. Architecture Detailed Design of Enterprise Distributed System Based on Raft Algorithm

3.1. Network Layer Design Based on ZMQ Message Queue

This paper implements a topology. As shown in Figure 1, the router server uses the ROUTER mode, and forms a request-response mode together with each node server using the DEALER mode. The router server can eliminate the need for each node to establish a connection with each other. On the contrary, each node only establishes a connection with the router when it starts up. The router saves the addresses and other information of all nodes, and regularly sends heartbeat packets to monitor the service status of each node. When communication between server nodes is required, the requester (Requst) initiates a communication request to the router, and the router server receives the request efficiently, and changes the first frame of data to the address of the requester on the original data, and then according to the address of the target node. Addressing and message forwarding. After the target node receives the message, it returns to the router node in the same way. The router continues addressing and returns to the requester.

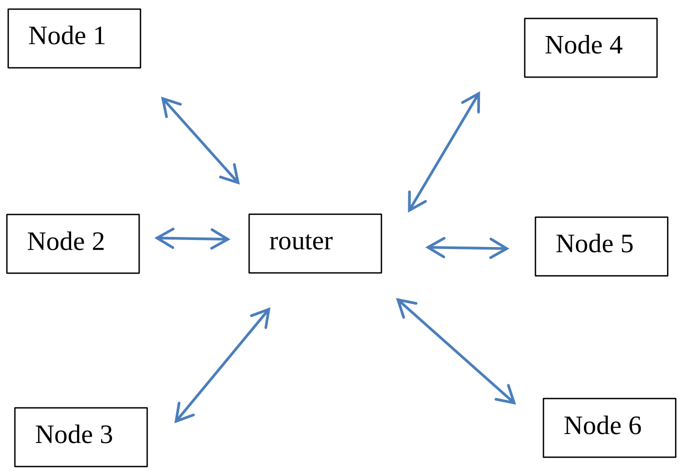

Figure 1. DEALER-ROUTER mode topology

3.2. Application Layer Design Based on Raft Consensus Algorithm

According to the working principle of the Raft consensus algorithm, this paper designs a Raft module for the public business framework in the application layer. The design idea of the Raft module is: deploy a group of public servers of the same type to carry public services, and select the most idle server as the main server to carry the business according to the consumption degree of the business module (other public servers vote to approve or reject, when the If the number exceeds half of the servers + 1, it can be the master server, otherwise it will time out). When the main server is down, the business Raft module on the server will automatically re-select a public server to continue hosting, and still select the public server with the most idle resource consumption. After the main server is selected, the Raft module will notify all servers to record which business module Which public server to run on (the public server ID registered by the RPC registered business module), and will call back the initialization function registered before the module and the initialization of the dependent business module.

3.3. Service Layer Design Based on Coroutine Technology

In this project, coroutine technology is used to design synchronous RPC - when you need to wait for the response result, suspend the logic currently being processed, and continue the current operation when the response result is returned.

Figure 2 shows the workflow of synchronous invocation. A coroutine object has three states: idle, suspended, and executing. When a coroutine object is created, the coroutine is in an idle state, waiting to be executed; the communication protocol, event, and timer trigger the coroutine, and the coroutine is in the running state at this time; after the execution of the coroutine, the coroutine is suspended, waiting for the next coroutine program or remote protocol trigger.

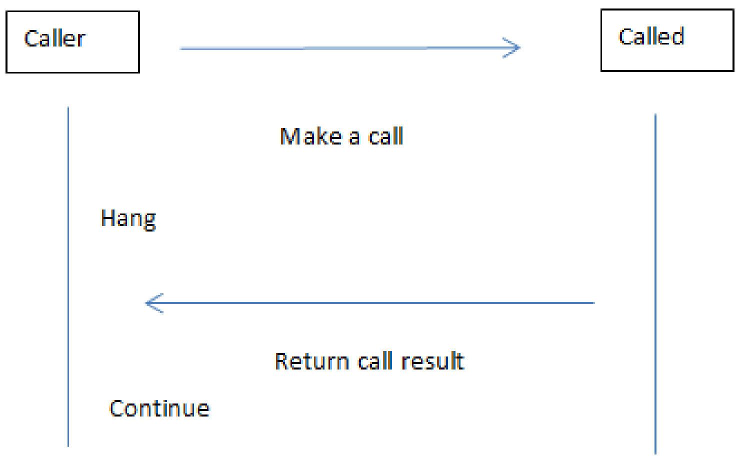

Figure 2. Synchronous RPC workflow

4. Experiment and Analysis of Enterprise Distributed System Based on Raft Algorithm

4.1. Experimental Environment

The experimental process needs to consider the impact of the number of cluster nodes on the performance of the Raft algorithm. In a real production environment, a Raft node is a physical machine. The number of nodes involved in the experiment is large. Due to the limitations of the experimental conditions, there is no Instead of deploying Raft nodes on multiple physical machines, a single-machine multi-copy deployment method is used, and multiple Raft nodes are deployed on multiple physical machines, which is not convenient for experimental testing. In the single-machine multi-instance mode, the IP address of each node is the same. In order to avoid conflicts, different port numbers are assigned to the network communication of different nodes. All Raft nodes share single-machine system resources, and the local operating system is responsible for resource allocation. The hardware and software configuration is shown in Table 1.

Table 1. The hardware and software configuration of the experimental machine

Software and hardware	Version/Specification
Processor	2.2 GHz Six cores Intel Core i7
Memory	16 GB 2400 MHz DDR4
Hard disk	APPLE SSD 500G AP0512M
Operating system	Mac OS Catalina 10.15.3
JDK	JDK 1.8.0_161

4.2. Leader Election Results

First start the Balanced_Raft cluster with 6 service nodes, and check the cluster leader node and term value at this time. Then stop one service node of the cluster at a time, so that the number of surviving nodes in the cluster does not reach the majority of the number of cluster nodes. After maintaining the cluster failure for a period of time, restore the cluster failure node to work, and check whether the cluster elects a new leader at this time. The term value of the leader node and the new leader node.

Analysis of the results: Before the cluster leader fails, node 1 is elected as the cluster leader node, and the Term value of this node is 1. When the cluster recovers from the failure, node 5 is elected as the new leader, and the term Term value is increased from 1 to 12. At the same time, the new leader node initiates heartbeat detection and continues to provide normal services to the cluster.

4.3. Log Replication Write Performance

In the experiment, the result of the experiment is obtained by calculating the average time spent writing 800 records in the serialization scenario and the multi-client scenario. The variable is the number of nodes in the cluster, and the number of clients in the multi-client scenario is For 5 and 10, the written data is a simple string key-value pair, and its character length does not exceed 10. Each group of experiments is repeated for 50 rounds, and the average writing time is calculated. The experimental results obtained are shown in Figure 3.

Table 2. Write time

Number of nodes	Serialize	5 clients	10 clients
2	513	201	185
4	625	332	306
6	758	462	459
8	889	501	496
10	1069	607	526

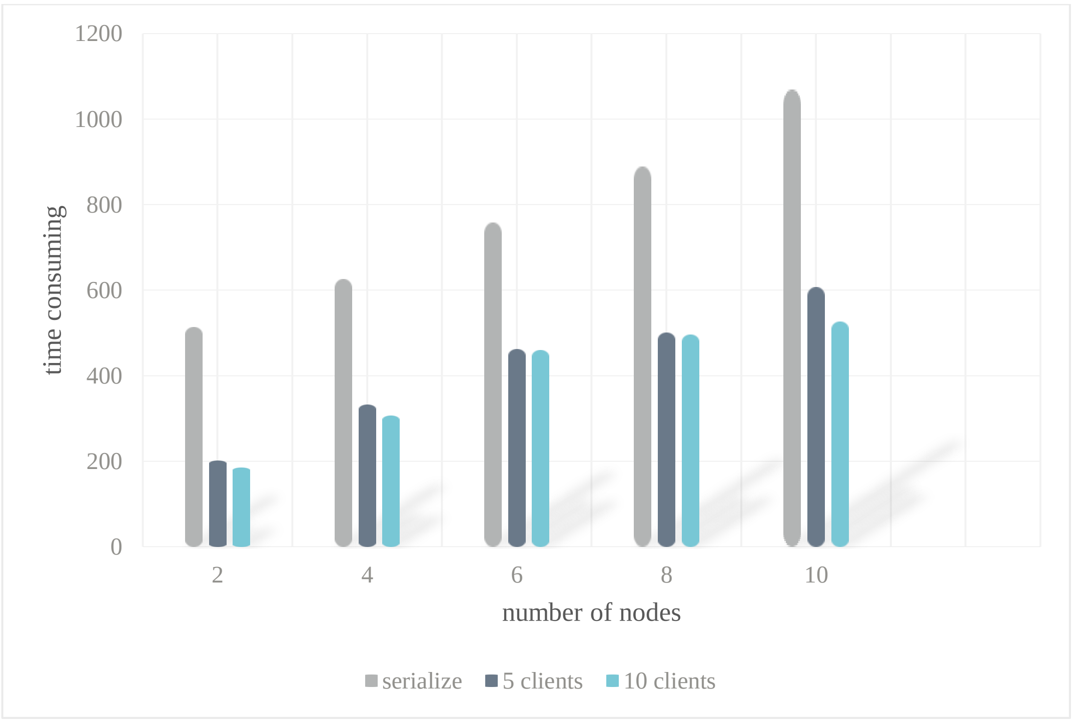

Figure 3. Data write performance comparison

According to the experimental results in Table 2, it can be seen that under different numbers of nodes, to complete the writing of 800 pieces of data, the multi-client writing scheme takes about half the time of the serial writing scheme, which means that in the same In a unit time, the multi-client concurrent write scheme can handle more client requests, so the performance is better.

5.Conclusion

Due to the limited cache application capacity of enterprise single-computer systems, it is increasingly unable to undertake large-scale cache storage tasks. Therefore, in current practice, cache data is mostly stored in multiple computers through specific methods. Since multiple computers need to be connected to each other and backed up with each other, a cache system composed of multiple computers is called a cache cluster. In order to properly manage and schedule cache clusters, people have further researched and implemented distributed cache systems. What this article describes is an enterprise distributed system. Based on the research content of this paper, the next research on this topic can be carried out from the following two aspects: the next step is to improve their applicability in enterprise distributed systems by combining Raft algorithm and BLS signature. The next step can be to segment the overly long log list according to certain rules. When sequencing, you can first determine the segment, and then perform log sequencing in the segment by linear detection sequencing. This segmented The linear detection sequencing method is beneficial to speed up the sequencing speed of the cluster log and improve the performance of the algorithm.

Funding

This article is not supported by any foundation.

Data Availability

Data sharing is not applicable to this article as no new data were created or analysed in this study.

Conflict of Interest

The author states that this article has no conflict of interest.

References

[1] Golubev Y F, Koryanov V V. Shipping Cargo on a Raft by an Insectomorphic Robot. Journal of computer & systems sciences international, 2018, 57(5):813-821.

[2] Werner G, Hanka L. Optimization of Artificial Neural Networks with Genetic Algorithms for Biometric Pattern Recognition. Land Forces Academy Review, 2019, 24(3):256-264. https://doi.org/10.2478/raft-2019-0031

[3] Basinya E A. Distributed system of collecting, processing and analysis of security information events of the enterprise network infrastructure. Bezopasnost Informacionnyh Tehnology, 2018, 25(4):43-52. https://doi.org/10.26583/bit.2018.4.04

[4] Tarhini A, Balozain P, Srour F J. Emergency management system design for accurate data: a cognitive analytics management approach. Journal of Enterprise Information Management,

2020, 34(2):697-717.

[5] Pfaff M, Krcmar H. A web-based system architecture for ontology-based data integration in the domain of IT benchmarking. Enterprise Information Systems, 2018, 12(1-5):236-258.

[6] Kisielnicki J, Markowski M M. Real Time Enterprise as a Platform of Support Management Systems. Foundations of Management, 2020, 13(1):7-20.

[7] Venkatesh M, Raj V M, Suresh Y. Mining massive online location-based services from user activity using best first gradient boosted distributed decision tree. International journal of enterprise network management, 2020, 11(1):3-13. https://doi.org/10.1504/IJENM.2020.103880

[8] Plonskiy V Y, Chistyakova T B. System Of Dynamic Redistribution Of Warehouse Resources Of Industrial Enterprise. Vestnik Of Astrakhan State Technical University Series Management Computer Science And Informatics, 2020, 2020(4):18-28.

[9] Ghannam A N, Mansour H, El-Bastawissy A, et al. Perspectives of an Enterprise Integration Plug-in System Architecture for Networked Manufacturing Systems. Engineering, Technology and Applied Science Research, 2019, 9(2):4075-4078. https://doi.org/10.48084/etasr.2739

[10] Basinya E A. Distributed system of collecting, processing and analysis of security information events of the enterprise network infrastructure. Bezopasnost Informacionnyh Tehnology, 2018, 25(4):43-52. https://doi.org/10.26583/bit.2018.4.04

[11] Oleynikov A. Conceptual Structure Of The Management System In The Territorally Distributed Enterprise Of The Shipbuilding Industry. Vestnik Of Astrakhan State Technical University Series Management Computer Science And Informatics, 2020, 2020(3):26-33.

[12] Adam A. Research On Multi Agent Distributed Application System Based On Www Platform. Acta Electronica Malaysia, 2020, 4(2):31-34.

[13] M Kantšukov, Sander P. A lesson in valuation from Estonia: The difference between the fundamental value of equity under distributed and traditional profit taxation systems. Verslas Teorija Ir Praktika, 2018, 19(1):146-156. https://doi.org/10.3846/btp.2018.15

[14] Bocciarelli P, D'Ambrogio A, Falcone A, et al. A model-driven approach to enable the simulation of complex systems on distributed architectures. Simulation, 2019, 95(12):1185-1211.

[15] Shah B, Khanzode V. Designing a lean storage allocation policy for non-uniform unit loads in a forward-reserve model. Journal of Enterprise Information Management, 2018, 31(1):112-145. https://doi.org/10.1108/JEIM-01-2017-0018