Design and Research of Multi-user Distributed Configuration Management Based on Zookeeper

Ming Zhang[1](✉), Zhaojian Shen[2], Bin Yin[1], Li Cui[2], and Fan Xu[1]

[1] Beijing Spatial Information Transmission Centerion, Beijing 100094, China
2663931027@qq.com

[2] Beijing Institute of Remote Sensing Information, Beijing 100094, China

Abstract. The aerospace system can be divided into multiple types of users according to the division of functions. Different users maintain a set of their own configuration files and store them on different servers. Since there may be logical crossover between services between users, there are situations where different users maintain the same set of configuration files. For the problem of fast synchronization of the same configuration file among multiple servers and simultaneous modification of the same configuration file by multiple users, using the Zookeeper distributed synchronization mechanism and the multi-user dismantling and marking method of configuration files, we propose a multi-user distributed configuration management scheme based on Zookeeper. Store the content of the configuration file in Zookeeper according to the node, use the Zookeeper publish/subscribe mechanism to synchronize the configuration file between the configuration management center and each user application center, realize version management through Bitkeeper technology, and use B/S architecture to realize daily configuration file maintenance. The synchronization performance is tested by synchronizing time, and the test results show that the solution optimizes the configuration file management process and improves the efficiency of configuration file management and maintenance.

Keywords: Multi-user · Zookeeper · Node · Update synchronization

1 Introduction

With the continuous development of aerospace technology, its benefits in agricultural disaster prevention, weather forecasting, geological exploration, and military applications have become more and more significant [1–3]. Space system is a large and complex system, including spacecraft, space launch system and ground application system [4]. It involves many kinds of users, and the applications of different users are generally distributed on different servers. When the program starts and runs, it needs to call the corresponding configuration files. As a common way of aerospace system maintenance, configuration files are widely used in aerospace systems and play a very important role in the entire aerospace system software life cycle [5, 6]. There are also the following two problems in the use of aerospace system configuration files:

1) Because the configuration files are deployed on different servers, if a configuration file is modified, it needs to be sent to each server by file transfer, and then the configuration files can be updated synchronously by logging in to different servers, which leads to long synchronization time, low efficiency and easy errors in the synchronization process.
2) There is a high degree of business coupling among users in the aerospace system, and different users work together, so different users operate the same configuration file at the same time. In the existing mode, the same configuration file is usually operated sequentially by setting restrictions by user rights, and then the configuration file is synchronously updated by file sending and server login. There are also some problems in this mode, such as low modification efficiency, long time consumption and error-prone synchronization process.

Zookeeper is an application service framework with distributed and open source architecture. It has the functions of version management, distributed synchronization, configuration maintenance. Its high availability, real-time, atomicity and other characteristics can also ensure the consistency of data and the reliability of the system [7–10] In order to effectively solve the above two problems, this paper proposes a multi-user distributed configuration management scheme based on Zookeeper. It realizes the synchronous update of configuration files and the operation of the same configuration file by different users at the same time, and improves the management efficiency of configuration files.

2 Zookeeper Introduction

Zookeeper is a file management system based on distributed storage. It builds the corresponding data structure model by establishing a tree-like directory. Each node in the tree-like structure is called a Znode node, which also has its own child nodes and leaf nodes [11–13]. Each node will store several KB of data, and maintain a set of its own data structure, which mainly includes information such as time stamp and version number to identify the changes of nodes. The atomicity of a node determines that its reading and writing operations are to read, write and replace all the contents of the node.

Zookeeper operations mainly include node creation, node deletion, node setting, node viewing, node obtaining, etc. Where create means creating a node, delete means deleting a node, setdata means setting a node, getdata means getting a node, exists means checking whether a node already exists, and getChildren means getting a child node. Zookeeper is generally used with Watcher to monitor nodes. When the node information changes (such as adding, deleting and modifying), it will be reported in time. Operations such as getdata、exists、getChildren can register Watcher on nodes. While create, delete, setdata and other operations can trigger Watcher set on nodes.

The high availability of Zookeeper is mainly ensured by the election mechanism [14, 15]. Its roles mainly include leaders, learners and customers, among which learners are divided into followers and observers, and leaders are responsible for initiating and deciding decision-making voting; Followers receive customer requests and vote in the whole election process; Observers are not responsible for voting, but only for synchronizing the status of leaders. Usually, once a server gets more than half of the votes, it will become the new leader. We assume that there are 7 identical servers, labeled ID1–ID7, which will become the new leader only when the number of votes exceeds half, that is, 4 votes. ID1 will vote for itself when it starts first, and ask other servers. Because the number of votes is less than 4, it is not elected; After ID2 is started, it will vote for itself first, and then ask other servers. ID1 replies to ID2 with approval because it has been started. At this time, ID2 won 2 votes and was not elected; By analogy, when ID4 gets 4 votes after it is started, ID4 is elected; Since ID4 has been elected, ID5 ~ ID7 can only be followers after they are started. Once the leader quits or doesn't respond, this campaign mode will be resumed until a new leader is elected.

3 Implementation Scheme

At present, for the generation and management of configuration files based on web, configuration maintenance personnel usually use B/S mode to maintain their configuration management center, and can add, delete, modify and view the configuration files accordingly. The configuration center stores the configuration files of n users (user1, user 2, …, userN). According to different functions, each user is configured by multiple application nodes, for configuration application nodes that need to be modified frequently, join Zookeeper node, which is represented by Znode1, Znode2, …, ZnodeN, and these nodes are registered with watcher events in the client. One configuration file corresponds to one Znode node as for the configuration files managed by users alone. One configuration file corresponds to multiple Znode child nodes for the configuration files jointly managed by users.

When the application starts, the configuration file will be called, that is, the configuration application node data will be obtained. When the data content of the configuration application node monitored by watch changes, the Zookeeper client will obtain the change through the watch event. At the same time, for software maintainers, they want to record and manage the details of different versions and changes during the development process. Bitkeeper is a commonly used version management tool based on linux system at present. Compared with SVN technology, Bitkeeper will extract all the changed configuration files without considering which files have been specifically modified. In design, it will consider displaying key information such as change times, change dates and users. The interaction between configuration management center and each application is shown in the following Fig. 1:

(1) Configure the application node design
Configuration files of different user middleware are stored in each node of Zookeeper. For the configuration files managed by users alone, the nth user node is represented as Znode1_userN, Znode2_user2, …ZnodeN_userN. The design of configuration file in Zookeeper server is shown in Fig. 2, where < > represents directory. The directory at the top level is represented as <middle_ware>, and the directory at the next level represents different user application apps, such as <user1_name>, <user2_name> , <usern_name>, every user middleware application app contains two subdirectories, which are respectively represented as, <conf>

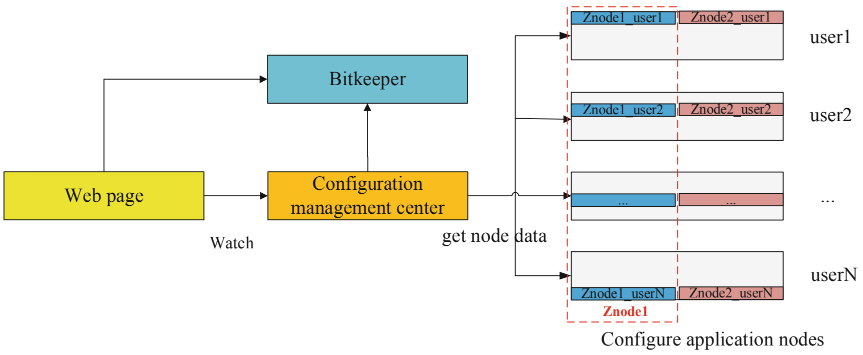

Fig. 1. Schematic diagram of interaction between configuration management center and various applications

and <version>. Where <conf> represents the identified frequently modified configuration file content, which is the final node, and its content is only the final value of user1 management configuration, including interface information interfaces, user id, etc. <version> contains version history history and current version current.

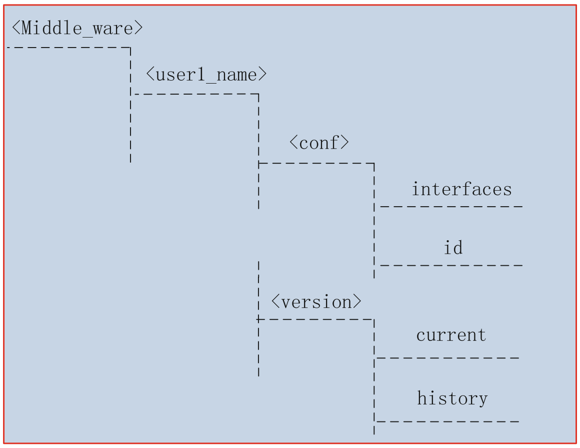

Fig. 2. Configuration of application node structure-individual management by users

For the configuration files jointly managed by users, the configuration file multi-user disassembly marking method is adopted. Assuming that the number of users managed by the same profile is N, the profile node consists of child nodes of Znode1_user1, Znode1_user2, …Znode1_userN. In the design of Zookeeper server, as shown in Fig. 3, the top-level directory remains unchanged, and the next-level directory is represented by <common_name>, which means that the configuration file is managed by different users. Its subdirectories contain different users <user>, which are divided into <label> according to the configuration content. The coarse granularity of label setting is determined by the logic of the actual configuration file, which is generally determined when the configuration file is generated. Each child node of the configuration file is stored in Zookeeper, and its path is represented as /Middle_ware/common_name/user_n/label_n.

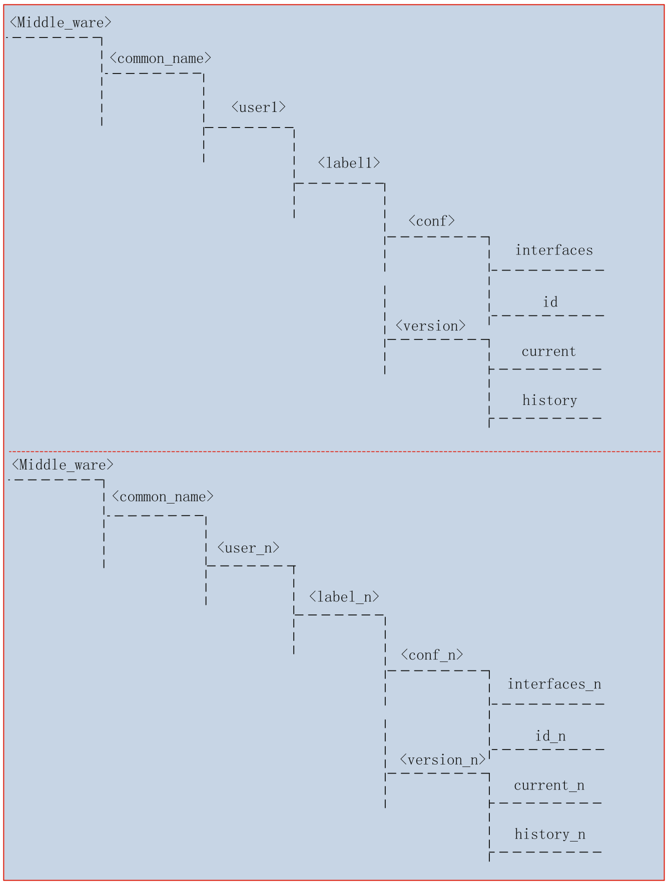

Fig. 3. Configuration Application Node Structure-User Common Management

(2) Implementation of profile update synchronization
Each user application needs to load the local configuration file before starting, and the configuration file can be started normally only after the initialization parameter verification, and each time the configuration file is updated, it needs to be synchronized on each user server. In order to automatically obtain configuration files from Zookeeper, the design takes into account the need to read the data of Znode nodes

before the application starts. The watch option is set in all reading operations of Zookeeper, such as getdata, exists and getchildren, and registered on the Znode node. As long as the data content of the node changes, the Zookeeper client will sense the event change at the first time, thus realizing the automatic acquisition of configuration files.

When the data content of Zookeeper node changes, the callback function of watch this Znode node will be triggered. This function re-parses the contents of the configuration file and refreshes the data values stored in the memory, thus realizing the real-time update of the configuration file. In order to prevent abnormal events such as downtime of Zookeeper server, the updated configuration file is generally permanently written into the local storage file.

Using Zookeeper publish/subscribe mechanism to realize the synchronization of configuration files between configuration management center and user application centers. The publisher publishes the data on the nodes or sub-nodes of Zookeeper, and each user application center acts as a subscriber to view and subscribe the published data. Subscribers register the node they need with the server, and if the data of this node changes, they will send the watch event to the client. After receiving the event change notification message, subscribers will get the latest data from the server. To ensure the consistency of configuration, Zookeeper needs to support real-time synchronization of distributed configuration application nodes on each server. At the same time, because of the single view feature of Zookeeper, that is, no matter which server subscribers connect from, they will get the same configuration information, thus solving the single point failure problem of Zookeeper server. Once a server fails, other servers in the cluster can still guarantee the reliability of the system through the campaign mode.

4 Test

In order to verify the reliability and synchronization time of the scheme, the test environment deployed the same Zookeeper service on 8 servers with the same performance in a cluster mode, selected 8 users, established a connection with Zookeeper instance through user middleware, and then created a configuration application node in the service cluster, and set the node as listening state. When the maintainer needs to modify the content of the application file configured by the user, he only needs to select the corresponding configuration file through the Web page and modify the content of the configuration file accordingly. It can be seen that the content of the corresponding configuration application node in Zookeeper has been modified, while the latest modification time of the configuration file of this user node has been updated to the current time, and the content is consistent with the modified one.

(1) User-managed profile separately
Assuming that users manage their own configuration files separately, and each user manages three configuration files, the new synchronization method is almost completed in seconds, while the traditional file transfer to each server takes nearly 5 min to complete a synchronization.

(2) Users jointly manage the same profile
In this mode, eight users log on the Web page at the same time to operate the same configuration file. It can be seen that the content of the corresponding configuration application node in Zookeeper has been modified at the same time, and the latest modification time of monitoring this node is the time when the last user finished the configuration, and the content is consistent with that after modification. The synchronization time is also seconds, while in the traditional mode, each user needs to operate the same file in turn, and then transmit it to each server through the file. It takes nearly 20 min to complete a synchronization.

5 Summary

With the rapid increase of the number of servers, in complex aerospace system applications, the traditional way of logging in different servers by maintenance personnel to update configuration files synchronously is inefficient and prone to errors. At the same time, the coupling degree between the services of each user is getting higher and higher, so it is difficult to ensure that each user configures and manages a configuration file separately. This paper presents a distributed configuration management scheme for multi-users based on Zookeeper, the content of configuration files are stored in Zookeeper according to nodes, and the publishing/subscribing mechanism of Zookeeper is used. The synchronization of configuration files between configuration management center and each user application center is realized, version management is realized by Bitkeeper technology, and daily configuration file maintenance is realized by B/S architecture. Its server campaign mode improves the usability of the system. This scheme not only ensures the consistency of user profiles, but also ensures that there is no need to restart the service during operation. Under the condition that the configuration files managed by users alone and the same configuration files managed by users together, the second-level synchronous update can be realized, which greatly improves the efficiency of operation and maintenance.

References

1. Zhang, K.: Space technology changes life. Sino-Foreign Exch. **5**, 350 (2019)
2. Xie, Y., Qin, Z., Huang, H.: Review and prospect of military aerospace technology. Flight Missile 15–19 (2002). 10.3969/J. ISSN.1009-1319.2002.11.009
3. Chen, J., et al.: Inventory of foreign aerospace technology development in 2019. Natl. Defense Sci. Technol. Ind. **12**, 19–21 (2019)
4. Feng, K., Huang, J., Huang, Z.: Development and application of space system safety risk analysis technology. Sci. Technol. Inform. **22**, 393–395 (2010). https://doi.org/10.3969/J.ISSN.1001-9960.2010.22.344
5. Meng, R.: Reliability and security of aerospace system software. Qual. Reliab. **4**, 10–13 (1992)
6. Xing, Z., et al.: Architecture design of software defined spacecraft system. Spacecraft Eng. **30**(5), 1–8 (2021). https://doi.org/10.3969/J.ISSN.1673-8748.2021.05.001
7. Chen, D., Chang, G.: Development and application of ZooKeeper. Comput. Programm. Skills Maintenance **2017**(21), 35–36, 42. https://doi.org/10.3969/J.ISSN.1006-4052.2017.21.014

8. Wang, Z.: Talk about zookeeper. Comput. Nerd **15**, 76 (2018). https://doi.org/10.3969/j.issn.1672-528X.2018.15.075
9. Ren, A., Feng, J., Zhu, Y.: Research and implementation of database synchronization based on Zookeeper service. Inform. Syst. Eng. **7**, 116–117 (2020). https://doi.org/10.3969/J.ISSN.1001-2362.2020.07.053
10. Feng, S., et al.: Research on distributed ICE middleware based on Zookeeper. Comput. Syst. Appl. **27**(12), 222–226 (2018). https://doi.org/10.15888/j.cnki.CsA.006693
11. Miao, F., Yan, Z., Dai, L.: Design and implementation of configuration management center based on Zookeeper. Railway Comput. Appl. **27**(10), 26–29 (2018). https://doi.org/10.3969/j.issn.1005-8451.2018.10.006
12. Li, R., Ben, Y.: High availability of services through tomcat manager and Zookeeper. Commun. Power Technol. **35**(9), 177–178 (2018). https://doi.org/10.19399/j.cnki.tpt.2018.09.070
13. He, H., Wang, Y., Shi, L.: Research on data synchronization based on ZooKeeper service in distributed environment. Inform. Netw. Secur. **9**, 227–230 (2015). https://doi.org/10.3969/j.issn.1671-1122.2015.09.050
14. Deng, C.: Research and Application of Distributed Framework Based on ZooKeeper. Three Gorges University, Hubei (2017). https://doi.org/10.7666/D.D01233894
15. Nanjing Linghua Microelectronics Technology Co., Ltd.: Distributed data exchange method based on ZooKeeper: CN202210232173.0. 6 May 2022