Multithreading-Based design and implementation of distributed file transfer system

Xianhui Li[1, 2, *], Zengrong Xu[1, 2, a] and Changping Gong[1, 2, b]

[1]Nari Group Corporation/State Grid Electric Power Research Institute, Nanjing 2111 06, China.
[2]China Realtime Database Co, Ltd., Nanjing 211106, China.

*Corresponding author: lixianhui@sgepri.sgcc.com.cn,
[a]xuzengrong@sgepri.sgcc.com.cn, [b]gongchangping@sgepri.sgcc.com.cn

Abstract. File is an important form of data exchange in application system, but It is restricted by the file size and the network environment between application systems, lacks of effective technical solutions for efficient and stable transmission of files, we consider the design principles of loose coupling, scalability and high availability, design a distributed file transfer system based on multithreading; The scheme designs the module of API, Broker, NameServer, contacts moudles by the way of Socket connection, forms a complete file transmission channel in the application system, and provides convenient and abundant file transfer process information to help application system know file transfer process, at the same time, adopts key technologies such as multithreading, distributed architecture, breakpoint renewal and so on, guarantees file transmission efficiency, satisfies the needs of file exchange between application systems better.

1. Introduction

One of the characteristics of the network is to realize information sharing, File transfer is one of the very important contents of information sharing, It realizes the interaction of data and completes the goal of data exchange. As more and more application systems interact with each other, the file size for interaction between application systems is increasing day by day, and the requirements for file transfer efficiency and stability between interactive service systems are increasing. There is a need for a secure, efficient, and reliable file transfer solution between application systems. How to efficiently and stably transfer large files to a designated application system becomes a key issue in the file transfer system.

Most of the existing technical solutions for file transfer are point-to-point transmission. File transfer between application systems is required in a through network environment. However, in actual situations, many application systems will use hardware devices such as a gatekeeper to cut off the network environment to a non-through-through network based on security reasons. Therefore, the common point-to-point transmission scheme is not applicable to non-through networks. Application systems need a set of file transmission solutions what can break the non-through network environment barriers.

This paper adoptses a distributed architecture to ensure the stability of the file transfer system, and uses multithreading technology and forwards file stream to ensure the efficiency of file transfer in non-through network environments, combineds with common techniques for file transfer, such as resuming breakpoints, retransmissions, and so on. Finally, a set of efficient and stable multi-threaded distributed file transfer system design solutions have been formed. The paper was funded by the Science and Technology Project of Nari Group Corporation. (SG-UEP And Its Installation)

2. Overall Architecture Design

2.1. Design Idea

File transmission system is designed to achieve efficient and stable transmission of files between application systems. As a middleware, file transmission system can be deployed in a single machine or multi machine (according to the actual requirements of file transmission). Single machine deployment provides file transmission services for multiple application systems in horizontal direction. Multiple machines are deployed between the upper and lower levels of the multiple application systems, through the network, the communication network is formed to provide the horizontal and vertical file transfer service for the application system.

File transmission components should solve the data island problem of application system, let application system realize data sharing and data centralized management. It is mainly used for data integration, data migration, data disaster and so on in large and medium enterprises.

2.2. System Overall Architecture

The specific architecture is shown as shown in the diagram below:

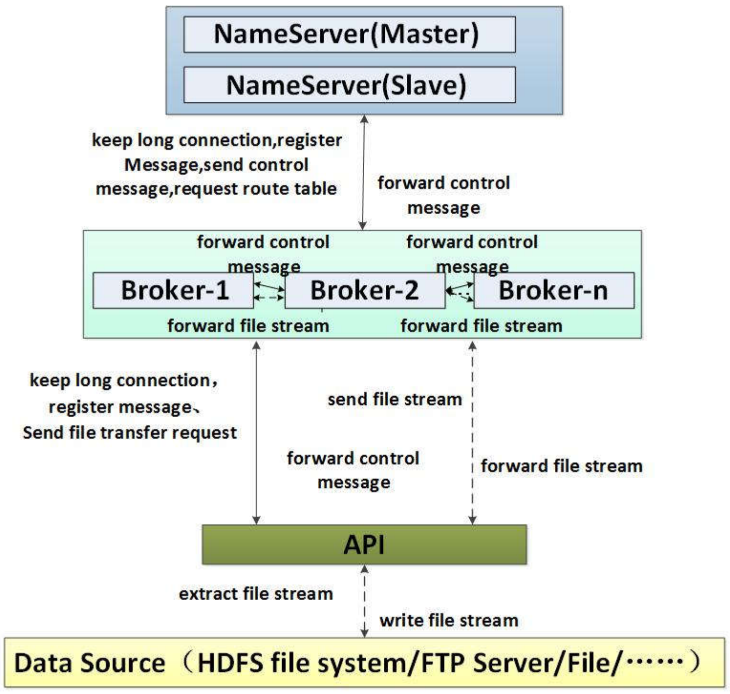

Figure 1. Overall architecture

As shown in Figure 1, the overall structure is divided into three main modules, namely:

API: responsible for extracting files from application system data sources, sending and receiving file operations, and communicating with Broker modules.

Broker: responsible for information control and file stream transfer operation. Routing refers to the network process [1] that determines the end to end path from a source to a destination. The Broker module is responsible for building the routing channel for file transfer.

Name Server: responsible for file transfer process control information analysis and file transfer process management.

3. File Transmission System Module Design

3.1. Broker Module

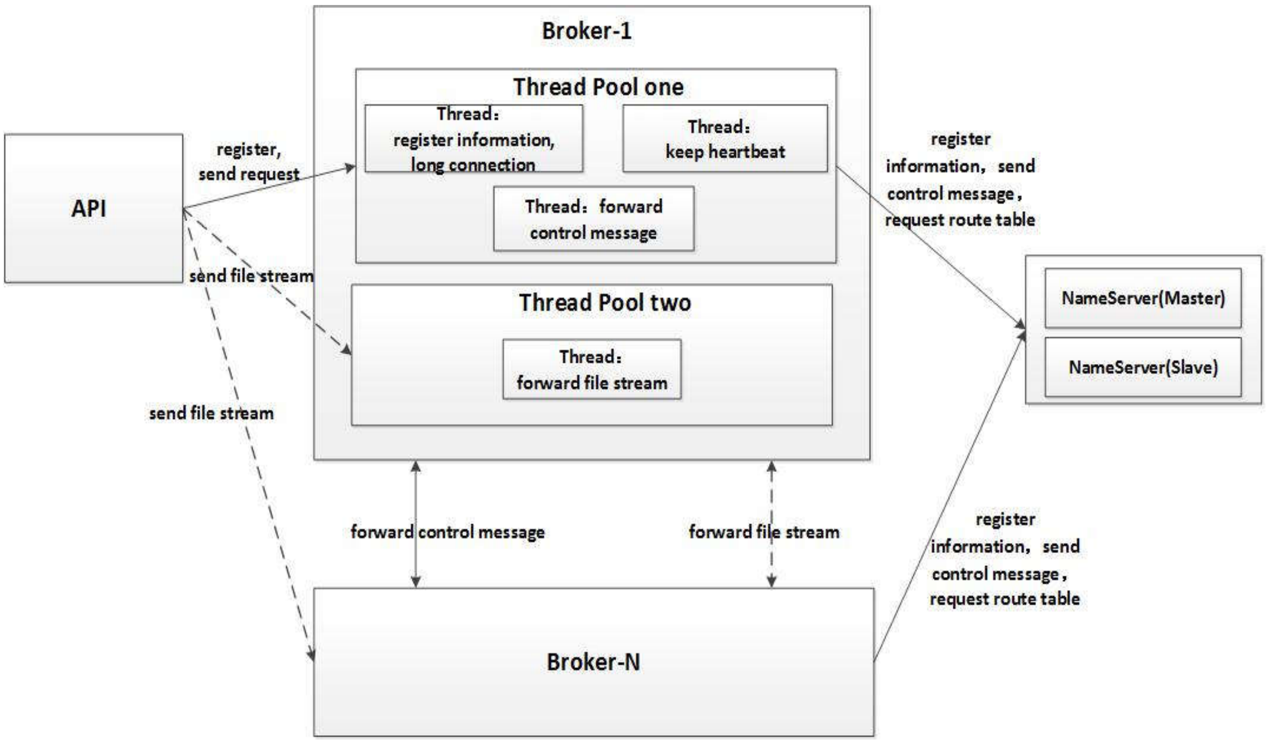
Broker-1
Thread Pool one
Thread: register information, long connection
Thread: keep heartbeat
Thread: forward control message
Thread Pool two
Thread: forward file stream
API
register, send request
send file stream
send file stream
register information, send control message, request route table
NameServer(Master)
NameServer(Slave)
forward control message
forward file stream
register information, send control message, request route table
Broker-N

Figure 2. Broker module schematic

As is shown above, Broker connects with API and Broker. The inside of Broker is divided into two thread pools: threads in thread pool one are responsible for managing registered connections, heartbeat retention, and forwarding control information; threads in thread pool two are responsible for forwarding file streams. The vertical transmission between Broker and Broker is mainly the forwarding control information and the file stream transmitted by forwarding API, and the load balancing mechanism between Broker nodes is mainly provided between them in the tansversal data transmission.

The main technology of Broker module is Socket programming and thread pool.

3.2. NameServer Module

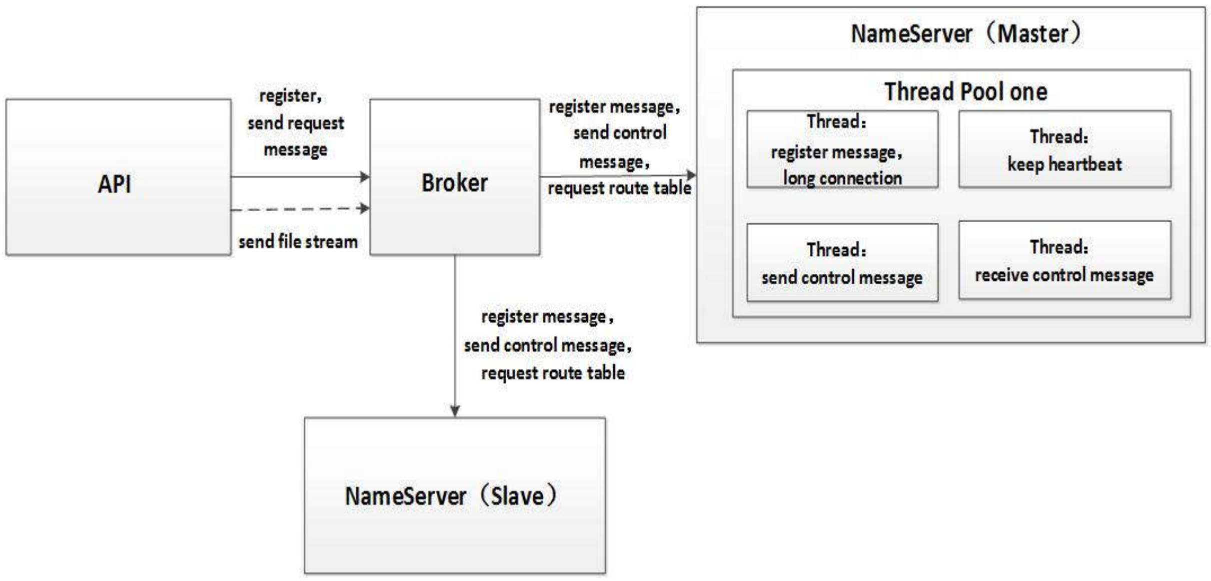
NameServer（Master）
Thread Pool one
Thread: register message, long connection
Thread: keep heartbeat
Thread: send control message
Thread: receive control message
API
register, send request message
send file stream
Broker
register message, send control message, request route table
register message, send control message, request route table
NameServer（Slave）

Figure 3. Name Server module schematic

As is shown above, Name Server connects with broker. There is a thread pool inside the Name Server: the thread in the thread pool is responsible for managing registration information (Routing Table), heartbeat maintenance, and sending and receiving control information. The main technology of Name Server module is Socket programming, timing frame Quartz and thread pool.

Because Name Server is very important in architecture, Name Server node needs to make Name Server node strategy design to prevent single point failure. Moreover, Name Server can configure the load balancing strategy for Broker, which can set switches and weights.

4. Key Technologies

4.1. File Stream Forwarding

In order to solve the problem of file transfer between application systems in a non-through network environment, the file transfer system uses the file stream forwarding function to implement cross-node file transfer.

The main realization of ideas: An indirect path is established through multiple Broker nodes, and the file is transmitted in the form of a file stream among multiple Broker nodes during the file transfer process, and finally reaches the target end, so that the non-through network file transmission is realized.

4.2. Broken-point Continuingly-transferring

For the file transfer process, for example, if the connection is interrupted, resulting in the file transfer is not completed, in general, re-transmission of files is a more common practice, but the re-transmission may occur when the file transfer is interrupted again. Therefore, the file transfer interrupt retransmission of the file is not efficient and practical. It is necessary to implement the broken-point continuingly-transferring function to reduce the impact of file transfer interruption and improve the file transfer efficiency.

The main idea is to record the progress of the file transfer, start the broken-point continuingly-transferring function, and then send the file from the file transfer progress position to improve the file transfer efficiency.

4.3. Error Retransmission
For the file transfer process, the file on the target side may be damaged after the file transfer is completed. If the file on the target side is found to be damaged through file verification, the source will be notified to resend the file so as not to affect the normal use of the file on the target side.

The main realization of ideas: When the file is sent, MD5 digest is performed on the source file, and this information is sent to the target. After the file is transmitted on the target side, a file is generated. MD5 digest is also performed on this file. When two digests are compared, if the digest information is consistent, the file transfer is completed successfully. If the digest file is not consistent, the file transfer is not successful and an error retransmission operation is required.

4.4. Distributed Architecture And Multi-thread Design
In order to solve the problem of high efficiency and stability in the transmission of large files between application systems, when sending large files, the commonly used strategy is to transfer the files in blocks, and it is necessary to divide the files in the API module. In order to ensure the efficiency of file transfer, it is necessary to synchronize the files after the block by the multi-thread design to improve the efficiency of file transmission. Multi-thread programs extend the concept of multitasking at a lower level: a program can perform multiple tasks. [2]

In the file transfer system scheme designed in this paper, the Broker nodes in the file transfer system are all connected to the NameServer. In order to ensure the load balancing of the Broker nodes in the file transfer process, the overall design adopts a distributed architecture. The file transfer route is assigned by the NameServer. The NameServer decomposes a task into multiple Broker nodes to complete the process. It can also ensure that a single Broker node fails and does not affect the file transfer process. In the face of hardware failures, distributed systems provide high availability. The availability of the system is a measure of the proportion of time available to the system.[3] The application of distributed architecture ensures the efficient and stable file transfer system.

5. Physical Deployment

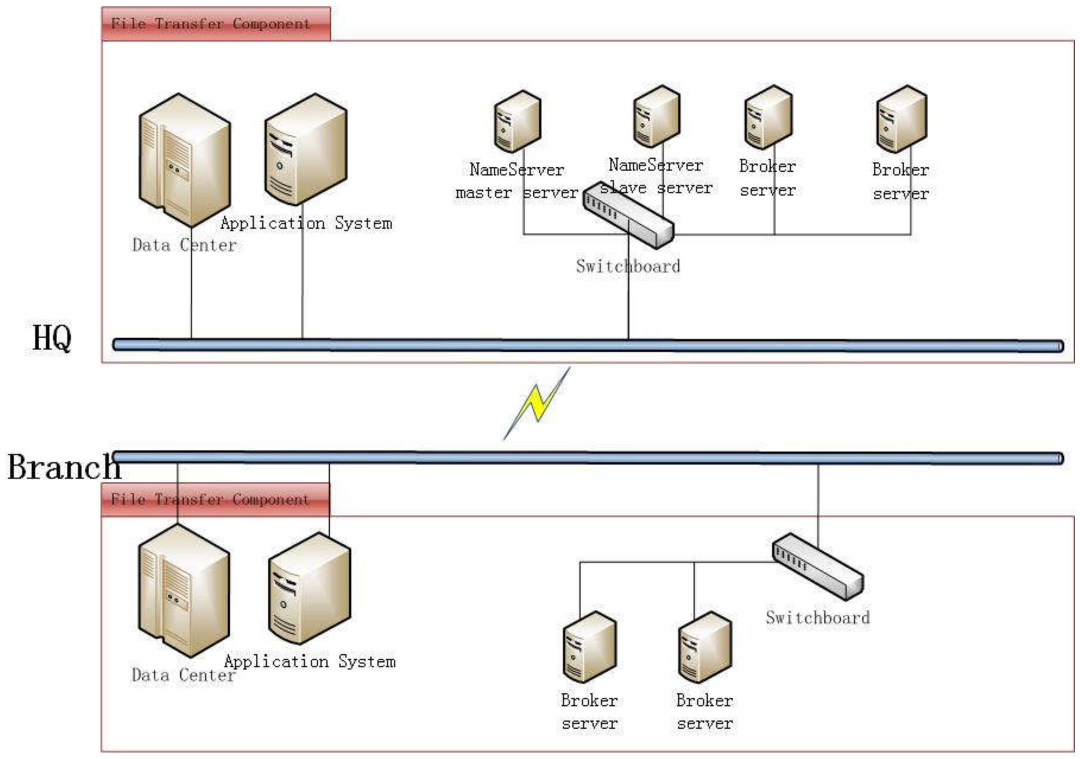

Figure 4. Physical Deployment Diagram

In the figure above, the API is integrated with the application system to facilitate the application system to send or receive files. Multiple Brokers deployed on multiple servers to build a file transfer routing channel; The Name Server is deployed on the active and standby servers to ensure the stability of the file transfer process. At the same time, the Name Server is deployed on the headquarters side and is responsible for file transfer transmission status and routing information management.

6. Conclusion

For the application system interaction process, there is a problem of non-through network and large file transfer. This paper designs a solution for distributed file transfer system based on multi-thread. The whole architecture of file transmission system is designed, the design ideas and main stream of API, Broker and Name Server modules are given in detail, and the physical deployment diagram of the file transmission system as the middleware is given, and the use method is clarified.

References

[1] James F. Kurose, Keith W. Ross. Chen Ming Translate. Computer Network [M]: CHINA MACHINE PRESS, 2009.
[2] Cay S. Horstmann, Cray Cornell. Zhou Lixin, ChenBo Translate. Core Java Volume I-Fundamentals (Ninth Edition) [M]. CHINA MACHINE PRESS, 2013:620.
[3] George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair. Jin Beihong, Ma Yinglong Translate. Distributed Systems: Concepts and Design, Fifth Edition [M]. CHINA MACHINE PRESS, 2013:13.
[4] Burg. Java in Distributed Systems [M]. CHINA MACHINE PRESS, 2003.
[5] Calvert, Kenneth L. TCP/IP Sockets in Java [M]. Morgan KaufmannPublishers, 1988.