Computers, Materials & Continua CMC, vol.65, no.2, pp.1739-1752, 2020

Research on Real-Time High Reliable Network File Distribution Technology

Chenglong Li[1], Peipeng Liu[1], Hewei Yu[1, *], Mengmeng Ge[2], Xiangzhan Yu[2], Yi Xin[2], Yuhang Wang[3] and Dongyu Zhang[4]

Abstract: The rapid development of Internet of Things (IoT) technology has made previously unavailable data available, and applications can take advantage of device data for people to visualize, explore, and build complex analyses. As the size of the network and the number of network users continue to increase, network requests tend to aggregate on a small number of network resources, which results in uneven load on network requests. Real-time, highly reliable network file distribution technology is of great importance in the Internet of Things. This paper studies real-time and highly reliable file distribution technology for large-scale networks. In response to this topic, this paper studies the current file distribution technology, proposes a file distribution model, and proposes a corresponding load balancing method based on the file distribution model. Experiments show that the system has achieved real-time and high reliability of network transmission.

Keywords: High reliable network, file distribution, load balancing.

1 Introduction

Nowadays, the fast-growing Internet has become an indispensable part of life. At the same time, the IoT is also developing rapidly in the direction of large-scale and large data volume. Therefore, how to achieve large-scale, real-time and highly reliable data transmission has become a hot trend in network research. Real-time and highly reliable file distribution technology can effectively guarantee the real-time response and stable operation of the IoT network, which is of great significance for applications deployed in the Internet of Things. At the beginning of the Internet, because of the small amount of data, large-scale data distribution does not take too much time and resources. However, with the continuous development of streaming media and the increasing number of network users, the distribution of Bluray video, software updates and online video streams is

[1] Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing, 100029, China.

[2] School of Computer Science and Technology, Harbin Institute of Technology, Harbin, 150001, China.

[3] Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou, 510006, China.

[4] Faculty of Business and Economics, University of Hong Kong, 999077, Hong Kong.

[*] Corresponding Author: Hewei Yu. Email: yhw@cert.org.cn.

Received: 03 November 2019; Accepted: 30 November 2019.

increasingly requiring more network resources, and content-based services and applications are exponentially growing. It is estimated that video traffic will account for 82% of all consumer Internet traffic [Zhou, Chen, Yang et al. (2015)], Network users need service providers to provide low latency, fast and reliable service. Through research, the academic community has proposed the concept of the Content Delivery Network (CDN) [Wang, Song, Han et al. (2013)], content distribution networks are receiving more and more attention from scholars. Every year, many articles about content distribution networks has been published in computer network-related journals and top-level conferences. Research covers various fields of application, such as 5G+ network [Boccardi, Heath, Lozano et al. (2014)], privacy protection [Chen and Zhao (2012)] , data-driven deep learning [Gligor, Meadows and Suri (2008)], lowering service costs [Zhang, Lu, Wu et al. (2018)], etc.

The content distribution network is a content distribution system that is deployed on different networks and regions through policies. It improves the quality of service by adding cache servers. Cache servers are typically deployed at the edge of the network and are typically accessed by users with a few hops. That is, the content distribution network is provided with a source server and a regional cache server. When the user requests a resource, the user request is redirected to the area cache server closest to the user, and the required content is obtained nearby, by which the response speed of the user visiting the website is improved. At the same time, the zone cache server is a cache image of the content provider ICP (Internet Content Provider) source server. In this way, content holders distribute content through content distribution network service providers, providing users with quality services.

2. Document distribution

2.1 Content distribution network

The main purpose of the content distribution network is to make the transmission of content on the network faster and more stable. The core idea is to reduce the relevant factors on the Internet that may reduce the speed of data transmission and destabilize stability [Duan, Xing, Tian et al. (2018); Riad, Elmogy and Shehab (2013)]. From a broader perspective, CDN represents a network application service system built on top of existing infrastructure with quality assurance, real-time efficiency, easy implementation and easy management [Mathew, Sitaraman and Shenoy (2012)]. The technical principle of CDN is to move the high-traffic service content from the central server to the edge of the backbone network closer to the user, and redirect the user request so that the user can access the resource nearby, providing the user a comfortable user experience. Spagna et al. [Spagna, Liebsch, Baldessari et al. (2013); Sung and Hung (2009)].

2.2 AMQP transmission model

AMQP (Advanced Message Queuing Protocol) is an advanced message transmission queue protocol that is deployed on the computer network application layer to provide unified messaging services [Ge and Liu (2017)]. AMQP is an open standards based messaging solution that allows anyone to interact with an MQ (Message Queue) server from any AMQP vendor via standard encoding. AMQP technology needs to set up a relay

server. When the system is running, the sender delivers the message to the relay server. Due to the different delivery keywords of the sender, the relay server distributes the message to different receivers. Implement asynchronous interconnection between the sender and the receiver. At the same time, the transit server is responsible for message maintenance and load balancing of the overall system. RabbitMQ is a typical open source messaging middleware based on the AMQP standard, an open source program for cluster message distribution. RabbitMQ has several different message distribution modes, namely direct message mode, broadcast/subscribing mode and receiving mode.

RabbitMQ service model. In the RabbitMQ transmission model, there are mainly three roles: RabbitMQ server, producer and consumer. RabbitMQ server maintains a path from producer to consumer. RabbitMQ has two delivery keywords, "exchange" and "queue", "exchange" means that the producer delivers the product, and "queue" is the receiving queue received by the receiver. There is a binding process from "exchange" to "queue". The producer can deliver the product directly to the "queue", or it can be delivered to the routing keyword "exchange". The RabbitMQ binding listens to the "queue" of the "exchange".

The producer is the sender of the message, and the message sent contains the message content and the message receiver. The recipient of the message can be set by routing the keyword "exchange name" or "queue name".

The consumer is the receiver of the message, and receives the message by binding the queue name. When the consumer receives the completion message, it will return an acknowledgement message to the RabbitMQ server. After receiving the message, the RabbitMQ server will delete the message from the listening queue.

RabbitMQ distribution model. RabbitMQ has three transmission types: Direct, Broadcast/Subscribing, and Receiving. Direct mode (Direct): In the direct transfer mode, the producer does not directly bind the route "exchange", but binds the queue keyword "routing_key". In this way, the producer directly distributes the message to the consumer's listening queue, and the consumer reads the message directly from the corresponding queue.

Broadcast/subscribing mode (Fanout): In the broadcast/subscribe mode, the producer does not directly declare the queue keyword "routing_key", but only the route "exchange". The consumer declares the bound route and binds the queue of the corresponding route (the name generated by the program). In this way, after the producer sends the message, all the subscribers corresponding to the route "exchange" will receive the message sent by the producer.

Receiving mode (Topic): In the receiving mode, the sender's declared queue keyword "routing key" is a list of words separated by a decimal point, which are generally set to meaningful words. The recipient selects some keywords to receive and binds these keywords to the "routing key" list. The RabbitMQ server will distribute the message to the message queue of the corresponding keyword listener according to the producer's keyword.

3 Load balancing of file distribution system

In a file distribution system for a large scale network, terminal nodes are connected to each other to form a structured P2P network. However, due to the configuration of each terminal node and the different network environments, the processing capability and transmission capability of each terminal node are different, which may be a great gap between nodes.

Studies have shown that in this structured network, there are cases where most service requests are concentrated on a small number of resources. 90% of service requests are concentrated on 10% of popular nodes. This phenomenon is called Zifp distribution [Yin, Liu, Min et al. (2010)]. In order to solve the problem of uneven load, this paper proposes a load balancing scheme based on multiple strategies. First, when the node requests the resource, the secondary server randomly selects a certain number of nodes that own the resource and feeds back to the resource requester to achieve the first balance when the resource is requested. In the subsequent resource request process, each node calculates its own load situation in real time, and predicts whether there will be a high load condition according to the download request situation received by itself, then returns the data to the secondary server. The secondary server selects the appropriate node, establishes a new resource owner, and implements the load mitigation [Handurukande, Kermarrec, Le et al. (2006); Ma (2016)]. Through the above two methods, a load balancing model is established to implement load balancing of the system network.

3.1 Concept definition

Definition 1: (physical utilization) the ratio of the load experienced by a node to its own capabilities

$$utl = \frac{load}{capacity} \tag{1}$$

Definition 2: (system imbalance) per node utilization and system node utilization mean (utl) variance in the system

$$dev = \frac{\sum_{i=1}^{n}(utl_i - utl)^2}{n} \tag{2}$$

Definition 3: (node load level) According to the physical utilization of the node, the node load status is divided into three levels: node low load load_l, node load normal load_m, and node high load load_h.

Definition 4: (High load prediction threshold) The difference value of the request file for predicting a high load is represented by ψ. If the value exceeds the distinguished value, the file is a file that causes a high load, and is also called a hotspot file.

Definition 5: (High-load file level) According to the prediction threshold, the file heat is divided into a low-heat file file_l and a high-heat file file_h.

Definition 6: (High load area and low load area) The area where the K-hop routing distance centered on a node is at most the N-node load is lower than the node high load threshold, the area is a high load area, and the central node is a high load area. Central node. To distinguish the load balancing thresholds between the high load zone and the low load zone, the threshold of the high load zone is referred to as the high load zone threshold, and the threshold of the low load zone is referred to as the low load zone threshold. Obviously, the high load zone threshold should be smaller than the low load zone.

3.2 Collect local load information for load zone judgment

When the node enters a high load state, the high load node broadcasts K-hop routing information to the surrounding network, requests the load status of the surrounding nodes, and calculates whether the node is in the high load area. When the surrounding node

receives the load request information, the node detects the node utilization of the node itself. If the node is in a low load state, the node utilization is less than the high load zone threshold, and the node returns the basic information of the node to the state requester. A timestamp is used to calculate the link delay. At the same time, the route hop count K is decremented by one, and the request is broadcast to the surrounding nodes. If the node is in a high load state, the message is forwarded only after the number of route hops is processed, and the message is not replied.

The local load balancing information collection steps are as follows: Step 1: When the node predicts that a hotspot file may be generated, according to the prediction model or the node load value which is higher than the high load area threshold, the node distributes broadcast information to the surrounding nodes to query the surrounding node load condition. Step 2: When the node receives the load request information, it immediately reads the load status of its own system. If the node utilization is higher than the high load area threshold, it jumps to Step 3; if the node utilization is lower than the high load area threshold, then the node information and the route hop count are fed back to the load information requesting node, and the transmission packet includes a timestamp of the information packet sent by the local node. Step 4: If the receiving node is the last hop receiving node of the route hop count, discard the broadcast packet; otherwise, the route hop count information TTL-K information is decremented by one, and then continues to broadcast to the surrounding nodes.

After the node broadcasts, the request message waits for the feedbacks, such as the node load utilization rate of the node.

$$W = (\mu_1 * ts + \mu_2 * k) * (\mu_3 * \frac{load}{capacity}) \tag{3}$$

$$E = \frac{\mu_1 * ts}{\mu_2 * k} * (\mu_3 * \frac{load}{capacity}) \tag{4}$$

Eq. (3) applies to the migration selection strategy of close range nodes, and Eq. (4) applies to the migration selection strategy of long distance nodes. μ_1, μ_2, and μ_3 are three user-adjustable parameters, which are used to adjust the proportion of operations occupied by different indicators. μ_1, μ_2, and μ_3 satisfy $\mu_i \subseteq [0,1]$, and $\sum_{i=1}^{n} \mu_i = 1$. ts represents the link delay,k represents the number of link route hops, and $\frac{load}{capacity}$ represents the node utilization. The smaller the calculated values W , E the better the node copy creation result.

It can be learned from the literature that a node can receive feedback from $N = m\frac{m^k - 1}{m-1}$ nodes at the maximum.The literature also indicates that the time complexity of the process is O(lgN), space complexity $O(N)$ [Yin, Liu, Min et al. (2010)]. After testing, it is found that when the number of route hops is set to 3, the number of received responses is set to 4, the test effect is ideal

3.3 High load area load migration strategy

When the node judges that it is located in the high load area through the feedback message, which means the number of returned messages is less than 4, the high load area is applied to the migration strategy. In the face of high load areas, node loads should be migrated to distant nodes as much as possible to reduce the number of high load nodes in the area and

reduce the degree of network congestion in the area. At the same time, try to migrate nodes to distant nodes and lower nodes. The possibility of migrating loads between each other. The migration process is performed by the load center node to send the migration node to the secondary server, the secondary server selects the load migration node, and the hot load file copy is established on the load migration node, so that the load migration is migrated outward from the load center of the high load area, so that the area is The high load area transitions to a low load area.

K is usually taken as 3. For all nodes, the load complexity of the high load area load migration node selection algorithm is $O(N)$, and the space complexity is $O(N)$, where N is the number of feedback received. In this article, n is usually set to have half the number of hot files and half of the number of high load nodes.

3.4 Low load area load migration strategy

When a node has a high load or predicts that a hotspot file will be generated, the node takes out the N nodes with the smallest W value from the storage linked list. Then the system will read the physical node information and load information, and send the node information to the secondary server. The secondary server obtains the information of the former n nodes according to the information of the nodes, sends a request to the nodes. The secondary server acquires load information and calculates the migration performance of the nodes. The next node in the linked list is taken out, and the migration performance of the extracted node is calculated. The migration performance of the extracted node and the previous optimal node is compared, until the migration performance of the newly extracted node is lower than the migration performance of the previous optimal node, or all nodes have been read and distributed the hotspot files in the high load node to the optimal node.

4 Design and implementation of document distribution system

At the architecture level, the system consists of two modules, namely the control module and the transmission module. The control module is responsible for the interactive control message processing of the system in the whole process. The transmission module is responsible for distributing the specific content after the negotiation of the interactive information. At the control level, the control information is transmitted and transmitted by AMQP, and is mainly implemented by RabbitMQ. The transmission module is divided into two parts, namely the AMQP transmission part and the P2P transmission part.

The system can be simply viewed as a three layer network topology, as shown in Fig. 1(a). In the network structure, multi source data is the producer of multimedia resources and is responsible for the production of multimedia resources. The first layer of the network topology is the central server, responsible for resource distribution of the entire system, which monitor resource distribution and store all multimedia resources. The second layer of the network topology is a transit server that stores copies of resources and is deployed in different regions to distribute resources to users. The third layer of the network topology is the end user, which is the requester and user of the multimedia resource.

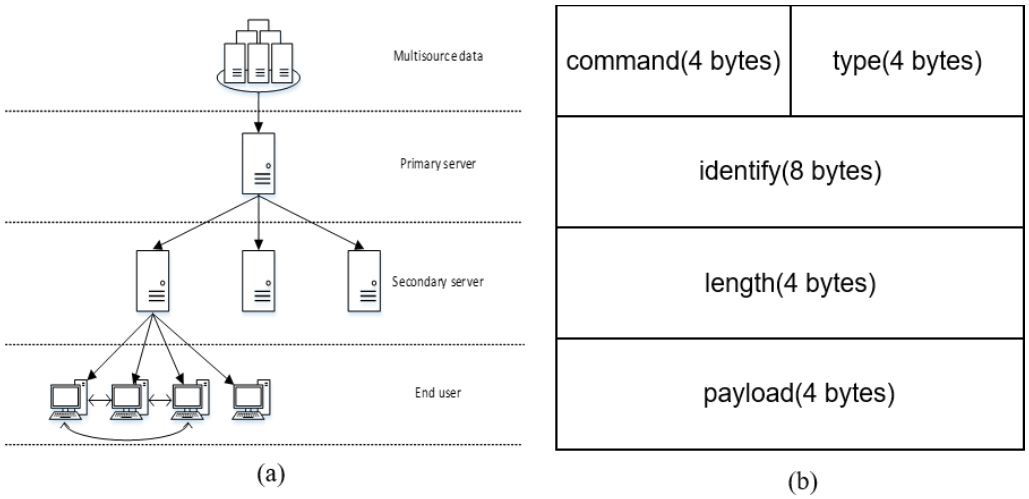

Figure 1: (a) Network architecture of the file distribution; (b) Command information

4.1 Control module

The file distribution system is based on the message driver. Each module communicates through the advanced message queue. The message format of the communication is agreed between the modules. The interactive commands mainly include session messages, broadcast messages, single point transmission messages, and error messages. Message protocol class messages and task class messages. The session message is responsible for the initiation and termination of the session, as well as the feedback reception of the session. Broadcast messages are responsible for the processing of some broadcast functions, such as scanning the current node online status. Single point transmission messages are responsible for file distribution for some non-receiving domains, when message content is only required to be distributed to a specific area. The error message is responsible for transmitting a message error, such as receiving a verification file message that does not match the source file. The task message is relatively simple and is responsible for loading various messages for transmission.

As shown in Fig. 1(b). The command field indicates the type of command. This field is used to distinguish which category the message belongs to. The type field indicates the specific interactive command type under the message class. Through the combination of the command field and the type field, the function of the command can be parsed. The identify field indicates the identification area of the command, and the client receives and processes the command only when the field matches its own node id. The length field indicates the length of the load. The payload field represents the message payload, the body portion of the content of the message. Through the above command format, various command interactions in the file distribution process can be realized.

In order to achieve better distribution speed of the entire system, it is necessary to make each server perform very high efficiency. The system uses the advanced message queue RabbitMQ to perform scheduling command transmission between each server and the client. During the running of the system, a message channel for controlling information is

maintained, and a control command is transmitted between the primary server (the primary server) and the regional server (the secondary server) through the channel.

4.2 Transmission module

File distribution mainly uses two transmission methods, the RabbitMQ transmission method based on the advanced message queuing standard and the BitTorrent transmission method based on the structured point-to-point transmission network. Based on the structured network, BitTorrent adopts many-to-many transmission mode in the transmission process, and it has more control transmission commands. When the file is small, only a few files are fragmented during the file fragment transmission process, which cannot be fully utilized. Out of the advantages of many-to-many slice transfer files. In this case, the RabbitMQ transmission mode has the characteristics of short and asynchronous transmission of control information, suitable for small file transmission, and complements the BitTorrent transmission mode.

File distribution transmission. RabbitMQ distributes small files in two modes, direct distribution mode and subscription/broadcast distribution mode. If you only need to transfer files to some nodes, the direct distribution mode is preferred. If you need to distribute files to all nodes, then the subscription/broadcast mode is preferred. In this case, the impact of broadcast control information on the entire network is reduced, and subscription/ Broadcast mode uses fewer file copies, reducing the amount of file disk operations

For larger files, if a one-to-many transmission method is used, the network bandwidth of a single node becomes a performance bottleneck of the entire system, resulting in a decrease in distribution efficiency. The structured P2P network BitTorrent adopts a many-to-many transmission mode. Each node also distributes the received file fragments [Li and Xie (2006); Petterson and Sirer (2009)], while requesting unsent file fragments. So although the receiving rate of a single node is reduced, the receiving time of the entire system is reduced.

Error retransmission. The implementation of the error retransmission mechanism is based on the consistency of the hash function, which means the result string obtained by the same file in different environments using the hash article function operation is the same. Here, the hash function MD5 operation is used to generate a 256-bit hash value to verify the consistency of the file [Xiao, Wang, Liu et al. (2018)]. During the transfer, the server sends the file name, file size, and file hash of the next sent file to the receiving node. After receiving the information, the receiving node will feed back the message to the server. After determining the content consistent with the sent content, the server sends a file to the receiving node. After receiving the file, the receiving node verifies the file size and the file hash value. If it is the same as the previously received one, the feedback receives the correct command; otherwise, the feedback receives the wrong file name. After receiving the error retransmission command, the server starts the error retransmission mechanism, establishes an error retransmission channel, resends the file information to the node separately, and then sends the file information to the node. If the file verification error occurs, the error loop module is called again until the file is received correctly.

Node Drops Continued. The terminal node offline retransmission module is designed as follows. Each time a node completes a file, and it will save a local record with time. Therefore, after the dropped node is re-online, it will scan the locally stored download data

and read the last download completion record that was downloaded before the line was deleted. If there is no local record, it may be that the node goes online for the first time or the node disk fails. In this case, all files need to be re-received. After reading this configuration, the file name of the file that was last received is sent to the central server. The central server queries all files distributed to the area after the last distribution time point according to the received file name, establishes a control channel and a data channel, and delivers the file information of the subsequent files to the dropped line through the control channel. The node and the file are delivered to the dropped node through the data channel, and the dropped node receives the file information through the control channel to verify the correctness of the received file, and correctly receives the correctly received command from the server; otherwise, returns the file name of the received error, and the error is passed. Retransmission module to re-receive

5 Introduction experiment of the file distribution system

This chapter first introduces the experimental environment configuration and the main tools used. Next, according to the system implementation in Chapter 4, experiment with each module, observe the distribution of the system under the condition of various parameters, to verify whether the proposed model can meet the actual needs in Chapter 3.

5.1 File distribution module experiment and result analysis

The experimental files used in this experiment are random files generated by the program, and the file sizes are randomly distributed in the file range of 1 MB to 1 GB. The file distribution experiment mainly selected the average download speed as the main measure of the test results. This parameter can reflect the overall distribution performance of the system and is less affected by fluctuations. The transfer distribution experiment was simulated between the three servers. The distribution speed is calculated as shown in Eq. (5).

$$Distribution = \frac{N*M_{i\_distribution\_file\_size}}{N*(t_{distributio\_end}-t_{distribution\_start})} \tag{5}$$

In the above formula, N represents the number of nodes receiving the file, $M_{i\_distribution\_file\_size}$ indicates the file size of the distribution file, $t_{distributio\_end}$ indicating the time when all the nodes receive the distribution file, $t_{distributio\_start}$ indicating the time when the system starts to distribute the file. The mean of the results of the operation $Distribution$ represents the average file distribution rate of the whole system, rather than the single node receiving the file. Below, this article extracts the distribution rate graph of several typical file sizes.

As shown in Fig. 2(a), is the transmission rate with different amounts of nodes using RabbitMQ or using BitTorrent network. As can be seen, when distributing smaller files (100 MB), RabbitMQ transmission has an absolute advantage when the number of nodes is small. When the number of request nodes increases, RabbitMQ transmission is better than BitTorrent transmission, but the advantage is smaller and smaller. As the number of nodes increases, the transmission rate decreases due to machine performance and network. This shows that RabbitMQ is suitable for small file transmission under a certain scale and has high transmission efficiency.

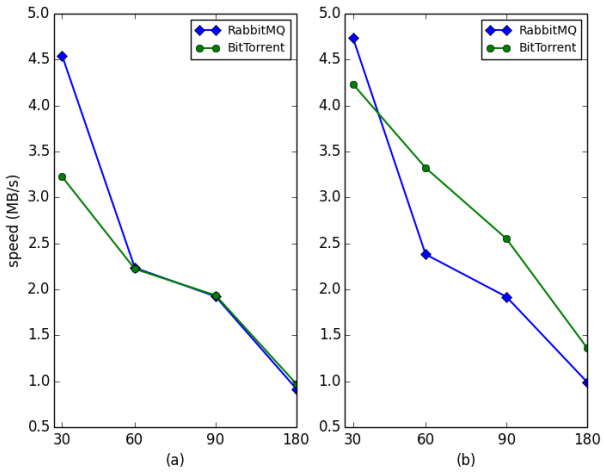

Figure 2: (a) 100 MB file resource distribution speed under different number of nodes; (b) 1000 MB file resource distribution rate under different number of nodes

As shown in Fig. 2(b), when the size of the transmission file is increased from 100 MB to 1000 MB, the RabbitMQ transmission rate is still higher than the BitTorrent transmission when the system resource utilization rate is not maximized at 30 nodes; In the case of system resource load balancing, the BitTorrent network's fragmented mutual transmission mechanism reflects its advantages and the BitTorrent network transmits more than the RabbitMQ network with the number of nodes keeps increasing. This shows that BitTorrent is more suitable for the transmission of large files, and the transmission efficiency is high in the case of many nodes.

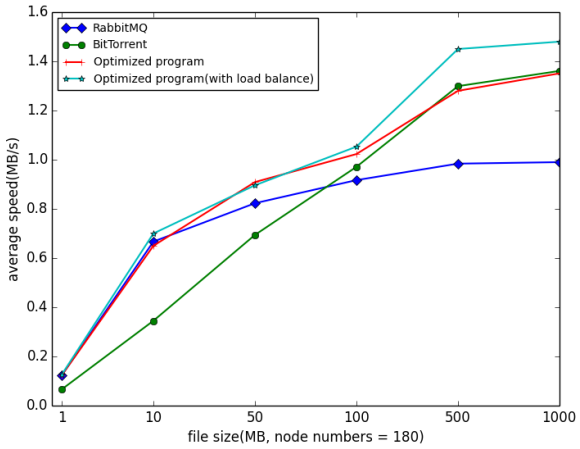

Figure 3: Distribution speed under different size of transmission file and different models

As shown in Fig. 3, it can be seen from the line graph that during the entire file distribution process, the RabbitMQ transmission and BitTorrent transmission are partially simplified and modified, and some transmission control is added on the basis of this, so the full scale is Segment files have good transfer speeds. In the case of transmitting files, the

transmission of control information before transmission accounts for a high proportion in the overall transmission process, so when the file is only 1 MB in size, the transmission speed is low. When the transmission file is large, the proportion of control messages is reduced, and the transmission speed fluctuation is based on smoothness. After the load balancing module is turned on, the transmission speed of the BitTorrent transmission module is significantly improved. This program fully utilizes the performance of the node, which reduces the load on the server to a certain extent and improves the efficiency of the entire transmission process.

5.2 Load balancing module experiment and result analysis.

Node Overload Rate. In this paper, five 200 MB files are distributed continuously through the program to analyze the overload condition of the node without using the load balancing algorithm and using load balancing. The node overload rate is shown in Fig. 4.

Fig. 3 shows how the node overload rate changes over time. As can be seen from the figure, for the program with the load balancing module turned on, the hotspot file prediction module predicts the generation of hotspot files shortly after the system starts running. After that, the system calls the load balancing policy. At this time, the overload node should be in the non-high load area. The migration strategy establishes a replica policy around the node, and the overload node in the area decreases, and then it is in a small fluctuation state. For the program without the load balancing module, after the program runs, the number of overloaded nodes increases continuously due to the increase of the request for individual files, and the overload rate of the system node increases continuously. After the receiving of some nodes is completed, the overload rate of the node begins to decrease. When a new file request arrives, the system node overload rate begins to fluctuate and then fluctuates periodically.

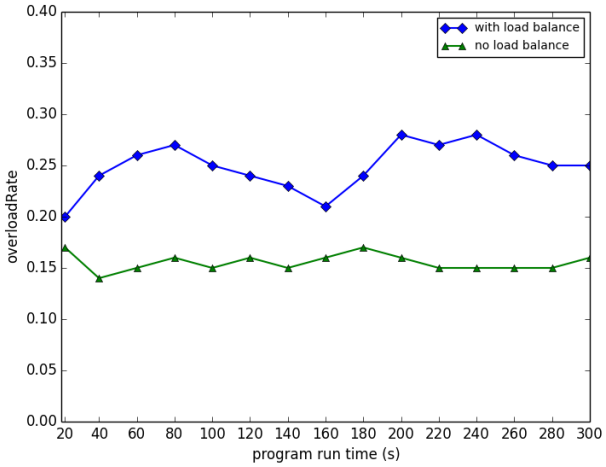

Figure 4: Comparison of node overload rate

High load node load fluctuations. When testing high-load node load fluctuations, this time also set up three nodes to have the requested resources, other nodes request resources from these nodes, and observe the load fluctuation of one of the nodes.

As shown in Fig. 5, it can be seen that the load fluctuation of the high load node changes with time. When the node is just running, the number of requests received by the node is about the same, and the load is similar. At 40 s, the load balancing module predicts the generation of hotspot files and then quickly creates a copy file. In the subsequent request peak, the load of the node is maintained in a relatively stable state because the copy file shares part of the request pressure. Then as the request decreases, the node's load is further reduced until there is no longer a low load state at the time of the request. For systems without load balancing, the load on the service node is rising, rising to 90%. After that, some nodes complete the reception, and the load of the service node begins to slowly decrease. Until 200 seconds, the request is no longer received and the system enters a low load state. The data can be analyzed and found that the load balancing system has better hot file prediction capability, can quickly execute the load migration strategy, and adjust the load of a single node

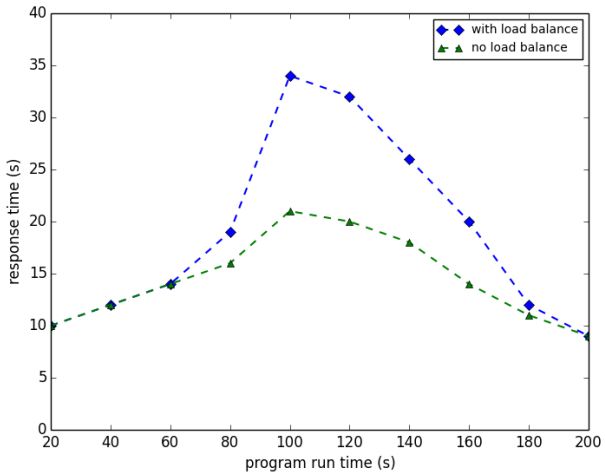

Figure 5: High load node load fluctuation graph

6 Conclusion

RabbitMQ transmission has an absolute advantage when the number of nodes is small. When the number of request nodes increases, RabbitMQ transmission is better than BitTorrent transmission, but the advantage is getting smaller and smaller. When the number of nodes is increasing, and the system resources are load-balanced, the BitTorrent network's fragmentation and mutual transmission mechanism shows its advantages. This shows that BitTorrent is more suitable for large file transmission, and the transmission efficiency is high in the case of multiple nodes. The data can be analyzed and found that the load balancing system has better hot file prediction capability, can quickly execute the load migration strategy, and adjust the load of a single node.

Funding Statement: This work was supported by National Key Research & Development Plan of China under Grant 2016QY05X1000, and National Natural Science Foundation of China under Grant No. 61771166, and CERNET Innovation Project (NGII20170412).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Boccardi, F.; Heath, R. W.; Lozano, A.; Marzetta, T. L.; Popovski, P. (2014): Five disruptive technology directions for 5G. *IEEE Communications Magazine*, vol. 52, no. 2, pp. 74-80.

Chen, D.; Zhao, H. (2012): Data security and privacy protection issues in cloud computing. *International Conference on Computer Science and Electronics Engineering*, vol. 1, pp. 647-651.

Duan, J.; Xing, Y.; Tian, R.; Zhao, G.; Zeng, S. et al. (2018): SCDN: A novel software-driven CDN for better content pricing and caching. *IEEE Communications Letters*, vol. 22, no. 4, pp. 704-707.

Ge, Z.; Liu, Y. (2017): Automatic discovery technology of real-time data distribution service. *Computer Technology and Development*, vol. 27, no. 1, pp. 25-29.

Gligor, V. D.; Meadows, C.; Suri, N. (2008): IEEE Transactions on Dependable a Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, vol. 5, no. 1, pp. 4-5.

Handurukande, S. B.; Kermarrec, A. M.; Le Fessant, F.; Massoulié, L.; Patarin, S. (2006): Peer sharing behaviour in the edonkey network, and implications for the design of server-less file sharing systems. *ACM*, vol. 40, no. 4, pp. 359-371.

Li, Z.; Xie, G. (2006): A load balancing algorithm for DHT-based P2P systems. *Computer Research and Development*, vol. 43, no. 9; pp. 1579-1585.

Ma, S. (2016): Research on application of P2P technology in large file sharing. *Experimental Technology and Management*, vol. 33, no. 3, pp. 147-150.

Mathew, V.; Sitaraman, R. K.; Shenoy, P. (2012): Energy-aware load balancing in content delivery networks. *Proceedings IEEE INFOCOM*, pp. 954-962.

Peterson, R.; Sirer, E. G. (2009): AntFarm: Efficient content distribution with managed swarms. *Networked Systems Design and Implementation*, vol. 9, no. 1, pp. 107-122.

Riad, A. M.; Elmogy, M.; Shehab, A. I. (2013): A framework for cloud P2P VoD system based on user's behavior analysis. *International Journal of Computer Applications*, vol. 76, no. 6, pp. 20-26.

Spagna, S.; Liebsch, M.; Baldessari, R.; Niccolini, S.; Schmid, S. et al. (2013): Design principles of an operator-owned highly distributed content delivery network. *IEEE Communications Magazine*, vol. 51, no. 4, pp. 132-140.

Sung, M. K.; Han, C. M. (2009): A study on architecture of CDN (Content Delivering Network) with content re-distribution function. *11th International Conference on Advanced Communication Technology*, vol. 1, pp. 772-777.

Wang, T.; Song, L.; Han, Z.; Jiao, B. (2013): Dynamic popular content distribution in vehicular networks using coalition formation games. *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 9, pp. 538-547.

Xiao, B.; Wang, Z.; Liu, Q.; Liu, X. (2018): SMK-means: an improved mini batch k-means algorithm based on mapreduce with big data. *Computers Materials & Continua*, vol.

56, no. 3, pp. 365-379.

Yin, H.; Liu, X.; Min, G.; Lin, C. (2010): Content delivery networks: a bridge between emerging applications and future IP networks. *IEEE Network*, vol. 24, no. 4, pp. 52-56.

Zhang, W.; Lu, Z.; Wu, Z.; Wu, J.; Zou, H. et al. (2018): Toy-IoT-Oriented data-driven CDN performance evaluation model with deep learning. *Journal of Systems Architecture*, vol. 88, pp. 13-22.

Zhou, Y.; Chen, L.; Yang, C.; Chiu, D. M. (2015): Video popularity dynamics and its implication for replication. *IEEE transactions on Multimedia*, vol. 17, no. 8, pp. 1273-1285.