Design of distributed storage system of digital media information based on Metadata

ZHU Xiao-Jun[1],LI Cai-hong[2]

{wz20160401@163.com[1],lgf147852@tom.com[2]}

（1.NanTong Normal College, Nantong 226010,China；

2.Changchun Institute of Technology, Changchun 130021,China)

Abstract. In view of the poor load balancing performance of traditional distributed storage system of digital media information, a distributed storage system of digital media information based on metadata is designed. The hardware configuration of the system is server module and content distribution module. The server module is mainly composed of database server, data acquisition device server, SCADA server and routing table server. The content distribution module is mainly composed of multiple content distributors. The software of the system consists of central node module, task scheduling module, network module and client module. Through the combination of hardware and software, the distributed storage of digital media information is realized. In order to prove that the load balancing performance of the system is better, the traditional system is compared with the system. The experimental results show that the load balancing performance of the system is better than the traditional system.

Keywords: Metadata;Digital media information;Distributed storage system;Load balancing performance;

1 Introduction

The fifth information revolution in the 20th century refers to the digital revolution, which is marked by the wide application of computers and the achievement of digital media represented by electronic technology and network technology, thus affecting the development of various industries in social life. For more than half a century, the application of digital art has become the mainstream of digital media art in entertainment culture and commercial display. Today, digital media art is widely used and flourishing. Digital media has fundamentally changed the social environment, economy and culture of people's lives. As an interdisciplinary subject of culture, art and technology, space design has also been affected unprecedentedly [2]. In this context, the creative industry economy has given birth to a new form of digital media art represented by digital animation, digital film and television, online advertising, online games. This kind of digital media art, with information science and digital technology as the main means, mass communication theory as the basis and modern art concept as the core, applies the information art form to the fields of military industry, culture, art, education, medicine and commerce, which is also a new media form of interdisciplinary integration of modern science and technology and art.

EAI MOBIMEDIA 2020, August 27-28, Harbin, People's Republic of China

DOI 10.4108/eai.27-8-2020.2294612

With the continuous development of digital society, the creative industry represented by digital media art animation and game has become a key industry in many countries. The development of digital media technology and the continuous integration of media provide new technical means for the content and form of art design, and form a new art form. Its application has become the most dynamic and creative emerging industry in the cultural and creative industry, and has formed a certain industrial model. People use digital media and information technology to integrate text, image, voice, image and other information, and generate a series of related products, technologies and services.

Twenty-one In the 21st century, driven by the context of "globalization", the era of digital information is coming. With the rapid development of information technology, the number of digital media information is increasing, the area of digital media information is expanding and the speed of transmission is speeding up. People's life has entered the era of digital media information explosion. Digital media information has penetrated into people's daily life and become the most common in our life a part. At the same time, the development of the network provides a wider range and real-time way for the dissemination of digital media information. In reference [3], a distributed data storage architecture for data storage and management of power transmission and transformation projects is proposed. Based on the metadata model, the architecture refines the three types of data, i.e. engineering geographic information, three-dimensional design model and document data of power transmission and transformation projects, and according to the different data storage modes, A distributed storage architecture is designed to deal with all kinds of data, but the load balancing performance of the storage system is poor.

In view of the problems of the above methods, this paper designs a digital media information distributed storage system based on metadata, which solves the problems of the traditional system.

2 Design a distributed storage system of digital media information based on Metadata

2.1 Hardware design

The hardware configuration of the metadata based digital media information distributed storage system is server module and content distribution module [4].

2.1.1 Server module

The server module is mainly composed of database server, data acquisition device server, SCADA server and routing table server. The access flow of digital media information data is as follows:

1. Access the local routing table server and query the routing table according to the IP

address;

2. Start from the forwarding server of the client, enter the next hop according to the routing table results, and continue to step 1 until reaching the target server or the query fails. If it fails, the unreachable information will be returned; otherwise, the corresponding operation processing flow will be executed. In the process of processing, if the client requires the query or processing result to be an intermediate value, the intermediate result data will be returned after corresponding calculation and processing at the target server;

3. After the client displays the corresponding results or integrates the intermediate results, continue to perform the follow-up tasks [5].

2.1.2 Content distribution module

The content distribution module is mainly composed of multiple content distributors, which can manage all activities of all user nodes in its domain. The content distributor only stores the basic information of the digital media information file and the index information of the digital media information file segments, not the actual content of the digital media information file resources. When a user node uploads a file, the content distributor splits the uploaded file, distributes it to the user node under its jurisdiction, and records the basic information and index information of the file locally; when a user node queries a file, the content distributor finds the corresponding information of the query file by querying the file information stored in the whole upper P2P network, and returns it to the user section When a user node requests to download a file, the content distributor that receives the download request requests the user for file segmentation according to the local partition index information, and reconstructs the original file and transmits it to the user node. In the upper P2P network, the content distributor will send the local file information and the partition index information to its neighbor nodes for redundant storage, so as to ensure the fault tolerance and invulnerability of the system [6]. The content distributor maintains the network topology of the system by sending Ping messages to each other and receiving heartbeat information from user nodes.

The content distribution module uses the message trigger mechanism, and the content distributor calls the corresponding function module according to the received message type. The functional partition of the content distributor is shown in Figure 1.

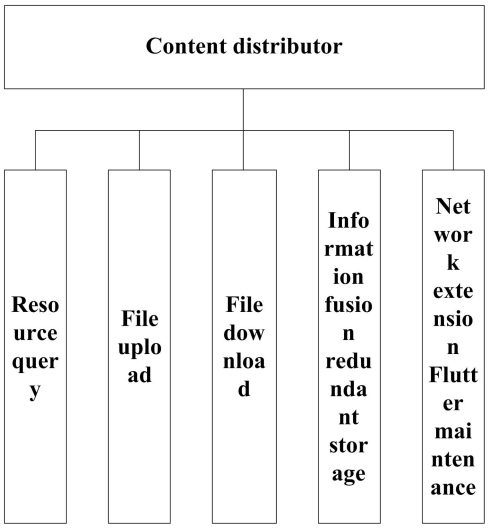

Fig. 1. Functional division of content distributor

According to the figure, the content distributor is composed of resource query function, file upload function, file download function, information fusion redundant storage function and network extended flutter maintenance function.

2.2 Software design

The software structure of the distributed storage system of digital media information based on metadata includes central node module, task scheduling module, network module and client module.

2.2.1 Central Node Module

The central node module is designed based on metadata. The main tasks of the central node module include processing the location information requests of clients, assigning new cache tasks and prefetch tasks to cache nodes, or assigning new storage tasks to persistent nodes, maintaining all the metadata of the cluster, managing all the read cache nodes in the cluster, writing and prefetch cache nodes and persistent node management node failures, etc [7]. The main classes related to the central node are shown in Figure 2.

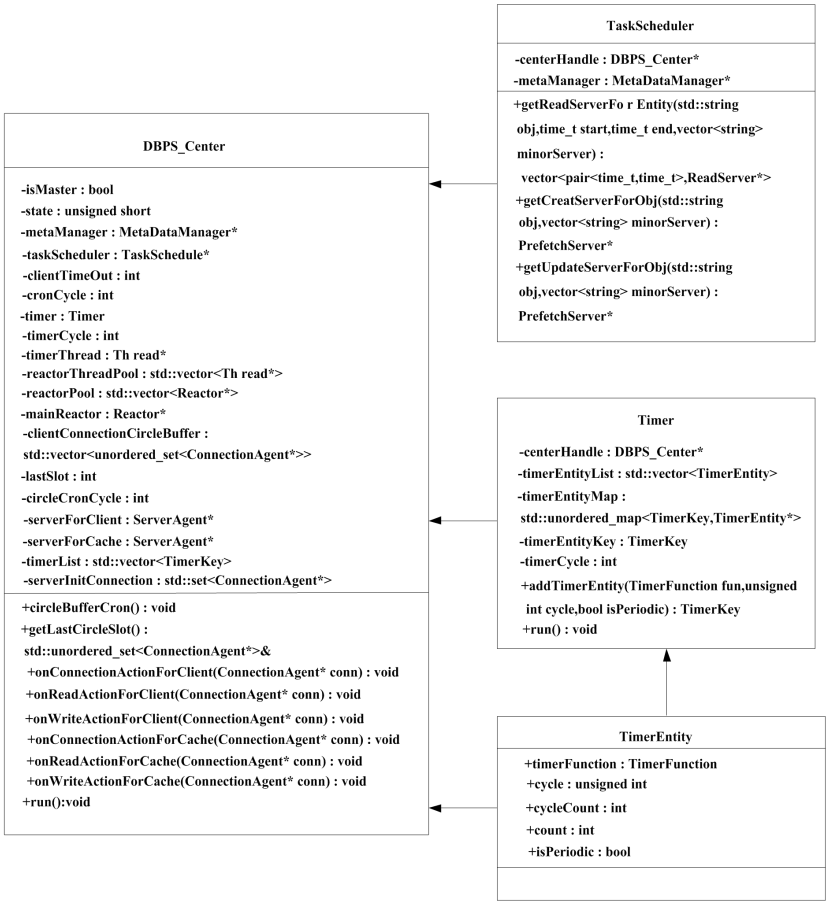

Fig. 2. Major classes associated with the central node

Each central node in the central node module corresponds to a DBPs ＜ center instance. Running its run() function service, it starts. It uses three service ports, one for processing the client's request, one for communicating with the data node, and the other for synchronization between the master and the standby. The events that nodes need to deal with are mainly divided into two categories: one is network events in digital media information, the other is timing events in digital media information.

Among them, network events are driven by messages. When the central node reads and parses the messages from the socket, it processes them according to the source and type of messages. These events include processing client's request, receiving cache node's heartbeat information, error report information and metadata update information, and processing cache node's data location request. For example, the network part of the node adopts the I / O multiplexer and non blocking I / O reactor mode. DBPs · center defines the callback function

onreadactionforclient to handle network events. The variable mainreactor is used to listen for newly initiated connections. Other reactor instances stored in the reactor pool run in multiple threads, mainly responsible for the central node and the storage section Communication between point and client [8].

Timer events are executed by timers, as shown in class timer in Figure 2. Timer only uses a real timer, and all timer events registered by nodes are realized by counters. For example, timer counts every 1ms, and timer events with a period of 1s will be executed when the count value reaches 1000, and each timer event is represented by a timerentity It records whether the functions and events to be executed at a fixed time are periodic events, and whether they have a long time or an execution cycle. The timing events mainly include the routine inspection procedures and cleaning procedures of the central node. For example, it checks whether the data node times out, does not contact with the central node, carries out corresponding processing, checks whether there is a timeout client connection, closes them, and regularly tracks the state time of the source object last written to the disk, regularly clears the invalid location information and node information, and also includes the timeout connection from the data node.

DBPS Center uses a circular buffer clientConnectionCircleBuffer to handle the Conn timeout of the client. Each buffer slot stores the client ConnectionAgent with corresponding time to the timeout. The pointer lastSlot always points to the slot with the longest time to the timeout. The adjacent slots are separated by a unit of time, so the previous slot of the lastSlot is the current timeout connection Section agent collection. Timing event will make lastslot move forward one slot in each unit time and close the connection in the slot. Each client connection will move itself from the last stored slot to the slot indicated by the current lastslot when it is active, so as to update the timeout.

2.2.2 Task scheduling module

Task scheduling module mainly includes task scheduler, whose tasks include allocating appropriate read cache nodes and write and prefetch cache nodes for client requests, allocating persistent nodes for new traceability objects or agents, selecting appropriate nodes to replace the tasks of the original nodes in case of node failure, etc. After receiving the client's request, the central node will parse the message, get the request type and parameters, and then call the corresponding task allocation function of taskscheduler according to the request type to get the allocation result and return it to the client. The whole process is shown in Figure 3.

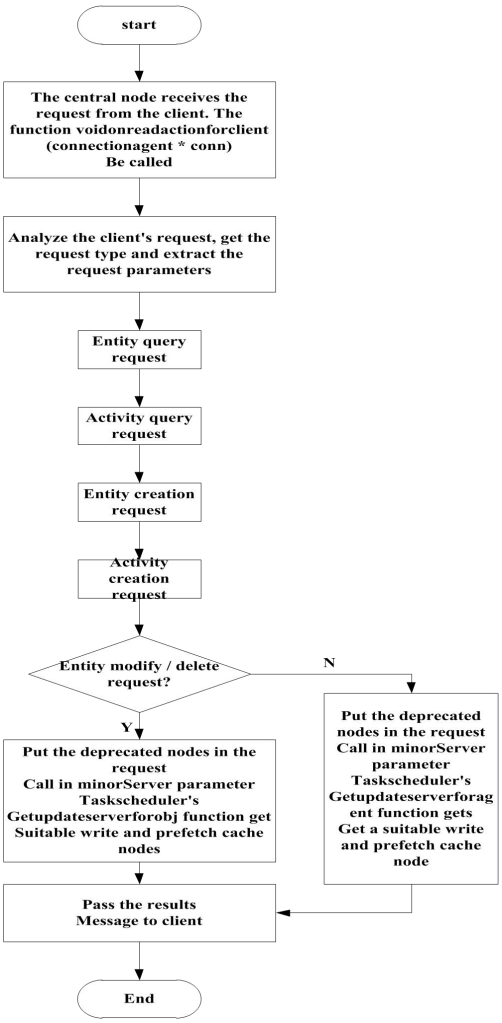

Fig. 3. Client request task scheduling

Taskscheduler defines the task scheduling function used by the central node when receiving the client location information request. According to the operation type and operation object type of the client request, it can be divided into: assigning the traceable object or agent in the query request to the read cache node where the data is stored in the uncached time period, creating or modifying / deleting the traceable object or agent in the request Configure the prefetch node that is responsible for processing the request, assign the persistent node that is responsible for managing its data to the newly added traceability object or agent [9]. According to the metadata maintained in metadatamanager, taskscheduler uses load balancing strategy to allocate tasks. In addition, it provides a node list parameter minorserver

in all task allocation functions. The list contains the nodes with which the client has recently failed to connect. It enables the client to participate in the selection of nodes and avoid getting invalid scheduling nodes again As a result, the task scheduler avoids the nodes in the list when selecting service nodes for clients.

First, call getreadlocationforentity to get several time periods and their corresponding cache nodes, then check the time periods that have not been cached and assign them appropriate service nodes. TaskScheduler segmenting the non buffering time segment according to the unit time sizePerSecond of the traceable object, then selecting multiple alternative nodes by function nth_optimalReadServer, and selecting the highest priority node to cache the data in a certain time period. It will call the MetaDataManager distributeReadServerForE after each assignment task is completed. The entity function updates the metadata.

2.2.3 Design network module

The network module consists of four main components. One is the I / O event source to be monitored. In network events, it mainly refers to the socket file. The other is the multiplexer, such as select, poll, epoll. The monitored event source needs to be registered with these multiplexers, and it is informed of the events to be monitored (readable, writable or with errors) and the handling to be called after the event occurs Program, the third is the manager that deals with the whole event driven process, which provides the user with the interface to register the event, and registers the event to the I / O multiplexer. It manages all event sources, circulates the execution of waiting and processing programs, and the fourth is the event processing function that contains the application logic. Reactor will associate the event source with its corresponding processing function, and adjust when the event occurs Use it.

The reactor is the manager. The reactorepoll is the I / O multiplexer using epoll. The reactorevent corresponds to an event source, and records the file descriptors related to the event source, the event types to be monitored and the event handling functions. It is registered to the reactorepoll through the reactor. The reactorepoll will create an epoll "event" according to the reactorevent and open it in the next event cycle To start listening, reactor contains a reactorepoll instance. It runs the event listening function of reactorepoll. When an event occurs, it obtains the event source related to it, reactorevent, and calls its event processing function. The event processing function is registered by the creator of reactorevent, usually serveragent or connectionagent, etc.

Serveragent represents a server agent, clientagent represents a client agent, and connectionagent represents a connection. The connectionagent instance corresponds to a successfully created connection, records the corresponding socket, and contains the event

source created for it, reactorevent, which provides the event handling functions of socket readability, writability and error to reactorevent, through which the user interacts with the opposite end, and provides the function of sending messages to the opposite end and the user registration callback function to the user Function. The callback function describes the processing that the user needs to do when different events occur, such as when the connection is successfully established, when the data is readable, when the message is sent, etc. it reads and writes the data of the socket through the buffer of the application layer, so that the user only needs to read and write the buffer, and does not need to interact with the kernel [10].

In the network module, there is communication between the client and each node. They send request, response or report information to each other. The message format is shown in Figure 4.

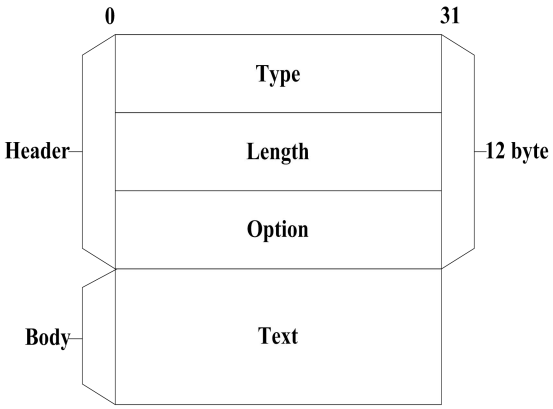

Fig. 4. Message format

The message header contains three fields, type indicates the type of message, and the node will parse the text part of the message body according to the message type after receiving the message; length indicates the number of bytes occupied by the text of the message body, and the node will judge whether the message received is complete according to it; option indicates optional parameter, in which the node can store some important information or simple message information Set text to null. For example, when a node returns error information for a request, it can store the error code in it. Text is the main part of the message, which is the byte order encoded by protobuf. After the node receives the message, it uses the deserialization function of protobuf to parse the text according to the message type, so as to read the content. The format of all message bodies needs to be in advance Defined in the protobuf file, the encoding and compression algorithm of protobuf reduces the message volume.

2.2.4 Design client module

The client module designs the API including crud function according to the user's demand. The user creates the instance of entity, activity or agent by executing the constructor of class entity, activity and agent, and sets their properties through the member function, including the object name and version number of entity, the agent it belongs to, the time when it begins to exist, the start and end time of activity, and the sending time Starting from its agent and action name, agent namespace and name, etc., by setting the relationship between them, create an association for them, and finally use the commit function to submit the new instance to the system. The API related to the creation operation includes the API shown in Table 1 in addition to the constructor and member function of the entity / activity / agent class.

Table 1 Create Action API

Serial number	Create operation API	API function
1	voidsetRelationship(ProvObj* obj, ProvObj* obj,rel_t type)	Set the relationship between two objects, including wasused, wasatrributedto, etc
2	int commit(ProvObj* obj)	Write objects to DBPs system

3 Simulation experiment design and result analysis

3.1 Design comparison experiment

The distributed storage system of digital media information based on metadata is used in the experiment of distributed storage of digital media information. Set up the experimental platform of the system, and build spark on Hadoop HDFS, which is jointly stored by HDFS and HBase, coordinated by zookeeper, and spark is responsible for the calculation part of the experiment. The experimental cluster is configured on 7 machines, and the operating system uses Ubuntu 14.04 LTS. Table 2 shows the specific configuration and version. You can expand the nodes according to the actual application. The configuration of all nodes of each machine in the cluster is shown in Table 3.

Table 2 Specific configuration of clusters

Serial number	Name	To configure
1	Hadoop version	Hadoop-2.6.0
2	Spark version	Spark-1.5.0
3	HBase version	HBase-1.1.4

4	Zookeeper version	Zookeepe-3.4.6
5	JDK	Jdk1.8.0_72
6	Scala	Scala-2.11.8
7	IDEA	ideaIC-15.0.2
8	Control node	Four core, 32g memory, 1t hard disk, quantity
9	Computing node	Four core, 16g memory, soog hard disk, quantity 7

Table 3 All node configurations for each machine in the cluster

Machine name	IP	Course
master	192.168.1.170	NameNode,Master,ResourceManager,DataNode,JournalNode,HMaster
slave 1	192.168.1.171	NameNode,Worker,DataNode,NodeManager,JournalNode,HRegionServer
slave 2	192.168.1.172	DataNode,Worker,NodeManager，JournalNode,HRegionServer
slave 3	192.168.1.173	DataNode,Worker,NodeManager，JournalNode，HRegionServer
slave 4	192.168.1.174	DataNode,Worker,NodeManager，JournalNode，HRegionServer
slave 5	192.168.1.175	DataNode,Worker,NodeManager，JournalNode，HRegionServer
slave 6	192.168.1.176	DataNode,Worker,NodeManager, JournalNode

In order to ensure the effectiveness of the experiment, the traditional digital media information distributed storage system is compared with the metadata based digital media information distributed storage system designed in this paper. Compare the load balancing

performance of the distributed storage system of digital media information. The judgment of load balancing performance is based on the fluctuation of load value of each system. The greater the fluctuation is, the better the load balancing performance is.

3.2 Analysis of experimental results

The experimental results of load balancing performance between the traditional digital media information distributed storage system and the metadata based digital media information distributed storage system designed in this paper are shown in Figure 5.

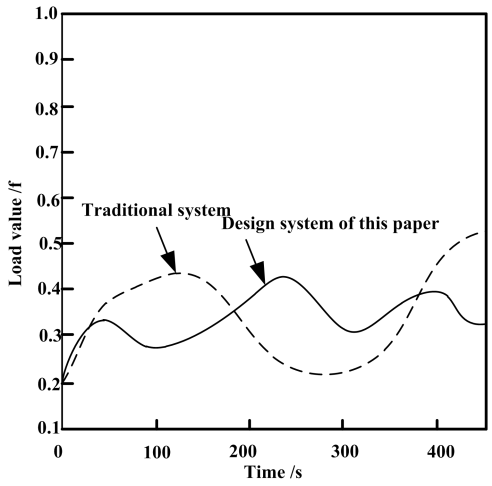

Fig. 5. Experimental results of load balancing performance

According to the experimental results of load balancing performance in Figure 5, the load balancing performance of the digital media information distributed storage system based on metadata is better than that of the traditional digital media information distributed storage system.

4 Concluding remarks

Due to the poor load balancing performance of traditional distributed storage system of digital media information, this paper designs a distributed storage system of digital media information based on metadata. Through the combination of hardware and software, the distributed storage of digital media information is realized. The distributed storage system of digital media information based on metadata can improve the load balance performance, which is of great significance for the distributed storage of digital media information.

References

[1]XING Hao-jian, QIAO Qiu-xiao, JIN Zhong-xie.: Monitoring Technology of Nuclear Power Primary Circuit Leakage Point Based on Distributed Temperature Sensor. Acta Photonica Sinica, Vol. 48, no. 5, pp. 161-167 (2019)

[2]JING Weizhe, LIU Yang, XIANG Yue, et al.: Energy Managemen Strategy of DCCHP Based on Demand Response. Electric Power Construction, Vol. 38, no. 12, pp. 68-76 (2017)

[3]HAN Wen-jun, YU Chun-sheng.: Distributed data storage architecture for data storage management of power transmission and transformation engineering. Journal of Shenyang University of Technology, Vol. 41, no. 4, pp. 366-371 (2019)

[4]DING Sheng, WU Kai-bin, RAO Yao, et al.: Research on the combined cycle energy supply system of solar-heat compensation for gas distributed energy under different organic working fluids. Journal of Thermal Science and Technology, Vol. 18, no. 2, pp. 137-142 (2019)

[5]WANG Jinwei, TANG Baofu.: Reliability Design of Distributed Temperature and Humidity Control System for Phased Array Radar.Modern Radar, Vol. 41, no. 7, pp. 85-89 (2019)

[6]ZHANG Zhigang, SUN Xintong, ZENG Zhihui.: An Intelligent Defogging System for Distributed Temperature Measurement of Vehicle Windows based on Wireless Transmission of A7139.Chinese Journal of Electron Devices, Vol. 42, no. 4, pp. 1076-1080 (2019)

[7]LI Xianglong, FU Xiao, ZHU Jie, et al.: Research on partition method and selection of dominant nodes for distribution network with distributed generation. Power System Protection and Control, Vol. 47, no. 13, pp. 24-29 (2019)

[8]WANG Dawei, HU Yanhui, LI Lin, et al.: Application of distributed automatic quantitative loading system in Myanmar Oil Depot. Oil & Gas Storage and Transportation, Vol. 37, no. 6, pp. 710-714 (2018)

[9]XU Mingdong, CHENG Yukai, ZHANG Huanyue, et al.: Design of distributed wi reless illumination measurement system. Journal of Dalian Dalian Polytechnic University, Vol. 37, no. 4, pp. 291-295 (2018)

[10]XIONG Xianming, CUI Xiangliang.: Improvement of vehicle identification method for φ-OTDR fully distributed optical vibration sensing system. Laser Journal, Vol. 39, no. 6, pp. 70-73 (2018)