Research and Implementation of Distributed Data Cache Based on JGroups Technology

Linliang Zhang[1, a *], Pengcheng Yue[1, b], Zhaoxia Li[1] and Huailiang Chen[2]

[1] Shanxi Transportation Research Institute, XueFu Street 79th, Taiyuan City, Shanxi Province, China

[2]Baidu Online Network Technology Co. Ltd, Beijing City, China

[a]zhanglinliang@163.com, [b]yuepengcheng241@126.com

Keywords: Distributed data cache; Data redundancy backup; Failure recovery; Peer-To-Peer; J Groups

Abstract. Distributed Data Caching Technology plays an important role in the system architecture. It can reduce the load of database access layer and improve overall system performance. Because of the shortage in data redundancy backup and failure recovery of existing products in the distributed data cache, this paper design and implement a Distributed Data Caching System, which has the function of data redundancy backup and failure recovery. This system uses the Peer-To-Peer topology, achieves two caches of different data distribution patterns, replicated cache and distributed cache, uses J Groups technology to achieve communication and data migration between cluster nodes, and completes the data redundancy backup and failure recovery depending on reliable service module. In the test, Distributed Data Caching System of the paper designed provides reliable data redundancy backup and failure recovery mechanisms, and the system is stable.

Introduction

In recent years, many cache products have appeared, such as the Memcached, JBoss Cache, OSCache, Ehcache, etc., but these distributed cache [1] products there exist certain disadvantages in terms of data redundancy backup and failure recovery. Memcached [2] does not support the redundant backup, its server cache content of all nodes is different, if a node fails, the data stored in this node is lost, so Memcached has single-node failure problem. JBoss Cache support two redundant policies: Global replication and Buddy replication. Global replication copies the data to all nodes in the cluster, ensuring the data can be transferred to any node in the cluster when it fails, but it limits the scalability of the system. Buddy replication choose specific nodes as the backup data nodes, but its redundant backup nodes are set through XML file, when the backup node fails, it can't start a new node as failure node. OSCache provides the functionality of the cluster is very limited, can't let the cache data replication between nodes. Ehcache is a simple cache system developed by Java language, don't provide redundancy backup and failure recovery function.

Therefore, it is significant to design and implement a distributed data cache system which has the function of data redundancy backup and failure recovery. This research comes from an electronic ticket verification website projects.

Caching framework design

Distributed cache system designed in this paper adopted the Peer-To-Peer [3] topology. Each cache node in the cluster knows the existence of all other nodes, can communicate with any node. This structure has a single hop data distribution and the characteristics of low latency. The cache system as the electronic ticket inspection data cache layer's official website, the function is complete the database query result cache, reduce customer request processing time, reduce the load on the database server and also with data redundancy backup and failure recovery of function. Caching system is mainly composed of the following modules: cache management module, replication, cache module, distributed cache module, data distribution module, communication module replacement algorithm,

the cache synchronization module, cache and reliability of service module. Caching system design framework as shown in Fig. 1.

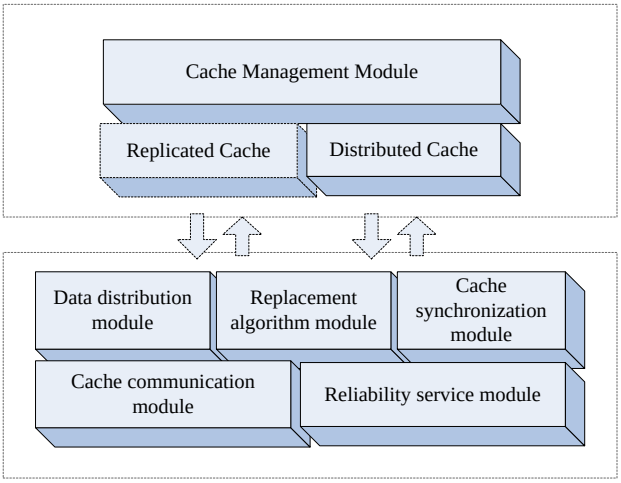

Figure 1. Caching System Design Framework Diagram

System main module design

Based on modular design idea and the object-oriented design method, the system developed by Java language. It has a good scalability. Design and realization of the main module is described below.

Cache management module. Cache management module in this system was implemented by the CacheManager class, it is the entry point of the caching system, is responsible for initializing all configuration information of system, establish and manage the cache object, and coordinate work among various modules. Initialization process is as follows: first, construct the CacheManager object, call the init method, this method first calls parseConfiguration method of ConfigurationFactory class to construct a Configuration class object, and then use the Configuration object construct a ConfigurationHelper class object, finally the ConfigurationHelper object as parameters call configure method of CacheManager class to complete the initialization of all configuration information.

Cache the data distribution and synchronization. Distributed data cache system is designed in this paper, data must be cached on multiple machines in a cluster. so it is very important that how to distribute the cached data among the machine nodes in the cluster. It guarantees the validity of the cache data on the cluster. When a node in the cluster has an expiration cache data or a data update, how to ensure that all cache data synchronization between nodes, which requires effective synchronization mechanism to ensure that the data between the cache node is the same. Caching system designed in this paper has realized the two kinds of distributed cache: replicated cache and distributed cache.

Copied cache is that the cache content of every machine in the cluster nodes is consistent, each node has a full backup of the cache data. When one copy in the cache data update, do not need data distribution algorithms, it can directly update the data to copied cache of the local machine, then call the communication module to update messages sent to the other nodes in the cluster, ensure that all nodes cache data synchronization. Update process as shown in Fig .2.

When querying data from the copied cache, the system directly lookup data from the local machine, the cache if present, direct return. Otherwise, query data from the local database, put the query results in the local cache, return the query results, at the same time call communication module cache to update messages sent to the other nodes in the cluster. Query process is shown in Fig .3.

Copy the cache synchronization mechanism is TTL mode and client failure mode. The client failure mode is when the local replication cache data update, call communication module send updates to other nodes in the cluster, ensure that all nodes cache data synchronization, the client failure mode is implemented by means of registered listeners to replicate the cache.

Copy the cache synchronization mechanism is TTL mode and client failure mode. The client failure mode is when the local replication cache data update, call communication module send updates to other nodes in the cluster, ensure that all nodes cache data synchronization, the client failure mode is implemented by means of registered listeners to replicate the cache.

TTL mode would provide a lifecycle (TTL) when a data object inserted in the cache. When get a object in the cache it would first check if the TTL expired. If overdue, TTL mode would delete it, query a new one from the local database, and put it into the cache. Then background will start a thread continuously detect whether the object in the local cache expiration, if overdue, it is removed from the cache data, finally also calls the communication module to update messages or failure messages sent to the other nodes in the cluster. Replicated Cache class implement the replication cache, and it provides all of the API interface that operated copied cache.

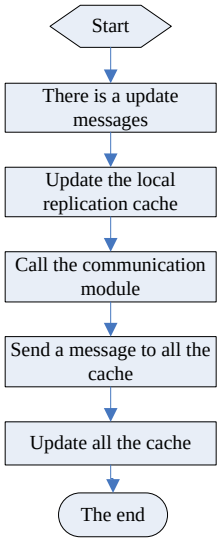

Figure 2. Replicated Cache Updating Process

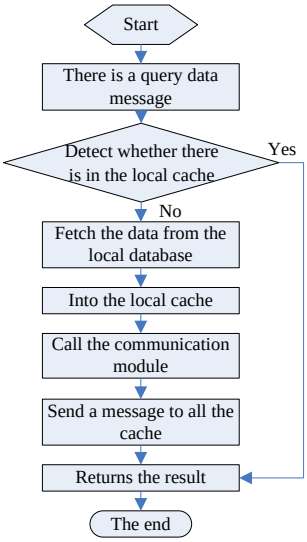

Figure 3. Replicated Cache Querying Process

Using a powerful data distribution algorithms, distributed [4] cache the data distribution of each node in the cluster. each node saves a part of data. The entire cache data scattered to each node in the cluster. At the same time backup data is also distributed of different nodes, which ensure the reliability of the system. First discuss distribution algorithm used by the distributed cache: Consistent Hashing [5] is a hash algorithm, a given data mapping for a 32 bit hash value, a numerical space between 0 to 232 and see it as a head-tail circle, the space as shown in Fig .4.

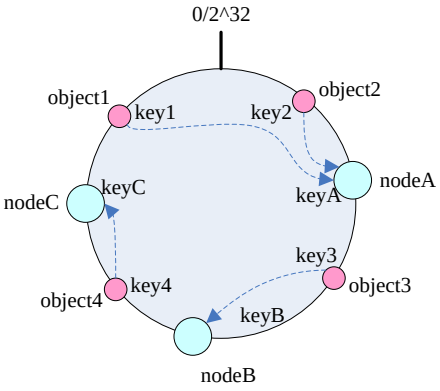

Figure 4. Consistent Hashing Algorithm(1)

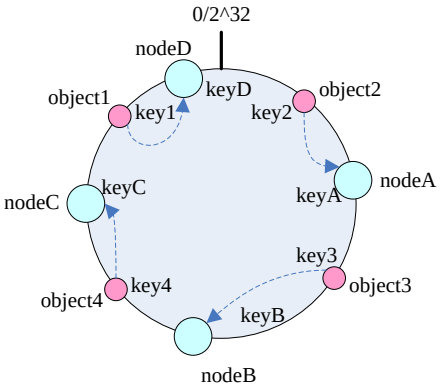

Figure 5. Consistent Hashing Algorithm(1)

Cache the data distribution process: first of all find the hash value of all the machine nodes in the cluster. Distributed caching uses the IP address of each machine as the key to the Hashing function.

Map all the hash values calculated to the 0 to 232 circle; Then use the same hash function to seek the hash value of the cached data keys, and map it to a circle. From the map to the location of the key to start a clockwise search, the data is saved in the first cache node to find; If more than 232 can't find the node, the cache data is saved on the first node. At the same time, the next node, which is the next node in the clockwise direction, is used as the backup node of the data. Shown in Fig .4, Object1 mapped to the location of more than 232 has not yet found the cache node, the object1 will be saved on the first node nodeA, nodeB as a backup node.

If add a new machine node D in the distributed cache system, and mapped to Fig .4 node C and node A, shown in Fig .5, the original mapping to nodeA object1 now to redistribution in node D, only the data between node C and node D to redistribution. It can be seen from the above, the consistency hash algorithm to maximize the suppression of the redistribution of the key, and increase the reliability of the cache system.

When inserting data into the distributed cache, the node of the data storage and the backup node are found according to the consistency hash algorithm, and the communication module is called to save the data to the machine node and the backup node. The process of updating and deleting is the same as the process of data insertion. To retrieve data from the distributed cache that you need invoke the consistent hash algorithm to find nodes storing the data and then call the cache communication module to read data. If there is data in the buffer of a node, call the communication module to return the result, else query data from the local database. the query result update to the nodes cache and the backup cache, results are re-turned. The query processing procedure is shown in Fig .6.

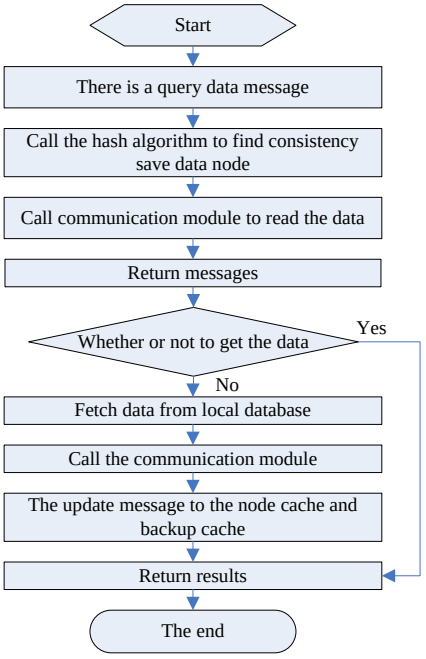

Figure 6. Distributed Cache Querying Process

Synchronization mechanism for distributed caching is TTL mode and client failure mode, but it is not entirely different from the cache copy. TTL mode would provide a lifecycle (TTL) when a data object inserted in the cache. When get a object in the cache it would first check if the TTL expired. If overdue, TTL mode would query a new one from the local database, and put it into the cache. Then background will start a thread continuously detect whether the object in the local cache expiration, if overdue, it is removed from the cache data, and sent to the other nodes in the cluster. Client failure mode is defined as the distributed cache data update or delete, first according to the consistent hashing algorithm to find the saved the data nodes and backup nodes and communication module is called to the message to the nodes and data backup nodes to achieve cache data to update or delete. Distributed

caching is implemented by the DistributedCache class, which provides all the API interfaces that operate the distributed cache.

Replacement algorithm module. Cache replacement algorithm points out how to select the object in the cache when the cache space is full. The cache system to achieve the three traditional replacement algorithms: first in first out algorithm (FIFO), the least recently used algorithm (LRU), the least frequently used algorithm (LFU).

Cache communication module. The cache communication module provides the interaction between nodes in the cluster. The system develops a set of communication protocol based on JGroups technology, which is used for the communication between nodes. The calling of communication module is realized by the way of copied cache and distributed cache registration monitor. The communication module is mainly implemented by the JGroupsManager class, and the process of sending the message by the cache is as follows:

If the local cache data update, then generate a data update event, the event will inform the listener class JgroupsReplicator;

Updated objects as a parameter, call the JgroupsReplicator class notifyElementUpdated () method, the method will update the message into a EventMessage class object;

EventMessage class object as a parameter, call the send JGroupsManager () method, the method of EventMessage class object encapsulated into class JGroupsMessage objects;

The final call to the JGroupsManager member variable NotificationBus class object sendNotification () method to send messages to other nodes in the cluster.

Replicated cache receive message process:

Cache system initialization, generation of JGroupsManager object according to the information in the configuration file, will monitor a port of the local machine. JGroupsManager realized interface org.jgroups.blocks.NotificationBus.Consumer;

When there is a message to the port, call the JGroupsManager class handleNotification() method, the received message is converted to the JGroupsMessage object;

JGroupsMessage class object as a parameter, and then call the JGroupsManager class handleJGroupNotification () method and the method extracts the relevant information from the JGroupsMessage class object, including name, event type, of the key of the data object and value;

According to the name to find the replicated cache object, and event to execute from the message.

The communication process of the distributed cache and replicated cache is slightly different, but the implementation of the bottom layer is both called the JGroupsManager class, which is not discussed here.

Reliability service module. Reliability service module main function is to provide cache nodes of data migration and failure recovery. When a new cache node joins the cluster or node removes from the cluster, reliability service module will be called dynamic and its reliability service is implemented by the Coordinator class. First discuss the reliability service of replicated cache, there is no need to do any processing when there is a cache node quits or failure in the Replicated Cache cluster, because all nodes in the cluster have a copy of the entire cache.

When a new cache node is added to the replicated cache cluster, the processing flow of the Coordinator class is called:

After the CacheManager class of the new join node is initialized, start Coordinator, it calls the communication module JGroupsManager class to send a message to all other nodes in the cluster. The message notify other nodes “I am a newly joined node” and the new one is to know the other nodes in the cluster;

After all the nodes that receive this message, first the information of the new node is added to the configuration information, and then return the new node a same type of news, the new node communication module JGroupsManager received this message, parses the message found it is a reply message, send the message to the coordinator, if receive the same message type again, no further treatment;

Parsing reply message Coordinator sends a request the migration data to the node that first reply message;

The node that receives the migrated data starts Coordinator class to prepare to the new node migrate data in the cache. Coordinator send data in the cache to the new node at intervals, until all of the cached data have moved so far, finally sent a migration data is complete message;

The coordinator of the new node constantly receiving cached data from the migrate nodes, putit into the local cache. In this process, the new node does not send update cached data message to other nodes until the data transfer is completed.

The distributed cache data distribution method is completely different from the replicated cache, so their data migration and failure recovery process is different. When a new cache node is added to the distributed cache cluster, the processing procedure of the Coordinator class is called as follows:

After the CacheManager class of the new join node is initialized, called consistent hashing algorithm to find the map their position on the circle. A new node nodeE, the map on the circle position is shown in Fig .7 shows, simply migrate the cached data on the nodeB release between nodeA and nodeE to the new node. Start Coordinator, call the communication module JGroupsManager class to send a message to all other nodes in the cluster. The message notify other node "I am a newly joined node", and then send a data transfer request to the nodeB node.

After all nodes receive this message, all the information of the new node is added to the configuration information, the nodeB receives the request of the migration data, start Coordinator, ready to migrate data to nodeE;

NodeB Coordinator to take data from its own cache, first determine the location of data mapping in the round, if it is in the scope of nodeE, the data object is encapsulated into a message send to nodeE. And put the data into your own backup cache in, and removed from the local cache(the data saved in the backup node of the node B has been deleted), until all cached data after test. Intermittent completion of this process to prevent network congestion;

NodeB Coordinator the next step to take the local backup cache data. To detect location of the data preservation, if the data of nodeE is not processed, otherwise sent the backup data to nodeE and tell it which is the backup data. nodeE put data into its backup cache, nodeB also delete these data from the local backup cache. The process is also performed intermittently until all the backup data is detected, sending a message to complete the migration;

At this point, the entire data migration work is completed.

See Fig .8 shows, assuming there are 5 nodes in a distributed cache clusters, now the nodeB is down suddenly, the process of failure recovery is as follows:

Assuming nodeA query a data, the data in the nodeB, nodeA sends a request message to nodeB. Wait for a period of time, if nodeA did not receive any reply, repeat sent two requests to the nodeB. If you still haven't received a reply, then to the backup node nodeC request data. NodeC sends the data to nodeA. After received data nodeA notice nodeC that nodeB maybe had been a failure.

NodeC sends a query message to nodeB, if nodeC received a reply message from nodeB, indicating that the nodeB is normal; if after waiting for a period of time and repeated several times to inquire data, have still not received the reply message, determining nodeB failure;

NodeC sends nodeB invalidation message to all other nodes, all nodes delete the information of nodeB node from configuration information;

NodeC Coordinator take the local backup cache data into the local cache for saving, at the same time to update the backup node cache of the nodeC, and delete data from the local backup cache until the process is completed;

NodeC Coordinator sends data backup request to the counter clockwise direction of nodeE. After received the messages nodeE check whether their backup node is nodeC. If it is, will start coordinator sends data backup to nodeC node, or no treatment.

After the completion of the backup data from nodeE to nodeC, the whole work of the failure to restore completed.

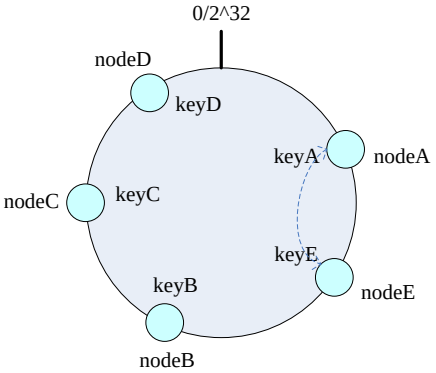

Figure 7. Distributed Cache Data Migration

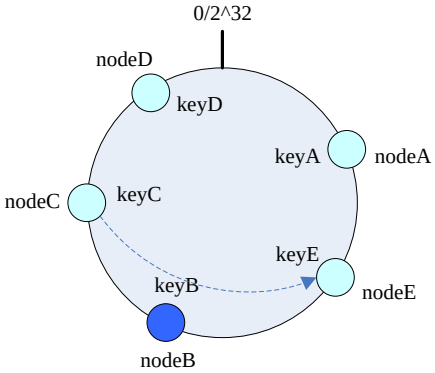

Figure 8. Distributed Cache Failure Recovery

Conclusion

This paper realizes two kinds of different distribution patterns: replicated cache and distributed cache. They all have the function of data redundancy backup and failure recovery, and have good stability. The replicated cache is adapted to be used in clusters with fewer nodes, and the distributed cache is adapted for use in a large-scale cluster server. At present, the cache system has been applied in the test environment of the data cache layer of the electronic ticket checking official website. The system runs stably, and it can meet the demand of the actual application. But there is a obvious shortcomings in the system to realize the distributed cache is when the system performs data redundancy backup or failure recovery function, significantly increases the amount of data transmission in the network, for network bandwidth requirements is very high.

References

[1] Joseph Issa, Silvia Figueira, Hadoop and memcached, Performance and power characterization and analysis, J. Journal of Cloud Computing. 1 (2012) 1-20.

[2] Rui Li, Jiapei Fan, Xinxing Wang, Zhen Zhou, Huayi Wu, Distributed cache replacement method for geospatial data using spatiotemporal locality-based sequence, J. Geo-spatial Information Science. 18 (2015) 171-182.

[3] Qiu Dongyu, R. Srikant, Modeling and performance analysis of BitTorrent-like peer-to-peer networks, J. Acm Sigcomm Computer Communication Review. 34 (2010) 367-378.

[4] X Yu, Z Kedem ,A distributed adaptive cache update algorithm for the dynamicsource routing protocol, J. Proceedings IEEE INFOCOM. 1 (2003) 730-739.

[5] Millar, Grant Paul, E. A. Panaousis, C. Politis. Distributed Hash Tables for Peer-to-Peer Mobile Ad-hoc Networks with Security Extensions, J. Journal of Networks. 7.2(2012).