Journal of Computational Science & Engineering 8 (2013) 310-313

Journal of Computational Science & Engineering

Available at http://www.asocse.org ISSN 1710-4068

Design and Realization of ActiveMQ Based High Performance Message Cluster

Guangxuan Chen[1, a], Yanhui Du[1,b], Panke Qin[2,c] and Lei Zhang[3,d]*

[1]*People's Public Security University of China, Beijing, China*
[2]*Beijing University of Posts and Telecommunications, Beijing, China*
[3]*The Logistics Academy, Beijing, China*

Article Information

Article history:
Received 7 August 2013
Revised 14 August 2013
Accepted 20 August 2013
Available online 5 September 2013

Keywords:
ActiveMQ
Data synchronization
Fault recovery
Load balancing
Message cluster

Abstract

In order to solve the performance bottleneck problems and enhance the reliability of the message cluster, this paper proposed a new solution. By introducing ActiveMQ component and message agent, the message cluster can extend the single middleware into the message middleware environment, and thus improved the availability of the data in the shared file system and database. Meanwhile, a new algorithm and mechanism of load balancing, data synchronization and fault recovery are introduced into the message cluster that greatly enhanced the performance and reliability of the cluster.

1. Introduction

With the rapid development of internet, network servers must possess strong network computing processing power and ability of providing a large number of concurrent access services so as to better deal with the sharp increase of accesses and even complicated transactions. However, the limit of processing power, memory, I/O and bandwidth of the single network server became the bottleneck in facing the large number of tasks. Traditional way to solve these problems is to adopt expensive high performance mainframe or SMP computer. This is not always effective when more processing requests and data surge to the server. In addition, the low efficiency and unreliable transmission among different hardware platform and different network

* Corresponding author: [a]ericcgx@qq.com, [b]dyh6889@gmail.com, [c]qinpanke@gmail.com, [d]zxcv123o@163.com

2013 © The American Computational Science Society. All rights reserved.

environment also bothered the customers [1].

In order to solve these problems, middleware and cluster are introduced in the study. Middleware is first used to solve distributed heterogeneous problems. Its main purpose is to shield the heterogeneity and complexity of the distributed environment and provide isomorphic distributed development platform for the users and developers.

Cluster refers to a group of independent servers in the network that provide reliable service for the users as a single system. Higher performance, higher performance-to-price ratio, better flexibility and higher availability made cluster system can effectively solve the bottlenecks and enhance the availability of the system in dealing with large scale requests and tasks. In the following text, we proposed a new solution of cluster, an ActiveMQ and message agent based cluster system that possesses high performance and reliability.

2. ActiveMQ Component

Apache ActiveMQ is a fast open source message component that supports clustering, lightweight container and Java virtual machine of most cross-language clients and can seamlessly integrates with the J2EE1.4 container. ActiveMQ supports two types of message transmission mode, i.e., Point-to-Point mode and Publisher/Subscriber mode [2]. This paper adopted Point-to-Point mode, its development flow is introduced as follows.

Producer's development flow. The Producer generates messages (shown in Figure 1):

● Create Connection: create a JMS Connection according to the url, user and password.

● Create Session: create a Session on the base of Connection and indicate whether support transaction and ACKNOWLEDGE.

● Create object Destination: specify the name of the subject; the producer and consumer will send and receive the corresponding message through the subject.

● Create MessageProducer: create object MessageProducer according to Destination, and set the persistent mode at the same time.

● Send message to Queue: encapsulate TextMessage and send out the message through the send method of MessageProducer.

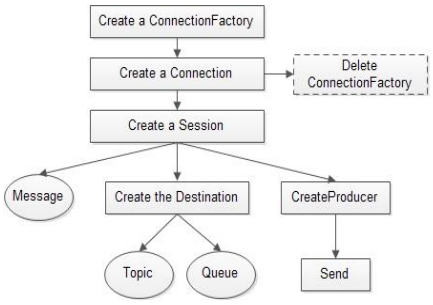

Fig. 1 Producer's workflow – Send message

Consumer's development flow. Consumer consumes the messages (shown in Figure 2):

● Implement interface MessageListener: class consumer should first implement the interface MessageListener, and then monitors and processes the messages through method onMessage().

● Create Connection: create a JMS Connection according to url, user and password. A clientID is needed for the Connection if the Connection is under durable mode.

● Create Session and Destination: similar to that in producer's development flow.

● Create replyProducer (optional): replyProducer is used to send replies to the producer.

● Create MessageConsumer: create Message Consumer according to the Destination.

● Consume message: receive the messages from producer through method on Message and send replies to producer through method replyProducer.

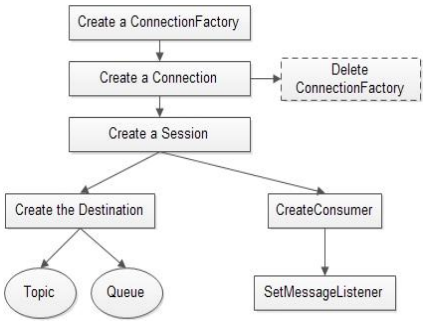

Fig. 2 Consumer's workflow – Receive message

2013 © The American Computational Science Society. All rights reserved.

ActiveMQ Cluster

Integral Structure of ActiveMQ based cluster. The ActiveMQ based cluster is consisted of client agent, load balancer and server.

● Agent: Agents are used to shield the underlying differences between the system and the networks so as to realize the transparent transmission among application layers. In addition, agents help the clients send the messages.

● Load balancer: Load balancer is used for maintaining the connections between agents and servers and recording the locations of the queues and subjects. Load balancers are connected through high-speed intranet and each of them is connected to all the servers in the system. Most of the time, load balancer transmits clients' tasks and calculates the servers' load. Load balancer forms a many-to-many relationship between clients and servers. This makes it possible that one client can access multiple queues on different servers, or one server can process multiple accesses from different clients.

● Server: Servers are responsible for the storage and transmission of the messages.

The whole structure between agents and servers can be seen as middleware. While, the agent, load balancer and server formed a message middleware system that supporting cluster. As can be seen from Figure 3, server 1 and server 2 share the queues and subjects that originally should accomplished by one server. Ideally, this new solution doubled the transmission efficiency [2].

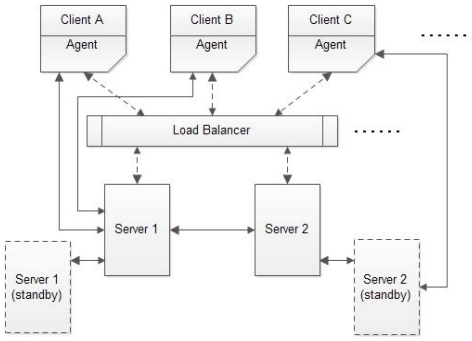

Fig. 3 Integral structure of ActiveMQ based message cluster

ActiveMQ's function. ActiveMQ implements the function of cluster server through message agent. The message agents in the same network are visible to the others and keep communication with each other. All the clients are equipped with agents and communicate with each other through agents.

On the whole, each client exhibits an agent to the outside. All the agents share the common file system and database. If one client downtime, the other agents will take over its tasks in order to guarantee the cluster run smoothly. Looking into the inside of the clients, each client equipped with multiple message agents, but only one agent will provide service for the outside in the same time period and the other agents will act as slave machines. Once the master server is downtime, the client will immediately enable the slave servers to take over master server's tasks. Sharing the file system and database between the master and slave servers can ensure the high availability and reliability of the data.

In order to guarantee the performance and rate of the transmission of the messages, each server is equipped with a standby. By transparently increasing the number of server to share the same number of queues and subjects, each server will bear less number of queues and subjects and thus the efficiency of accessing and transmitting the message will enhanced obviously.

3. Optimization mechanism for the cluster

In order to optimize the performance of the cluster, load balancing, data synchronization and fault recovery are introduced into the cluster.

Load balancing. Load balancing is achieved through two levels. The first level is load balancing among different load balancers. The user specifies one or multiple load balancers for the agent. The agent will first judge the load status of the load balancer and then choose the active one with the lightest load according to the load balancing algorithm.

2013 © The American Computational Science Society. All rights reserved.

The second level refers to the load balancing between servers. The load balancer will choose a server with the lightest load to establish queue/subject according to the weighted algorithm, shown in Eq. 1.

$$L(s) = k_1 \times L_{cpu} + k_2 \times L_{mem} + k_3 \times L_{I/O} + k_4 \times L_{net} . \quad (1)$$

Here, $L(s)$ refers to the load of the server; L_{cpu} means the cpu occupancy of the server; L_{mem} is the utilization ratio; $L_{I/O}$ is the I/O status and L_{net} is the bandwidth occupancy. k_1, k_2, k_3 and k_4 are weight factors. Bandwidth is relative important in the message cluster, so we can increase the weight of bandwidth accordingly. For example, the weight can be assigned as $k_1 = k_2 = k_3 = 0.2, k_4 = 0.4$.

Data synchronization. Data synchronization in the cluster system reflects in two aspects: data synchronization of queue/subject tables among load balancers and data synchronization between master servers and hot-standbys.

As for the first one, each load balancer maintains its own index. When the required queue is not available or failed to open the specific queue in the index, the data synchronization mechanism will maintain that the index is error and then retrieve the queue/subject tables from each servers and generate global queue/subject table.

As for the second one, only the master server provides services for the outside at a certain moment, while the hot-standby only stores the relative data. The master server and hot-standby both adopt duplex service software to monitor each other's running status. The well-run one will aware the failures of the one at the other end and take corresponding measures when one of them is downtime.

Fault recovery. Throughout the cluster system, load balancer and servers need fault recovery mechanism in case of downtime.

As for the load balancer, fault recovery is not necessary if there is multiple load balancers in the cluster as if one of them downtime will not affect the work of the entire cluster. The client will automatically search and reconnect to the load balancer with the lowest load in the cluster according to the optimization rule when they lose connection with the original load balancer. However, hot-standby is necessary if there is only one load balancer in the cluster system. The standby will maintain the content of queue/subject tables and connection tables in real time.

As for the servers, each server adopts hot-standby for fault recovery. The hot-standby will take over the master server's tasks when it is downtime.

4. Conclusion

The limit of processing power, memory, I/O and bandwidth of the single network cluster became the bottlenecks in facing large scale of requests and processing tasks. The ActiveMQ based message cluster we designed can effectively release the processing stress that bothered the clients. By introducing load balancing, data synchronization and fault recovery, the performance and reliability of the cluster enhanced greatly. Meanwhile, the availability and manageability of the data in the cluster also improved obviously.

References

[1] Y.T. Huang, Q.K. Chen, Computer Applications, 1 (2008) 52-55.

[2] B. Synder, D. Bosanac, R. Davies, ActiveMQ in Action, Manning, New York, 2011.

2013 © The American Computational Science Society. All rights reserved.