Research on Load Balancing and Database Replication based on Linux

Ou Li*, Yan Chen, Taoying Li

College of Transportation Management, Dalian Maritime University, China

*liou_dlmu@163.com

Keywords: load balancing; database replication; Apache; Tomcat; MySQL

Abstract. With development of computer and network technology, people access to the Internet grew exponentially, data become an integral part of people's life. More and more companies begin to use load balancing and database replication technology, to ease the pressure on access to network devices and ensure data integrity. In this paper, the structure of load balancing cluster is analyzed, the process of integrating Apache and Tomcat based on Linux OS is described. On this basis, MySQL database replication is completed. Test results show that the performance of the load balancing cluster system is significantly higher than that of the single server, to a certain extent, database replication technology can ensure the integrity and accuracy of data.

1 Introduction

With the continuous development of science and technology, network information resources are increasingly rich, and people have higher request to network service. Network devices, such as server and database and so on, are the basis for the user to have a good Internet experience. However, network devices that are used by some websites may not be able to load large amounts of concurrent access and data[1].

Load balancing cluster can share a large number of concurrent accesses to each sub-server, so as to improve the performance of the system[2]. Compared to traditional servers, the cluster system has the advantages of expansibility, fault tolerance, higher throughput, maintainability and more cost-effective[3].

The meaning of database replication is to keep two or more database content consistent, or keep part of content consistent as needed. Database replication is able to enhance the availability of the database, enhance the fault tolerance ability of the network. It meets the needs of the current application, and is widely used.

In this paper, we study load balancing cluster and database replication based on Linux operating system. The structure is organized as follows: Section 2 introduces the load balancing cluster architecture, method of integrating Apache and Tomcat, and establishes load balancing cluster. Section 3 presents the principle of MySQL database replication, and implements database synchronization function. Finally, Section 4 concludes the paper and points out further work.

2 Load Balancing Cluster

2.1 Load Balancing Cluster Architecture.

Load balancing cluster architecture of integrating Apache and Tomcat through JK plug-in is shown as Fig.1.

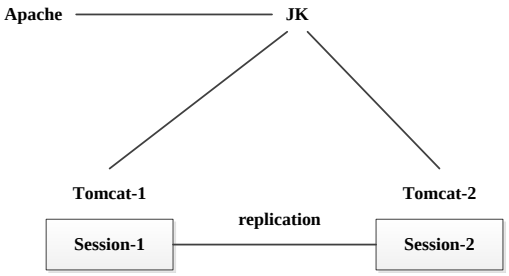

Figure 1. Load balancing cluster architecture

When a user sends a request through the browser, Apache first determines the type of request, if the request is static, then by the Apache analysis, and the results back to the client. If the request is dynamic, such as JSP, Apache will give the job to JK, JK will assign jobs to Tomcats according to load balancing algorithm. Tomcat analysis is completed, results back to the client by Apache. JK plays the role of scheduling in this process, avoids the situation occurrence, that one Tomcat too busy, another idle. So that Tomcats will achieve the purpose of cooperation and improve performance.

2.2 Establish Load Balancing Cluster

In the Linux operating system, after the correct installation of the configuration software, through the integrating Tomcat and Apache to establish the load balancing cluster.

Set Tomcat. Modify the configuration file server.xml of Tomcat. First, <Connector> is used when AJP is connected to Apache and Tomcat. Modify <Connector> content. Then, remove comments of <Engine >, named for the Tomcat that we have installed. That is, modify the value of the jvmRoute to "Tomcat instance name". In this paper, we install two Tomcats, and named as tomcat1 and tomcat2. Finally, add <Cluster> content to <Engine> or <Host> elements. Final configuration file of tomcat1 is shown as Fig.2.

```
<?xml version="1.0" encoding="UTF-8"?>
- <Server shutdown="SHUTDOWN" port="8005">
    <Listener SSLEngine="on" className="org.apache.catalina.core.AprLifecycleListener"/>
    <Listener className="org.apache.catalina.core.JasperListener"/>
    <Listener className="org.apache.catalina.mbeans.ServerLifecycleListener"/>
    <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener"/>
  - <GlobalNamingResources>
      <Resource pathname="conf/tomcat-users.xml" factory="org.apache.catalina.users.MemoryUserDatabaseFactory" description="User
        database that can be updated and saved" type="org.apache.catalina.UserDatabase" auth="Container" name="UserDatabase"/>
    </GlobalNamingResources>
  - <Service name="Catalina">
      <Connector port="8080" redirectPort="8443" connectionTimeout="20000" protocol="HTTP/1.1"/>
      <Connector port="8009" redirectPort="8443" protocol="AJP/1.3" protocolHandlerClassName="org.apache.jk.server.JkCoyoteHandler"
        maxSpareThreads="512" minSpareThreads="25" maxKeepAliveRequests="200" maxThreads="9216"/>
      <Engine name="Catalina" jvmRoute="tomcat1" defaultHost="localhost">
        <Realm className="org.apache.catalina.realm.UserDatabaseRealm" resourceName="UserDatabase"/>
      - <Host name="localhost" xmlNamespaceAware="false" xmlValidation="false" autoDeploy="true" unpackWARs="true" appBase="webapps">
        - <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster">
            <Manager className="org.apache.catalina.ha.session.DeltaManager" notifyListenersOnReplication="true"
              expireSessionsOnShutdown="false"/>
          - <Channel className="org.apache.catalina.tribes.group.GroupChannel">
              <Membership port="45564" className="org.apache.catalina.tribes.membership.McastService" dropTime="3000"
                frequency="500" address="228.0.1.99"/>
              <Receiver port="5000" className="org.apache.catalina.tribes.transport.nio.NioReceiver" maxThreads="6"
                address="172.18.145.104" selectorTimeout="100"/>
            - <Sender className="org.apache.catalina.tribes.transport.ReplicationTransmitter">
                <Transport className="org.apache.catalina.tribes.transport.nio.PooledParallelSender"/>
              </Sender>
              <Interceptor className="org.apache.catalina.tribes.group.interceptors.TcpFailureDetector"/>
              <Interceptor className="org.apache.catalina.tribes.group.interceptors.MessageDispatch15Interceptor"/>
              <Interceptor className="org.apache.catalina.tribes.group.interceptors.ThroughputInterceptor"/>
            </Channel>
            <Valve className="org.apache.catalina.ha.tcp.ReplicationValve"
              filter=".*\.gif;.*\.js;.*\.jpg;.*\.png;.*\.htm;.*\.html;.*\.css;.*\.txt;"/>
            <Deployer className="org.apache.catalina.ha.deploy.FarmWarDeployer" watchEnabled="false" watchDir="/tmp/war-listen/"
              deployDir="/tmp/war-deploy/" tempDir="/tmp/war-temp/"/>
            <ClusterListener className="org.apache.catalina.ha.session.ClusterSessionListener"/>
          </Cluster>
        </Host>
      </Engine>
    </Service>
</Server>
```

Figure 2. Configuration file of tomcat1

At the same time, we need to modify configuration file of tomcat2. Configuration file of tomcat1 and tomcat2 are a little different, as shown in Fig.3.

```
<?xml version="1.0" encoding="UTF-8"?>
<Server shutdown="SHUTDOWN" port="8005">
  <Listener SSLEngine="on" className="org.apache.catalina.core.AprLifecycleListener"/>
  <Listener className="org.apache.catalina.core.JasperListener"/>
  <Listener className="org.apache.catalina.mbeans.ServerLifecycleListener"/>
  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener"/>
  <GlobalNamingResources>
    <Resource pathname="conf/tomcat-users.xml" factory="org.apache.catalina.users.MemoryUserDatabaseFactory" description="User
      database that can be updated and saved" type="org.apache.catalina.UserDatabase" auth="Container" name="UserDatabase"/>
  </GlobalNamingResources>
  <Service name="Catalina">
    <Connector port="8080" redirectPort="8443" connectionTimeout="20000" protocol="HTTP/1.1"/>
    <Connector port="8009" redirectPort="8443" protocol="AJP/1.3" protocolHandlerClassName="org.apache.jk.server.JkCoyoteHandler"
      maxSpareThreads="512" minSpareThreads="25" maxKeepAliveRequests="200" maxThreads="9216"/>
    <Engine name="Catalina" jvmRoute="tomcat2" defaultHost="localhost">
      <Realm className="org.apache.catalina.realm.UserDatabaseRealm" resourceName="UserDatabase"/>
      <Host name="localhost" xmlNamespaceAware="false" xmlValidation="false" autoDeploy="true" unpackWARs="true" appBase="webapps">
        <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster">
          <Manager className="org.apache.catalina.ha.session.DeltaManager" notifyListenersOnReplication="true"
            expireSessionsOnShutdown="false"/>
          <Channel className="org.apache.catalina.tribes.group.GroupChannel">
            <Membership port="45564" className="org.apache.catalina.tribes.membership.McastService" dropTime="3000"
              frequency="500" address="228.0.1.99"/>
            <Receiver port="5000" className="org.apache.catalina.tribes.transport.nio.NioReceiver" maxThreads="6"
              address="172.18.145.117" selectorTimeout="100"/>
            <Sender className="org.apache.catalina.tribes.transport.ReplicationTransmitter">
              <Transport className="org.apache.catalina.tribes.transport.nio.PooledParallelSender"/>
            </Sender>
            <Interceptor className="org.apache.catalina.tribes.group.interceptors.TcpFailureDetector"/>
            <Interceptor className="org.apache.catalina.tribes.group.interceptors.MessageDispatch15Interceptor"/>
            <Interceptor className="org.apache.catalina.tribes.group.interceptors.ThroughputInterceptor"/>
          </Channel>
          <Valve className="org.apache.catalina.ha.tcp.ReplicationValve"
            filter=".*\.gif;.*\.js;.*\.jpg;.*\.png;.*\.htm;.*\.html;.*\.css;.*\.txt;"/>
          <Deployer className="org.apache.catalina.ha.deploy.FarmWarDeployer" watchEnabled="false" watchDir="/tmp/war-listen/"
            deployDir="/tmp/war-deploy/" tempDir="/tmp/war-temp/"/>
          <ClusterListener className="org.apache.catalina.ha.session.ClusterSessionListener"/>
        </Cluster>
      </Host>
    </Engine>
  </Service>
</Server>
```

Figure 3. Configuration file of tomcat2

Set Apache. Httpd.conf is the configuration file of Apache. At the end of the file, load mod_jk Module, specify the path to workers.properties, set jk log level and where to put the jk log, select the log format, and so on. Expanded content is shown as Fig.4.

Create workers.properties file.The workers.properties file that added to the httpd.conf file does not exist, so we need to create our own. The workers.properties file used to register the workload manager(Tomcat). In this paper, two Tomcats are registered in the file, as workload managers. Workers.properties file is shown as Fig.5.

```
LoadModule jk_module modules/mod_jk.so
  〈Ifmodule mod_jk.c〉
    JkWorkersFile /usr/local/apache/conf/workers.properties
    #Where to put jk logs
    JkLogFile /usr/local/apache/logs/mod_jk.log
    #Set the jk log level [debug/error/info]
    JkLogLevel info
    #Select the log format
    JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"
    #JkOptions indicate to send SSL KEY SIZE,
    JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories
    #JkRequestLogFormat set the request format
    JkRequestLogFormat "%w %V %T %q %U %R"
    #JkShmFile to put logs
    JkShmFile /usr/local/apache/logs/mod_jk.shm
  〈/IfModule〉
```

Figure 4. Expanded content of httpd.conf

```
workers.tomcat_home=/usr/local/apache-tomcat-7.0.47
workers.java_home=/usr/local/java/jdk1.7.0_45
ps=/
worker.list=tomcat1, tomcat2, loadbalancer, status
# ----------------------------------------------------
# the first tomcat
# ----------------------------------------------------
worker.tomcat1.port=8009
worker.tomcat1.host=172.18.145.104
worker.tomcat1.type=ajp13
worker.tomcat1.lbfactor=1

worker.tomcat1.cachesize=1000

worker.tomcat1.socket_keepalive=1
worker.tomcat1.socket_timeout=300
worker.tomcat1.local_worker=1
worker.tomcat1.retries=3
# ----------------------------------------------------
# the second tomcat
# ----------------------------------------------------
worker.tomcat2.port=8009
worker.tomcat2.host=172.18.145.117
worker.tomcat2.type=ajp13
worker.tomcat2.lbfactor=1
# ----------------------------------------------------
# load balancer worker
# ----------------------------------------------------
worker.loadbalancer.type=lb
worker.loadbalancer.balanced_workers=tomcat1,tomcat2
worker.loadbalancer.sticky_session=1
worker.loadbalancer.sticky_session_force=0
worker.status.type=status
```

Figure 5. Workers.properties file

2.3 Test performance.

Using JMeter as a testing tool for server performance, simulate 1000, 1500, 2000, 2500, 3000 users to access the server separately. Access results of single server are shown as Table 1. Access results of load balancing cluster are shown as Table 2.

Table I. ACCESS RESULTS OF SINGLE SERVER

Samples / Label	1000	1500	2000	2500	3000
Average	1432	3897	5626	9481	12568
Median	1022	3406	3499	6662	9945
90%Line	3267	9418	10750	21008	21015
Min	11	6	10	19	20
Max	6108	10311	21099	21123	95966
Error %	0.00%	0.00%	7.60%	26.16%	35.83%

Table II. ACCESS RESULTS OF LOAD BALANCING CLUSTER

Samples / Label	1000	1500	2000	2500	3000
Average	201	827	1041	2123	2915
Median	35	259	402	377	534
90%Line	616	3007	3232	9013	9326
Min	2	2	2	2	2
Max	1511	3199	3814	9408	21090
Error %	0.00%	0.00%	0.00%	0.00%	1.17%

By comparing the table 1 and table 2, in the same amount of traffic, the average response time, 50% user response time, 90% user response time, minimum response time, Maximum response time and error rate of the load balancing cluster are greatly improved compared with the single server. Experimental results show that, the performance of load balancing cluster is significantly higher than that of single server.

3 MySQL Database Replication

3.1 Principle of Database Replication

MySQL replication is a solution, in which multiple MySQL database can achieve master slave synchronization. Asynchronous is its feature. It is widely used in different occasion, which needs MySQL have a higher performance and higher reliability requirements.

MySQL replication is an asynchronous replication process. Copy from one MySQL instance (we call it Master) to another MySQL instance(we call it Slave). Implementation process of the entire synchronization between Master and Slave is mainly completed by the three threads. Two threads (SQL thread and IO thread) in the Slave, another thread(IO thread) in the Master[4].

MySQL replication process is shown as Fig.6.

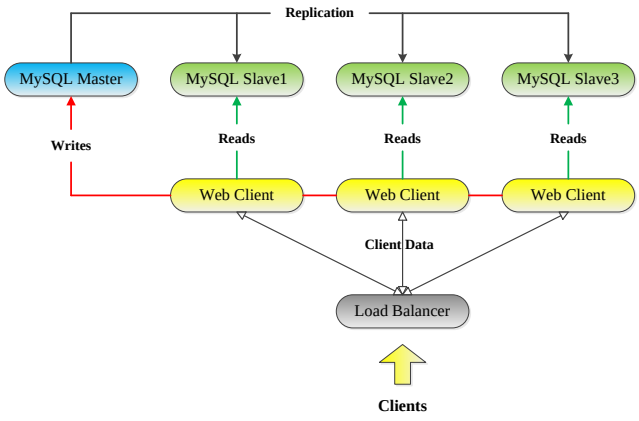

Figure 6. MySQL replication process

The basic process of MySQL replication is as follows[5]:

(1) Master records the binary log, after the completion, Master notification storage engine to commit transaction. Slave IO thread is connected to the Master, and requests to read the contents of the specified location in the specified log file.

(2) Master receives requests from Slave, reads the corresponding information according to the Slave request, and returns the result to Slave IO thread. The result includes log information, the name and location of the information returned, which is in the binary log file on the Master.

(3) After receiving the information, Slave IO thread will write the information to the end of the log relay file on the Slave, then, read the name and location of the bin-log file on the Master, and record the information to the master-info file.

(4) Slave SQL thread detects relay log file, after detection a new increase in the content of log relay, it will immediately parses the contents of the file. Slave SQL thread receives updates from the Master, and performs the copy of these updates. So the data on Master and Slave are exactly the same.

3.2 Implemention of Database Replication

Set Master. The Master's IP address is 172.18.145.104.

In the my.cnf file, which is the configuration file of MySQL, server-id is the unique identity of the database server, and usually server-id of Master is set to 1. Binlog-do-db is the name of the database that needs replication. MySQL replication is performed by using a binary log file, so need to open the MySQL log function by log-bin. In the paper, we set server-id = 1 , binlog-do-db=train, log-bin=mysql-bin.

After setup, grant replication permission to Slave. Then, show master status, the result is shown as Fig.7.

```
mysql> show master status;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000004 |       98 | train        |                  |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

Figure 7. Master status

Set Slave. The Slave's IP address is 172.18.145. 117.

In the configuration file of MySQL, set related information. In the paper, we set server-id = 2, master-host = 172.18.145.104 , master-user = root, master-password = root , master-port = 3306. Where master-host is the IP address of the Master, and master-user, master-password, master-port are account information for database replication, these correspond to grant replication permission account information.

Then, modify MySQL permissions. Finally, execute the contents shown in Fig.8. on the MySQL command line. Where master_log_file corresponds to Master’s File, and master_log_pos corresponds to Master’s Position.

3.3 Database Replication Test

Show Slave status, and the result is shown as Fig.9. MySQL replication uses Slave_IO_Running and Slave_SQL_Running, and status of them are “Yes”. This shows Slave IO thread and SQL thread run well, so MySQL replication is successful.

```
mysql> slave stop;
mysql> change master to
master_host ='172.18.145.104',
master_user='root',
master_password ='root',
master_port =3306,
master_log_file ='mysql-bin.000004',
master_log_pos =98,
master_connect_retry =10;
```

Figure 8. Contents to execute on the MySQL command line

```
mysql> show slave status \G;
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 172.18.145.104
                  Master_User: root
                  Master_Port: 3306
                Connect_Retry: 10
              Master_Log_File: mysql-bin.000004
          Read_Master_Log_Pos: 98
               Relay_Log_File: tomcat2-relay-bin.000002
                Relay_Log_Pos: 235
        Relay_Master_Log_File: mysql-bin.000004
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 98
              Relay_Log_Space: 235
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
1 row in set (0.00 sec)
```

Figure 9. Slave status

4 Conclusions

In this paper, we introduce the structure and process of load balancing, implement load balancing cluster by integrating Tomcat and Apache, and provide related configuration files. Then, we introduce the principle of database replication, implement MySQL replication by setting Mater and Slave. Finally, we prove that the performance of load balancing cluster is superior by simulation test. The future research work is doing research on load balancing algorithm, to improve working efficiency of the cluster.

Acknowledgment

This work is partially supported by National Key Technology Research and Development Program of the Ministry of Science and Technology of China(No. 2014BAH24F04), National Natural Science Foundation of China (No. 71271034), the National Social Science Foundation of China (Grant No.15CGL031).

References

[1] T Schroeder T, Goddard S, Ramamurthy B. Scalable web server clustering technologies. Network, IEEE, 2000, 14(3): 38-45.

[2] Z. Gao, X.J. Kang, “Research on Enhancing the Running Performance of Tomcat Server,” Computer and Digital Engineering, vol.36, issue 10, pp. 203-205, 2008.

[3] B.Q. Zeng, Z.G. Chen, “Research of Server Cluster System,” Application Research of Computers, vol.21, issue 3, pp. 186-187, 2004.

[4] G.W. Tian, “MySQL Replication Technology,” Nature Science Journal of Harbin Normal University, vol.31, issue 4, pp. 45-48, 2015

[5] W.L. Zhang, C.H. Jiang, J.C. Wei, “MySQL Replication Technology and its Applications,” Computer Science, vol.39, issue 11A, pp. 168-170, 2012.