The Comet-Based Implementation of Real-Time Drawing Board

Rui Yang, Chongwen Wang

Department of Software Engineering, Beijing Institute of Technology, Beijing, 100081, China

Abstract - Comet technique can transfer web data from a client to another in real-time, with the comet server as the message freight station. Using this technology to transfer messages has real-time and efficient advantages. In this paper, a web version real-time share drawing system is designed and implemented, according to the comet server push technology. This system support to transmit drawing information between users, and support a large number of concurrent access. The development in the client side use the flex technology and the system adopts ajax to request and send data. In the server side, system use java to develop service program, and the system is deployed on the tomcat server.

Index Terms - comet, flex, ajax, drawing board

Summary

Traditional drawing board is divided into two, one is based on the c / s architecture which needs to be installed before use, the other is based on media stream technology. However, the real-time sharing drawing board of the b / s structure is blank. With the extensive use of web application development, more and more users make such a demand that they hope to use a web-based real-time shared drawing board, making the drawing board combined with the existing web products tightly. When the user browse the webpage, they can use the drawing board at the same time , without having to install any software. As the comet technology is more and more mature, it is possible to realize such a system.

Although the Ajax technology can make internet applications faster and more friendly, the real-time shared drawing board in the b/s architecture need to transmit the changed information in the background to the client without the customer-side non-stop to refresh and send request. The comet server push technology can be a good solution to this problem. Comet technology, the server does not passively wait for user requests, but initiatively send the message to the client, and the client need not too often to send the request to the server to obtain information. So that the server can take the initiative to pass information to customers, and this technology has immediate, efficient, good performance and able to support a large number of users advantages.

System Architecture

The system is based on the comet server push technology. The system uses Adobe Flex technology, JSP technology, Ajax technology for the design and the implementation. Firstly, the system complete the drawing board with Flex, and then let the sketchpad client embedded in the webpage. The comet client send the drawing information to the comet server-side through the a Ajax long connection. Then the server-side parse the message, package the message and send the message to the destination (the drawing information receiver). The system architecture is shown in Figure 1.

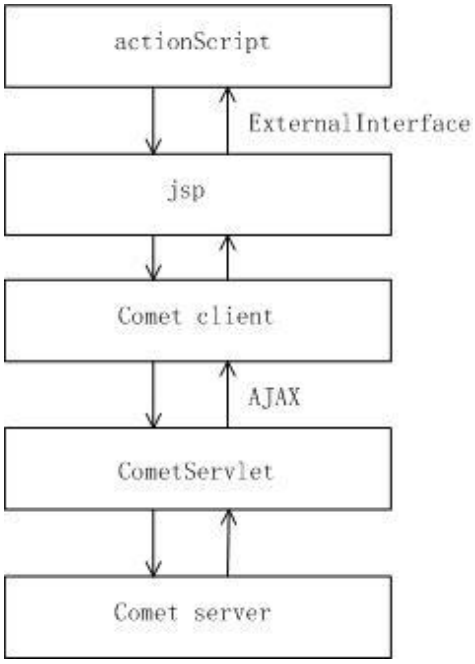

Fig.1 system architecture

Comet of the system mainly uses the Bayeux message protocol and java NIO technology. Bayeux based on Publish / Subscribe mode is a communication protocol for asynchronous and low latency message transmission between the browser and the server. Client and server are connected by long polling with Ajax technology, after the connection the return of the server-side data and client data requests is asynchronous (server-side asynchronous push message to the client side). The Bayeux support the message asynchronous transfer well, so the Web system can achieve high response for the user interaction to overcome the shortcomings of the traditional Web mode of communication. In the system, the client messages are initially accepted by the web container, and the connection between the client and the server is established by the http long connection, so web container io performance is very important. NIO is to supplement the lack of the traditional I / O that one thread handling only one connection. But with NIO, only one thread will be able to manage all the connections. When the request occurred, the request will be handed out to the threading, and it can take full advantage of

the system resources (threads). The system uses non-blocking I / O to manage the connection between the client and the server to improve server performance.

Comet is divided into the comet client and the comet server. Comet client and server side send handshake Bayeux messages to reach a consensual agreement, then the comet client sends a connection request for the long connection, after that the server-side asynchronous transfer data to the client. Comet control long connection through the NIO, so that the system has good performance.

System Achievement

Page Sketchpad

The Sketchpad client with as3 development requires basic canvas, buttons, and other information, as well as the button corresponding to the event handler. After the Graffiti button is selected, the process of moving the mouse start recording point information, and the point information is stored in the array. When the mouse is lifted, call the js function by the external interface, and send the point information to the js. Js access the array, the array sent to another client through the comet. Js needs to call function in as3 by the external interface when other users send draw information, then drawing board obtain the point group information for display. The key code are as follows.

```
private function onMouseUp(e : MouseEvent) : void {
    flag = false;
  ExternalInterface.call("jsGetObj",huiarray);
  huiz = 0;
  huiarray=new Array();
}
private function drawInit() : void {
    ExternalInterface.addCallback("getObj",getObj);
    .....
    }
```

Comet Client

Comet client and server-side interaction is shown in figure 2. Handshake ->long connection -> transmission of information.

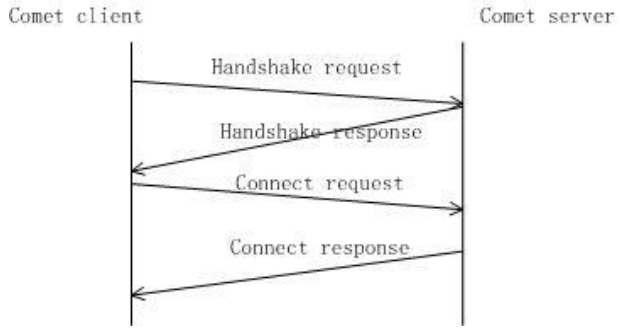

Fig.2 comet interaction

When connected to the web page of the drawing board, the comet client first initializes the the cometd configuration information. And then handshake with the comet server to negotiate the protocol version and the supported type of connection information. The key js code are as follows.

```
var cometdURL = location.protocol + "//" + location.host
            + config.contextPath + "/cometd";
        $.cometd.configure({
                url : cometdURL,
                logLevel : 'info'
            });
$.cometd.handshake();
```

After the handshake, the comet client access the handshake response message, and get the connection type and version information, then send long connection to the server. After that,the client and server can send JSON format messages.

```
function _connect()
{      var message = {
          channel: '/meta/connect',
          connectionType: _transport.getType()
       };
       _setStatus('connecting');
       _debug('Connect sent', message);
       _send([message], true, 'connect');
       _setStatus('connected');
}
```

After user draw on the drawing board, the comet client will send the point information to the comet server. The "chat" property stores points group chat attribute, the "lineVal" property stores line thickness information, and the "colorVal" properties stores color information.

```
    $.cometd.publish(_privateSendChanel, {
                type : 'senddraw_a',
                userId : _userId,
                userName : _userName,
                room : _targetRoom,
                peer : peerUserId,
                chat : config.drawArray,
                lineVal : config.lineVal,
                colorVal : config.colorVal,
                deliver : 'A-V',
                tenantid : config.tenantId
            });
```

Comet Server

Comet server subscribe the same channel as the client, and bind the appropriate method. When the service receives the user messages, it need to parse the message, assembly message and send the message to the comet client. Client receives the message and forward to the flash drawing board for display. The main code are as follows.

```
subscribe("/service/privatechat", "privateChat");
public void privateChat(Client client, Map<String, Object> data) {
Map<String, Object> message = new HashMap<String, Object>();
message.put("scope", "private");
```

```
message.put("userId", fromUserId);
message.put("userName", data.get("userName"));
message.put("chat", data.get("chat"));
message.put("lineVal", data.get("lineVal"));
message.put("colorVal", data.get("colorVal"));
message.put("time", time);
message.put("fontStyle", data.get("fontStyle"));
message.put("type", type);
message.put("visitorIp", visitorIp);
......
client.deliver(getClient(), selfChanel, message, null);
......
}
```

The Underlying NIO Treatment

As multiple users can use drawing board system, taking into account the server performance problems, the system can not use one connection handle one thread. In this system, NIO manage the connection between the comet client and the comet server. The NIO class diagram is shown in Figure 3 under.

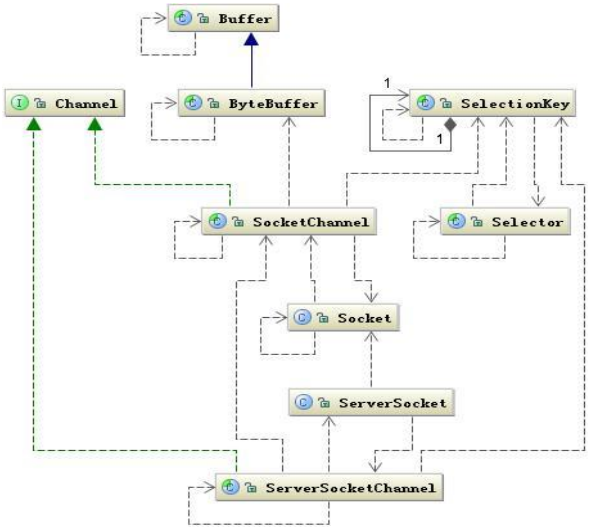

Fig.3 NIO class

The NIO kit consists of three parts. 1) Selector and SelectionKey: the Selector is a very important class to achieve the non-blocking I/O, and all of the channels and the corresponding events of interest are registered to the Selector object, and there is a loop continuously to monitor Socket channels registered on the Selector. When monitoring the event of interest has occurred on the Channel, it automatic produce a SelectionKey object, then handle of the event. 2) Buffer: Buffer object is equivalent to a data container, and it can be seen as a large array of memory used to store and extract the data of all the basic types. The core of the Buffer class is a memory area, and use operating system features and capabilities to enhance and improve the Java legacy I/O performance. 3) Channel: Channel is considered NIO toolkit an innovation. It is the channel between the buffer (Buffer) and I/O services. It both can be read can also write, and can transmit data more efficiently.

NIO of the system uses the event mechanism. First, the server accept client connection requests and after receive connections the channel and read event registered to the selector. When the data sent to the server, the master thread will sent to the read the thread pool, and then assign the thread reads the data coming from the client; then return the data to the master thread, and the server processes business; after that, if the data needs to be returned to the client, the data and the channel will be sent to the write thread pool, a write thread will respond the data to sent to the client.

System use select mechanism. It need not to start a new thread for each client connect to the server, but register all of the connections to the selector object using the single-threaded manage a large number of concurrent network connections to make the system fully use of system resources; and the non-blocking I/O can return without waiting for I/O operation complete, thereby reducing the system overhead and increasing the system performance. NIO processing key code are as follows.

```
public class Server implements Runnable {
public Server(int port) throws Exception {
  selector = Selector.open();
  sschannel = ServerSocketChannel.open();
  sschannel.configureBlocking(false);
  sschannel.socket().bind(new InetSocketAssress(8080));
  sschannel.register(selector, SelectionKey.OP_ACCEPT);
   }
  public void run() {
  while (true) {
   Set selectedKeys = selector.selectedKeys();
   Iterator it = selectedKeys.iterator();
   while (it.hasNext()) {
     SelectionKey key = (SelectionKey) it.next();
      {     ServerSocketChannel ssc =
(ServerSocketChannel) key.channel();
            SocketChannel sc = ssc.accept();
               sc.configureBlocking(false);
               sc.register(selector,
          SelectionKey.OP_READ,request);              }
 else if (read) { Reader.processRequest(key);  .....}
 else if ( write ) { Writer.processRequest(key);  .....}}}
 else {   addRegister();  ....
```

System Operation Demo

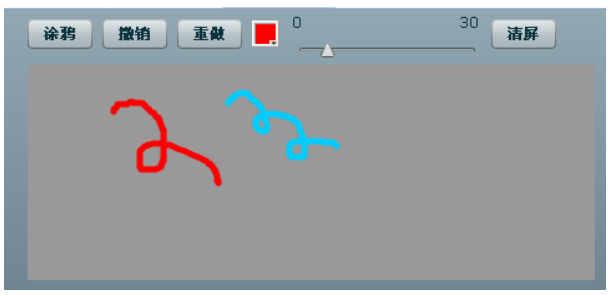

Fig.4 user1's drawing board

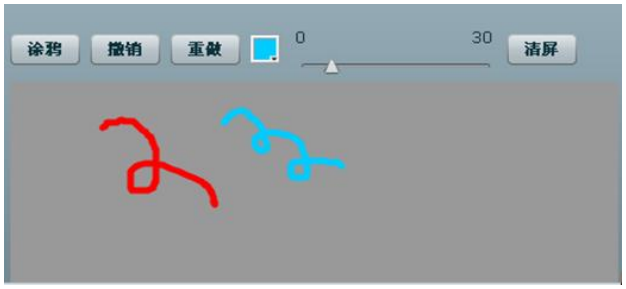

Fig.5 user2's drawing board

As shown in Figure 4 and Figure 5, two users can use draw way to communicate. After user set the pen color and pen thickness, user can make the drawing. When the mouse lift, drawing information will be sharing to the other users in real-time.

Conclusion

This paper analyzes the advantages of the comet server push technology in the web real-time message transmission, and use this technology to achieve real-time shared drawing board program, and analyze the system architecture and processes. The system is based on the advanced technology, and in real-time, concurrent terms it has superior performance. At present, the process of this system is being tested and run well.

References

[1] Dylan Schiemann. An Introduction to Bayeux. 2007.
[2] Alex Russell, Greg Wilkins, David Davis. Bayeux—a json protocol for publish/subscribe event delivery protocol 0.1 draft3. The Dojo Foundation.2007.
[3] P Greg Wilkins. Ajax, Comet and Jetty. Webtide. (2006-09-20) [2012-4-16].
[4] Engin Bozdag. Push solutions for AJAX tehnolohy. Master's Thesis. Delft University of Technology.1-43.2007.
[5] Cometd Project. The Dojo Foundation. (2008-9) [2010-09] http://www.cometd.com/.
[6] Xuefeng Liao. Flex integrated into a JavaEE application of best practices. (2009-08)[2010-09].
[7] Michael Galpin. not visible Flash. (2010-7)[2011-10].
[8] Nicholas Chase, Geronimo. Traitor: Apache Geronimo's JMS achievement: ActiveMQ . (2008)[2012-2-3].