ATLANTIS PRESS
Advances in Computer Science Research, volume 62
2nd Joint International Information Technology, Mechanical and Electronic Engineering Conference (JIMEC 2017)

Realtime Mobile Video Conference Design Base On Reactive Architecture

Haidong Lv, Yongkang Deng, Yuexing Zhong, Rui Xiao, Wenbo Li

City Institute Dalian University of Technology, Dalian, 116600, China

Keywords: video conferencing; real-time application; Node.js; SocketCluster; video and audio collection.

Abstract

Due to the traditional software architecture can not meet the performance requirements of large concurrent real-time applications, using the SocketCluster real-time transmission frame of the new non blocking asynchronous mode of Node.js platform and on the basis of the realization of the function of the cluster support Web video conferencing system work on mobile platform. The system can support the scalability of CPU multi core and multi server cluster, so as to meet the requirement of real-time application system.The test proves that the real time system developed by the above platform is superior to the experience and foundation for the future development of real time application.

1 Introduction

Currently on the market nearlly all mainstream video conferencing system[1] only when was built based on the internal enterprise platform or private cloud platform can it to ensure the smooth transmission of video and audio to meet the peformance requirement. And further the video conference client terminal basically run on the mobile platform's native app mode and need to download and manully installed in the mobile phone or tablet by their own. It is difficult to update and upgrade to the new release version.

At the same time in the design and implementation of video conferencing mostly were built with traditional server side framework such as JavaEE or MS.NET which using traditional multi thread blocking mode, so it is difficult to meet in the condition of high concurrency requirements on real-time performance of video conference.

Nowadays reactive programming[2] has gradually replaced the traditional programming method to become the mainstream of the project development. The Reactive Systems are more flexible, loosely-coupled and scalable. It can make software application easier to develop and amenable to change. They are significantly more tolerant of failure and when failure does occur they meet it with elegance rather than disaster. Reactive Systems are highly responsive, giving users effective interactive feedback.

At mainwhile the reactive programmming model combines the event-driven[3], non-blocking I/O model[4] that makes application system lightweight and efficient has beed widely used in moden software development. The Node.js[5] is one of the best framework for this new programming model.

Using the reactive event-driven and non-blocking model, The new kind of realtime mobile video conference application was designed and implemented based on Node.js and SocketCluster[6]. The system can support the scalability of CPU multi core and multi server cluster, so as to meet the requirement of real-time application system. The test proves that the real time system developed by the above platform is superior to the experience and foundation for the future development of real time application

2 System Archirecture Design

The mobile video conferencing system works with using web mode instead of native app mode. The server platform uses Node.js which runs on the Linux Ubuntu Server, the web server uses Express to feed the web page to the mobile client.

The SocketCluster framework is used as video and audio data transmission between all of the conference participants. SocketCluster is an open source real-time WebSocket[7] framework for Node.js. It supports both direct client-server communication like Socket.io[8] and group communication via pub/sub channels[9]. SocketCluster is designed to scale both vertically across multiple CPU cores and horizontally across multiple machines/instances, so it can support large concurrent and real-time high density data transmission system.

The core functions of the system is the transmission of each member to participate in the conference of video and audio, for real-time data transmission to support multiple users, using the SocketCluster framework to support Socket transmission clusters to achieve two-way real-time transmission of video and audio data between server and client.

The clients which are mobile phone, tablet or PCs use the built-in browser which supports HTML5 standard. The client side functionality is implemented by combing Bootstrap, AngularJS to perform response page rendering and two-way data binding.

The SocketCluster client is responsible for sending the local users of video and audio, and other users of the sending and receiving of audio and video, and use the HTML5 Audio API[10] the realization of audio synthesis and playback. Figure 1 shows the mobile video conferecece system architecture.

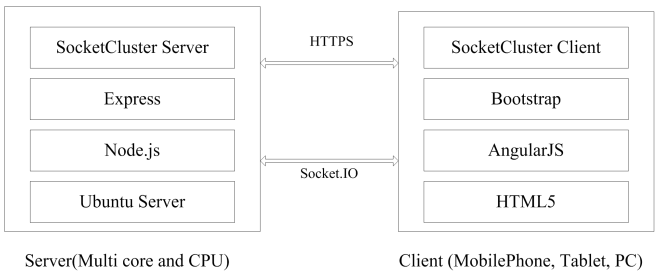

fig.1 System architecture

3 Reactive message divern implemented

The Reactive programming model rely on asynchronous message-passing to establish a boundary between components that ensures loose coupling, isolation and location transparency and the boundary provides the means to delegate failures as messages which can be capture to handle this kind of exception conditions.

SocketCluster with real-time pub/sub channels can be used to implement the reactive event driven model, either the client or the server can use JavaScript to interact with these real-time pub/sub channels to publish and receive any kind of data. In this video conference system the video and audio data is transfered using these pub/sub chanel. The application reactive programming model implementation is shown as Figure 2.

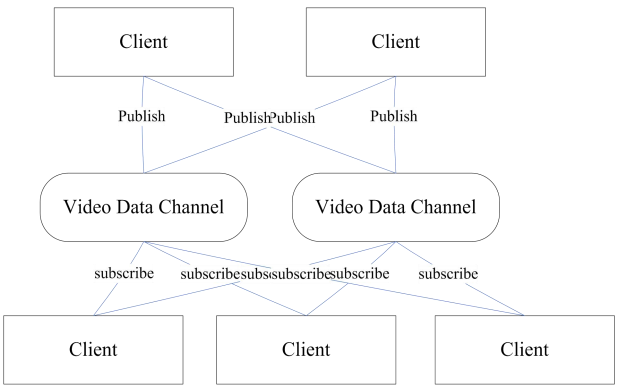

fig.2 System reactive model implemeting

Inside of the SocketCluter server there is socket server engine which can be used to handdle the data transmission for the server and all clients. The SocketClutser main process start worker process on each CPU core and each worker process strat the socket server to act as data transmission center in the system.

With worker instance the SocketCluter socket server can be create with the following snippet code show.

```
module.exports.run = function (worker) {
  //start the socket server
  var scServer = worker.scServer;
  //start video and audo transfer module
  meetingvideo.start(scServer);
  meetingaudio.start(scServer);
};
```

The socket server instance then is injected into video and audio modules. The video and audio module use the injected socket server to listene the data arriving event in the data channel and react the event with the callback hanlder function which is programmed in JavaScript which is showed in following snippet code.

```
module.exports.start=function(scServer){
  scServer.on('connection', function (socket) {
    socket.on("meeting.video",function(data){
      scServer.exchange.publish("meetimg.video.data",data);
    });
  });
};
```

The SocketClutser socket server support both peer-to-peer by emit event and grouping data transferring by publishing and subscribing to channel, in this system peer-to-peer is used in management functionality and grouping mode is used in video and audio transmission.
By publishing to a channel, the data can be send to multiple clients at once. A client which is subscribed to a channel will receive all data published to that channel. Both clients and servers can publish to a channel. Channels are intended primarily for client to client communication but SC also offers a way to listen to channels from the server.

In the server side, when the socket server is created the data can be publish to the specific channel like following code.

```
scServer.exchange.publish('videodata',videoDate);
```

Any client which is connected to SocketCluster socket server and is subscribed and watching the specific channel would receive the data which is published from other clients or server. On each client, the video data receiving code might look like the following code.

```
var videoChannel = socket.subscribe('videodata');
videoChannel.watch(function (videoData) {
  //do video processing work
});
```

SocketCluster also support data publishing to socket chanel from client, thus can peform client-client data transferring model. The client can use following snippet code to publish video and audio data to socket server channel.

```
socket.publish('videodata, videoData);
```

With the support of SocketCluster client framework the client side socket can be obtained directly without any programming code.

Finnaly socket.publish(event, data) and channel.publish(data) functions allow to send group messages between multiple client sockets (n client sockets ⇄ n client sockets - Many to many communication directly between clients, thus it is flawless and perfectcan to fit the online video conference.

All abvoe data transferring works in the event drivern and non-blocking communication model which is called reactive programming model. This model allows recipients to only consume resources while active, leading to less system overhead.

4 Scale implemented with Clutser

SocketCluster supports scale both vertically (across multiple CPU cores on a machine/host) and horizontally (across multiple hosts).

For vertically scale, SocketCluster runs as a cluster of different kinds of processes; each kind of process has its own 'controller' file which can configure each process' behaviour - There is a controller for load balancers, one for workers and one for broker processes. The most important one (where most of your application logic should go) is the workerController. The following snippet code shows how to using SocketCluster multi processing to implement scale on multi core in one CPU.

```
var SocketCluster = require('socketcluster').Socket Cluster;
var socketCluster = new SocketCluster({
  workers: 16,
  brokers: 1,
  port:8000,
  appName: "mobilemeeting",
  workerController: __dirname + '/worker.js',
  brokerController: __dirname + '/broker.js',
});
```

In above snippet code there are 16 workers processes, 1 broker process and the man server process on the server The scale of system designing and implementing shown as Figure 3.

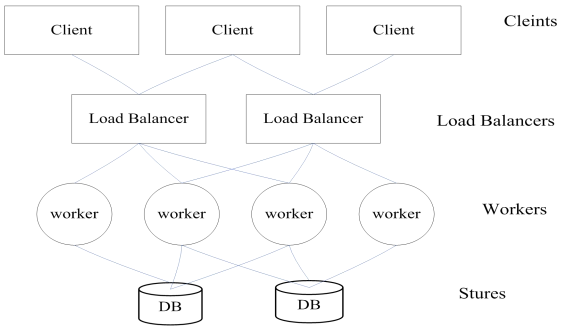

fig.3 Scale implemented with SocketCluster

And each worker process start the node.js instance and its related framework instance such as Express and our customized modules mainly video and audio transmission program. The following snippet code shows each worker process can start Express Web server instance and system application modules.

```
var express = require('express');
var serveStatic = require('serve-static');
var path = require('path');
var meetingvideo=require("./business/meetingvideo");
var meetingaudio=require("./business/meetingaudio");
module.exports.run = function (worker) {
   var app = express ();
   //start the http server
   var httpServer = worker.httpServer;
   //start the socket server and get its instance
   var scServer = worker.scServer;
   app.use(serveStatic(path.resolve(__dirname, 'web')));
   //HTTP handled by Express server
   httpServer.on('request', app);
   //start video transmission module
   meetingvideo.start(scServer);
   //start audio transmission module
   meetingaudio.start(scServer);
};
```

5 Client video data receiving and showing

In the client side browser which sopports HTML5 standard is used to view the conference GUI, the real-time video is captured using HTML5 GetUserMedia API from camera and then is sent to the local page in the <video> element and uses the AngularJS $apply() method to trigger the daynamic updating of web page, this is showed as the following code.

```
navigator.getUserMedia({video:true },function(stream){
video.src =window.URL.createObjectURL(stream);
video.play();
$scope.$apply();
```

After video image data is gained and send to the <video> element in the web page, then the video data is captured by using HTML5 canvas and Angular $interval to achieve timing service of 60 frames per second, and then the data is sent to the SocketServer video data channel, thus all the clients and server which has subscribe the channel can receiving the data on real time. The schematic is shown in the following code.

```
$scope.videotimer=$interval(function(){
canvasContext.drawImage(video, 0, 0, 100, 90);
meetingVideoData=canvas.toDataURL();
socket.sendVideoData({meetingNo:$scope.meetingNo,userid:
$rootScope.loginuserid,videodata:meetingVideoData}); },100
0/60);
```

SocketCluster provides a socket client plugin which support web client to connect the socket server in SocketCluster

server. When the web page is loaded, the socket client would connect to socket server through this plugin.

The socket client then can send data to socket channel or monitoring data arrival event from channel and receive the data from other clients or server .In the video conference system the data which is sent or received all contain meeting user id and video or audio data.

With the user info the web page can show the video on the specific user video element. The user's video displaying implementing code is shown below.

```
//monitoring the channel arriving data
socket.watchVideoData(function(data){
var senduserid=data.userid;
//specify the video element for the send data user
var videoimage=document.querySelector("img#img_"+
senduserid);
 videoimage.src=data.videodata;
});
```

With the same working mode, the audio data can be obtained with same HTML5 API. After auido data is obtained, the system use Audio API to perform filter processing and analysis according to the sampling rate of the packing for binary data, then the auido data is sent with bulk mode to SocketCluster channel to all of the participates in the meeting, finally the web client receives the audio sample data from audio channel through subscribing it and use the Audio API to generate the audio wave to to play on speaker. As the implemention code is similar with video so it is ommited here. The conference system client showing like the Figure 4.

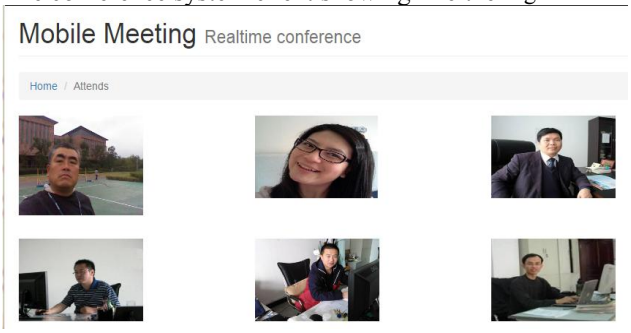

fig.4 mobile video conference client showing

6 Conclusion

Reactive architecture design means that problems may be detected quickly and dealt with effectively. Reactive model systems focus on providing rapid and consistent response times, establishing reliable upper bounds so it can deliver a consistent quality of service. This consistent behaviour in turn simplifies error handling, builds end user confidence, and encourages further interaction.

The reactive architecture which is based on Node.js and its SocketCluster framework can develop a variety of real-time applications and will become the mainstream technology of real-time application development.

Acknowledgements

This system was supported by the project of innovation and entrepreneurship training program for college students in Liaoning Province in 2015(201513198000001).

References

[1] Jason Procyk, Carman Neustaedter, Carolyn Pang, Anthony Tang, Tejinder K. Judge. "Shared geocaching over distance with mobile video streaming", *CSCW Companion '14: Proceedings of the companion publication of the 17th ACM conference on Computer supported cooperative work & social computing,* pp. 110-120, (2014)

[2] Zhaolin Li,Fang Wang. "Modeling and Simulating Multiclock Reactive Systems in Java", *TProceedings of 2012 IEEE International Conference on Computer Science and Automation Engineering（CSAE 2012）VOL02* (2012).

[3] Ling Tang. The discuss of the dead-lock issue in event-driven system. Proceedings of the 2015 3rd International Conference on Information Technology and Career Education(ICITCE 2015).

[4] Yu Ge, Guojin Wang, Huaiyuan Zhen, Taiyong Jin. "A Non-Blocking Locking Method and Performance Evaluation on Network of Workstations", *Journal of Computer Science and Technology*, **01**, pp. 22-25, (2001).

[5] J. Bermúdez-Ortega, E. Besada-Portas, J.A. López-Orozco, J.A. Bonache-Seco, J.M. de la Cruz. Remote Web-based Control Laboratory for Mobile Devices based on EJsS, Raspberry Pi and Node.js Original Research Article. IFAC-PapersOnLine, Volume 48, Issue 29, PP.158-163,(2015).

[6] Huimin Cui,Qing Yi,Jingling Xue,Xiaobing Feng. Layout-oblivious compiler optimization for matrix computations[J]. ACM Transactions on Architecture and Code Optimization (TACO) . 2013 (4).

[7] FETTE I,MELNIKOV A.The WebSocket Protocol. RFC 6455 . 2011.

[8] Gomaa,Hassan.Software Modeling and Design:UML, Use Cases, Patterns, and Software Architectures. . 2011.

[9] Castro, Miguel,Druschel, Peter,Kermarrec, Anne-Marie,Rowstron, Antony I.T.Scribe: A large-scale and decentralized application-level multicast infrastructure. IEEE Journal on Selected Areas in Communications . 2002.

[10] Wilcox M.HTML5 Video Introduction. http://academic.research.microsoft.com/Publication/694577/data-compression-methods-and-theory . 2011.