Development of IoT Application Based Socket Cluster and Cylon

Haidong Lv[1, a], Ribo Ge[2, b] and Zhigang Zhu[2, c]

[1]Software engineering department, City Institute of Dalian University of Technology, Dalian, China

[2]Engineering practice center, City Institute of Dalian University of Technology, Dalian, China

[a]haidonglu@126.com, [b]newcomer_68@163.com, [c]zhi_gang_zhu@126.com

Keywords: IoT; Message queue; Socket cluster; Node.js; Cylon.js

Abstract. Internet of Things which directly developed with traditional embed technology are characterized by increasing size and complexity, so that the IoT application developer will still not be able to adequately manage them for a long time to come. As a response to this trend, the new approach computing paradigm based on Node.js and its related framework aims to design and develop IoT systems easily and quickly. This paper describes a new architecture which is based on Node.js, Socket Cluster and Cylon framework to be able to provide a pretty-good solution for developing IoT application which worked in reactive message queue mechanism to satisfy real-time processing restriction

Introduction

The Internet of Things (IoT) has recently gained massive traction [1]. IoT challenges enterprises, small companies, and developers with new problems to solve. The real value of the IoT lies in the data generated by connected products, real-world feedback on how products are performing and how customers are using them.

Achieving above value and the full benefits of IoT connectivity [2] means building connected products on top of a comprehensive IoT platform which spans the full spectrum of technologies required for the IoT [4].

The key to develop IoT application system is to build an IoT platform [5]. An IoT platform can be thought of as a jewel with three distinct facets: cloud, connected device, and mobile or web control and it would be the best the three facets should be developed with the same technology and framework to simplify the development of the IoT system and increased developer productivity to meet the demand of rapid deployment and delivery of projects. The final solution is the JavaScript and its related framework, the main platform would be the Node.js.

Node.js has very low resource requirements and works with event driven, nonblocking I/O mode. These features are leveraging in data-intensive IoT scenarios, from wearables to M2M [6]. Node.js sets itself up as an appropriate pairing for IoT. The bottom line is that developers building data-intensive, real-time IoT applications [7] often find Node.js a natural fit.

In the IoT platform cloud, in order to collect the device data in real-time, the related protocol should be used instead of the HTTP which is use in Web, such as MQTT, WebSocket etc. In this field SocketCluster is the best choice to used as data transmission framework to used between Iot deices and cloud platform. SocketCluster is a scalable framework for real-time applications and microservices. It supports direct client-server communication or group communication via pub/sub channels and is designed to scale both vertically across multiple CPU cores and horizontally across multiple machines/instances (via pub/sub channel synchronization).

The page outlines the new approach using JavaScript and its frameworks such as Node.js, SocketCluster, Cylon.js and AngularJS both in cloud, sensor devices and web client [8] to develop the reality IoT application.

Application Architecture Design

In this IoT platform, the server cloud, device layer, web and mobile client both are developed with JavaScript language and its framework. The back end and front developer can use the same language and technology to rapidly speed up the IoT application development and deployment and improve the system maintenance.

In the cloud server, the Node.js is used as main server platform and Socket Cluster is used as both web server and device data transmission server for gathering the IoT sensor device data and push to web client which within PC or mobile phone.

In the IoI devices layer, the micro controller computer [9] such as Raspberry Pis [10], Beagle Bones, Arduino, Intel Edison, Intel Galieo etc connect all kind of sensor devices to collect environment monitoring data. In order to support all above micro computers the innovated Cylon.js is selected as device controlling platform to simplified device programming. Cylon.js is a JavaScript framework for robotics, physical computing, and the Internet of Things. It makes it incredibly easy to command sensor devices.

In the client side in order to support both normal PC and mobile phone, tablet with the same code without writing different for each one, the AngularJS and Bootstrap is used as IoT web client side framework. The application uses the SPA (Single Page Application) model which AngularJS mainly support property to reduce the data transmission between cloud and client to improve the real-time performance to meet IoT application demand. The Fig. 1 showed general architecture of system.

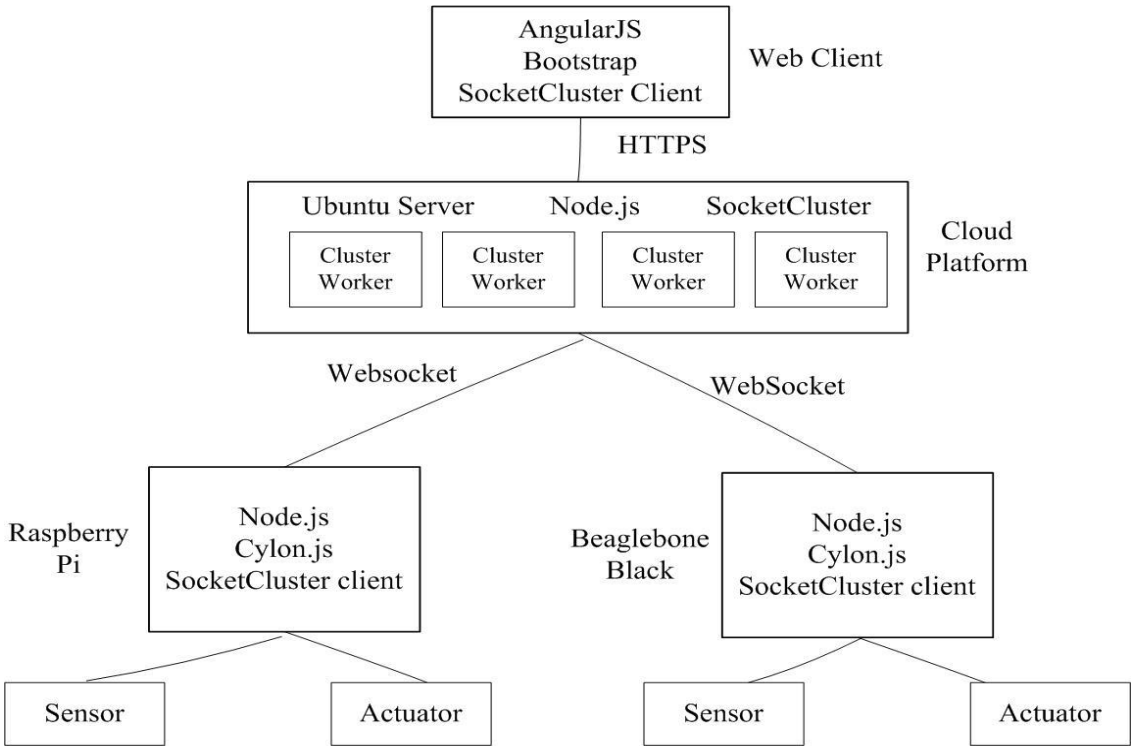

Figure 1. System architecture diagram

Cloud Platform Data Collecting Design

The cloud uses the Ubuntu Server as server platform and Node.js as base run-time framework to support the Express web server framework and SocketCluster data transmission server which works in message publish and subscribe mode as reactive methodology.

For data security the SSL is used for device sensor data transferring between device and cloud.

To improve the processing performance SocketCluster is designed to scale both vertically across multiple CPU cores and horizontally across multiple machines/instances (via pub/sub channel synchronization). In the IoT system vertically scaling is used to run each CPU core in the server. The

following code showed the server app will automatically detect the server CPU cores count then start each work process for each CPU.

```
var fs=require("fs");
var SocketCluster = require('socketcluster').SocketCluster;
var os=require('os');
var cpus=os.cpus(); // Get the CPU cores
var socketCluster = new SocketCluster({
  workers: cpus, //start process on each cpu
  brokers:  1,
  port:  8000,
  protocol: 'https', // using HTTPS
  protocolOptions: {
    key: fs.readFileSync(__dirname + '/sslkey/22863591-192.168.1.100.key'),
    cert: fs.readFileSync(__dirname + '/sslkey/22863591-192.168.1.100.cert'),
    passphrase: 'passphase4privkey'
  },
  appName: "IoT",
  workerController: __dirname + '/worker.js',
  brokerController: __dirname + '/broker.js',
  socketChannelLimit: 1000,
  rebootWorkerOnCrash: true
});
```

The SocketCluter property workers used to specify the worker process need to be started as SocketCluster server started. The worker code is showed as following.

```
module.exports.run = function (worker) {
    var fs=require("fs");
    var https = require('https');
    var express=require("express");
    var path = require('path');
    var app = express ();
    var server = https.createServer(app);
    app.use(require('body-parser').json());
    app.use(express.static(__dirname+'/web'));
    var httpServer = worker.httpServer;
    var scServer = worker.scServer;
    httpServer.on('request', app);
};
```

The worker process will run in each CPU core and SocketCluster built in load balance module will treat each message send/receiving to span each worker.

Sensor Data Gathering and Sending Design

At device layer the microcomputer connect to various sensors for gathering its monitoring data and send to cloud platform. Current time the micro-computer likes raspberrypi, BeagleBone Black mainly used to be acted broker between cloud and sensor devices.

To simplified development and improve maintenance, the key is selecting the correct technology and platform. Through comparing many open source framework which responsible for operating each kinds of sensor devices, the Cylon.js is selected to be main solution for this task. It has an extensible system for connecting to hardware devices and support for many devices that use General Purpose Input/Output (GPIO) have a shared set of drivers provided using the cylon-gpio module and lead to rapidly speed up IoT project development and ease the maintenance in the future.

Meanwhile the microcomputer works as the SocketCluster client to send sensor data to SocketCluster server and receive the actuator command which is sent from web client through the server. The following code illustrates how to use Cylon and SocketCluster to gather device data and send to SocketCluster channel.

```
var Cylon = require('cylon');
var client    = require('socketcluster-client');
var client  = client.connect(https://210.30.108.30:8000');
Cylon.robot({
  connections: {    arduino: { adaptor: "firmata", port: "/dev/ttyACM0" }  },
  devices: {    sensor: { driver: "analogSensor", pin: 0}   },
  work: function(site) {
    site.sensor.on("analogRead", function(val) {
      var monitordata=site.sensor.analogRead();
      var infodata={site:"T1011", data:monitordata };
      client.publish('tdata',infodata);    });
  }
}).start();
```

Web Client Dashboard Design

Nowadays all kinds of application need to support different client and device to access the information it supplied, for example the PC desktop, mobile phone, tablet and even wearable devices.

In order to accommodate different client screen size and automatic reactive view displaying, the web client is developed using Bootstrap and AngularJS framework also combined with Socket Cluster client.

Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web. It's made for folks of all skill levels, devices of all shapes, and projects of all sizes. Bootstrap easily and efficiently scales web applications with a single code base, from phones to tablets to desktops with CSS media queries.

AngularJS is an open-source web application framework that offers quite a bit to a developer through a stable code base, vibrant community, and rich ecosystem. Its two-way data binding is the supermodel of the feature set and best suit for real-time application for synchronizing data with the server side.

AngularJS mainly support SPA application architecture to reduce the data transmission between server and client and result to improve the IoT application real-time performance.

AngularJS implements the MVC pattern to separate the role for web client, the model presents the data which will be showed in the view, the view responsible for data showing and controller acts broker between model and view component.

AngularJS uses the route to implement SPA pattern, due to the built in router can not support the inner view template, the plugin module UI-router is used the main router, the following code showed the web client dashboard implementing with Angular module and ui-router.

```
angular.module("IoTApp",['ui.bootstrap',"ui.router","ngResource","smart-table",'ngDialog','ngCoo
kies',"IoTApp.main","IoTApp.user","IoTApp.admin","IoTApp.dashboard","IoTApp.datacenter"]);
// Web main route definition
angular.module("IoTApp").config(function($stateProvider,$urlRouterProvider,$locationProvider){
  $stateProvider.state("main",{
     templateUrl:"../iot/views/main.html"
  }).state("user",{
      templateUrl:"../user/views/main.html"
  }).state("datacenter",{
     templateUrl:"../datacenter/views/main.html"
```

```
    }).state("dashboard",{
        url:"/dashboard",
        templateUrl:"../dashboard/views/main.html"
    });
    $urlRouterProvider.otherwise('main');
});
// main controller
```

In the IoT web data dish board the SocketCluster client connect socket server and subscribe the sensor data channel and actuator command channel which can be receive sensor data to display on client view and send command to actuator from web client action.

Using AngularJS two-way data binding when the mode data represent the sensor gathering data change the view will be update automatically. The following code illustrates how AngularJS service can receive the sensor data through SocketCluster client by subscribing the specific channel at real-time fashion.

```
angular.module("mobileMeetingApp.realmeeting.service").factory('socket', function($rootScope){
    var options = {  protocol: 'https',hostname: '210.30.108.30', port: 8000   };
    var socket = socketCluster.connect(options);
    var sensorChannel = socket.subscribe('sensor.data');
    var actuatorChannel=socket.subscribe('actuator.command');
    return {
        on:function(eventName, callback){
            socket.on(eventName, function () {
                var args = arguments;
                $rootScope.$apply(function () {    callback.apply(socket, args);    });
            });
        },
        emit:function(eventName, data, callback){
            socket.emit(eventName, data, function () {
                var args = arguments;
                $rootScope.$apply(function () {
                    if (callback) {    callback.apply(socket, args);        }
                });
            });
        },
        actuatorCommand:function(data){
            socket.publish("actuator.command",data);
        },
        watchSensor:function(callback){
            sensorChannel.watch(function(){
                var args = arguments;
                $rootScope.$apply(function () {   callback.apply(socket,args);  });
          });
        },
    };
});
```

Through this two-way data binding mechanism there is no need to writer code to manipulate the DOM API to update the data on the web page.

Summary

Using the latest technology and framework to develop and deploy IoT application system will be a development trend of IoT industry. With comparing with traditional IoT application development with C, C++ and CGI language and technology, this approach can simplify the development, quicken developing speed and improve quality and maintainability of the IoT system software, greatly reduce the investment of the system. In the future the span multiple machines cluster framework based on Node.js will be used to develop large scale IoT application system to used in all kind industries internet applications andsmart city.

Acknowledgements

In the process of development and deployment of the project, it has been greatly assisted by the engineers of Dalian Linktime Development Co.Ltd and JiuDing Heating Co.Ltd。Special thanks to Mr.Lu Yonglin for helping all the issues of system testing and deploying.

References

[1] Hagen Völzer, Daniele Varacca. Defining Fairness in Reactive and Concurrent Systems. May 2012 Journal of the ACM (JACM): Volume 59 Issue 3, June 2012.

[2] Zhenhuan Zhu, S. Olutunde Oyadiji. Structure Design of Wireless Sensor Nodes with Energy and Cost Awareness for Multichannel Signal Measurement. February 2016 ACM Transactions on Embedded Computing Systems (TECS): Volume 15 Issue 1, February 2016.

[3] Souleiman Hasan, Edward Curry. Approximate Semantic Matching of Events for the Internet of Things.August 2014 ACM Transactions on Internet Technology (TOIT) - Special Issue on Event Recognition: Volume 14 Issue 1, July 2014.

[4] Mario Strasser, Boris Danev, Srdjan Č apkun. Detection of reactive jamming in sensor networks.September 2010 ACM Transactions on Sensor Networks (TOSN): Volume 7 Issue 2, August 2010.

[5] Jian-Min Jiang, Huibiao Zhu, Qin Li. Analyzing Event-Based Scheduling in Concurrent Reactive Systems. September 2015 ACM Transactions on Embedded Computing Systems (TECS): Volume 14 Issue 4, December 2015.

[6] Alejandro Talaminos-Barroso, Miguel A. Estudillo- Valderrama, Laura M. Roa, Javier Reina-Tosina, Francisco Ortega-Ruiz. A Machine-to-Machine protocol benchmark for eHealth applications – Use case: Respiratory rehabilitationOriginal Research Article. Computer Methods and Programs in Biomedicine, Volume 129, June 2016, Pages 1-11.

[7] Daniel Barata, Gonçalo Louzada, Andreia Carreiro, António Damasceno. System of Acquisition, Transmission, Storage and Visualization of Pulse Oximeter and ECG Data Using Android and MQTTOriginal Research Article. Procedia Technology, Volume 9, 2013, Pages 1265-1272.

[8] 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, 2015, Pages 158-163.

[9] Bryan M. Kowal, Travis R. Schreier, Joseph T. Dauer, Tomáš Helikar. Programmatic access to logical models in the Cell Collective modeling environment via a REST API Original Research Article. Biosystems, Volume 139, January 2016, Pages 12-16.

[10] Dhvani Shah, Vinayak haradi. IoT Based Biometrics Implementation on Raspberry PiOriginal Research Article. Procedia Computer Science, Volume 79, 2016, Pages 328-336.n