ATLANTIS PRESS

Advances in Computer Science Research, volume 80

International Conference on Computer Science, Electronics and Communication Engineering (CSECE 2018)

Designing of IoT Platform Based on Functional Reactive Pattern

Haidong Lv, Xiaolong Ge, Hongzhi Zhu, Zhiwei Yuan, Zhen Wang and Yongkang Zhu
City Institute of Dalian University of Technology, Dalian China

***Abstract*—To meet the real-time requirement and data fast processing has become a significant challenge in the Internet of Things platform. In an attempt to design and implement new reactive paradigm , this paper presents an innovative functional reactive programming model to design and implement IoT platform using Node.js, Kafka, MQTT, Angular and RxJS etc technologies to process high speed device collecting data. In this IoT platform system the functional reactive approach has been pushing both on the back-end and the front-end to simplified IoT platform system designing and programming and reduce the development time and effort. Future research will focus on the practice validation of this work by implementing this IoT platform into a big data system and artificial intelligence application.**

Keywords—internet of thing; functional reactive programming; MQTT; kafka

I. INTRODUCTION

At present the IoT connected devices are turning up everywhere, according to IoT analysts, the number of networked electronics is projected to exceed 20-50 billion devices by 2020. The Internet of Things is giving rise to a new era of networked computing where business applications intelligently monitor and control remote sensors, mobile devices, and smart machines and where devices such as actuators, valves and switches are connected and communicating.

Almost every major IT companies are offering their own IoT platform to manage these IoT devices. And hundreds of technology companies are offering capabilities for IoT use case implementation.

Normally most IoT application platforms[1] are developed by using the monolith architecture and synchronizing working model, due to it mainly uses thread technique to process multiple concurrent IoT devices connection, so it can not suit to manage vast devices and meet the high performance requirement.

To overcome those monolith system shortcoming, the new trend technique functional reactive architecture which based on micro-services and non-blocking design patterns appeared and developed for resolving these problem, this innovative solutions that intelligently monitor and control remote sensors and devices are revolutionizing enterprise computing.

Reactive programming is an important concept that provides a lot of advantages: it naturally handles asynchrony and provides a model for dealing with complex data and time flow while also lessening the need to resort to shared mutable state.

Reactive programming is specific to suit for design and implement applications such as interactive UIs and animation, client-server communication, robotics, IoT, sensor networks, etc.

Reactive Programming raises the level of abstraction of all kind of system code so developer can then focus on the interdependence of events that define the business logic, rather than having to constantly fiddle with a large amount of implementation details.

Reactive streams is suited to manage time-related complexity well, it makes reasoning about asynchronous events, controlling their timing, and combining them simpler. It is much simpler than using callbacks, and even simpler than using Promises. Since the server logic naturally deals with multiple identical request events, a stream where the events can flow and be processed is a better mental model compared to single-execution Promise abstraction.

Reactive programming main purpose is moving of everything that is outside of our control to inputs or outputs, leaving the system main logic inside a pure function.

Functional reactive programming (FRP) is a programming paradigm for reactive programming, specifically asynchronous data flow programming using the building blocks of functional programming such as map, reduce, filter etc.

This paper main focus on how to design non-blocking, highly responsive, resilient, elastic, and message-driven IoT management platform application.

In order to make FRP programming more easily and high productivity, a lot of new technology and languages has been invented.

In the server side development the Node.js and Vert.x are mainly used to develop reactive event-base application, The micro service framework Seneca, SocketCluster which based on Node.js can develop reactive micro service simply and efficiently. Another framework Zetta is an open source platform built on Node.js for creating Internet of Things servers that run across geographical distributed computers and the cloud, it combines REST APIs, WebSockets and reactive programming and is perfect for assembling many devices into data-intensive, real-time applications.

In the front side the famous framework Angular, React and Vue.js are all mainly used to develop reactive UI application,

this paper uses the Google Angular as the IoT platform front end UI development technology to simplified complex front-end designing and programming.

With these latest technologies and functional reactive programming, the advanced IoT platform was designed and implemented in the real industry field, this Innovative solutions that intelligently monitor and control remote sensors and devices are revolutionizing enterprise computing.

II. IoT Platform Architecture Design

The IoT platform architecture from a high-level perspective is showed in Figure 1, the whole platform is consist of four subsystem: backed end server, data exchange center, sensor side and client UI side.

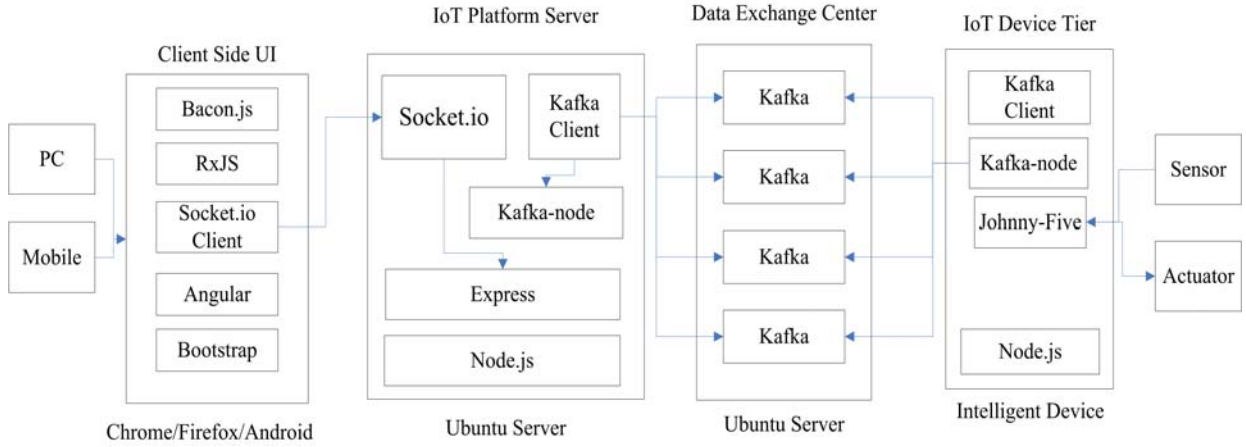

FIGURE I. IOT PLATFORM SYSEM ARCHITECTURE

The whole platform is built based on Ubuntu server 16.04 and Node.js 8.9.4. Due to Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient[2], it is perfect programming platform to develop real time application such as automatically control system and IoT system.

The figure.1 above shows reactions among the different parts inside the platform. It is mainly based on NodeJS. The NodeJS platform has a variety of modules that are perfect for the Internet of Things. This platform be mainly built upon event driven, reactive steam, MQTT[3] and Kafka[4][5], it works in asynchronous non-blocking development model, which does not mean to be multiple threading, but the non-blocking I/O for handling network, file system and others, it uses the callbacks system, so it can possible to handle a huge number of events per second using a single thread. This feature is exactly what this IoT system needs.

The key for implementing the IoT platform[6] is the reactive stream[7]. At device tier the data detected by the sensor is transformed into stream and send with Kafka-Node platform to Kafka cluster center in the reactive mode. The platform server side receives the data by subscribing the topic of Kafka in asynchronous reactive mechanism and then send to other data processing and analysis component by send emit with Node's socket.io or event. Finally the user interface with web or mobile at the client site receives the data though Angular and Sokcet.io client in the event stream way[8].

III. Reactive Message Exchange Center Design

In large companies and industry, the IoT platform need to manage vast numbers of IoT device endpoints and to collect and act upon massive volumes of raw business data generated by the devices. With the traditional architecture system, it can not transform and process so vast data in the real time, result in the data which can be stored in time has been lost.

In order to meet the requirement, the platform use Apache Kafka framework as device message exchange center.Apache Kafka is a streaming platform for collecting, storing, and processing high volumes of data in real time. Before Apache Kafka, there was not a solution that perfectly met the IoT vast data processing in real time. Traditional messaging systems are real-time, but were not designed to handle data at scale. Kafka typically serves as a central data hub in which all data within an enterprise is collected.

The data can then be used for continuous processing or fed into other systems and applications in real time[9].

A Kafka cluster was built in the IoT platform to meet the vast data processing requirement. All the distributed IoT devices data was collected and transformed with MQTT protocol to Kafka Pub/Sub queue.

In the platform when use setup a new IoT device, the platform will automatically create a specific topic in Kafka with using its stream API, there after all the data which is collected from this device will be send to the topic to store. The platform can acted as Kafka consumer to subscribe this topic and can get the data with streaming API automatically in real time. Each IoT device will have a corresponding topic in Kafka cluster.

The platform use Kafka-node framework which is a Node.js client with Zookeeper integration for Apache Kafka. In the platform the following snippet code is illustrated how to use Kafka-node within Node.js to connect Kafka as its client as

producer to send data to Kafka or as consumer to receive data of device[10].

```
var kafka = require('kafka-node');
var KeyedMessage = kafka.KeyedMessage;
var Producer = kafka.Producer;
var client = new kafka.Client('210.30.108.30:12001, 210.30.108.31:12001, 210.30.108.32:12001');
var producer = new Producer(client);
producer.send(payloads, function (err, data) {
    cb(data);
});
```

For transforming data, the JSON has been the standard data format in nearly all kind of applications and systems, in this IoT platform all the data from or to devices need to convert to JSON format, the following snippet code shows how to implement the task.

```
var express = require('express');
var app = express();
var kafka = require('./Kafka.js');
var bodyParser = require('body-parser');
// create application/x-www-form-urlencoded encoding
var urlencodedParser = bodyParser.urlencoded({ extended: false })
app.post('/toKafka', urlencodedParser, function (req, res) {
   // concert to JSON format
   kafka.produce(req.body.key,req.body.message, function (result) {
      res.send(result)
   });
});
```

IV. IoT Devices Tier Data Collection and Transformation Design

The device tier encompasses a wide variety of intelligent endpoints, including mobile computing devices, wearable technology, remote sensors and controls, and autonomous machines and appliances. Normally the IoT platform might manages hundreds of thousands or even millions of devices, so the processing capability is very important, only the function reactive paradigm can meet this requirement.

Under the functional reactive programming pattern the whole IoT system need to be to work as stream event driven and synchronized model which about collecting streams of measurement values, storing them for later use and visualization as well as transforming and combining data into message streams that can then be fed to actuators, such as pumps and valves.

This platform use the MQTT protocol for transfer device's data between platform server and IoT devices. MQTT is special with super lightweight architecture and is ideal for scenarios where bandwidth is not optimal.

This platform supports two kinds of IoT devices, one is worked as MQTT broker, the other is worked as MQTT client. When the device is MQTT broker, the platform can configure the management device as client to subscribe the specific.

The platform server use the MQTT broker framework Mosca, it is a node.js MQTT broker, which can be used as standalone or embedded in another Node.js application. The IoT device can be acted as MQTT client endpoint to publish or subscribe data to and from MQTT broker.

The following snippet code shows how an IoT device which acted as MQTT client to measure data from sensor with Johnny-Five framework which can used in Node.js platform using JavaScript programming language and then send to server side with MQTT technology. Johnny-Five has grown from a passion project into a tool for inspiring learning and creativity from all across the world.

```
var mqtt  = require('mqtt');
var five = require("johnny-five");
var board = new five.Board();
var mqttclient=mqtt.connect('mqtt://210.30.108.30',{
   username:'iotplatform',
   password:'xxxxxx',
   clientId:'device#1'
});
mqttclient.on('connect', function () {
   board.on("ready", function() {
      var DeviceSensor = new five.Thermometer({
      controller: "BMP085"
   });
  //event for temperature change
  DeviceSensor.on("change", function() {
   var sendData={deviceNo:1201, data:this.celsius};
   mqttclient.publish('device#1', sendData); //send data to server broker
  });
});.
```

V. IoT Device Management Center Design

The IoT platform provides web and mobile UI for customer to manage their IoT devices. By comparing the most popular framework products on the market: Angular, React and Vue.js. The Google Angular is selected to used in platform client side, due to its reusing one code and abilities to build apps for any deployment target such as web, mobile web, native mobile and

native desktop. Further more the Angular main works in functional reactive pattern, it puts IoT platform in control over scalability and meet huge data requirements by building data models on RxJS, Immutable.js or another reactive push-model. So the platform both server, client and device tier all work in the same reactive work model.

Angular is reactive (non-blocking) by using observable or promises programming paradigm, the result in the front end of IoT platform is in reactive also.

At current time, Kafka does not have any JavaScript web client support, a middle tier has to implement between Kafka message center and Angular client, the socket.io framework is the best choice for this purposes.

In server side the platform use Kafka-Node as client to connect to Kafka and then send the device data through socket.io to Angular client, include web and mobile.

With Node.js, Kafka and socket.io in server side, Angular gets are a reactive partner, the front end UI use the same and familiar syntax with a functional programming model in the server side.

VI. CONCLUSIONS

In the paper a advanced and innovated IoT platform system which combined functional reactive programming and event-based, no-blocking architecture has been designed and implemented. Compared with the traditional platform, the new kind of system has short development time and high development efficiency. Further it also keep the IoT platform more maintainable and extensible due to its micro-service architecture. With FRP the IoT application system can be easy designed and implemented with all the benefits of a mature and vibrant both back-end and front-end framework ecosystem.

In the future the reactive paradigm will influence and shape the world of IoT application for many years to come, and it will be the future direction of IoT platform designing and developing.

ACKNOWLEDGMENT

This project was supported by the innovative and entrepreneurial training program for national college students in local colleges and universities in 2017 published by Ministry of education of China (No:201713198003).

REFERENCES

[1] Muhammad Suryanegara; Ajib Setyo Arifin; Muhamad Asvial; Gunawan Wibisono: A system engineering approach to the implementation of the Internet of Things (IoT) in a country. 2017 4th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), Year: 2017;Pages:20-23. *)*

[2] Martin K. Mwila: Design and implementation of a Node.js based communication framework for an unmanned autonomous ground vehicle.2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech),Year:2017,Pages:74-79.

[3] Manel Houimli; Laid Kahloul; Sihem Benaoun: Formal specification, verification and evaluation of the MQTT protocol in the Internet of Things, 2017 International Conference on Mathematics and Information Technology (ICMIT). Year:2017, Pages:214-221, IEEE Conferences.

[4] Godson Michael D'silva; Azharuddin Khan; Gaurav; Siddhesh Bari, in: Real-time processing of IoT events with historic data using Apache Kafka and Apache Spark with dashing framework, 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT),Year:2017,Pages:1804-1809.IEEE Conference.

[5] Zhenghe Wang; Wei Dai; Feng Wang; Hui Deng; Shoulin Wei; Xiaoli Zhang; Bo Liang: Kafka and Its Using in High-throughput and Reliable Message Distribution, 2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS), Year:2015,Pages:117-120.IEEE Conferences.

[6] Alireza Kahrobaeian; Yasser Abdel-Rady I. Mohamed: Interactive Distributed Generation Interface for Flexible Micro-Grid Operation in Smart Distribution Systems, IEEE Transactions on Sustainable Energy,Year: 2012, Volume: 3, Issue: 2,Pages: 295 - 305,Cited by: Papers (38).

[7] M Abdul Salam; S. Sethulakshmi: Control for grid connected and intentional islanded operation of distributed generation, 2017 Innovations in Power and Advanced Computing Technologies (i-PACT),Year: 2017,Pages:1-6. IEEE Conferences.

[8] Ashwini Raut: Real time monitoring of substation by using cloud computing, 2017 Third International Conference on Sensing, Signal Processing and Security (ICSSS),Year: 2017,Pages:138-147,IEEE Conferences.

[9] Washington Velásquez Vargas; Andres Munoz-Arcentales; Joaquín Salvachúa Rodríguez: A distributed system model for managing data ingestion in a wireless sensor network, 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC),Year: 2017,Pages:1 - 5,IEEE Conferences.

[10] Evgeny L. Romanov; Galina V. Troshina, The IoT-architecture on the principles of reactive programming. 2017 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON),Year:2017,Pages:317-322.