Journal of Intelligent & Fuzzy Systems 32 (2017) 3069–3080
DOI:10.3233/JIFS-169250
IOS Press

IoE-MPP: A mobile portal platform for internet of everything

Xin Chen[a,*], Pengfei Yang[a], Tie Qiu[b], Hao Yin[c] and Jianwei Ji[a]
[a]*School of Computer Science, Beijing Information Science and Technology University, Beijing, China*
[b]*School of Software, Dalian University of Technology, Dalian, China*
[c]*Research Institute of Information Technology, Tsinghua University, Beijing, China*

Abstract. With the development of network technology, the Internet portals have been constantly changing. However, such portals cannot meet the requirements of Internet of Everything (IoE) communication between people and things or between things themselves. Inspired by the ideal of container, Mobile Cross-platform Application Development Framework (MCADF) and Platform as a Service (PaaS), we designed a "Cloud + Container" portal platform for IoE. The cloud provides application development, testing, deployment, operation, management, and other functions. It is also responsible for data storage, management and analysis. The terminal of user container is used to carry and manage mass applications and IoE data, and provide user access entry. In this paper, we first introduce the background of IoE and related problems, then give the detailed design of the platform. Finally, we evaluate the performance of the platform comparing with other platforms.

Keywords: IoE, container, MCADF, PaaS, cloud platform

1. Introduction

The Internet of Things (IoT) [1, 2] is the network of physical devices, vehicles, buildings and other items embedded with electronics, software, sensors, and network connectivity that enables these objects to collect and exchange data. And many researchers apply themselves to improve the performance of the Internet of Things [3, 4]. However, with a huge number of smart wearable devices, intelligent furniture and other new equipments used in IoT, people will enter Internet of Everything (IoE) era. IoE is referred as the Internet that connect people, data, processes and objects together [5].

The Internet portals, such as Google, Yahoo and other search engines, as well as Facebook, Twitter and other social networks, have been constantly changing. However, such portals cannot meet the requirement of Internet of Everything (IoE) communication between people and things or between things themselves: the data of IoE, application quantity has produced a spurt of growth [6], which brought great challenges to the application developers and ordinary users. Besides, heterogeneous data is largely comprised of structured and unstructured data, and even the semi-structured data, which increased the difficulty of the data storage.

The mobile native application can make interaction with smart devices [7], realizing the management of intelligent device, but users cannot effectively manage a large amount of applications. Further more, the development costs are relatively large by the native language. Container entry [8] can manage applications, however, the internal applications actually use the Web application to replace the original applications, for part of the applications is feasible, but also cannot connect the offline applications, and the performance of the intelligent devices affect the fluency of the poor native applications.

*Corresponding author. Xin Chen, School of Computer Science, Beijing Information Science and Technology University, Beijing 100101, China. E-mail: chenxin@bistu.edu.cn.

Aiming at the above challenges, this paper presents a "cloud + container" pattern of IoE mobile portal platform (IoE-MPP) according to the idea of container, MCADF and PaaS technologies. In addition, we adopted a set of reliable access control mechanism [9] to prevent illegal resources sharing. According to the ideas of the transparent computing [10, 11] on demand loading, users can load different applications in line with the needs of different scenarios.

The rest of this paper is structured as follows. Section 2 provides the related work about MCADF and PaaS. The platform architecture and its components are presented in section 3. Section 4 introduces the key technologies of the platform from four aspects. Section 5 focuses on evaluating the performance comparison of two MCADFs and the stress test. Conclusion and future work are addressed in the last section.

2. Related work

Most of the research in the domain of mobile portal platform focuses on mobile cross-platform application development framework and PaaS, so the related work can be separated into two parts.

MCADF is designed to support the development of phone applications which are written as the same language and can run on different platforms including iOS, Android, Windows Phone and BlackBerry, etc [12]. The cross-platform development tools can be classified into two types based on the way of achieving platform-independence. One allows implementing a mobile application with web technologies like Hyper Text Markup Language 5 (HTML5), Cascading Style Sheets (CSS) and JavaScript, the typical representative is PhoneGap created by Nitobi [13]. The other is represented by Titanium, which packages the native API into JavaScript interface, the source code and data can be converted to equivalent native code (e.g. Objective C, Java) [14].

PaaS is a category of cloud computing services that provides a platform allowing developers to develop, run and manage applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an application [15]. Initiated in 2012, mobile PaaS (mPaaS) provides development capabilities for mobile application designers and developers [16], such as IBM Bluemix [17], APICloud, AppCan, ExMobi etc. These platforms adopt the mobile cross-platform development framework which focuses on HTML5 technology. The combination of MCADF and PaaS shortens the application development cycle, reduces the difficulty of application deployment and operation.

However, shortages also exist in the current mPaaS platforms. Firstly, the user experience of using applications developed by HTML5 is limited by native phone capabilities. Secondly, mPaaS solved the problem of the mobile developers, but for ordinary users, it is still difficult to manage applications. These mPaaS platforms do not provide management entry which is similar to the container.

3. Architecture

Entity-Relationship Model [18] provides a method and graphic symbol that express entity, relation and their properties for users, which support a conceptual description for the real world. This paper illustrates the basic entities and interactive relationship of IoE-MPP by means of this model, represented as,

$$I = \{E, R\} \tag{1}$$

Among them, I represents IoE-MPP, composed of binary group, as shown in Fig. 1. E is the entity set, R is the link set. $E = \{e_{Cl}, e_{Co}, e_D, e_S, \}$, the basic function entities are respectively shown:

(1) e_{Cl} is the cloud-end, which refers to the PaaS cloud platform, providing some functions for App's development, testing, deployment, operation and management etc. It is also responsible for the storage, management and analysis of data.

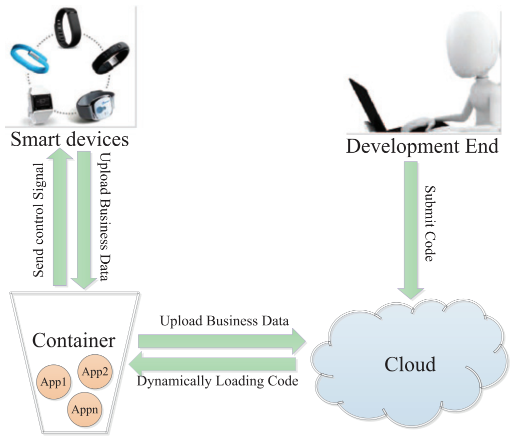

Fig. 1. "IoE-MPP" model.

(2) e_{Co} is the container, which is the mobile application container, in essence, a mobile native application, which can be considered as an Apps shell or a lightweight virtual machine. The container can abstract the resource of the system, and the traditional virtual machine is the abstract of hardware resources. Referenced by the ideas of resource container, the developers package the sharing resources which are required by APPs with similar functions, into the application containers. The application container which dynamically loads Apps from the cloud on demand can be directly installed on the mobile operating system by users. Apps share the underlying resources and realize the communication between themsleves. According to the ideas of safe container, each App is placed in a separate running environment and identified by ID, running his own business.
(3) e_D is the development-end, Developers can easily develop applications in the local browser or IDE by means of the libraries provided from the PaaS platform.
(4) e_S are smart devices, which refers to all kinds of information sensing device in IoE, including the wearable devices, intelligent household equipment, etc. It can real-timely record the related data and synchronize data with mobile phone, tablet, etc.

$$R = \{r | r =< e_i, e_j >\},\ i, j \in E,\ e_i \neq e_j.$$

Two connected entities is not symmetrical, which has directional and reveals the the input and output of the calculation, and the flow of information or control.

(1) $< e_D, e_{Cl} >$ represents submitting code, Developers submit applications code to the cloud platform.
(2) $< e_{Cl}, e_{Co} >$ represents dynamically loading code. The user can dynamically load JavaScript to container from the cloud platform on demand.
(3) $< e_{Co}, e_{Cl} >$ represents uploading application business data. The business data of the applications can be uploaded to the cloud platform.
(4) $< e_S, e_{Co} >$ represents uploading device business data. The business data of the smart devices can be uploaded to the cloud platform.
(5) $< e_{Co}, e_S >$ represents sending control signal. The application of the container controls the operation of the equipment by sending signal.

3.1. Mobile end

Mobile application container is the entrance that users enter IoE. In order to realize the unified management and scheduling for the resources in the container, the container is designed as a hierarchical module, which is shown in Fig. 2, a total of five modules. Application Management realizes the operations of installing/updating/uninstalling applications. Data Storage achieves the mass hierarchical data storage. The Application Engine provides the application running environment. Shared Module can be shared by all applications of the container. Mobile Application Cross-platform Development Kit offers a series of cross-platform API—-Titanium.

3.1.1. Data storage

Data Storage is the core of container, which includes executable code, application components and business data.

Executable code module stores the resource of the application itself. In essence, the application is a kind of data resources which is written by JavaScript, and those JavaScript codes define the layout of user interface and the business process. Users can dynamically load JavaScript from the cloud platform on demand. Every application bundle is usually only a few KB to hundreds of KB, which it is very lightweight and efficient.

Application components module encapsulates some shared resources that supporting the application

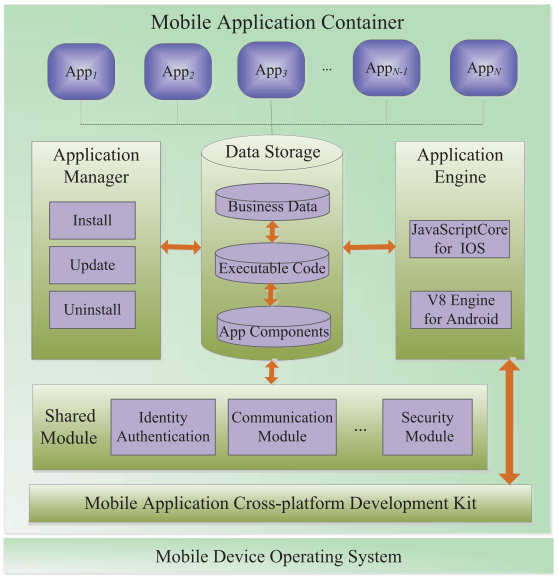

Fig. 2. Mobile application container architecture.

running, such as blue tooth, maps and payment function module.

Business data module includes the business data of every application from the communication management module, storing in the file and SQLite database [19]. The business data source is diversified, mainly divided into five categories: (1) The IoE equipment, (2) The cloud platform application interface, (3) The third-party interface, (4) The user input, (5) The log operation form the terminal.

Executable code and business data are separately stored according to application ID, however, application components are centrally stored in one piece.

3.1.2. Application manager

The Application Manager (AM) shown in Fig. 2 has several responsibilities including loading the desktop on the first instantiation of the container.

Normally, all the application data are hosted on the cloud platform. When an application is needed to be loaded, the corresponding application executable code and business data will be fetched from the cloud platform and converted to the application in the container, and application components will be selectively downloaded. It is necessary to check whether there exists the corresponding application in local container before these are downloaded.

The AM handles the installation of new applications within the container and also periodic version checks for both installed applications and the container itself. Version checking are performed by making an HTTP request to a server based version component in the application store, and applications can be updated dynamically and restarted within the container. When the application is no longer needed, the corresponding application data can be removed from the data sharing platform.

3.1.3. Mobile application cross-platform development kit

Mobile application cross-platform development kit includes a cross-platform API for accessing native UI components such as navigation bars, menus, and dialog boxes and native device functionality including the file system, network, geolocation and accelerometer.

3.1.4. Application engine

Application Engine encapsulates the application run support environment. Application Engine reads the application executable code and business data, setting up the UI of the application, responding to users' trigger events, executing the business logic.

The entire application from start to finish is now controlled by JavaScript, which can interact over the engine with native code and components. It figures out what dependencies the script has on the Titanium API and application components. On the JavaScript runtime front, we've shipped two separate JavaScript runtime with our applications, respectively, JavaScript Core on iOS and V8 engine on Android, running the JavaScript code directly without a WebView. The running applications can provide a perfect user experience and its performance is closed to the native application.

3.1.5. Shared module

Usually, the applications installed in the same mobile application container should be released by the same enterprise. These applications usually have similar public module, such as identity authentication mechanism, the communication module, the security module and so on.

(1) Identity Authentication

The object of authentication has two kinds, one is for tenants, the concept of tenant will be introduced in section 4.3, the tenant has carried on the limits for the application resources of container and container can only load the corresponding application under the tenant. The other is for users, the permissions of normal users is low, which only could get the public resources of tenant, and for some special users, the permission level is higher, they can use more applications under the tenant.

(2) Communication Management

Communication Management consists of an IoE equipment interface module, a third-party service interface module, and a cloud platform service interface module. The IoE equipment interface module supports the intelligent hardware and IoT, including the function of the three aspects:

* Automatically scanning and perceiving the user's place (the station, shops, restaurants, etc.) and intelligent objects (air conditioning, coffee machine, etc.) via Bluetooth or Wi-Fi, automatic loading and starting the corresponding application.
* Being responsible for configuring, monitoring, collecting and receiving data that the IoE devices send, such as location information, heart rate, etc.

* Sending a signal to the smart devices and operating them, such as intelligent vehicle driving.

The third-party service interface module mainly realizes the management of background data by means of the third-party interface outside of cloud platform. The common interface service method includes REST. The data that the third-party service interface module receives needs to be converted and analyzed according to the business data access rules. The cloud platform service interface module directly manipulate the cloud platform data, its interface service mode is also REST, which provides a flexible data services support for the application in the mobile scene.

(3) Security Management

The user privacy protection and the safety of the data transmission are mainly responsible for the security management module. The module can take different security policies for the different types data. For example, the game data can be transmitted in the form of plaintext because of the low security level, and the electronic case belong to personal privacy, should be encrypted. In this paper, referring to the technology security mechanism of HIPAA [20], protecting the information and restricting the data access in the network.

3.2. Cloud-end

The cloud-end is composed of IaaS (Infrastructure as a Service) based cloud and PaaS (Platform as a Service) cloud. IaaS provides the hosting environment, as shown in Fig. 3. PaaS offers the services for developers and ordinary users, including application development environment, application runtime environment, platform runtime library, data storage environment and management platform.

3.2.1. Hosting environment

Computing resources, storage resources and network resources can be virtualized in IaaS and used by users in the form of infrastructure services through the network. Application hosting environment is the physical environment to deploy and run applications. Completing application test, developers can apply for the necessary resources and services for application deployment according to actual needs of the application running, including the number of application examples that need to be deployed, the dynamic adjustment strategy of application example and the life cycle of application, etc. After being approved,

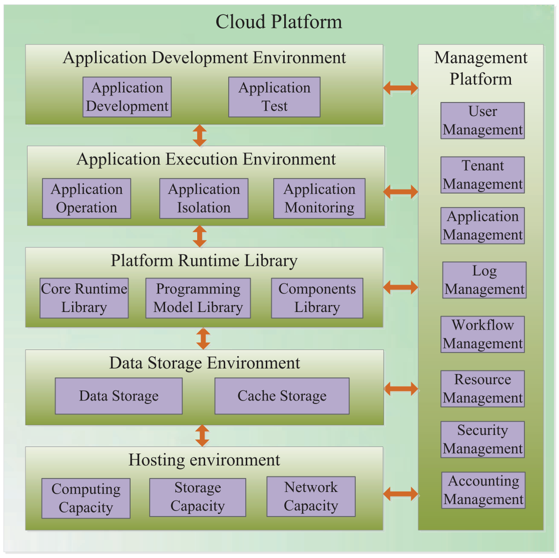

Fig. 3. Cloud platform architecture.

the system automatically packages and generates the virtual machine template (image) of the application according to the type of application and the required execution environment, and loads into the virtual machine. Through these simple and convenient operations, the application can deployed with the fastest and immediately begins to provide services.

3.2.2. Application development environment

In the development and management modules, platform provides users with application development and testing environment. There are two kinds of implementation environment. The one is an online application development test environment, in which the software developers need not to install and configure software, all development tasks are completed in the browser. However, it is necessary that the platform provides a good development experience and the network developers locate at should be keep stability and has enough bandwidth. The other is offline integrated development environment, in whcih the software developers should install and configure software developers in the local, programming in the software, and at the end, the development application will be uploaded to the cloud end.

3.2.3. Application execution environment

Application execution environment is responsible for providing the container or sandbox, which is the application runtime environment and also is the guarantee of realizing application isolation. Application execution environment controls and monitors applications, uploading parameters such as the application execution status, resource usage performance to management platform.

3.2.4. Platform runtime library

The cloud end provides the own platform runtime library, including platform API and general service API. Platform API is the core runtime platform, providing API service for each function module. Common service APIs includes applications programming model and runtime, components and templates library. Applications programming model and runtime are aimed at developers, traditional mobile applications need packaging and uploading to the application stores, afterwards, users can download and install it. The mobile applications of this platform need not packaging, and users only need load application code in the terminal container, realizing the installation of the application. The platform development framework is based on the Model- View-Control (MVC) model [21], adding files, libraries and other modules for application development. The developers realize the application development according to this model. The components and templates library contains common components of application module, servicing for the programming model module.

3.2.5. Data storage environment

Data storage environment includes cloud storage and cloud storage cache module. The traditional relational database mainly aimed at the structured data in the data storage, and it is not suitable for the semi-structured, and unstructured mass data. The relational model also restricts the fast access ability to huge data [22]. We choose MongoDB which is a non-relational database [23] as the the cloud storage module of this platform to package the data operation interface and provide data access management to the upper application. In order to avoid repeatedly accessing the same resources, cloud storage cache module provides a caching service. The commonly data is stored in the cache, and the application can read it at any time which could reduce the load of the data server and improve the performance of data access.

3.2.6. Management platform

Management platform provides background management services including user management, tenant management, application management, log management, workflow management, resource management, security management and accounting management, etc functions. It implements a series of management for the application from the submission, audit, deployment, to release.

Tenancy management can realize resource repeatability and data isolation between applications. User management can configure permissions for developers. With the change of the business and user requirements, application need to be distributed, upgraded, which is operated by application management module. The platform can monitor the application running state, such as response time, throughput and workload real-time information. The developer optimizes the application according to those informations. Security management implements the applications management from the perspective of security, and the core function is distributing the authorized applications, as well as allowing or refusing to execute certain applications.

4. Key technology

4.1. Mobile cross-platform application development frameworks

Native applications can provide better user experience while they require in-depth knowledge of each platform and the related SDKs. This increases the cost of development and updating and the time to market. In order to reduce the cost of development and reach out to maximum user across several platforms, developers are migrating to cross platform development tools.

Relative to PhoneGap, Titanium packages the native API into JavaScript interface. Its performance is better than PhoneGap, which can be reflected in the experiment of the 5 section.

The platform is built up on Titanium, which provides UI and phone APIs and other optional modules for developing. Figure 4 shows the technical architecture of Titanium.

The Titanium platform contains the middle two layers in Fig. 4. It provides both the UI components and phone APIs to access the resources like contact, location, keyboard, etc. Furthermore, it also provides optional modules to use third-party services like facebook login, wechat payment, etc. These interfaces can meet most needs for mobile development. In summary, the major contribution of Titanium is providing identical interfaces for both iPhone Operating System (iOS) and Android, which enables the cross-platform capability.

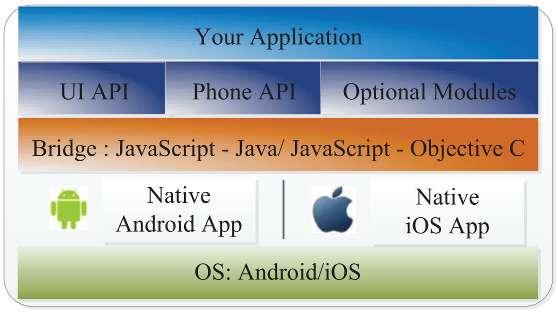

Fig. 4. Titanium architecture.

What the developers actually do with JS source is dependent on the platform, but generally it breaks up like this:

* Source is statically analyzed and interpreted to find references to Titanium API.
* Localization strings (strings.xml), application metadata (tiapp.xml), and density specific images all generate platform specific analogs.
* In iOS, JS source is base64'd and inlined as a variable into a generated C file, generated the final native binaries by xcodebuild. In Android, JS source is packaged as APK assets, compiled to Java bytecode.
* The program interpreted and compiled can run on the mobile device.

The key element of Titanium is the JavaScript to Objective C or Java Bridge, which converts the JavaScript code and maps calls into the respective SDKs.

4.2. Application isolation

Application isolation is the key to guaranteeing the safe operation of the application execution environment, for the third-party applications in the untrusted domain. It is necessary to ensure that a certain application occuring errors at runtime will not cause the system failure. As shown in Fig. 5, the platform is mainly from three levels to achieve application isolation, respectively are: the Docker container isolation, quarantine container isolation and application instance isolation.

4.2.1. Docker container isolation

Docker is a lightweight virtualization tool that can package an application and its dependencies in a virtual container that can run on any Linux server, which is great for the isolation and superior to virtual

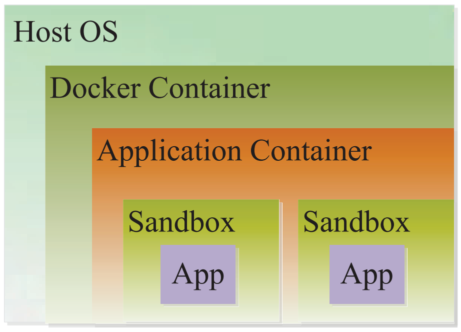

Fig. 5. Application isolation environment.

machines in the performance overhead for translation happening from Guest to Host OS. Deploying Application, the system will obtain the corresponding Docker container and application execution environment from the Docker resource pool according to the application type, and to deploy applications on the Docker container.

4.2.2. Application container isolation

Application container isolation regards the application running container as a unit to quarantine application, the common application container has Weblogic, Websphere and Tomcat, etc., this platform selects the lightweight Tomcat container, which is more convenient and quick. An application deployed in a Docker container independently, a Docker container or a physical machine can deploy multiple application containers.

4.2.3. Application instance isolation

Application instance isolation refers to that an application running container can deploy multiple applications, each application instance is running in an independent sandbox, which controls the security of the application runtime, as shown in Fig. 5. This method occurs in an inner process, which takes up the least system resources. At the same time, the sandbox technology can also control resource utilization of the single application instance.

4.3. Multi-tenancy management (Data isolation)

Application is provided to users, for applications, the tenant is its user [24]. PaaS service is used to the application system and the developers, for this platform, users and tenants belong to two different levels, is many – to – many relationships. The combination

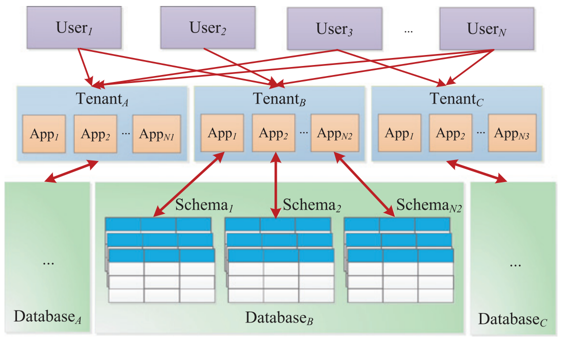

Fig. 6. Multi-tenancy management.

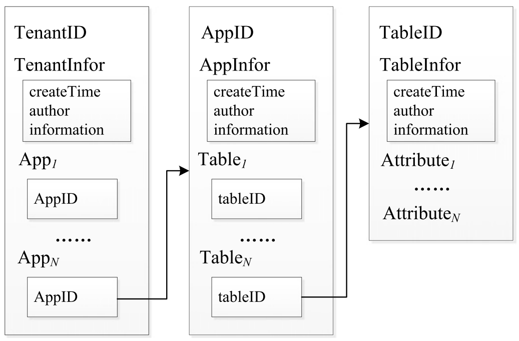

Fig. 7. Data model based on MongoDB.

of the relevant application is the platform's tenants, as shown in the Fig. 6.

$$Tenant = \{App_1, App_2, ..., App_N\} \qquad (2)$$

The tenant limits the resources, the same type or the same enterprise's applications locate in the same tenant, the resources of this tenant can be shared. In the same tenant, each application has an ID in order to realize the separation of data.

The user is the developer. Each developer can get different permissions of tenants and develop applications. Each tenant can contain many developers, this means that the project is responsible for the members under the tenant. As shown in Fig. 6, the $Tenant_A$ have three developers, which are $User_1$, $User_3$, $User_N$. $User_2$ develops applications only in the $Tenant_B$.

Each tenant has its own set of data that remains logically isolated from data that belongs to all other tenants. Metadata associates each database with the correct tenant, and database security prevents any tenant from accidentally or maliciously accessing other tenants' data. In a same tenant, business resources and application code are generally shared between all the applications, and the applications are located in the same database. However, in order to guarantee the data isolation between applications, each application has its own separate set of tables that are grouped into a schema, as shown in Fig. 6. The separate schema approach can typically accommodate more applications per server than the separate-database approach.

4.4. *Mass data storage*

Multi-tenancy management technique can ensure the data sharing and data isolation between different applications in the same tenant. On the basis of the multi-tenant management, it is necessary for the huge amounts of data storage to select a data storage model. This paper chooses directing a non-relational database, using the nested data structure to maintain the integrity of the data logic, a whole data as a whole will be to manipulate. An application itself is data, each application has its own business data, combining multi-tenancy management, design nested data model as follows Fig. 7

TenantID represents the tenant, which contains a large amount of applications. Applications are identified by appID. In each application, the developers define freely the corresponding table according to business requirements. TableID is the primary key for the table, and each table can nest any attributes. The loose nested data model reduces the huge time cost data in the process of retrieval as a result of connection between tables. It is more easily for together data to store it on the disk, so the data reading and writing are more quickly.

Cloud storage cache module mainly improves the speed of data reading. The platform adopts MemCache [25], which is an high performance distributed in-memory key-value store system for small chunks of arbitrary data (strings, objects) from results of database calls, API calls, or page rendering, by maintaining a unified and huge hash tables in memory. In simple terms, the applications call the commonly data into memory, and then read it from the memory. Its simple design promotes quick deployment, ease of development, and solves many problems facing large data caches.

In the face of the mass data processing tasks, the platform adopts MapReduce programming model that is put forward by Google [26]. MapReduce model divides a task into many more granular subtasks and these subtasks can be scheduled between the spare processing node, which makes the faster processing node process the more tasks, so as to avoid that the slow processing nodes extend the task completion time.

4.5. *Service interface standand*

Interface encapsulation is based on resource-oriented architecture (ROA), and adopts the generic Internet interface service form , providing RESTful webservice interface. REST stresses on the standard and general operation, which are GET, PUT, POST and DELETE methods provided by HTTP, making a unified handling for the identifies resources based on uniform resource identifier (URI) through the above interface [27]. The followings are the specific interface design.

5. Evaluation

In order to validate and evaluate our platform, we have done an intensive set of experiments executed in Cloud Foundry [28] which is a real cloud computing platform. This section has been divided into two subsections. The first subsection shows the comparison of two mobile application development frameworks: Titanium and PhoneGap, which gives the reason that the platform chooses Titanium as the application development frameworks. The second subsection presents preliminary performance analysis tests conducted in order to evaluate some fundamental operations from the platform.

5.1. *Comparison of MCADFs: Titanium and PhoneGap*

Titanium and PhoneGap can implement the function by calling their respective APIs, which focuses on the hardware access control of mobile devices. The test mainly aims to the two framework joint function, including the access speed to the same hardware equipment, and the time to read and write files, and tests the performance difference between them.

The test performance comparison of Titanium and PhoneGap mainly make use of two kinds of mainstream platform, namely the Android and iOS platforms. Table 1 is the specific test environment.

The same test cases wrote respectively using Titanium and PhoneGap, can specifically illustrate the performance difference of the API calls. Two kinds of code are completely equivalent on the semantic. For the same function we add timestamp in the front and back of the calling API JS statements, respectively

Table 1
Object interface

URL	HTTP	Function
$/api/\langle className\rangle$	POST	Create Object
$/api/\langle className\rangle/\langle ObjectId\rangle$	GET	Get Object
$/api/\langle className\rangle/\langle ObjectId\rangle$	PUT	Update Object
$/api/\langle className\rangle$	GET	Query Object
$/api/\langle className\rangle/\langle ObjectId\rangle$	DELETE	Delete Object
$/api/\langle className\rangle/count$	GET	Count Object
$/api/\langle className\rangle/\langle ObjectId\rangle/exists$	GET	Exists Object

Table 2
User interface

URL	HTTP	Function
$/api/user$	POST	Create User
$/api/user/login$	POST	Log in
$/api/user/logout$	POST	Log out
$/api/user/verfyEmail$	POST	Verfy Email
$/api/user/resetPassword$	POST	Reset Password
$/api/user/\langle ObjectId\rangle$	GET	Get User
$/api/user/\langle ObjectId\rangle$	PUT	Update User
$/api/user/\langle ObjectId\rangle$	DELETE	Delete User

Table 3
Test environment

Related Parameters	Equipment 1	Equipment 2
Operate System	IOS 8.0	Android 4.2
CPU Type (1.3 GHz)	ARMv8 A7	ARM MT6582
Memory Size	1 G	1 G
Network Bandwidth	500 KB/s	500 KB/s

and calculate the different timestamp to achieve the access time.

The test results are averaged for the continuous 10 times tests in each function point, as shown in Fig. 8.

From the Fig. 8, we can see that: the access time of PhoneGap is less than Titanium, but the other function is exactly in the opposite. If the test platform is changed, the access performance keeps a good stability. To summarize, the execution performance of the Titanium surpasses the PhoneGap, which is the reason that the platform chooses Titanium as the application development frameworks.

5.2. *Stress test performance*

The previous section make a comparison on the application development framework Titanium and PhoneGap for the mobile terminal to evaluate their performance. This section will conduct stress tests on the cloud platform. The several operations supported by the platform results from four specific illustrative methods are presented, namely: i) loginPlatform, ii) createApp, iii) getAppStatus and iv) deleteApp.

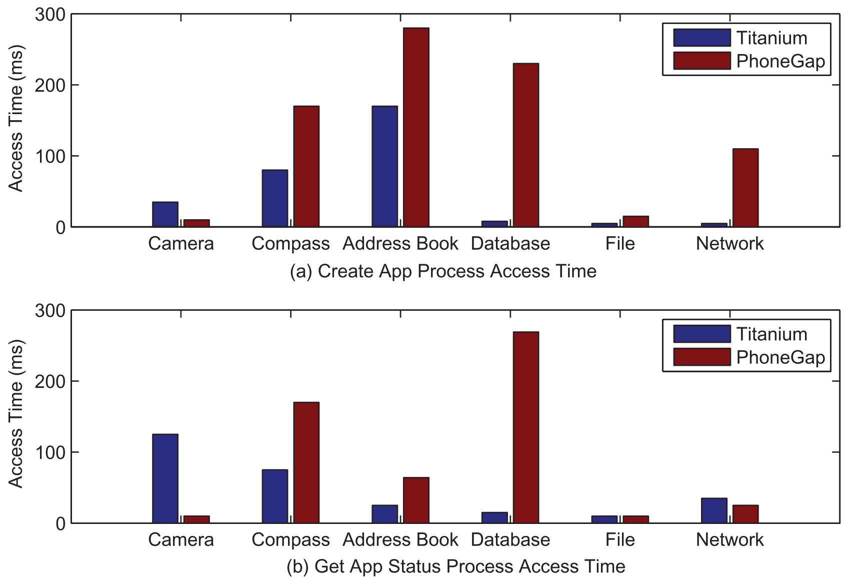

Fig. 8. Comparison of two mobile application development frameworks: Titanium and PhoneGap.

The test is divided into two phases. The first phase tests and compares the performance of the four platforms: IoE-MPP, APICloud, ExMobi, AppCan, which is in the same stress condition. The second phase mainly tests the performance of IoE-MPP under the different pressure environments.

The Apache JMeter tool [29] is used for performing load tests simulating the simultaneous access of several users to the platform. The main analysis metric is the response time. The experiments follow a black box approach measuring the time from request parameters to receiving data. Obviously, this metric is highly variable and depends on several factors such as network conditions, the application server, computational resources, intrinsic PaaS characteristics, etc. The test machine configuration is an Intel(R) Core(TM) 2.53 GHz with 6GB of RAM and Windows 7 as operating system.

In the first phase, IoE-MPP and the other platform are respectively measured under the same stress situation, the number of threads is set to 1. The test bed has been executed 10 times and Fig. 9 shows the average times for the different request arrival times.

As can be seen in Fig. 9(a), the createApp process initializes an application and prepares the execution environment in the specific PaaS provider. IoE-MPP and ExMobi have response times under 500 ms and APICloud has average response times around a value of 1500 ms while especially the response time of AppCan is more than 4000 ms. This discrepancy exists because the internal logic of the operation createApp

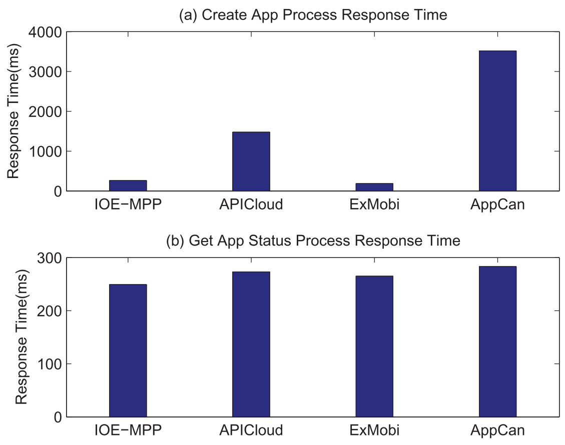

Fig. 9. Comparison of four mobile application development platforms.

is different for each platform. The get AppStatus process, acquires state information about an application, thus providing essential information for developers. The results are presented in Fig. 9(b). Tests show that there are little difference among the four platforms in the aspect of response time values which roughly vary from 250 ms to 300 ms, This is because that the internal logic of adding data is more complex than acquiring data, however, it is obvious that the performance of IoE-MPP is superior to APICloud and AppCan.

Through the comparison of the above four platform, we can see that IoE-MPP is not inferior to

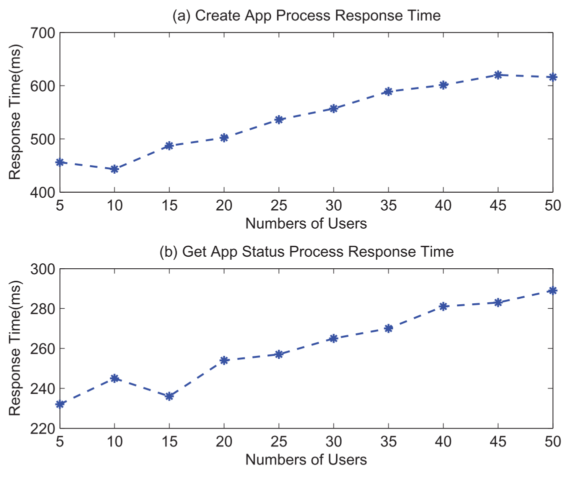

Fig. 10. Response time for different simultaneous users.

the performance of APICloud and AppCan in the case of low load. The next will test the variation of the platform performance under different load conditions. When multi-users concomitantly access the platform, fluctuations in the platform performance should appear, and the size of the fluctuations shows the stability of the platform. For some of the selected methods, tests are performed with different simultaneous users. In the tests of JMeter, the number of thread group represents a certain number of concurrent users. The thread group value t varies from 1 to 50, and the ramp-up period $r = 0.3t$. On each figure, the plotted numbers are the average values obtained among all the users, complemented by the corresponding standard deviation interval.

As can be seen in Fig. 10, whether it is creating App process or getting App status process, the performance of the platform decreases with the increase of the number of the simultaneous users. The reason why there are performance fluctuations is partly caused by the lack of physical hardware, which can be solved by increasing the number of servers, and the other part is resulted from the internal logic, including resource allocation and task scheduling, which requires the optimization in the later work.

6. Conclusion

In this paper, we present a "cloud + container" pattern of IoE mobile portal platform according to the idea of container, MCADF and PaaS technologies, respectively test the performance of mobile end and cloud end via practical deploying and experimenting. The platform can provide the application development, deployment, operations functions for developers, and it is convenient for users to manage the mass applications. When the concurrent increases, the performance needs to be improved. In the future, we will optimize the platform from aspects of resource allocation and task scheduling.

References

[1] D. Zhang, S. Zhao, L.T. Yang, M. Chen, Y. Wang and L. Huazhong, Nextme: Localization using cellular traces in internet of things, *IEEE Transactions on Industrial Informatics* **11**(2) (2015), 302–312.

[2] H. Ning, H. Liu and L.T. Yang, Aggregated-proof based hierarchical authentication scheme for the internet of things, *IEEE Transactions on Parallel and Distributed Systems* **26**(3) (2015), 657–667.

[3] T. Qiu, L. Chi, W. Guo and Y. Zhang, Stets: A novel energy-efficient time synchronization scheme based on embedded networking devices, *Microprocessors and Microsystems* **39**(8) (2015), 1285–1295.

[4] T. Qiu, D. Luo, F. Xia, N. Deonauth, W. Si and A. Tolba, A greedy model with small world for improving the robustness of heterogeneous internet of things, *Computer Networks* **101** (2016), 127–143.

[5] A. Bujari and C.E. Palazzi, Opportunistic communication for the internet of everything. In *Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th*, 2014, pp. 502–507. IEEE.

[6] H. Zhu, E. Chen, H. Xiong, H. Cao and T. Jilei, Mobile app classification with enriched contextual information, *Mobile Computing, IEEE Transactions on* **13**(7) (2014), 1550–1563.

[7] A. Charland and B. Leroux, Mobile application development: Web vs. native, *Communications of the ACM* **54**(5) (2011), 49–53.

[8] Y. Jinfang, A review of the researches on wechat. In *2015 International Conference on Social Science and Technology Education*, Atlantis Press, 2015.

[9] R. Rosen, Linux containers and the future cloud, *Linux J* **240** (2014).

[10] Z. Yaoxue, Transparence computing: Concept, architecture and example [j], *Acta Electronica Sinica* **32**(12A) (2004), 169–173.

[11] Y. Zhang and Z. Yuezhi, Transos: A transparent computing-based operating system for the cloud, *International Journal of Cloud Computing* **1**(4) (2012), 287–301.

[12] P. Smutnỳ, Mobile development tools and cross-platform solutions. In *Carpathian Control Conference (ICCC), 2012 13th International*, 2012, pp. 653–656. IEEE.

[13] J.M. Wargo, PhoneGap Essentials: Building Cross-Platform Mobile Apps. Addison-Wesley, 2012.

[14] I. Dalmasso, S.K. Datta, C. Bonnet and N. Nikaein, Survey, comparison and evaluation of cross platform mobile application development tools. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*, 2013, pp. 323–328. IEEE.

[15] L.-J. Zhang, Editorial: Big services era: Global trends of cloud computing and big data, *IEEE Transactions on Services Computing* (4) (2012), 467–468.

[16] R.-S. Chang, J. Gao, V. Gruhn, J. He, G. Roussos and W.-T. Tsai, Mobile cloud computing research-issues, challenges and needs. In *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*, 2013, pp. 442–453. IEEE.

[17] S. Moghal, S. GharapetDikaleh, O. Sheikh and J. Marini, Mobile application development with ibm worklight and ibm bluemix. In *Proceedings of 24th Annual International Conference on Computer Science and Software Engineering*, 2014, pp. 301–302. IBM Corp.

[18] P.P.-S. Chen, The entity-relationship model-toward a unified view of data, *ACM Transactions on Database Systems (TODS)* **1**(1) (1976), 9–36.

[19] L. Vogel, Android sqlite database and content providertutorial, *Database* **9** (2010), 2.

[20] R. Wu, G.-J. Ahn and H. Hu, Towards hipaacompliant healthcare systems. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, 2012, pp. 593–602. ACM.

[21] X. Zhang and R. Deters, Mvc apps in the personal cloud. In *Cloud Computing and Communications (LATINCLOUD), 2012 IEEE Latin America Conference on*, 2012, pp. 7–12. IEEE.

[22] R. Cattell, Scalable sql and nosql data stores, *ACM SIGMOD Record* **39**(4) (2011), 12–27.

[23] P. Membrey, E. Plugge and D.T. Hawkins, The definitive guide to MongoDB: The noSQL database for cloud and desktop computing, *Apress* (2010).

[24] C.-P. Bezemer and A. Zaidman, Multi-tenant saas applications: Maintenance dream or nightmare? In *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE)*, 2010, pp. 88–92. ACM.

[25] R. Nishtala, H. Fugal, S. Grimm, M.K. Wiatkowski, H. Lee, H.C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab, et al., Scaling memcache at facebook. In *nsdi*, volume 13, 2013, pp. 385–398.

[26] J. Dean and S. Ghemawat, Mapreduce: Simplified data processing on large clusters, *Communications of the ACM* **51**(1) (2008), 107–113.

[27] H. Han, S. Kim, H. Jung, H.Y. Yeom, C. Yoon, J. Park and Y. Lee, A restful approach to the management of cloud infrastructure. In *Cloud Computing, 2009 CLOUD'09 IEEE International Conference on*, 2009, pp. 139–142. IEEE.

[28] CloudFoundry. Cloud foundry. https://www.cloudfoundry.org/, 2016.

[29] E.H. Halili, Apache JMeter: A practical beginner's guide to automated testing and performance measurement for your websites. Packt Publishing Ltd, 2008.