Porting mobile web application engine to the Android platform

Yonghong Wu, Jianchao Luo, Lei Luo
School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054
benbshmily@126.com, andy.luoma@163.com, lluo@uestc.edu.cn

Abstract

Android which Google released as an open-source mobile phone operating system is a Linux-based platform; it consists of the operating system, middleware, and user interface and application software. xFace is a cross-platform web application engine for mobile Internet. As a global leader in mobile web application engine, xFace [1] has the advantages of versatility, easy to master, universal. It significantly reduces the effort of development; while the services of testing which porting layer provides, will greatly reduce the developer's development costs. A perfect transplant will greatly reduce the Android platform migration workload, but also to make widget's operating efficiency to be improved. This paper describes xFace of native code (c++) runs on the Android (java) platform through JNI called, xFace porting layer(c++) also calls the Android platform APIs(JAVA) by JNI, such as Graphics, HTTP and other related system modules.

Keywords—Android, JNI, Mobile web Application Engine, Porting

1. Introduction

Along with the continuous development of communication industry, today's mobile terminals have extended from single service with telephone to multiple services with voice, data, images, video and games [2]. Communications, computing and Internet convergence will be trend of the information and communication technology development in the future. Nowadays, operators’ services have erected one by one in the open data services, a unified service --- the platform for mobile Internet. The step into data transfer is an important stage in the development of the 3G. It provides rich data services also contributed to the development of smart mobile devices. Under the impact of the 3G, mobile terminals gradually turn to be intelligent. Under a variety of 3G services such as mobile instant messaging, mobile email, mobile search, online browsing, mobile TV and mobile games are the hot spot type of value-added services, 3G mobile terminals under a variety of value-added services would be the integration of up to provide users with chain services, it will be a technical major breakthrough. In addition, along with the WEB2.0 maturity, improving, it makes mobile Internet to become an inevitable trend in the future, while the development of next-generation mobile communication terminal is the power of mobile Internet development.

xFace is a kind of engine of the web application. It uses HTML, CSS, and JavaScript and other Web technologies to develop lightweight mobile applications. Mobile applications, which are developed by using the technology of xFace, can run on different operating systems without any modification, such as windows Mobile, BREW, Symbian and other mobile phone platforms. It greatly reduces development efforts and costs of the mobile web application software. Android as an open-source mobile phone platform, is favored by the major mobile phone Manufacturers, its large rate of market share is also increasing. In this context, for mobile Internet and mobile terminals, this paper puts forward the way to solve the xFace transplanted to the Android platform that is very valuable and practical significance.

2. xFace transplant programs in the Android platform

2.1. Android platform features

The architecture of Android system [3], similar to other operating systems, use a hierarchical structure. From the chart to see, Android is divided into four layers; from the top to the lower level are the application layer, application framework layer, system layer and the Linux runtime core layer.

In figure 1, Applications with some of the essential applications, such as email, address book, SMS and browser, etc. All applications are done by the Java language. Application Framework provides java class library for Android applications, Android application
developers developed applications, like developing those core applications, with full permission to access to the APIs provided by framework. Libraries layer was developed by C/C++ language, those libraries are packaged by Application Framework section. The bottom section is the Linux Kernel which is developed by C language, it provide the core system services, such as security, memory management, file system management, process management, network group, Driver Model and so on. It is also an abstract layer between the hardware and system software.

Each of Android applications runs in its own process, to enjoy the Dalvik virtual machine for its exclusive distribution of instances. Dalvik virtual machine runs a Dalvik-format-Executable file (.Dex), the file of this format is optimized so as to make the Android applications occupy less memory when it’s running. Java compiler compiles Java source files into class files, class files has been built into the dex format file by a kind of DEX tool, only this file can run in Dalvik virtual machine after registered in the Dalvik. Thus, all of the Android applications must use java language to develop. This situation gets C/C++ developers into trouble.

2.2. xFace mobile web application engine

xFace is composed of xFace engine and platform adaptation layer (also known as Porting layer), xFace engine is mainly responsible for page layout, script analysis, logic and other functions, the porting layer is mainly responsible for abstracting the operating system-related functions into the system-independent in order to make xFace run on different platform.

xFace engine and porting layer are developed by C/C++, and therefore it’s a big trouble to port the xFace to Android platform.

**Feasibility Analysis:** xFace is developed by C/C++, but the Android application is developed by java language, which allows us to have to use JNI. JNI's called the Java Native Interface, which is provided by JAVA to allow JAVA and the C/C++ interface to call each other. In view of this, JNI provides feasibility for porting xFace to the Android platform. Native code (interface) of xFace can be generated the dynamic library called by JNI in JAVA. Also, because of the Android system functions are supported by JAVA packages (such as SMS, e-mail module, address book, phone calls, local capacity, etc.). Because of all of the Android APIs are based on java package. To finish the operating system-related functions of xFace porting layer, we have to use JNI to call the JAVA classes and methods in C/C++.

**Transplant Program:** According to the feasibility analysis, we know that xFace is ported to Android platform is entirely feasible, so it requires an effective plan to implement. An efficient migration plan can reduce a lot of workload of transplant. Simultaneously, the operating efficiency of widget is also improved.

According to the specificity of Android applications, application framework and start-up code must be written in JAVA language, and then called C/C++ program entry of xFace by JNI in that start-up code. However, in the xFace platform porting layer, some of the features also need to adopt a similar approach to call the Android APIs which are provided as JAVA packages by Android. Event-dispatching can also be the way through the JNI passing from JAVA to C/C++ layer. And then we formed a transplant plan which is mixed-programming language. As shown in Figure 2.
3. Porting the xFace native Code to Android

3.1 Build the environment

Transplanting xFace to the Android platform, because of mixture of programming language, we use Eclipse and the Android environment NDK as the environment of compilation. NDK is primarily used to compile xFace core and porting layer, and it compiles the starting-entry(C/C++) of xFace into dynamic library. Eclipse is mainly used to build Android application development environment, needed ADT plug-in. At last, we use Eclipse to package the program and the dynamic library, which is generated by NDK, into Android executable files – DEX files.

Android NDK: Android NDK is the acronym for Native Development Kit. Android application developers can use NDK to develop C/C++ native code. In this plan, we also need to use it to develop C/C++ native code and call the java classes and methods by JNI. Android NDK has developed to provide a way of developing dynamic library for application developer. Developers need to make their project folder into the folder of “NDK_ROOT / apps /” and write the Makefile in accordance with the specification of NDK. For example, first, we should create a new folder as the name of “xFace” under the directory of “NDK_ROOT / apps /”. Next, we create a new file named “application.mk” and a new directory named “project” under the directory which is created just now. Create a new folder named “jni” in the “project” folder, our xFace Native Code and Makefiles are placed inside this folder. Due to space reasons, we can not do introduce the specifications of the Makefile for the NDK. NDK compiler works under the Linux environment, so if you use Windows, you have to install the software such as Cygwin.

Installing eclipse: After install eclipse software, you should install the ADT plug-in into the eclipse, select the ADT plug-in options and create a new Android emulator. After that you can develop the Android application with it.

3.2 Implementation of the porting layer

Porting layer is an abstraction of operating system functions-related; the implementation of porting layer is the main work of transplantation. It provides interfaces for xFace engines, due to different operating systems platform layer functions do correspond to abstraction, which makes xFace can be run on different platform.

Platform adaptation layer implementation can be divided into two parts. One, it is completed entirely by the C/C++ code, such as threads, semaphores, memory [5] and file operations modules. And local capacity of the other part is related to points in two ways, first by the C/C++ to prepare the framework, and it called Android APIs which are provided by JAVA packages (such as access and set the system time, MMI implementation, call the local function such as phone calls, send text messages and other system calls); the second is a direct implementation by the java function, and then C/C++ side call them which have been achieved by the java (such as HTTP module, etc). As shown in Figure 3.

![Figure 3 Implementation plan of porting layer](image)

3.3 Integration of the porting layer and core

After porting layer of each module completed, we can integrate it with xFace engine. Each module of xFace engine also needs to NDK environment to be compiled into static libraries. All of those static libraries are supplied for xFace program entry module calling. xFace program entry module which is called by JAVA is responsible for the performance of xFace capacity module. This module depends on the xFace engine and platform adaptation layer, and must follow the norms of java call native code compiled into dynamic link libraries and by the Eclipse environment, written in java program packaged with the framework to Android emulator to run.

4. Details of the design

According to the above design, we can divide xFace program into three layers: application framework entrance layer, xFace Native entrance, xFace Native layer (shown as below in figure 4). Following segments elaborate the completion of xFace.
4.1 Application framework entrance

Application framework entrance is primarily responsible for starting the xFace; Developed by JAVA language. And its main function is that prepare for running xFace in the Android platform, to provide a running environment, and the creation of the window and dispatch the event information to xFace native layer. As the Android applications are running on Dalvik virtual machine, so the program must start by the way of JAVA. It is a bridge of communication between xFace Native framework and Android, so it is essential.

4.2 xFace Native entrance

xFace Native entrance is mainly responsible for achieving xFace local businesses, such as loading the page, parsing the page, initialization xFace widget runtime environment, as well as the release of related resources and so on. Associated with xFace business-logic is organized in this layer, which is the work of xFace entrance. Only for Application framework entrance called.

4.3 xFace Native Code

This layer is mainly to achieve graft-layer specific functions, xFace all of the features are in this layer, which coordinates xFace engine and porting layer; and to make porting layer and the underlying of operating system, porting layer and the Android application programming interface layer in the layer of packaging. It exposes the interfaces to xFace application developers and xFace engines. This layer of xFace is the kernel, all of the performance capabilities of xFace application are implemented in this layer.

5. Testing

According to the solution of this paper proposing, the function of porting layer has been implemented. After integrating the porting layer and core, it can run web applications. In order to test the validity and effectiveness of this solution, the following are some simple web application to run screenshot (as shown in figure 5).

There are no exceptions and very steady when the program is running. At this point, it demonstrates the feasibility and correctness of the solution of this paper.

6. Summary

Currently almost all of the transplants, the source build-environment of transplant and the target platform build-environment are basically the same. This migration plan of mixed programming language provides a general method for the situation that the source and the target platform of build-environment of transplant are different; furthermore, this solution also makes code-reuse to maximize and reduces development costs and cycle.

This article based on the Android platform, while providing a set of transplantation program, but the program is also a place to shortcomings in the drawing mechanism, because it is provided through the JNI calling java class library Android than any other platform, the efficiency of direct system call mapping is slightly low. Therefore, it is also a need for further study and improvement to improve efficiency.
7. References