INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

ANDROID OS RELIABLE FOR REAL-TIME APPLICATIONS

DR. SHEEL GHULE

Persistent Systems Limited, Nagpur, India

Accepted Date: 27/02/2014 ; Published Date: 01/05/2014

Abstract: Android platform is a new generation of smart mobile phone platform launched by Google. Android is free and open, providing an easy-to-use development kit containing flexible map display and control functions. The Android operating system (OS) is widely used within several types of embedded & mobile platforms, including mobile phones and tablets, and the industry is exploring the ability of Android within other embedded platforms, i.e., automotive or military, that require real-time guarantees and the ability to meet deadlines as a pre-requisite for reliable operation. This paper gives highlights on the use of real-time applications.

Keywords: Android, Reliable, Real-Time

PAPER-QR CODE

Corresponding Author: DR. SHEEL GHULE

Access Online On:

www.ijpret.com

How to Cite This Article:

Sheel Ghule, IJPRET, 2014; Volume 2 (9): 731-738

INTRODUCTION

Traditional studies on the reliability of software focus on functional failures, and do not emphasize the time-related behavior of systems that can also cause the software to fail. The ability to meet deadlines and time constraints is critical to embedded systems software that mandate response to stimuli within prespecified real-time design specifications, and reliability considerations require a detailed evaluation of the ability of the system to meet these specifications. Android provides the support of mobile map and location service, which is probably a concern of vast numbers of developers. So far, the development of mobile map and location applications is complex and difficult, and is often required to pay high copyright fees to map makers. Android is free and open, providing an easy-to-use development kit containing flexible map display and control functions.

The Android OS is an operating system primarily designed for mobile platforms by Google. It is an open source OS based on LINUX kernel. Android is finding widespread acceptance in the mobile and portable computing market, and this study examines, for the first time, its performance & reliability in more demanding embedded real-time applications.

The Android operating environment can be labeled as:

- An open platform for mobile development
- A hardware reference design for mobile devices
- A system powered by a modified Linux 2.6 kernel
- A run time environment
- An application and user interface (UI) framework

II. ANDROID ARCHITECTURE

Android platform is open system architecture, with versatile development and debugging environment, but also supports a variety of scalable user experience, which has optimized graphics systems, rich media support and a very powerful browser.

The Android system supports background processing, provides a rich user interface library, supports 2-D and 3-D graphics using the OpenGL-ES standard and grants access to the file system as well as an embedded SQLite database.

An Android application typically consists out of different visual and nonvisual components and it enables reuse and replacement of components and an efficient database support and support various wireless communication means.

The Android system is typically divided into the four areas as depicted in the following graphic.

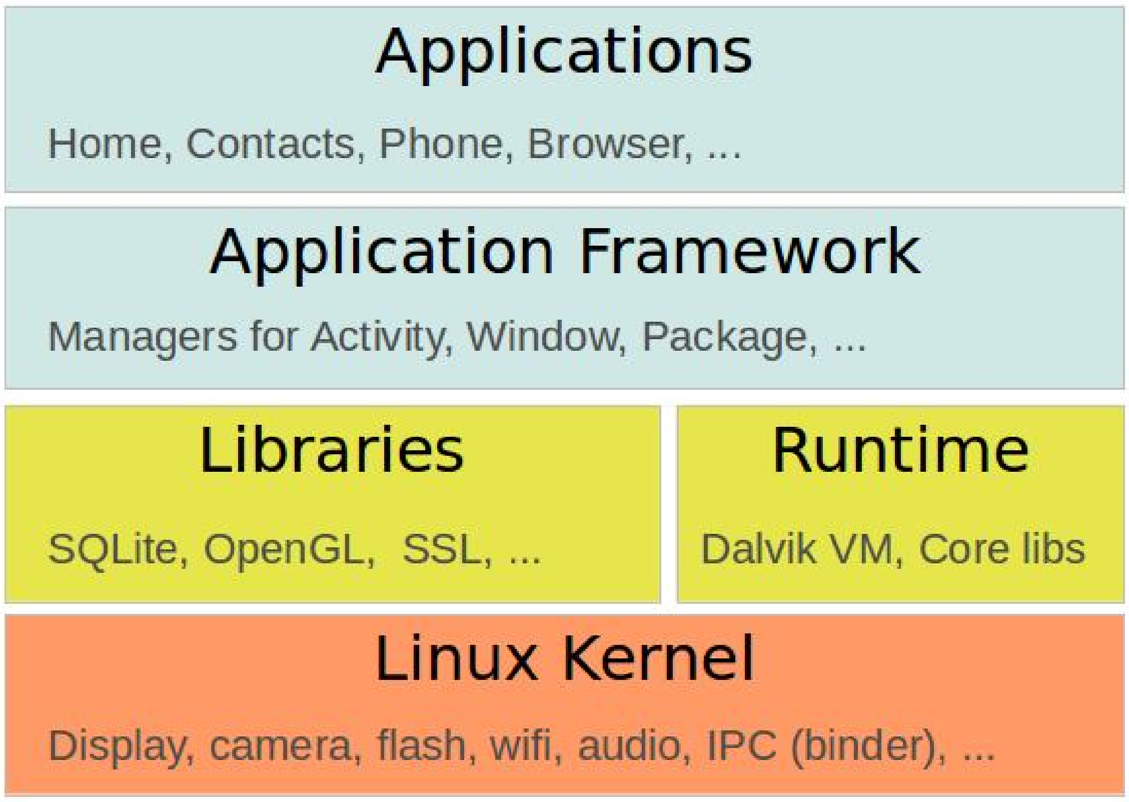

Applications
Home, Contacts, Phone, Browser, ...
Application Framework
Managers for Activity, Window, Package, ...
Libraries
SQLite, OpenGL, SSL, ...
Runtime
Dalvik VM, Core libs
Linux Kernel
Display, camera, flash, wifi, audio, IPC (binder), ...

Fig. Android OS Architecture

A. Applications:

The Android Open Source Project contains several default application, like the Browser, Camera, Gallery, Music, Phone and more. A set of core applications are on the top level in the framework, including an email client, a SMS app, a calendar, a maps-application, web browser, contacts-app, and many more.

B. Application framework:

API which allows high-level interactions with the Android system from Android applications. The application architecture is designed to simplify the reusing of all components. This mechanism allows every component to be replaced by the user.

C. Libraries and runtime:

Libraries for the Application Framework for many common functions (graphic rendering, data storage, web browsing, etc.) as well as the Dalvik runtime and the core Java libraries for running Android applications.

Every Android application runs in its own process given by the OS, and owns its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM is executing files in the .dex (Dalvik Executable) format which was optimized for minimal cpu-and-memory-usage.

D. Linux kernel:

Communication layer for the underlying hardware. Android relies on Linux (Kernel version 2.6) for core system services such as memory management, process management, network stack, security, and driver model. The core also acts as a hardware abstraction layer between the applications and all the hardware.

III. ADVANTAGES OF DALVIK VIRTUAL MACHINE

Android based systems utilize their own virtual machine (VM), which is known as the Dalvik Virtual Machine (DVM) [4]. The DVM uses special byte-code, hence native Java byte-code cannot directly be executed on Android systems. The Android community provides a tool (dx) that allows converting Java Dominique A. Heger (www.dhtusa.com) 3 class files into Dalvik executables (dex). The DVM implementation is highly optimized in order to perform as efficiently and as effectively as possible on mobile devices that are normally equipped with a rather modest (these days normally a dual, or quad) CPU subsystem, limited memory resources, no OS swap space, and limited battery capacity. The DVM has been implemented in a way that allows a device to execute multiple VM's in a rather efficient manner. It also has to be pointed out that the DVM relies on the modified Linux kernel for any potential threading and low-level memory management functionalities. With Android 2.2, some major changes to the JVM infrastructure were implemented. Up to version 2.2, the JVM was an actual interpreter, similar to the original JVM solution deployed with Java 1.0. While the Android solution always reflected a very efficient interpreter, it was still an interpreter and hence, no native code was generated. With the release of Android 2.2, a just-in-time (JIT) compiler was incorporated into the solution stack, which translates the Dalvik byte-code into much more efficient machine code (similar to a C compiler). Currently, Android version 4 (Ice Cream Sandwich) and 4.1/4.2 (Jelly Bean) is deployed on some devices. It has to be pointed out though that currently, only a few devices are actually running version 4.1/4.2 or 4.0, while most devices are still operating on older

Android versions. Down the road, additional JIT and garbage collection (GC) features will be deployed with Android, further busting aggregate systems performance.

IV. ANDROID APPLICATION COMPONENTS

A. Activity

An activity is the visual representation of an Android application. An Android application can have several activities.

Activities use views and fragments to create the user interface and to interact with the user.

An activity is usually a single screen in your application. Each activity is implemented as a single class that extends the Activity base class. Your class will display a user interface composed of Views and respond to events.

B. Broadcast Receiver

A broadcast receiver is a component that responds to system-wide broadcast announcements. A broadcast receiver (receiver) can be registered to listen to system messages and intents. A receiver gets notified by the Android system if the specified event occurs.

For example, you can register a receiver for the event that the Android system finished the boot process. Or you can register for the event that the state of the phone changes, e.g., someone is calling.

C. Service

A Service is code that is long-lived and runs without a UI. A service performs tasks without providing a user interface. They can communicate with other Android components, for example, via broadcast receivers and notify the user via the notification framework in Android.

A good example of this is a media player playing songs from a play list. In a media player application, there would probably be one or more activities that allow the user to choose songs and start playing them. However, the music playback itself should not be handled by an activity because the user will expect the music to keep playing even after navigating to a new screen. In this case, the media player activity could start a service using the Context. start Service method to run in the background to keep the music going. The system will then keep the music playback service running until it has finished. Note that you can connect to a service with the Context. bind Service method. When connected to a service, you can communicate with it through an interface exposed by the service. For the music service, this might allow you to pause, rewind, etc.

D. Content Provider

A content provider (provider) defines a structured interface to application data. A provider can be used for accessing data within one application, but can also be used to share data with other applications.

Android contains a SQLite database which is frequently used in conjunction with a content provider. The SQLite database would store the data, which would be accessed via the provider.

V. POWER MANAGEMENT, SECURITY AND MARKET SHARE

A. Power Management

In the mobile device arena, power management is the most important area. That does not imply though that power management should be neglected on any other system. To illustrate, to reduce and manage power consumption, Linux based systems provide power-saving features such as clock gating, voltage scaling, activating sleep modes, or disabling memory cache. Each of these features reduces the system's power consumption (normally at the expense of an increased latency behavior). Most Linux based systems manage power consumption via the Advanced Configuration and Power Interface (ACPI). Android based systems provide their own power management infrastructure (labeled Power Manager) that was designed based on the premise that a processor should not consume any power if no applications or services actually require power. Android demands that applications and services request CPU resources via wake locks through the Android application framework and native Linux libraries. If there are no active wake locks,

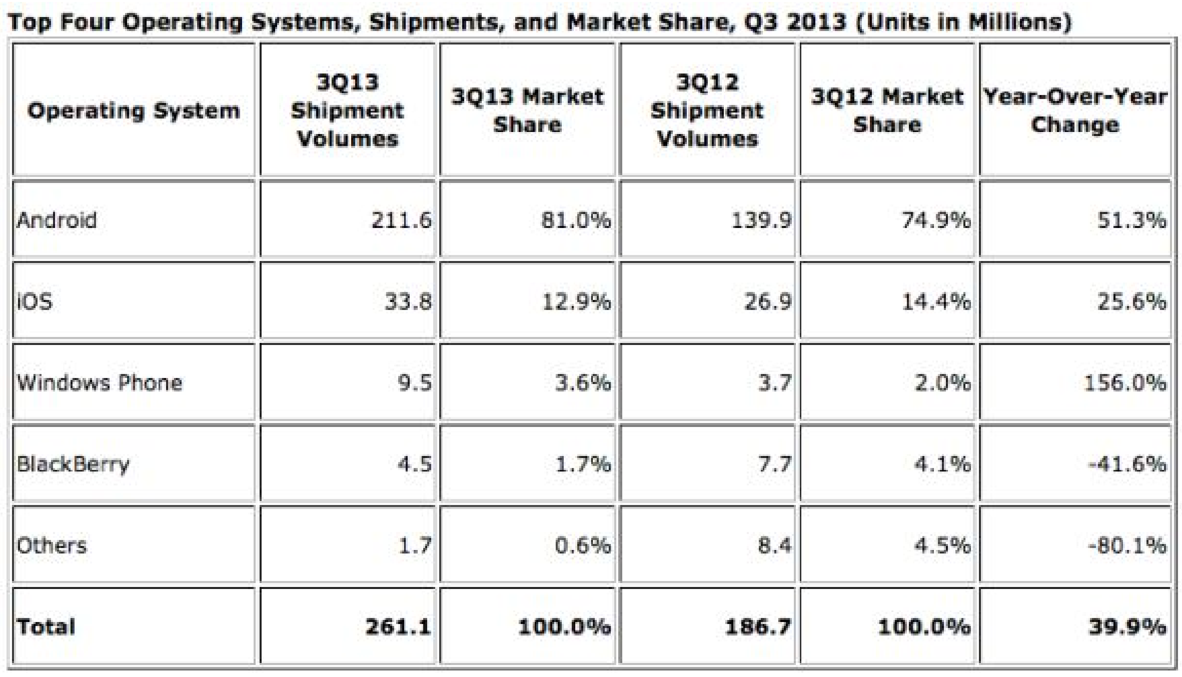
Android will shut down the processor.

B. Security

Each application runs in sandbox environment to enforce security in Android and this is done by assigning each application a unique user ID and running that application as that user in separate process.

C. Market Share

According to the International Data Corporation (IDC) Worldwide Quarterly Mobile Phone Tracker, the Android platform just crossed 80% market share in the third quarter of 2013. With 211.6 million smart phones shipped in this quarter, Android continues to maintain its dominant position in the market.



Top Four Operating Systems, Shipments, and Market Share, Q3 2013 (Units in Millions)

Operating System	3Q13 Shipment Volumes	3Q13 Market Share	3Q12 Shipment Volumes	3Q12 Market Share	Year-Over-Year Change
Android	211.6	81.0%	139.9	74.9%	51.3%
iOS	33.8	12.9%	26.9	14.4%	25.6%
Windows Phone	9.5	3.6%	3.7	2.0%	156.0%
BlackBerry	4.5	1.7%	7.7	4.1%	-41.6%
Others	1.7	0.6%	8.4	4.5%	-80.1%
Total	**261.1**	**100.0%**	**186.7**	**100.0%**	**39.9%**

(Credit: IDC)

VI. CONCLUSIONS

The very impressive, rapid evolution of Android resembles the great work done by the Linux community over the years. Android is not a Linux solution per se, but does utilize a modified Linux 2.6 kernel that is incorporated into the Android operating environment.

Use of Android OS should conduct measurements of its behavior carefully to gauge the combined effects of slippage, its frequency and value, and its accumulation, on the reliability of their system.

REFERENCES

1. Android - An Open Handset Alliance Project, http://code.google.com-/intl/zh-CN/android/..

2. Android Developers, http://www.androidin.com/.

3. N. Gramlich, Android Programming, PDF Electronic Book, 2008. Available from: http://androidos.cc/dev/index.php.

4. Android Tutorial, http://www.vogella.com/tutorials/Android/article.html

5. Introduction to Android https://code.google.com/p/androidgroup/downloads/detail?name=Introduction%20to%20Android.pdf

6. Android Architecture http://www.android-app-market.com/android-architecture.html

7. Application Fundamentals http://developer.android.com/guide/components/fundamentals.html

8. Maker, F., Chan, Y., "A Survey on Android vs. Linux", University of California, 2009

9. Liang, "System Integration for the Android Operating System", National Taipei University, 2010

10. Brady, P., "Android Anatomy and Physiology", Google I/O Developer Conference, 2008

11. Bornstein, D., "Dalvik VM Internals", Google I/O Developer Conference, 2008

12. Toshiba, "NAND vs. NOR Flash Memory: Technology Overview", Toshiba, 2006

13. Johnson, "Performance Tuning for Linux Servers", IBM Press, 2005

14. Heger, D., "Quantifying IT Stability – 2nd Edition, Instant Publisher, 2010

15. Android Wikipedia, 2011

16. Linux Wikipedia, 2011

17. Learn Android, http://developer.android.com/guide/basics/what-is-android.html

18. Dalvik architecture: www.cnet.com