Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 6, June 2013, pg.46 – 52

SURVEY ARTICLE

Android Optimization: A Survey

Ms. Debosmita Sen Purkayastha[1], Mr. Nitin Singhla[2]
[1]Department of Computer Science and Engineering & M.M. University, Mullana, India
[2]Asst Prof. Department of Computer Science and Engineering & M.M. University, Mullana, India

[1] debos.senp@gmail.com; [2] nnmailin@yahoo.co.in

Abstract— Android can be rightly defined as a software stack as just like any pile of objects arranged in a stack, this software stack includes an operating system at its bottom layer, a middleware layer and a bunch of key applications along with a collection of APIs for writing applications at its upper layer. In one of these layers there is an android runtime which actually empowers an android device and differentiates it from any other mobile development platform. The runtime platform for android is chosen to be Dalvik virtual machine instead of java virtual machine. The choice made by Google has strong reasons for this variation. This paper tries to take a broad look at the architecture and design of android runtime.

Key Terms: - Dalvik Virtual Machine; Android Runtime Optimization; Bytecode optimization; dex file

I. INTRODUCTION

Android has a layered architecture. Because of its open source nature [6] and an extremely well designed and efficient architectural model android is touching high skies in the market. It is assumed to overtake the i-phone market very soon. In 2013, android market share has increased drastically. Every small and big company is choosing android as their device's operating system.

A walk through in past decades shall take us to the era of embedded device programming where there was an extensive use of proprietary development tools. It somewhere restricted the knowledge of mobile application development to hand full folk. The solution to this problem was very expensive. But then arrived a dream come true platform called ANDROID.

The operating system approximately consists of 12 million lines of code includes 1.75 million lines of C++, 2.1 million lines of Java, 2.8 million lines of C and 3 million lines of XML code.

II. BRIEF HISTORY OF EVOLUTION OF ANDROID AS AN OPEN SOURCE

Android Inc was founded in Palo Alto, California, United States, October 2003.

The people behind this platform development are namely Andy Rabin, Chris White, Rich Miner and Nick Sears,

Android Inc acquired by Google in 2005. The initial developers then worked under Google for further development of Android. Google has selected java as a language for application development and chosen a non-standard virtual machine for deployment.

Open Handset Alliance is a consortium of telecom, hardware and software companies which came together to build an open environment for mobile development. The first product to be released under OHA was Android by Google [7].

Android License

Licensing is done under two open sources namely:

1. GNU General Public License (GPL) for licensing the release of Linux Kernel.
2. Apache Software License (ASL) for licensing Android Platform excluding Linux kernel [10].

III. ANDROID ARCHITECTURE

Android architecture includes five layers:

A. Applications layer

It includes all the native applications which were shipped with device and the third party applications

B. Application Framework layer

It includes the application managers like windows manager, location manager, notification manager etc. These allow developers to use various capabilities of Android operating system

C. Libraries

It includes the core libraries written in C/C++ such as SGL for graphics, SQLite for native database support etc.

D. Android Runtime

It is the engine which empowers applications. It includes Dalvik virtual machine and core android libraries.

E. Linux kernel

It provides abstraction layer between hardware and remaining stack. It includes device drivers and is responsible for handling core services such as power management, memory management etc.

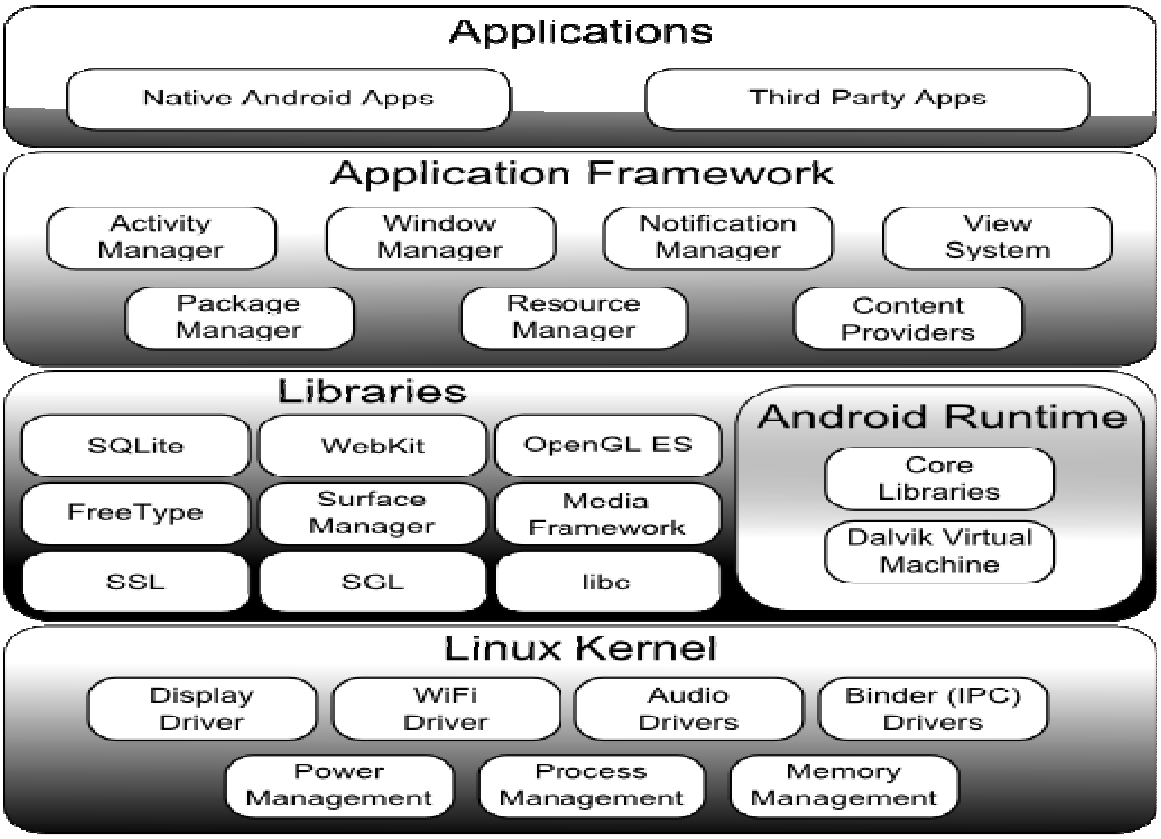

Fig. 1 Android Architecture

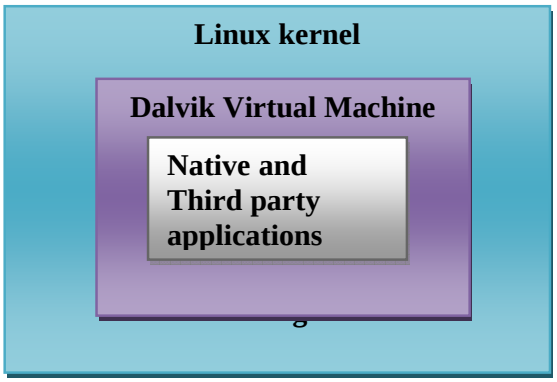

Fig 2 Android software environment

IV. ANDROID RUNTIME

A. Virtual Machine

It can be rightly defined as a software or virtual implementation of a machine which executes like physical machine. [11] Based on the degree of use and similarity to real machine, virtual machine can be broadly divided into two categories:

1) *System Virtual Machine***:** It can be defined or thought of as an execution environment which is virtually a complete operating system. An existing architecture is emulated which provides the functionality of running multiple instances of virtual machines without the presence of real hardware. Guest OS allows different operating systems to run in the same computer

2) *Process Virtual Machine* *It* can be thought of as a language virtual machine which, in general, is used to execute a single program i.e., aimed to handle a single process. It is created when that process is started and destroyed when it exits. It provides a platform independent programming environment which in turn allows a program to execute in multiple platforms alike. Java Virtual Machine or JVM is of this kind. It is also called an Application Virtual Machine.

B. Java Virtual Machine (JVM)

JVM is particular about the operating system i.e., there is a JVM which meant to execute on a specific operating system. For Linux, say there is a particular flavour of JVM. Similar is the case with Windows or any other OS. Java programs are compiled by the java compiler into bytecode. The bytecode is interpreted and executed. JVM provides run-time environment in which java bytecode is executed. Hence it is rightly tagged as "Write Once, Run Anywhere".

JVM runtime is going to execute .class files or .jar files by emulating the .JVM instruction set by either interpreting it or using a JIT (Just-In-Time) compiler. JVM has a stack-based architecture. JVM is a platform of many different languages such as Scala, Fantom, Clojure etc.

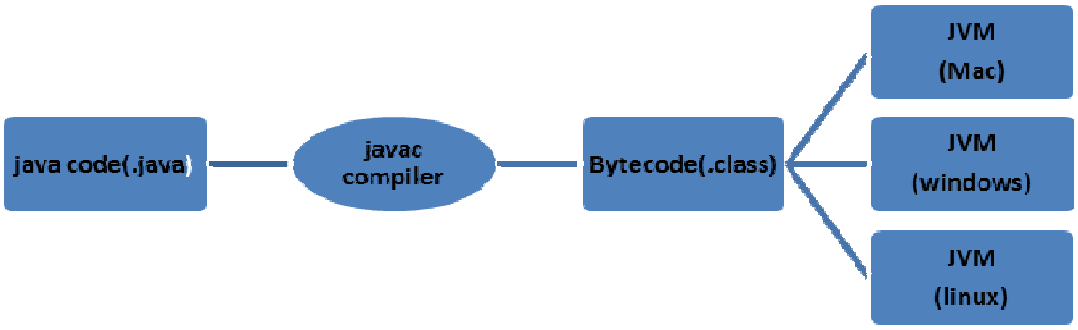

Fig 3 Java Virtual Machine Architecture

C. Dalvik Virtual Machine (DVM)

DVM is a process virtual machine (VM) targeted for Android operating system. DVM uses register-based architecture. It is designed in a way that each application runs on its own instance of DVM. So that crash of one

application doesn't affect the execution of any other application running in device. The software which runs applications (apps) on a android device is DVM.

Dalvik is open-source software and got its name from a place called Dalvik in Iceland. Google has chosen Dalvik Virtual Machine for android instead of any standard virtual machine as DVM was designed in a way so that a device can successfully run multiple instances with efficiency.

An android application need to run on different flavours of devices ranging from a mobile phone to a tablet computer efficiently and accurately. Google has decided java as the language for application development but it is also possible to write applications in C/C++. DVM uses Linux kernel for managing the low-level functionalities such as threading, security etc [1]. So a C/C++ application will run directly upon the Linux kernel OS. Android provides a native development kit (NDK) which allows a developer to create libraries in C++.

D. The dx tool

Java source code is converted into .class files. As soon as ".class" files are generated, they are converted into a ".dex"(Dalvik executables) by "dx" tool. A .class file contains only one class whereas a single .dex file contains multiple classes. The .dex file is executed in DVM. Redundant strings and constants are included only once in the .dex so that space can be conserved. They can share data from a single copy.

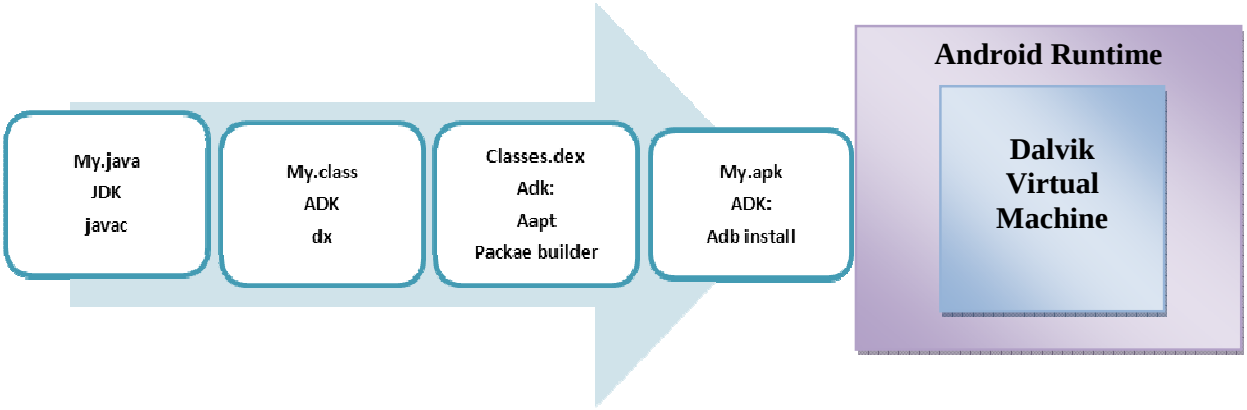

Fig 4 Execution path

E. The dex file format

In standard java environment, the source code is converted into bytecode which in turn is stored into .class files. If have say hello.java file which contains two public classes, two static classes and three other classes then compiler will generate seven .class files altogether. The .class files are executed in JVM.

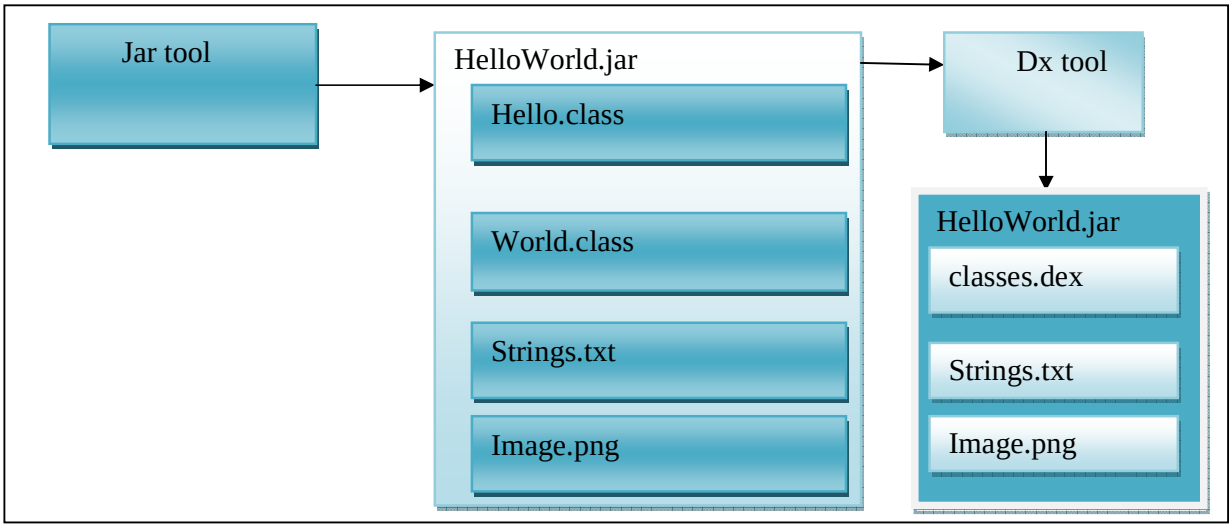

Fig 5 dx tool converts .class files into classes.dex

Android further compiles the compiled classes and convert into a more compressed file called the .dex file. In the above given example, seven .class files will be compressed into one .dex file (Dalvik executable file) i.e., a .dex file can contain multiple classes whereas a .class file can contain only one class. The .dex file is executed in Dalvik virtual machine.

A constant pool contains literal constants within class. These values are field, class, interface and method names as well as string values. The constant pool is called heterogeneous i.e., all types of constants are mixed together. There is a duplication of constants across .class files. In case of .dex file a single type-specific constants [12] pool which is shared by multiple classes and hence supports data sharing. The values are always referred to by their indexes in the constant pool.

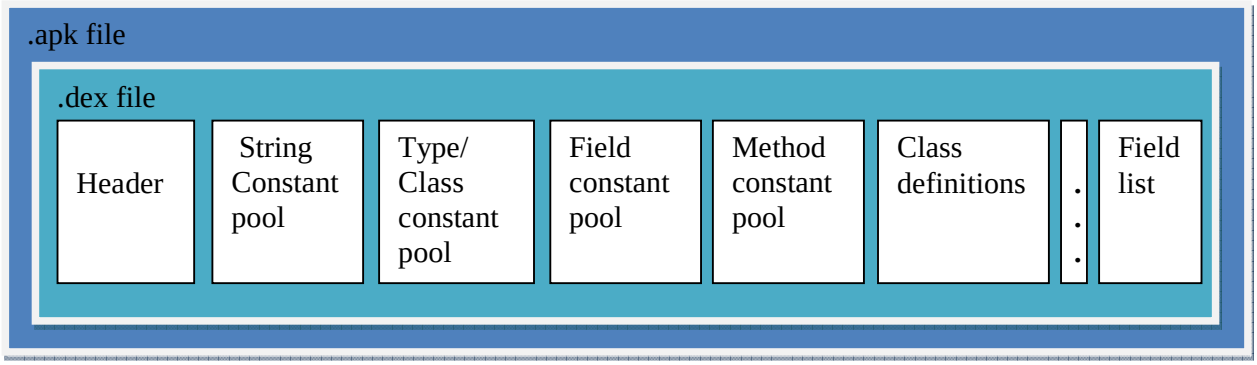

Fig 6 .dex file

Since multiple classes share same constant pool hence redundancy is removed. Less repetition results into more logical and comprehensive pointers within a .dex file as compared to .class file. Thus optimization of constant pool resulted in an effective memory management.

F. The Zygote

Zygote is a virtual machine process which starts at system boot time. As soon as zygote starts, Dalvik VM is initialized. It in turn preloads and pre initializes core library classes. These core libraries are read-only files and hence can be easily shared across processes. Zygote starts up at init.

After initialization zygote waits for socket requests from runtime processes so that it could fork a new instance of virtual machine. After an application is started, the zygote is forked, so there are two VM(s).These pre-loaded libraries are read-only. This memory will be copied only if the new process tries to modify it. Fig 8 shows an example of zygote.

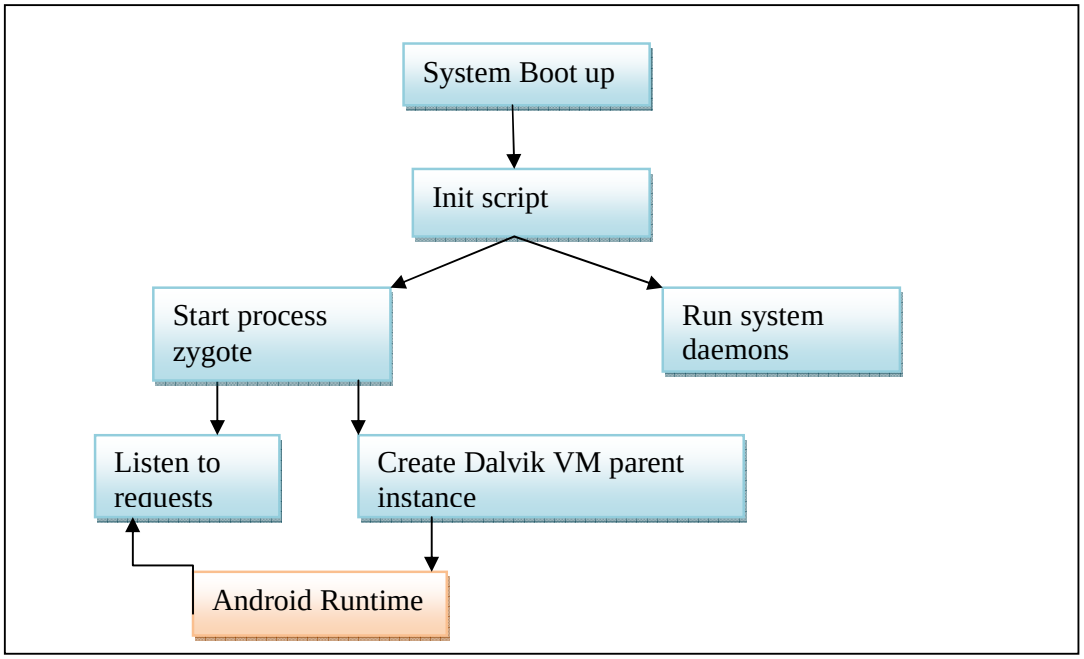

Fig 7 Zygote

In JVM, each instance of VM contains an entire copy of core libraries [8] and associated heap. Hence memory is not shared across Instances.

G. Register-Based Architecture

In general, VM has stack-based architecture and the reason behind this is that implementation is very easy and writing a compiler for stack based architecture is very convenient [4]. But performance is at risk then. Fig 8 shows a general example of register-based architecture example for the instruction where the operand in R1 and R2 are added and the result is stored in R3

ADD R1, R2, R3

Study shows register-based architecture requires approx 47% less VM instructions as compared to stack-based machines. Problem associated with this is that the size increases by almost 25%. [5].It is because average instruction size is larger as operations are carried out at register level and there is no push pop operation as in stack-based architecture.

DVM is a register-based virtual machine. The increase in code size is compensated by 50 % with the use of .dex file. It is made possible by sharing pool constants. Performance is quiet high when execution is to be carried out in mobile devices

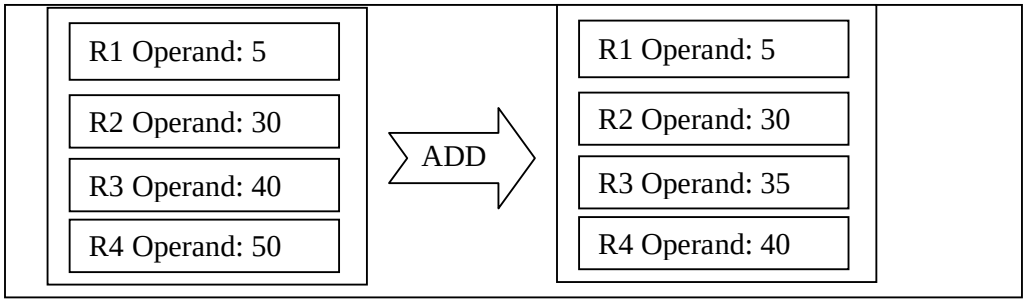

Fig 8 Example of ADD R1, R2, R3

H. Core Libraries

It includes the libraries which are specific to Dalvik VM, java compatibility libraries and Third-party utility libraries.

1) *Dalvik VM Specific Libraries:* These include system information, debugging etc. They allow to request or modify VM-specific information. It includes the VM-specific packages which are dalvik.annotation, dalvik.bytecode and dalvik.system
2) *Java Compatibility Libraries:* These include base and utility classes. These libraries are responsible for providing a familiar environment for programming in Java language.
3) *Third Party Utility Libraries:* These include Apache Http Client 4.0.These libraries provide utilities for encoding and decoding, Bluetooth support, JavaScript Object Notation etc

V. CONCLUSIONS

Dalvik Virtual Machine is a very efficient byte code interpreter. It uses a register-based architecture. It executes .dex files which are transformed java bytecode. Core libraries provided utility to the apps. Thus, Google has selected a justified platform for android execution.

REFERENCES

[1] What is Android? "Android Developers <http://developer.android.com/guide/basics/what-is-android.html>
[2] Lindholm, Tim and Yellin, Frank. The Java Virtual Machine Specification. Second Edition. Sun Microsystems, 1999: Chapter 4
[3] Pavone, Michael, "Dex flle Format"<http://www.retrodev.com/android/dexformat.html>
[4] Jones, Derek. "Register vs. stack based VMs" http://shape-of-code.coding-guidelines.com/2009/09/register-vs-stack-based-vms/
[5] Security Engineering Research Group, Institute of Management Sciences. "Analysis of Dalvik Virtual Machine and Class Path Library" 2009
[6] http://en.wikipedia.org/wiki/Android_(operating_system)

[7] Open Handset Alliance, – Android overview, http://www.openhandsetalliance.com/android_overview.html.
[8] Bornstein
[9] Android Official Webpage, http://www.android.com
[10] Ed Burnette, Hello Android, 3rd edition, The Pragmatic Bookshelf Raleigh, North Carolina Dallas, Texas.
[11] Virtual Machine, http://en.wikipedia.org/wiki/Virtual_machine
[12] Bornstein, Dan. "Dalvik VM Internals" 2008 Google I/O Session Videos and Slides<http://sites.google.com/site/io/dalvik-vm-internals>

