Volume: 10 Issue: VIII Month of publication: August 2022
DOI: https://doi.org/10.22214/ijraset.2022.46478

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VIII August 2022- Available at www.ijraset.com

An Empirical Study on Architecture and Efficiency of Dalvik VM and Android Runtime (ART)

Nishanthan M R
B. Tech Computer science and Engineering, Lovely Professional University

Abstract: Many mobile platforms have developed as a result of the development of mobile technology. Rapid technological advancement has led to the widespread adoption of mobile devices with excellent graphics. Since its inception, the Open Handset Alliance project known as Android has undergone numerous revisions. Being centred on efficiency and pace, it's development. Android OS is driven by the Dalvik microkernel. The Android OS is sluggish due to the runtime as it is a virtual system. Android's multitasking capability leads to its latency. Earlier, there have been efforts to speed up Dalvik, but they were inefficacious.

Increasing Android's speed and optimization was severely hampered by this. The Android Runtime (ART), which takes the role of the Dalvik Virtual Machine, made its debut when Android 4.4 was released by Google. A fast gadget was made possible by ART, which skillfully blends portability and performance, which eventually enhanced graphics optimization. A pre-compiled program was maintained for the Android applications by ART, which was not feasible with Dalvik Virtual machine. A speed of double has been observed on the Android Operating system, in several experiments.

This paper discloses the architecture of android virtual machines (Dalvik and ART), memory optimization and efficiency of both in various devices. Also lists out the for and against of the latter one over the former and the basic functions of the runtimes.

Keywords: Android Runtime, Dalvik, Virtual Machine, Efficiency, Android, Mobile platform, DVM, ART

I. INTRODUCTION

Google's Android is a robust operating system competing with Apple's iOS, with great and enhanced features like an attractive user interface, connectivity, storage, resizable widgets, google cloud messaging (GCM), etc., A runtime is a layer of a computer program in which the code is executed and handles the tasks necessary to run the main program. Android introduced two runtimes called Dalvik and ART, which are utilized by applications and some android services, that enhance the performance, memory and battery of the system. [1]

Android is an operating system for mobile devices like smartphones and tablet computers that is open source and based on Linux. The Open Handset Alliance, headed by Google, and other businesses created Android. Android provides a standardized approach to mobile application development, so developers simply need to build for Android and their applications should be able to run on various Android-powered devices.

Google launched the first beta version of the Android Software Development Kit (SDK) in 2007, but Android 1.0, the first commercialized version, wasn't released until September 2008. The table below represents various versions of Android till now.

NAME	VERSION NUMBER	API LEVEL
Android 1.0	1.0	1
Android 1.1	1.1	2
Cupcake	1.5	3
Donut	1.6	4
Éclair	2.0 – 2.1	5 – 7
Froyo	2.2 – 2.2.3	8
Gingerbread	2.3 – 2.3.7	9 – 10
Honeycomb	3.0 – 3.2.6	11 – 13
Ice Cream Sandwich	4.0 – 4.0.4	14 – 15
Jelly Bean	4.1 – 4.3.1	16 – 18
KitKat	4.4 – 4.4W.2	19 – 20
Lollipop	5.0 – 5.1.1	21 – 22

Marshmallow	6.0 – 6.0.1	23
Nougat	7.0 – 7.1.2	24 – 25
Oreo	8.0 – 8.1	26 – 27
Pie	9	28
Android 10	10	29
Android 11	11	30
Android 12	12	31
Android 12L	12.1	32
Android 13	13	33

Legend: 🟧 Old version 🟨 Older, still maintained 🟩 Latest version

II. ANDROID VIRTUAL MACHINE

Google developers choose to use Android with Virtual machines for a plethora of reasons, but the key ones are as follows,

1) *Security:* The guest system cannot potentially escape the VM because it is completely separated by the virtual machine and cannot even "see" the host, much less attack it. Of course, it has occasionally occurred in practice. A security flaw—a programming error that turns out to have unfavourable effects—must be exploited in the VM implementation or, conceivably, the hardware features the VM depends on. There aren't many ways to get data out of the virtual machine; for instance, while accessing the Internet, the VM simulates a virtual network card that only handles the simplest packets and not complete TCP/IP, therefore the majority of IP-stack problems are contained within the VM. As a result, bugs that cause VM to break tend to be quite rare. [2]
2) *Platform Independent:* Different devices with various architectures can run the Android platform. The role of virtual machines (VM) is to abstract the requirement to build binaries for each architecture.

a) *Dalvik Virtual Machine:* Dalvik is a dropped VM in Android OS which was used to execute the android applications. It is a register-based virtual machine which was used by earlier versions of Android (before Android 5.0 – API 21) to optimize the mobile environment. It was developed by Dan Bornstein and Google technicians who built the Android mobile platform. The name Dalvik is derived from the name of the town in Iceland, where Dan's predecessors used to live.

b) *Android Runtime:* ART is the runtime which was introduced in the Android KitKat, the eleventh Android mobile operating system, which represents version 4.4. This came as the optional one then, which users can toggle between Dalvik and ART runtimes. ART is written in C, and C++ and licensed to Apache 2.0. It compiles the application's bytecode into native code that is then executed by the runtime. The very first version of Android which included only ART as a runtime was the 5.0 version, Lollipop. [3]

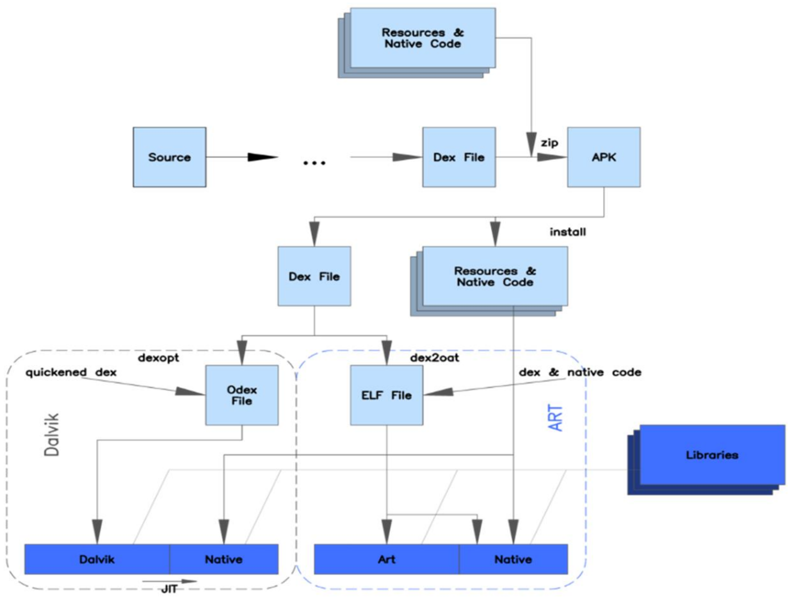


Fig. 1 Comparison of Dalvik and ART architectures

III. ATTRIBUTES OF DALVIK AND ART

A. Dalvik

1) Dalvik virtual machine uses the Just-In-Time (JIT) strategy, which uses the least amount of RAM but causes higher application wait times. [4]
2) It takes less time to start up because the cache makes its updates over time.
3) As there is greater room, it works better for low internal memory.
4) Time is a robust virtual machine that is good for engineers.

B. ART

1) ART uses the Ahead-Of-Time (AOT) technique, which compiles applications as they are installed, reduces memory requirements on the processor and speeds up waiting times.
2) The initial boot creates a cache, however restarting the device requires more time.
3) Consume more internal memory because ART optimizes the APK.

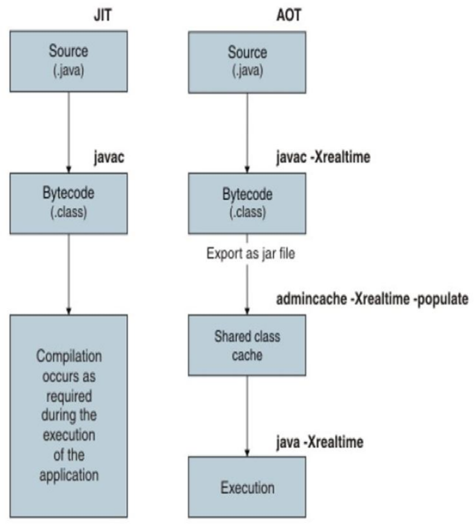


Fig. 2 Just-In-Time vs Ahead-Of-Time

C. Optimization and Memory

1) *DVM:* Although it is no longer used during runtime in more recent versions of Android, Dalvik bytecode format is still used as a distribution format. Dalvik is a Linux operating system that runs on top of Android. DVM converts Java code from Android apps into bytecode that the Linux operating system can execute. In essence, a simple Java program is translated into bytecode during runtime by the JIT compiler and executed on the computer. This can cause a slowdown because runtime compilation, especially during runtime, takes time. As a result, OEMs and manufacturers occasionally provide their software as odexed. Dalvik has two executable files named .dex(Dalvik Executable file) and .odex. The former file contains the compiled code for an android. Then, a single .apk file made from these .dex files is compressed. The Android operating system produces .odex files to reduce storage requirements and speed up an app's startup (.apk file). ODEX (optimized DEX), which contains the optimized bytecode, is created by using dexopt to optimize DEX. For devices with less storage, DVM is preferable. However, because compilation happens after installation, it takes longer.

Consequently, the entire DVM procedure can be summed up as Java source code(.java) – Bytecode(.dex) – DVM

2) *ART:* When apps are installed on a device, ART makes sure that they are fully compiled. Higher performance as there is no longer a requirement to translate code to bytecode and then compile. The drawback is that since it must constantly reside on the device due to compilation during installation, you need more storage space and it takes a little longer to install. Consequently, we have greater bytecode/machine code instead of relatively small java code. odexed and de-odexed are words you may have come across. In this case, a tiny bit of the application is precompiled so that the developer can go ahead and construct a part of it that is optimized for their device. The remainder of the program is now compiled at runtime, whereas that portion has been precompiled. This, therefore makes it slightly faster and more effective than in Dalvik.

However, this method requires a little bit more storage space. The dex2oat tool is used to optimize and compile .dex files into .oat files that may include ELF-formatted machine code. ART uses the on-device dex2oat tool to assemble programmes. This tool creates a built app executable for the target device from DEX files as input. Android automatically optimizes app data and generates an OAT file when an app is installed. The Android operating system generates an OAT file to expedite the loading of an Android app (.apk file). Android uses this file to speed up program loading and improve user experience. [5]

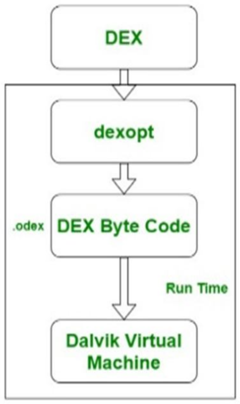


Fig. 3 Dalvik working

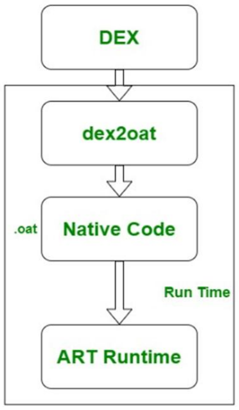
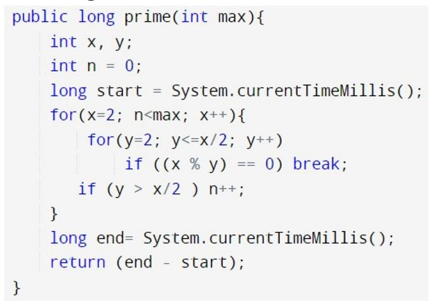


Fig. 4 ART working

3) *Performance:* Let us test the performance of both runtimes with the help of the prime number calculation algorithm.[6] In order to demonstrate the potential for performance comparisons between runtimes on the fundamental operations of integers, the prime number calculation testing is intended to develop an algorithm that conducts calculations with numbers. Numbers are calculated starting at zero and ending when the highest number has been reached.



```
public long prime(int max){
    int x, y;
    int n = 0;
    long start = System.currentTimeMillis();
    for(x=2; n<max; x++){
        for(y=2; y<=x/2; y++)
            if ((x % y) == 0) break;
        if (y > x/2 ) n++;
    }
    long end= System.currentTimeMillis();
    return (end - start);
}
```

Fig. 5 Prime number calculation Algorithm

Figure 5 indicates the code of calculation of the prime number, using the method of Eratosthenes. Although this method can quickly provide us with a list of prime numbers, it is not exactly the same as Eratosthenes' method for determining whether there is a prime number. When a non-prime number is deleted, the code will start with 2 and delete every item on the second list of divisible numbers and will evaluate the rest of the list as shown below.

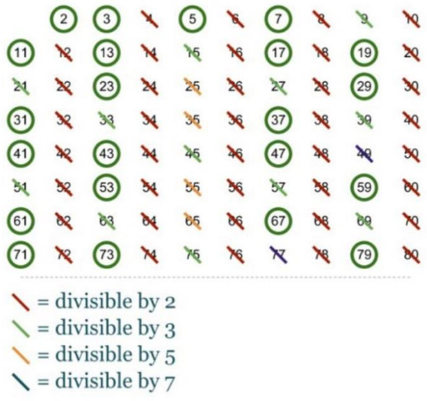


Fig. 6 Eratosthenes Method

A better compilation time has been observed by the above method in Dalvik on versions 4.2 and 4.4 compared to ART version 4.4 (KitKat) and 6.0 (Marshmallow) which clearly delineates that in versions 4.2 and 4.4, Dalvik compiles Java code as good as Android runtime. However, there are some exemptions in which the compilations were not as expected in both runtimes.[7] Hence, it can be summed up that, the Android Jelly Bean version runs relatively slower, and version 4.4 which runs on Dalvik has a slower time compared to Android runtime 4.4. The research conducted on the KitKat version later proves that it performed well on the prime number Java algorithm and its time almost matches the other ART runtime.

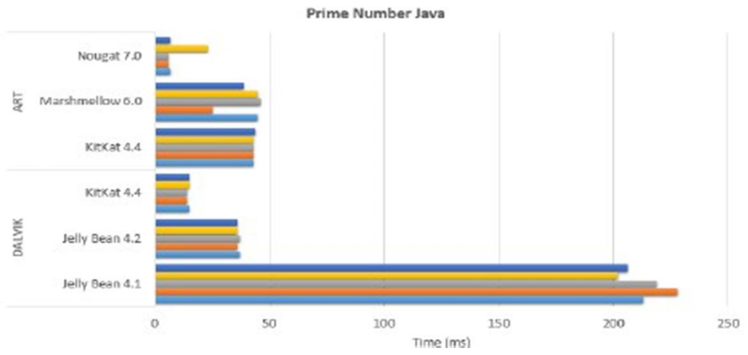


Fig. 7 Prime number calculation result

IV. SIGNIFICANT DIFFERENCES

DALVIK VIRTUAL MACHINE	ANDROID RUNTIME
It has a faster booting time	The rebooting process is notably longer
Cache builds up over time	In the initial boot, the cache is built up
Takes less memory because of JIT	Occupies a lot of memory storage implicitly due to AOT
Best for devices with small ROM	Best for devices with large ROM
App installation time is lower due to later compilation	App installation time is longer due to the compilation

V. CONCLUSION

Depending on the device, the user's own device, and preferences, ART is ultimately preferable, however with time, the apps get larger and take up more storage space, unlike Flappy Bird, which was only 1MB and took up less space. Therefore, if ART is to rule the market in the coming years, subsequent devices will need to increase storage. DVM has been replaced with ART in Android Lollipop. due to the fact that DVM converts bytecode each time you run a particular app. However, ART only translates it once while the program is being installed. This facilitates CPU execution. DVM/ART runs on top of ARM (architecture), just like MIPS, x86, etc. They cannot be replaced by one another. With enhancements, ART will have a few negligible drawbacks and lessen lag on Android devices. According to some benchmark research, the runtime will prolong battery life. JNI codes are supported by ART, which will help to better optimize the applications. More Google technologies, such as Google wearables, NFC, IR Blaster, and Bluetooth Additions, are made possible by ART. This is made better by the dramatically faster execution times seen on Android devices. We need to conduct some studies in the near future to compare and benchmark the performance of Android applications running on ART and JNI Android applications running on ART due to the ability to execute JNI applications. It seems obvious that Google has picked Ahead-of-Time (AOT) as the future compilation procedure. The trends suggest that ART may wind up becoming the default runtime for Android latest versions.

REFERENCES

[1] (Android introduction) https://www.tutorialspoint.com/android/android_overview.htm, Retrieved on August 2022
[2] Tom Leek, Stack exchange, https://security.stackexchange.com/questions/9011/does-a-virtual-machine-stop-malware-from-doing-harm, Retrieved on August 2022
[3] Laban Ndwaru, Introduction to Android ART, The next generation of Android Runtime, Thesis paper, June 2014
[4] Android source, Implementing ART Just-in-time (JIT) Compiler, https://source.android.com/devices/tech/dalvik/jit-compiler, Retrieved on August 2022
[5] Jelon, ART and Dalvik significant differences, GeeksforGeeks article, April 2021
[6] Radhakrishnan Yadav, Robin Singh Bhadoria, Performance analysis for Android Runtime Environment, published: 1 October 2015
[7] Alfan Presekal, Ruki Harwahyu, Riri Fitri Sari, Performance Comparison of Dalvik and ART on different Android based mobile devices, Conference paper, February 2019