Ankit Sinhal Follow
Payment SDK Developer at Mastercard. Dreamer and Achiever..
Apr 6 · 5 min read

Closer Look At Android Runtime: DVM vs ART

Before directly move to Android Runtime, we need to understand what runtime environment is and also understand some basic stuff i.e. the functioning of JVM and Dalvik VM.

What is runtime?

In a simplest term it is a system used by operating system which takes care of converting the code that you write in a high level language

like Java to machine code and understand by CPU/Processor.

Runtime comprises of software instructions that execute when your program is running, even if they're not essentially are a part of the code of that piece of software in particular.

CPUs or more general term our computers understand only machine language (binary codes) so to make it run on CPU, the code must be converted to machine code, which is done by translator.

So following are the generation of translator in a sequence-

1. Assemblers :

It directly translate assembly codes to machine codes so it was very fast.

2. Compilers :

It translates the code into assembly codes and then use assemblers to translate the code into binary. Using this compilation was slow but execution was fast. But the biggest problem with compiler is that the resulted machine code was platform dependent. In other words the code which runs on one machine may not run on different machine.

3. Interpreters :

It translates the code while executing it. Since the translation happens at runtime, the execution was slow.

How JAVA code execution works?

To maintain the platform independency of the code, JAVA developed JVM i.e. Java Virtual Machine. It developed JVM specific to every platform means JVM is dependency on the platform. The Java compiler converts the .java files into .class files, which is called byte code. This byte code is given to the JVM which converts it into machine code.

This is faster than interpretation but slower than C++ compilation.

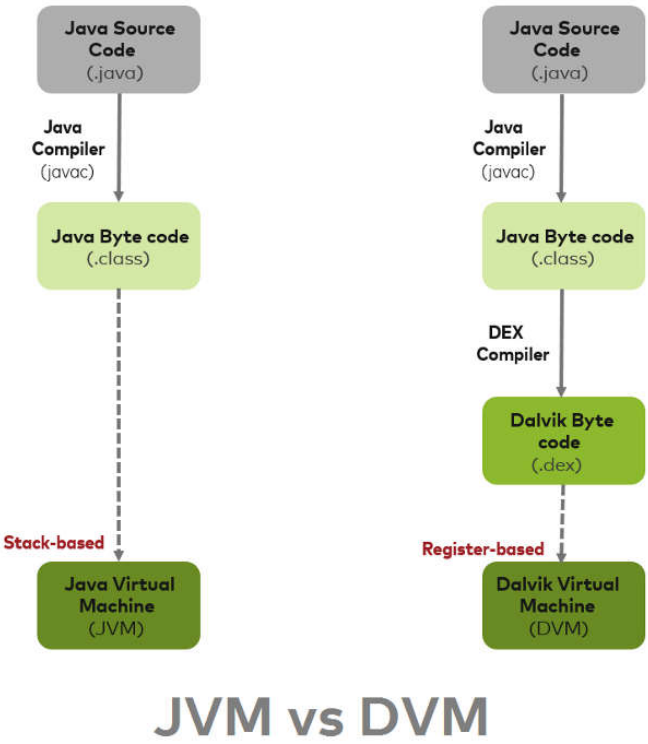
How Android code execution works?

In Android Java classes converted into DEX bytecode. The DEX bytecode format is translated to native machine code via either ART or the Dalvik runtimes. Here DEX bytecode is independent of device architecture.

Dalvik is a JIT (Just in time) compilation based engine. There were drawbacks to use Dalvik hence from Android 4.4 (kitkat) ART was introduced as a runtime and from Android 5.0 (Lollipop) it has completely replaced Dalvik. Android 7.0 adds a just-in-time (JIT) compiler with code profiling to Android runtime (ART) that constantly improves the performance of Android apps as they run.

Key Point: Dalvik used JIT (Just in time) compilation whereas ART uses AOT (Ahead of time) compilation.

Below are the code snippet explaining the difference between Dalvik Virtual Machine and Java Virtual Machine.



JVM vs DVM

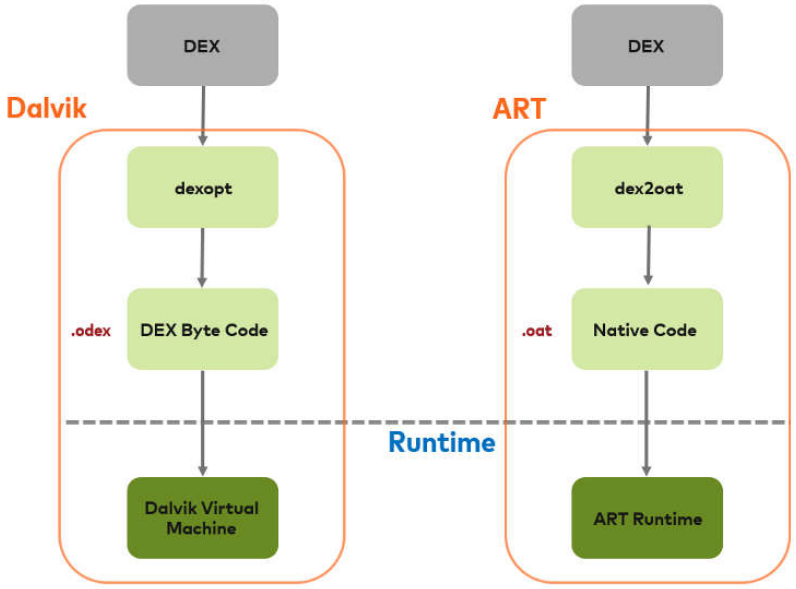
Just In Time (JIT)

With the Dalvik JIT compiler, each time when the app is run, it dynamically translates a part of the Dalvik bytecode into machine code. As the execution progresses, more bytecode is compiled and cached. Since JIT compiles only a part of the code, it has a smaller memory footprint and uses less physical space on the device.

Ahead Of Time (AOT)

ART is equipped with an Ahead-of-Time compiler. During the app's installation phase, it statically translates the DEX bytecode into machine code and stores in the device's storage. This is a one-time event which happens when the app is installed on the device. With no need for JIT compilation, the code executes much faster.

As ART runs app machine code directly (native execution), it doesn't hit the CPU as hard as just-in-time code compiling on Dalvik. Because of less CPU usage results in less battery drain.



Dalvik vs ART

ART also uses same DEX bytecode as input for Dalvik. An application compiled using ART requires additional time for compilation when an application is installed and take up slightly larger amounts of space to store the compiled code.

Why Android use Virtual Machine?

Android makes use of a virtual machine as its runtime environment in order to run the APK files that constitute an Android application. Below are the advantages:

· The application code is isolated from the core OS. So even if any code contains some malicious code won't directly affect the system files. It makes the Android OS more stable and reliable.

· It provides cross compatibility or platform independency. It meaning even if an app is compiled on platform such as a PC, it can still be executed on the mobile platform using the virtual machine.

Benefits of ART

· Apps run faster as DEX bytecode translation done during installation.

· Reduces startup time of applications as native code is directly executed.

· Improves battery performance as power utilized to interpreted byte codes line by line is saved.

· Improved garbage collector.

· Improved developer tool.

Drawbacks of ART

· App Installation takes more time because of DEX bytecodes conversion into machine code during installation.

· As the native machine code generated on installation is stored in internal storage, more internal storage is required.

Conclusion

ART is written to run multiple virtual machines on low-memory devices by executing DEX files, a bytecode format designed specially for Android that's optimized for minimal memory footprint. It makes the UI feel more responsive. That's all from my side. For more details on ART and Dalvik you can go through official Android document.

Thanks for reading. To help others please click ❤ to recommend this article if you found it helpful.

Check out my blogger page for more interesting topics on Software development.

You can also follow me at Twitter GitHub