Dalvik Virtual Machine 분석

조영필, 권용인, 양승준, 백윤흥
서울대학교공과대학전기컴퓨터공학부
e-mail : ypcho, yikwon, sjyang@sor.snu.ac.kr, ypaek@snu.ac.kr

Analysis of Dalvik Virtual Machine

Yeongpil Cho, Yongin Kwon, Seungjun Yang, Yunheung Paek
Dept. of Electrical Engineering, Seoul National University

요약

현재 스마트폰에서 가장 널리 쓰이는 OS 는 안드로이드이다. 안드로이드는 오픈 소스 플랫폼이기 때문에 이를 수정하여 기존 기술을 개선하거나 새로운 기술을 개발하려는 시도가 많이 이루어지고 있다. 이를 위해 필수적으로 시행해야 할 것이 Dalvik Virtual Machine 을 분석하는 작업이다. 이에, 본 연구는 Dalvik Virtual Machine 의 주요 요소를 분석하였다.

1. 서론

최근 IT 산업의 무게중심은 스마트폰에 놓여있다. 사실 과거에도 윈도우 모바일이나 심비안을 OS 채용한 스마트폰이 존재했으며, 그 이전에는 윈도우 CE 나 리눅스를 기반으로 하는 PDA (Persnal Digital Assistant)가 존재하였다. 하지만 이들의 시장 파급력은 현재의 스마트폰에 비하지 못했다.

2007 년 아이폰의 출시는 스마트폰 시장에 새로운 패러다임을 제시하였다. 첫 번째는 UX (User Experience)를 바탕으로 하는 사용성 강화이며, 두 번째는 스마트폰 OS 와 앱을 바탕으로 하는 ECO 시스템이다. 이러한 패러다임은 시장에 성공적으로 정착하게 되었으며, 수 많은 기업들이 패권을 노리고 시장에 진출하게 되었다.

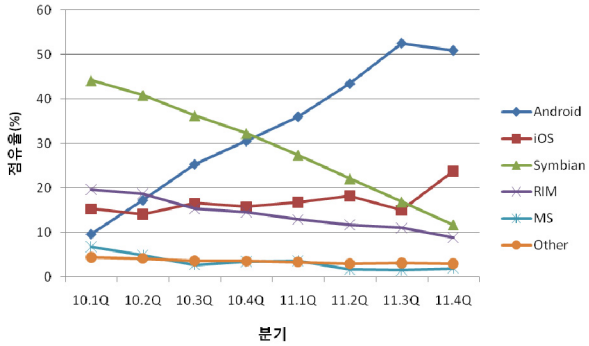

그림1 스마트폰 OS 별 시장 점유율[1]

그림 1 에 2010 년 부터 2011 년 까지 주요 스마트폰 OS 의 시장점유율이 나타나있는데, 이를 보면 안드로이드의 시장 점유율이 꾸준히 높아지는 것을 확인할 수 있다. 이러한 것에는 많은 이유가 있겠지만, 가장 큰 원인으로는 안드로이드가 오픈 소스 OS 라는 점을 들 수 있다.

이처럼 안드로이드의 시장 점유율이 높아짐에 따라 안드로이드를 기반으로 하는 수많은 응용과 기법이 개발되고 있으며, 이러한 개발을 위해 안드로이드의 내부 구조, 특히 Dalvik VM (Virtual Machine) 의 구조를 파악해야 할 필요성이 있다. 따라서 본 연구는 Dalvik VM 의 구조를 각 요소 별로 살펴보기 위해, 먼저 2 장에서 Dalvik VM 이 무엇인지 살펴보고, 3 장에서 Dalvik VM 의 주요 요소를 분석한 뒤, 4 장에서 마무리를 할 것이다.

2. Dalvik VM

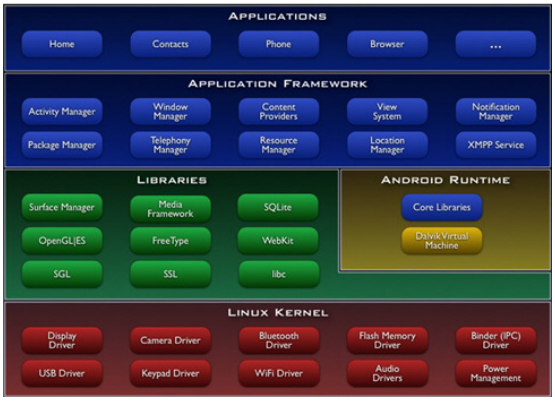

그림 2 안드로이드 프레임워크[2]

그림 2 에 나타나 있는 것처럼, DVM (Dalvik VM)은 안드로이드 프레임워크의 런타임 환경을 담당하고 있다. DVM 은 JVM (Java Virtual Machine)과 유사하지만 몇몇 차이점을 가지고 있다.

DVM	JVMs
DEX code	Java Bytecode
Register based	Stack Based
Concurrent GC	Various GCs

표2 DVM 과 JVM 비교

3. Dalvik VM 의 주요 요소 분석

(1) 메모리 할당

DVM 은 메모리 관리를 위해 내부적으로 mspace 를 사용하고 있으며, 이들 mspace 중 GC (Garbage Collection)의 대상이 되는 mspace 를 GcHeap 이라는 자료구조로 관리하고 있다. DVM 에서 GcHeap 에 메모리를 할당할 때 사용하는 함수는 dvmMalloc(size, flag) 이며, 이는 객체나 배열의 할당 등에 널리 사용된다. 메모리가 부족하여 할당에 실패할 경우 dvmMalloc 은 우선적으로 GC 를 호출하여 soft/weak/phantom 레퍼런스를 제거한다. 그 후에도 할당에 실패할 경우 GC 를 호출하여 finalizable 레퍼런스를 제거하게 되며, 계속해서 할당에 실패할 경우 GcHeap 을 확장하게 된다.

(2) 스레드 할당

안드로이드에서 사용되는 모든 스레드는 리눅스에 직접 관리되는 스레드이며, 스레드는 앱에서 Java API 를 통해 할당하는 방식과 DVM 내부에서 할당하는 방식이 있다. DVM 내부에서 스레드를 할당할 경우에는 dvmCreateInternalThread(pthread_t, name, func, func_args)를 통해 할당하게 된다. 할당된 스레드는 DVM 내부 전역 자료구조인 DvmGlobals 에서 threadList 변수를 통해 관리된다. 이를 통해 GC 를 수행할 때, 특정 스레드를 정지하는 등의 작업을 할 수 있다.

(3) 클래스, 메소드 resolve

DVM 에서 객체를 생성하기 위해서는 해당 객체의 클래스가 resolve 되어 있어야 한다. 마찬가지로, DVM 에서 특정 클래스의 메소드를 호출하기 위해서는 해당 메소드가 resolve 되어 있어야 한다. DVM 은 resolve 를 통해 해당 클래스 및 메소드를 메모리에 로딩하게 된다. resolve 에 사용되는 함수는 dvmResolveClass(referrer , classIdx, fromUnverifiedConstant) 와 dvmResolveMethod(referrer, methodIdx, methodType)이 있다. 두 함수에 공통적으로 사용되는 referrer 은 해당 클래스 및 메소드를 호출하는 클래스를 의미하며, resolve 함수는 referrer 로부터 DEX 파일의 정보 및 클래스로더 정보를 획득하게 된다. 그리고 classIdx, methodIdx 는 DEX 파일 내, 해당 클래스 혹은 메소드 정보가 저장된 인덱스를 의미한다.

(4) 메소드 호출

메소드가 resolve 되면 호출을 할 수 있다. 안드로이드에서 메소드를 호출하는 방법은 다음과 같이 구분할 수 있다.

(a) 인터프리트 메소드에서 인터프리트 메소드 호출
(b) 인터프리트 메소드에서 네이티브 메소드 호출
(c) 네이티브 메소드에서 인터프리트 메소드 호출

이처럼 차이가 발생하는 이유는 각 메소드의 스택 프레임이 위치하는 곳이 다르기 때문이다. 인터프리트 메소드의 스택 프레임은 DVM 내부의 가상 스택에 생성되며 네이티브 메소드의 스택 프레임은 일반적인 어셈블러 프로시저와 마찬가지로 SP (Stack Point)를 바탕으로 생성된다. 이 때문에, (a)의 경우는 DVM 인터프리터 내부에 가상 스택 위에 스택 프레임을 쌓는 코드가 존재한다. (b)의 경우에는 dvmInvokeMethod(obj, method, argList, params, returnType, noAccessCheck)함수를 사용한다. 이때 obj 는 메소드를 호출하는 객체를 의미하며 정적 메소드를 호출할 경우에 obj 는 NULL 이 된다. 마지막으로 (c)의 경우에는 JNI 인터페이스를 사용해 메소드를 호출하게 된다.

4. 마무리

DVM 의 주요 요소를 간략히 살펴보았다. DVM 은 여러 부분에서 JVM 과 유사한 점이 많다. 이는 DVM 이 ASF (Apache Software Foundation)의 오픈 자바 프로젝트인 하모니[3]를 기반으로 하여 개발되었기 때문이며, 이러한 까닭에 DVM 의 모든 소스코드는 아파치 라이선스 규정을 바탕으로 공개되어 있다. 그럼에도 불구하고 모바일 환경이라는 특성상 DVM 은 Zygote 의 사용이나 DEX 파일의 사용 등 많은 부분에서 독창성을 보이고 있다.

Acknowledgement

본연구는교육과학기술부/한국과학재단우수연구센터육성사업(과제번호 2012-0000470), 2011 년도정부(교육과학기술부)의재원으로한국과학재단의국가지정연구실사업(No.2011-0018609)및 IDEC 의지원을받아수행되었습니다.

참고문헌

[1] Gartner, Inc
[2] http://developer.android.com/guide/basics/what-is-android.html
[3] http://harmony.apache.org/