안드로이드 달빅 가상 머신을 위한 Class-to-Dex 런타임 변환 방법

정동헌○ 문수묵
서울대학교
clamp@altair.snu.ac.kr, smoon@altair.snu.ac.kr

Class-to-Dex Runtime Translation for Android Dalvik Virtual Machine

Dong-Heon Jung○ Soo-Mook Moon
Seoul National University

안드로이드는 구글의 기술력을 활용할 수 있고, 또한 비용이 거의 들지 않으며, 자바 언어를 이용하여 어플리케이션을 작성할 수 있다는 점에서 많은 인기를 얻고 있다. 안드로이드는 자바의 클래스 파일이 아닌 안드로이드의 고유 포맷인 덱스 포맷으로 변환한 후에 안드로이드 수행엔진인 달빅 가상 머신을 통해 수행한다. 달빅 가상 머신은 자바 어플리케이션을 수행할 수 있다는 점에서 자바 가상 머신을 대체할 수 있는 좋은 대안이 되고 있다. 하지만 안드로이드는 수행 시간 이전에 자바 언어로 구현된 어플리케이션을 미리 덱스 포맷으로 변환한 후에 달빅 가상 머신을 통해 수행되므로 수행 시간에 전송되는 자바 클래스 파일을 달빅 가상 머신에서 수행하기 어렵다는 문제가 있다. 우리는 수행 시간에 전송되는 자바 클래스 파일을 즉시 덱스 포맷으로 변환하는 Class-to-Dex 런타임 변환기를 구현하였다.

1. 서론

최근 휴대전화에서 인터넷 사용 및 다양한 사용자 어플리케이션을 설치할 수 있는 스마트폰이 많은 인기를 얻고 있다. 특히 구글에서 출시하여 오픈소스로 배포되는 안드로이드[1]는 국내외의 많은 내장형 기기 생산 업체들의 소프트웨어 플랫폼으로 각광을 받고 있으며, 북미에서는 아이폰을 제치고 점유율 1위를 차지하는 등 폭발적인 성장을 하고 있다. 그 성장의 이유 중의 하나는 자바[2] 언어를 이용하여 어플리케이션을 개발하기 때문에, 자바 개발자들이 쉽게 안드로이드용 어플리케이션을 작성할 수 있기 때문이다.

기존의 자바 어플리케이션의 수행 방법은 자바 언어로 작성된 어플리케이션을 자바 클래스 파일로 변환한 후에, 자바 가상 머신 환경에서 수행한다. 하지만 안드로이드는 자바 클래스 파일을 수행 시간 이전에 미리 안드로이드의 고유 포맷인 덱스 포맷[3,4]으로 변환한다. 변환된 덱스 포맷을 안드로이드에서 제안한 새로운 수행 엔진인 달빅 가상 머신에서 수행한다. 달빅 가상 머신은 기존의 자바 가상 머신을 대체할 수 있는 대안으로 사용될 수 있다.

2. 본론

기존의 자바 가상 머신을 달빅 가상 머신으로 대체하기 위하여 Class-to-Dex 런타임 변환기가 필요하다. 디지털 TV, 휴대폰, 블루레이 플레이어등 대부분의 내장형 기기에선 런타임에 자바 클래스 파일이 전송되며, 전송된 파일을 즉시 덱스 포맷으로 변환해 줄 수 있는 Class-to-Dex 런타임 변환기가 필수적이다. 우리는 그림 1에서와 같이 Class-to-Dex 런타임 변환기를 추가하여 기존 자바 가상 머신의 수행 엔진을 안드로이드의 달빅 가상 머신으로 교체할 수 있다.

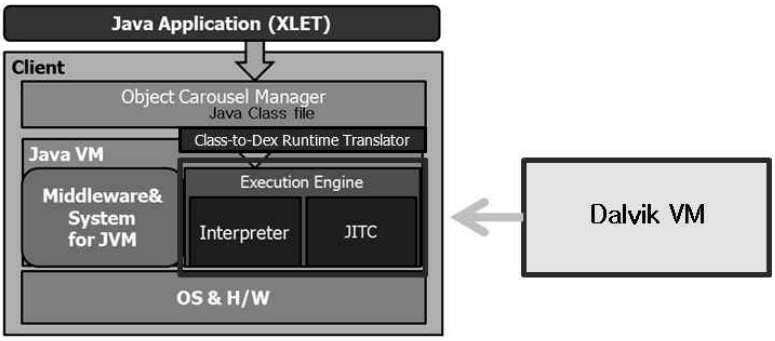

그림 1. Class-to-Dex 런타임 변환기

본 연구는 LG전자의 지원과 지식경제부 및 한국산업기술평가관리원의 산업원천기술개발사업의 일환으로 수행하였음. [KI002119, 고성능 가상머신 규격 및 기술 개발]

여러 개의 자바 클래스 파일이 하나의 덱스 파일로 변환될 수 있지만, 우리는 런타임에 전송되는 클래스파일을 필요할 때 마다 변환할 수 있도록 하나의 클래스 파일을 하나의 덱스 포맷으로 변환하는 방법을 선택하였다.

먼저 기존의 덱스 로더에서 확장자를 확인하여 확장자가 class인 경우에 우리의 변환기를 수행하도록 덱스 로더를 수정한다. 그리고 우리의 변환기는 class파일을 읽어 덱스 자료 구조로 변환하여 덱스 로더에 다시 넘겨준다.

우리의 변환기에서는 자바 클래스 파일의 컨스턴트 풀 정보를 이용하여 달빅 ID 리스트와 문자열, 클래스, 필드, 메소드에 대한 정보를 생성한다.

그리고 메소드의 각각의 바이트코드들을 변환한다. 자바 클래스 파일은 모든 값을 스택에 옮긴 후에 수행하는 스택에 기반한 수행을 하는 반면, 덱스 포맷은 레지스터를 명시적으로 표시하는 레지스터 기반의 수행방법을 채택하고 있다. 그러므로 덱스 포맷에서 사용할 레지스터를 결정한 후에, 각각의 자바 바이트코드를 달빅 바이트코드로 변환한다. 우리는 빠른 변환을 위하여 자바 바이트코드의 load와 store를 모두 달빅의 move 바이트코드로 변환하고, 간단한 copy propagation 최적화를 적용하였다. 이외의 다른 자바 바이트코드들도 덱스의 바이트코 코드로 변환한다.

그리고 이외의 자바에서 지원하는 모든 특징들을 지원하기 위하여 예외처리 리스트 및 접근 플래그 필드를 설정하고, 자바 클래스에는 없는 헤더와 매핑 테이블을 생성하여 완전한 형태의 덱스 포맷을 생성한다.

실험은 Linux 2.6.18의 운영체제가 설치된 400 MHz의 MIPS 기기에서 진행하였으며, 실험에 사용한 안드로이드는 MIPS 버전 Eclair Android이다. 그리고 성능 비교를 위하여 Phone Me Advanced MR2 자바 가상 머신[5]의 인터프리터의 성능도 함께 측정하였다. 벤치마크는 임베디드 기기에서 성능 측정에 많이 사용되는 caffeinemark를 실험하였다. 아래 실험 결과에서 알 수 있듯이 String 라이브러리의 차이로 인해 성능이 매우 나빠진 String 벤치마크을 제외한 나머지 벤치마크에서 우리의 변환기를 포함한 달빅 가상 머신이 자바 가상 머신의 인터프리터와 거의 유사한 성능을 보이는 것을 확인할 수 있다.

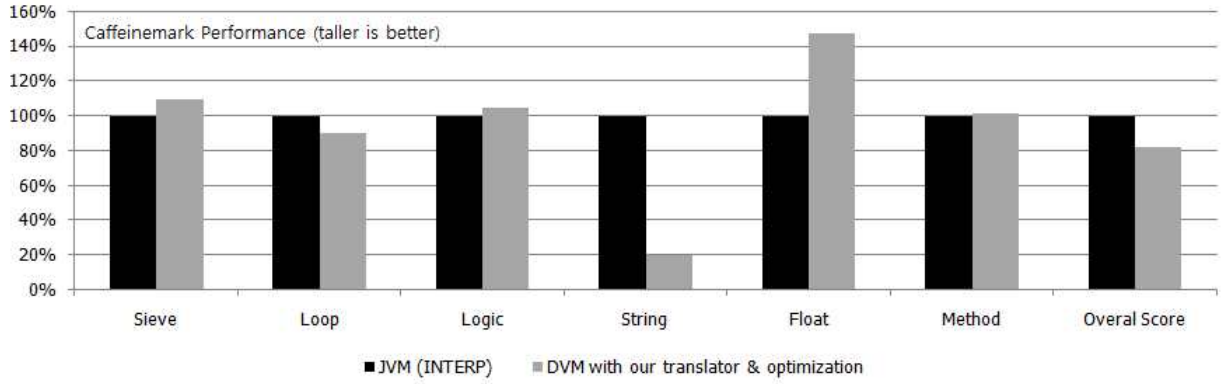

그림 2. 벤치마크 성능 결과

3. 결론

우리는 수행 시간에 전송되는 자바 클래스파일을 안드로이드의 달빅 가상머신에서 수행하기 위하여 class-to-dex 런타임 변환기를 구현하였다. 자바 가상 머신의 해석기에 비해서 성능이 크게 떨어지지 않으며 벤치마크와 테스트 어플리케이션을 수행할 수 있는 안정성을 가지고 있어, 달빅 가상 머신을 사용하는 다양한 내장형 기기에서 활용할 수 있을 것이다.

Reference

[1] http://www.android.com/

[2] J. Gosling, B. Joy, and G. Steele, The Java Language Specification Reading, Addison-Wesley, 1996.

[3] Bytecode for the Dalvik VM,
http://android-dalvik-vm-on-java.googlecode.com/svn-history/r11/trunk/dalvikvm/doc/google/dalvik-bytecode.html

[4] .dex Dalvik Executable Format,
http://android-dalvik-vm-on-java.googlecode.com/svn-history/r11/trunk/dalvikvm/doc/google/dex-format.html

[5] Sun Microsystems, White Paper " CDC: An Application Framework for Personal Mobile Devices"