내장형 자바가상기계를 위한 클래스 파일 변환기의 설계 및 구현

지정훈°, 양희재

경성대학교 컴퓨터공학과

jhji@conet.ks.ac.kr, hjyang@star.ks.ac.kr

Design and Implementation of a Class File Converter for Embedded Java Virtual Machine

Junghoon Ji and Heejae Yang

Dept, of Computer Engineering, Kyungsung University

요 약

자바는 서로 다른 종류의 컴퓨터 시스템에서 동일하게 동작하는 플랫폼 독립적인 특성을 가지고 있다. 자바가상기계(JVM)는 클래스 파일을 읽어 들여 인터프리팅하여 실행한다. 보조기억장치가 없는 내장형 시스템에서는 메모리에 클래스 파일이 위치하는데 클래스 파일에는 디버깅등의 목적으로 사용하는 정보와 클래스, 상수, 필드, 메소드 등의 정보들을 포함하고 있기 때문에 내장형 시스템에서 사용하기에 적합하지 않다.

본 논문에서는 클래스 파일을 변환하여 내장형 시스템에서 시스템의 효율적인 자원 사용과 성능을 향상시킬 수 있도록 클래스 파일을 변환 해주는 도구인 cls2bin을 설계 및 구현하였다. cls2bin은 클래스 파일에서 동작에 필요하지 않은 정보들을 제거하고 인터프리팅 될 수 있는 새로운 이미지(bin) 파일을 생성한다. cls2bin의 동작과정과 bin 파일 포맷을 살펴봄으로서 내장형 시스템에서 효율적인 자원사용과 내부정보의 접근 방법을 고찰하였으며 그 결과로 내장형 자바 시스템에서의 개선된 클래스 파일의 형태를 정의하고자 한다.

1 서 론

한번 작성된 자바 프로그램은 특정한 하드웨어나 운영체제에 상관없이 어디서든 동일한 동작을 하는 플랫폼 독립성과 높은 이식성을 가지고 있다. 이것은 다양한 하드웨어와 운영체제를 사용하는 내장형 시스템에서 효율적인 성능을 발휘할 수 있어 최근 자바를 이용한 내장형 시스템에 대한 많은 관심을 받고 있다.

자바 프로그램은 자바 언어로 작성된 코드를 중간 단계의 기계어 코드인 바이트 코드로 컴파일 하여 자바가상기계(Java virtual machine)라는 실행환경에서 실행된다. 내장형 시스템을 위한 JVM으로는 클래스 간 동적 연결 방식을 지원하는 썬 마이크로시스템사의 J2ME의 KVM[1]이 있으며, 클래스 파일 변환을 통한 정적 연결을 지원하는 simpleRTJ[2]등이 있다. 이외에도 여러 회사들이 많은 제품들을 출시하고 있다.

본 논문은 보조기억장치 없이 메모리나 ROM을 이용하여 동작을 수행하는 내장형 시스템에 알맞게 클래스 파일을 변환하여 효율적인 자원관리와 실행속도 향상시키는데 있다.

모든 자바 프로그램은 자바 원천코드를 자바 컴파일러에 의해 클래스 파일로 변형되며, JVM은 클래스 파일을 읽어 들여 실행하게 된다. 내장형 시스템은 자원이 한정되어 있으며 클래스 파일에는 여러 가지 링크 정보 및 디버깅 정보를 포함하고 있기 때문에 내장형 시스템에서 사용하기에는 크기다 크고 클래스 파일내의 상수나 필드, 메소드 등에 접근하는데 비효율적이다.

내장형 시스템에서는 클래스 파일을 새로운 형태로 변형하여 한정된 자원 사용과 필드 및 메소드 접근 등의 효율성을 높여야 한다. 본 논문에서는 자바 컴파일러로 컴파일 되어 생성된 클래스 파일을 새로운 이미지 파일 포맷인 bin 파일 포맷으로 변형하여 클래스 정보를 유지하면서 효율적인 자원사용과 성능향상을 위한 클래스 파일 변환기인 cls2bin을 설계, 구현하였다.

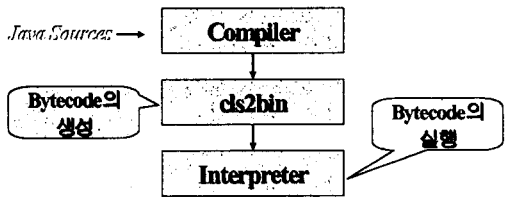

그림 1 cls2bin 흐름도

cls2bin은 호스트 컴퓨터상에 있으며, 호스트 컴퓨터에서 개발된 자바 클래스 파일을 이용하여 정적인 안전성 검사 및 클래스 연결 작업을 거쳐 하나의 이미지 파일인 bin 파일을 생성한다. bin파일은 ROM을 이용하여 내장형 컴퓨터에 다운로드 되어 JVM을 이용하여 실행한다.

본 논문의 구조는 다음과 같다. 2절에서는 cls2bin을 이용한 클래스 파일의 변형에 대해서 알아보고 3절에서는 bin파일의 구조에 대해서 알아본다. 4절에서는 cls2bin의 효과와 향후 연구방향 및 결론을 짓는다.

2 cls2bin의 클래스 파일 변환.

자바에서의 플랫폼 독립성과 높은 이식성을 가능하게 하는 것은 바이트 코드에 의하여 가능하다. 그러나 클래스 파일이 JVM에 의하여 읽혀져서 실행되기 때문에 실행 속도가 느리고 클래스 파일 내에는 여러 가지 링크 정보 및 디버깅 정보등 바이트 코드 실행에 있어서 없어도 될 정보들을 포함하고 있

어 크기가 크다. 내장형 시스템에서는 바이트 코드 실행과 관계없는 불필요한 정보를 제거하여 클래스 파일의 크기를 줄여서 사용하여야 한다. cls2bin은 호스트 컴퓨터에서 작성한 클래스 파일을 변환하여 내장형 시스템에서 사용할 수 있는 이미지 파일인 bin파일을 생성시킨다. cls2bin은 동작은 다음과 같다. (그림 2)

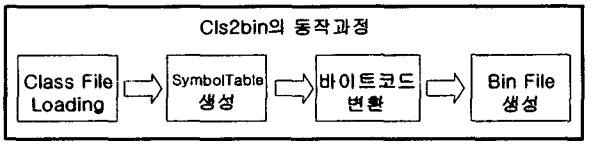

그림 2 cls2bin의 클래스 파일 변환작업.

cls2bin은 그림 2와 같은 과정을 거쳐 이미지 파일인 bin파일을 생성한다. 먼저 cls2bin은 클래스 파일을 읽어 심볼 테이블을 생성한다. 심볼 테이블은 클래스들간 링크를 위한 각종 심볼명들(클래스, 필드, 메소드)이 들어있다. 심볼 테이블 정보는 bin 파일 내에는 포함되지 않는다. 심볼 테이블 생성을 마치면 클래스의 바이트 코드를 검색하여 다른 클래스 참조나 클래스간 연결정보가 필요한 특정 바이트 코드(ldc, invokevirtual, get/put field등)들의 참조 인덱스 값들을 bin 파일 형식에 맞도록 변환한다. 이러한 변환작업이 필요한 이유는 JVM에서는 클래스들이 동적으로 연결되어 있지만, bin 파일 형식은 최적화와 빠른 정보 접근을 위하여 클래스 정보들을 정적으로 연결을 하기 때문에 변환작업을 수행하여 필요한 클래스들의 정의와 링크가 이루어져야 한다.

3 bin 파일의 구조 및 예제.

bin 파일에는 바이트 코드들의 동작에 꼭 필요한 정보들만 포함되어 있다. bin 파일의 정보들은 크게 클래스 정보, 상수 정보, 필드 정보, 메소드 정보 4가지로 나뉜다. 그림3은 bin 파일의 구조를 보여준다.

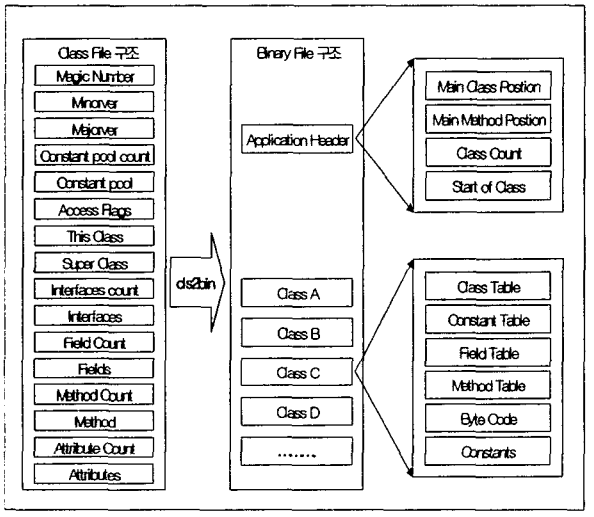

그림 3 bin 파일의 구조

3.1 헤더 및 클래스 테이블

bin 파일의 헤더 부분에는 bin파일 내에 포함되어 있는 클래스들의 개수와 각 클래스들의 시작 위치, 실행될 때 가장 먼저 호출되어질 메인 메소드에 대한 위치 정보가 들어있다. 클래스 정보는 new 바이트 코드에 의해 인스턴스가 생성될 때 참조된다. new #i에 의해 클래스 테이블이 참조되어 질 때 bin 파일의 클래스 테이블에서 i번째 테이블 정보를 참조하면 된다.

클래스 테이블은 다음과 같이 정의된다.

```
typedef struct _class {
    u2    flags;        // access flag
    u2    id;           // class id
    u2    iface;        // interface
    u2    super;        // super class
    u2    mtd_cnt;      // method count
    u2    vrt_fld;      // virtual field
    u2    stat_fld;     // static field
    u2    reserved;     // reserved
    u1    *field_tbl;   // field table
    u1    *meth_tbl;    // method table
} jvm_class;
```

클래스 테이블의 클래스 id와 수퍼클래스 id는 bin 파일의 클래스 인덱스이며, 이 인덱스는 바이트 코드 변환 작업에 의해서 결정되어 진다. 클래스 정보에 대한 접근 또한 구조가 정적으로 이루어져 있어 직접 접근이 가능하다. virtual field/static field는 인스턴스 변수에 대한 정보를 나타내고, field_tbl/meth_tbl은 클래스의 필드/메소드 테이블의 시작위치를 나타낸다.

3.2 상수 테이블.

상수 테이블은 ldc 바이트 코드에 의하여 상수 값이 스택에 올려질 때 참조된다. 형식은 ldc #i와 같다. i 값은 테이블의 오프셋을 나타내는 인덱스 값이다.

상수 테이블의 구조는 다음과 같이 정의된다..

```
typedef struct {
    i4    loc;    // location of the constant
    i4    len;    // length of the constant
} jvm_const;
```

접근하려는 상수가 UTF8 문자열이 아니라 Integer/Float라면 len에는 0이 들어가고 실제값은 loc에 형식 변환되어 저장된다. Long/Double은 len에는 -1이 들어가고 실제값은 메모리 상의 어느곳에 위치한다. loc에 그 위치가 들어간다. cls2bin은 이 구조체는 상수의 자료형에 의해 구분하는 것이 아니라 단지 4바이트인지 8바이트 인지의 크기만 구별한다.

3.3 필드 테이블.

필드 테이블은 필드 접근 바이트 코드인 getfield/putfield에 의하여 접근 된다.

필드 테이블의 구조는 다음과 같이 정의된다.

```
typedef struct {
    u2    len_id;  // field id
} jvm_field;
```

len_id의 15번째 비트는 이 필드의 크기를 (32bit / 64bit) 나타내며 나머지 비트들은 필드의 고유번호를 나타낸다. 필드는 최대 32K개까지 올 수 있다. 필드의 고유번호는 0부터 시작된다. 필드 테이블 엔트리에는 static인지 virtual인지 구분되지 않지만 코드 부분에서 getstatic/putstatic으로 호출되는 필드는 static 필드이고 그 외 getfield/putfield로 호출되는 것은 virtual 필드이다.

3.4 메소드 테이블

메소드 테이블은 클래스 내의 메소드를 호출하는 바이트 코드 명령어인 invokevirtual에 의하여 참조되며 구조는 다음과 같다. 형식은 invokevirtual #i이며, i는 클래스 파일내의 상수풀에 대한 인덱스 이지만 바이트 코드 변환 작업에 의해서 bin파일 내의 메소드 테이블 참조 인덱스로 변환되었다.

메소드 테이블은 다음과 같이 정의된다.

```
typedef struct {
    u2    stksz;    // stack size
    u2    localsz;  // local variable size
    u2    params;   // size of params (in 32bit count)
    u2    codesz;   // code size
    u1    *code;    // code position
} jvm_method;
```

메소드 테이블에는 현재 메소드 수행에 필요한 스택의 크기, 지역변수 풀의 크기와 메소드로 넘어오는 파라미터들이 들어있다. 위의 값들은 변환 전 클래스 파일의 항목들과 동일한 값을 가진다. 그리고 *code는 메소드의 실행 코드가 있는 bin 파일 내의 위치 값이 들어있다. 클래스를 상속과 같이 다른 클래스의 메소드 정보를 가질 경우에는 codesz는 0을 두고, stksz는 다른 클래스의 id값을, localsz는 그 클래스 내의 메소드 인덱스 값을 가리키도록 하였다.

3.5 bin 파일 생성 예제.

실제 자바 테스트 프로그램을 작성하여 cls2bin을 이용하여 bin 파일을 생성하여 보고 클래스 파일의 어떻게 변환되어 저장되는지 살펴보자.

테스트 프로그램은 4개의 클래스로 구성되어 있다.

> Object1.class, Parent.class, Child.class, Test.class

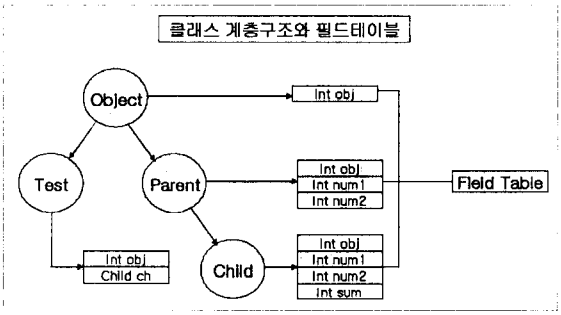

그림 4 테스트 클래스 계층구조와 필드 테이블.

위의 클래스들 사이의 연결 구조를 보면 최상위에 Object1이 있고, Parent와 메인 클래스인 Test에서 상속받았다. Child는 Parent 클래스를 상속받음으로서 Object1과 Parent 클래스 정보를 가지게 된다. cls2bin은 클래스들 사이의 정적인 연결을 지원하므로 bin 파일의 각각의 클래스에는 그림과 같이 부모 클래스의 필드 정보들도 함께 가진다.

그림 5를 보면 Test 클래스의 바이트 코드와 변경 후의 바이트 코드를 볼 수 있다.

run 메소드의 변환 전 바이트 코드를 보면 getfield, invokevirtual 등의 바이트 코드들은 클래스의 상수 풀들을 참조하였다. 그러나 bin 파일의 바이트 코드들을 보면 클래스 파일의 상수 풀의 심볼릭 정보들을 제거하였기 때문에 인덱스들이 정적인 형태로 바뀌었다.

run() 바이트코드	심볼 테이블
Method void run() 0 aload_0 1 getfield #5 <Field Child ch> 4 bipush 10 6 invokevirtual #6 <Method null> 9 aload_0 10 getfield #5 <Field Child ch> 13 bipush 20 15 invokevirtual #7 <Method null> 18 aload_0 19 getfield #5 <Field Child ch> 22 invokevirtual #8 <Method void sum_data()> 25 aload_0 26 getfield #5 <Field Child ch> 29 getfield #9 <Field int sum> 32 istore_1 33 return	This Class : <Test> Super Class : <Object1> Field Cnt : <1> Method Cnt : <3> Fields <0> : ch LChild;<0> <0> Methods <0>:<init> ()V <0> <0> Methods <1>:main ()V <0> <1> Methods <2>:run ()V <0> <2>
코드 변환 전 run()	코드 변환 후 run()
2a b4 00 05 10 0a b6 00 06 2a b4 05 10 14 b6 00 07 2a b4 00 05 b6 00 08 2a b4 00 05 b4 00 09 3c b1	2a b4 00 00 10 0a b6 00 00 2a b4 00 00 10 14 b6 00 07 2a b4 00 00 b6 00 00 2a b4 00 00 b4 00 00 3c b1

그림 5 바이트 코드 변환.

4 결론 및 향후 연구 과제.

본 논문은 내장형 시스템에서 클래스 파일을 변환하여 시스템의 효율적인 자원 사용과 성능을 향상시키는 것이다. cls2bin의 수행 결과 클래스 파일에서 동적연결을 위한 심볼릭 정보들과 바이트 코드의 실행에 사용되지 않는 정보들을 제거함으로서 클래스 파일과 비교하여 약42%의 절감효과를 볼 수 있었다.[6]

본 논문에서는 클래스 파일 변환기인 cls2bin의 설계와 구현을 통하여 내장형 자바가상기계를 위한 새로운 클래스 이미지 파일인 bin 파일을 생성하였다. bin 파일을 이용한 자바가상기계는 휴대폰, PDA 같은 모바일 기기와 셋톱박스 같은 내장형 기기에 탑재되어 유용하게 사용될 것이다.

향후 계획은 bin파일의 포맷을 개선하여 보다 효율적인 자원 사용과 빠른 내부정보 접근에 대하여 연구할 것이다.

참 고 문 헌

[1] Sun Microsystems, Java2 Platform Micro Edition (J2ME) Technology for reating Mobile Device, http://java.sun.com

[2] RTJ Computing, simpleRTJ: a small footfrint Java VM for embedded and consumer devices, http://www.rtjcom.com

[3] 양희재, "simpleRTJ 자바가상기계에서 클래스 파일의 메모리 상 배치", *한국정보과학회*, 제 29회 추계 학술대회, 2002. 10.

[4] T. Lindholm and F Yellin, *The Java Virtual Machine Specification Second Edition*, Addison-Wesley, 1999.

[5] J. Engel, *Programming for the Java Virtual Machine*, Addison-Wesley, 1999.

[6] 김성수, 김세영, 양희재 "내장형 자바가상기계를 위한 클래스 이미지 파일의 분석과 비교" *한국정보과학회*, 제30회 춘계학술 발표회 발표예정, 2003.