임베디드 자바 시스템을 위한 효율적인 Pre-resolution Method에 관한 연구

서정배°, 양윤심, 정민수
경남대학교 컴퓨터공학과

A Study on an Efficient Pre-resolution Method for Embedded Java System

CheongBae Seo, Yoon-Sim Yang, Min-Soo Jung
[1] Dept. of Computer Engineering, Kyungnam University

요 약

자바는 임베디드 시스템을 프로그램 하기에 매력적인 언어와 플랫폼으로 인식되어져 왔다. 그러나 임베디드 자바의 메모리와 프로세서 한계점을 가지고 임베디드 자바에 표준 자바 클래스 파일을 적용하기에는 적당하지 않다. 본 논문에서는 타겟디바이스에서 바이트 코드를 수행하기 전에 심볼레퍼런스 정보를 실제 주소로 바꾸기 위해 프리레졸루션을 사용하여 실행 시간을 줄일 수 있는 효율적인 메소드를 제안하였다. 클래스 파일에서 컨스턴트풀의 사이즈를 알기 위해서 13개의 클래스 파일들을 시험하였다. 본 프리레졸루션은 원래 사이즈의 92% 정도 전체적인 메모리 footprint를 줄였다. 또한, 메모리 참조 횟수도 감소시켰다.

1. 서론

자바는 임베디드 시스템을 프로그램 하기에 매력적인 언어와 플랫폼으로 인식 되어져 왔다. 그러나 메모리와 프로세서의 한계점을 가진 임베디드 자바에 표준자바 클래스 파일을 적용하는 것은 적당하지가 않다. 자바 소스코드는 클래스 파일 포맷으로 컴파일 된다. 표준 자바에서 클래스 파일은 완전히 분석되고 자바 메소드를 실행 하는데 필요한 모든 정보를 포함하고 있다. 클래스 파일의 주된 컨텐츠는 컨스턴트풀, 필드 선언, 메소드 선언 그리고 클래스와 메소드에 관한 일반적인 데이터라 불리는 심볼 테이블이다. 심볼레퍼런스는 컨스턴트풀에 수집되고 타겟 요소들을 보여주는 인덱스 중심의 간접참조들이다. 수행시 간접참조의 대상이 되는 컨스턴트풀의 심볼레퍼런스를 직접참조 정보로 변환하는 과정을 레졸루션이라고 한다.

컨스턴트풀의 레졸루션은 많은 실행 시간과 메모리 공간을 필요로 한다. 이런 이유들로 본 논문에서 프리레졸루션을 사용하여 실행 시간을 줄일 수 있는 효과적인 메소드를 제안하였다.

프리레졸루션은 타겟디바이스 상에서 바이트 코드를 실행하기 전에 심볼레퍼런스 정보를 직접참조가 아닌 실제주소로 변환한다. 프리레졸루션은 두 단계로 구성되어 있다. 첫 번째 단계는 스트링을 스트링이 해당하는 인덱스로 변환하는 Tokenization이고

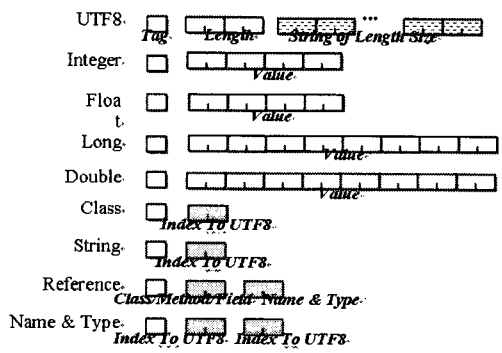
두 번째는 스트링의 인덱스를 메모리의 주소로 변환하는 Memorization이다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 심볼레퍼런스의 대상이 되는 컨스턴트풀에 대해 살펴보고, 3장에서는 Tokenization와 Memorization를 통하여 임베디드 JVM 변환 클래스파일 포맷에 있는 클래스 파일에 관한 프리레졸루션 메소드를 보여준다. 4장에서는 프리레졸루션 메소드의 평가를 살펴본다. 마지막으로 5장에서는 클래스 파일에 적용한 본 프리레졸루션 기술의 장점과 단점을 표준 자바의 클래스 파일과 비교 한다.

2. 관련연구

2.1 클래스파일의 컨스턴트풀

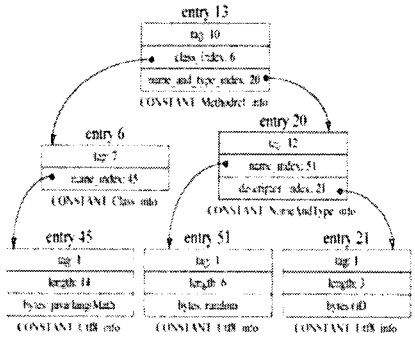
그림 1은 클래스 파일에서 컨스턴트풀의 포맷을 보여준다. 클래스 파일에서 인덱스 중심의 간접 참조는 컨스턴트풀에서 두드러지게 나타나는데, 컨스턴트풀의 구조는 그림 1와 같다. 컨스턴트풀은 테이블 형태로 구성되어서 각각의 엔트리들은 인덱스를 가진다. 컨스턴트풀 내의 많은 엔트리들은 컨스턴트풀의 다른 엔트리에 대한 인덱스 값을 자신의 내용으로 가진다. 그림 1에서 Class, String, Reference, Name & Type형 엔트리는 모두 다른 컨스턴트풀 엔트리에 대한 인덱스를 참조하는 구조를 가지고 있다[2][3].



[그림 1] 클래스 파일의 컨스턴트풀 포맷

2.2 레졸루션

클래스 파일을 통틀어서 컨스턴트풀 엔트리는 컨스턴트풀 리스트에서 각각의 엔트리의 위치를 알려주는 인덱스에 의해서 참조된다. 실제로 클래스 파일 내에서 클래스나, 필드, 메소드 정보를 접근할 때에는 각각의 Name & Descriptor에 대한 컨스턴트풀 인덱스 정보를 가지고 접근하게 된다[4].



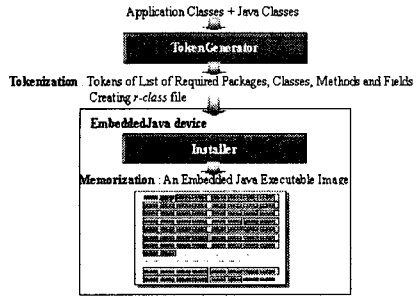
[그림 2] 메소드 컨스턴트의 심볼레퍼런스

그림 2는 Math 클래스에서 random() 메소드의 심볼레퍼런스다. JVM은 java.lang.Math클래스를 로드, 링크, 초기화하는 심볼레퍼런스를 검증한다. 그림 2와 같이 레졸루션은 클래스, 필드 이름, 메소드 타입을 알기 위해서 다섯 번의 엑세스가 필요하다.

3. 클래스 파일의 프리레졸루션

3.1 프리레졸루션 시스템의 개요

본 논문의 프리레졸루션 시스템은 기존 클래스 파일과는 달리 컨스턴트풀내의 모든 심볼레퍼런스를 인덱스로 변환된 r-class파일을 생성한다. 이 r-class는 프리레졸루션을 위한 클래스 파일을 나타낸다. r-class는 후에 타겟디바이스의 인스톨러에 의해 ROM메모리 주소로 바뀌게 된다.



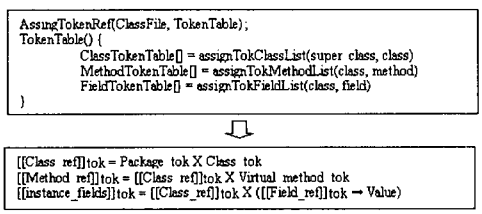
[그림 3] 프리레졸루션 시스템의 과정

그림 3과 같이 프리레졸루션 시스템은 심볼레퍼런스부터의 Tokenization 단계와 토큰으로부터의 Memorization의 단계로 나눈다.

3.2 Tokenization

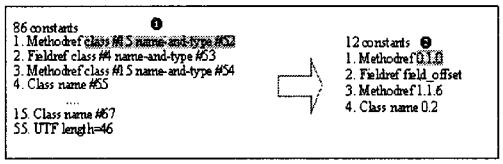
클래스 파일내 컨스턴트풀로부터 많은 양의 스트링 중심의 심볼레퍼런스를 제거한 후 각 레퍼런스와 관련된 인덱스 중심의 토큰을 할당하여 심볼레퍼런스를 토큰으로 바꾸는 단계이다. 첫 번째로 주어진 클래스 파일에서 필요로 하는 각 클래스와 메소드 그리고 필드의 토큰 테이블을 생성한다.

생성된 모든 토큰 테이블이 스트링 기반의 컨스턴트풀에서 토큰 기반으로 바꾸는 과정에 사용된다. 그림 4는 스트링 기반의 콘스턴트풀이 토큰 기반의 콘스턴트풀로 변환되는 알고리즘을 보여준다. 토큰 기반의 컨스턴트풀과정을 보여준다.



[그림 4] 심볼레퍼런스의 Tokenization 알고리즘

그림 5는 심볼레퍼런스의 Tokenization 과정을 나타낸다. 회색부분 "15 name-and-type #52"는 클래스 이름, 메소드 이름, 메소드 타입을 알려주는 인덱스를 가지고 있다. "0.1.0" 회색부분은 컨스턴트풀내에서 첫 번째 값(0)이 패키지 인덱스에 대응하는 토큰을 가지고 있고, 두 번째 값(1)은 클래스 인덱스에 대응하는 토큰을 가지고 있고는 의미이다. 그리고 세 번째 값(0)은 메소드 인덱스에 대응하는 토큰을 가지고 있다는 의미이다.

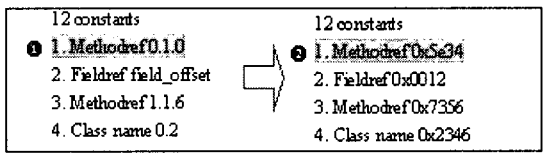


[그림 5] 심볼레퍼런스의 Tokenization 과정

3.3. Memorization

본 논문의 인스톨러는 타겟디바이스상에 요구된 패키지, 클래스, 메소드와 필드의 모든 오프셋 정보를 가지고 있다. 타겟디바이스의 인스톨러는 r-class를 인스톨하는 과정에서 각 토큰에 해당하는 주소를 할당할 수 있다.

그림 6은 각 토큰에 대응하는 타켓 메모리 주소 할당 과정을 보여준다.



[그림 6] 토큰의 메모리 어드레싱 과정

그림 6는 "0.1.0" 회색 부분은 패키지 인덱스, 클래스 인덱스와 메소드 인덱스 3개의 토큰으로 구성된다. "Ox5e34"는 타겟디바이스 내에서 대응하는 토큰의 실제 주소를 가리킨다.

4. 평가

기존 클래스 파일내 메소드와 필드의 심볼레퍼런스는 일반적으로 UTF-8 스트링으로 구성되어 있다.

본 프리레졸루션의 r-class는 컨스턴트풀내의 모든 스트링이 제거된 클래스파일이다. 기존 클래스파

일과 본 시스템의 r-class 파일내의 컨스턴트풀 사이즈를 비교하기 위하여 13개의 클래스 파일을 시험한 결과 r-class가 약 92%정도 감소되었음을 알 수 있다. 표 1의 기존 클래스파일과 본 시스템의 r-class 파일의 컨스턴트풀의 사이즈를 보여준다.

[표 1] 기존 클래스와 r-class의 컨스턴트풀 사이즈

	Constant Pool of traditional Class files	Constant Pool of our r-class files
HelloWorld	973	61
JavaLoyalty	1237	61
JavaPulse	5069	369
Wallet	1413	97
Atreenode	3413	273
BTreenode	3127	265
NullApp	539	33
Samplehibrary	166	5
CVM	431	5
Op20	1418	124
Op20Example	2231	188
SecurityDom	3073	366
Simple-Example	816	66
Total size	**23,906**	**1913**
Average size	**1839**	**147**
Effective Ratio		**92%↓**

기존 레졸루션은 메소드와 필드 컨스턴트풀에 대해서 6번 엑세스 해야한다. 그러나 본 시스템의 프리레졸루션은 단 한번의 엑세스로 각 명령에 해당하는 주소에 접근 할 수 있다. 이것은 컨스턴트풀과 메모리의 접근 횟수를 줄임으로써 전체적인 접근시간을 감소시킨다.

표 2는 프리레졸루션과 기존레졸루션의 컨스턴트풀과 메모리 접근 횟수를 비교한 결과를 나타낸다.

[표 2] 컨스턴트풀과 메모리 접근 횟수 비교

	Class file of traditional resolution			r-class file of our pre-resolution		
	constant pool	target address	Total	constant pool	target address	Total
Class	2	2	4	1	1	2
Method	6	2	8	1	1	2
field	6	2	8	1	1	2

그러나 프리레졸루션 시스템은 동적 바인딩을 제공하지 못한다. 하지만 임베디드 시스템은 고정된 주소를 기반으로 한 메모리 구조를 가지고 있기 때문에 동적 바인딩 없이 임베디드 시스템에 프리레졸루션 시스템을 적용할 수 있다.

5. 결론

본 논문의 프리레졸루션의 r-class는 전형적인 기존 클래스파일의 컨스턴트풀 사이즈와 비교하여 약 92%의 감소가 이루어 졌다.

본 프리레졸루션 시스템은 임베디드 자바가상시계가 클래스 파일을 로딩할 때 이미 ROM에 마스크된 주소정보를 바탕으로 심볼레퍼런스를 물리 주소로 바꾼다. 이것은 현 레졸루션과 비교할 때 클래스 파일 로딩 시간의 오버헤드를 가질 수 있다.

그러나 프리레졸루션 기술은 실행시 레졸루션의 비용와 메모리 참조 횟수를 줄임으로서 임베디드 시스템에 유용하다.

[참고문헌]

[1]Microsystems. The K Virtual Machine. Sun Microsystems, http://java.sun.com/j2me

[2] Mary Campione, Kathy Walrath, Alison Huml, Tutuorial Team, The Java(TM) Tutorial Continued: The Rest of the JDK(TM) (The Java(TM) Series). Sun Microsystems. The Java Archive file format. Sun Microsystems(1998)

[3] Doo-Jin Kang, Hye-Seon Maeng, Young-Min Lee, Tack-Don Han, Shin-Dug Kim, Class File Pre-Resolution for Embedded Java System. Domestic Conferences KISS(1999), Vol. 3, No. 1, 385-387

[4] Bill Venners, The Linking Model, Inside the Java Virtual Machine, McGraw-Hill(1998)

[5] Sun (1997), Java Card 2.0 Language Subset and Virtual Machine Specification, revision 1.0 final edn. ftp://ftp.javasoft.com/docs/javacard/JC20-Language.ps.

[6] W. Pugh. Compressing Java Class Files. Proceedings of ACM/SIGPLAN Conference on Programming Language Design and Implementation (PLDI)'99, May(1999)

[7] F. Tip, C. Laffra and P. F. Sweeney. Practical Experience with an Application Extractor for Java. OOPSLA '99, Nov(1999)