제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003.5)

임베디드 자바 시스템을 위한 핵심 클래스 파일에서 상수풀 항목의 해석

양희재
경성대학교 컴퓨터공학과
e-mail:hjyang@star.kyungsung.ac.kr

Analysis of the Constant Pool Entries in Core Class File of Embedded Java System

Heejae Yang
Dept of Computer Engineering, Kyungsung University

요 약

본 논문에서는 임베디드 자바 시스템을 위한 핵심 클래스 파일에서 상수풀(constant pool)의 각 항목들에 대해 통계를 내고 분석해 보았다. 분석 대상 클래스 파일들은 썬 마이크로 시스템사의 J2ME/CLDC 클래스 파일들과, RTJ Computing 사의 simpleRTJ 시스템의 클래스 파일들이다. 이들 파일들에 대한 분석 결과 임베디드 자바 시스템을 위한 핵심 클래스 파일에서 상수풀은 전체 파일 크기의 거의 절반에 해당되는 46%를 차지하고 있음을 알 수 있었다. 또한 상수풀에는 평균 44개의 상수들이 있으며, 이들 중 실제 바이트코드 실행에 사용되는 상수들은 단지 6퍼센트에 불과한 3개에 지나지 않았다. 나머지 78퍼센트의 상수들은 단지 형식 확인과 클래스 링크 목적으로만 사용되는 것들이었다. 이 결과는 실행 시간시 동적인 형식 확인과 클래스 링크를 하지 않는 환경이라면 매우 큰 메모리 절감을 이룰 수 있음을 보여주고 있는 것이다. 본 연구의 결과는 클래스 파일이 ROM 등에 탑재되어있는 임베디드 시스템 환경에 적용될 수 있다.

1 서 론

임베디드 시스템에 자바 기술을 적용하는 사례가 최근 급격히 증가하고 있다 [1]. 임베디드 시스템은 데스크톱 시스템과 달리 사용하는 프로세서의 종류나 운영체제의 종류 등 플랫폼이 매우 다양하므로 어떤 응용 프로그램을 임의의 다른 환경에 이식하는 것이 매우 어려웠다. 자바는 플랫폼 독립성을 제공해 줄 뿐 아니라 안전성도 제공하고 있으므로 이런 어려움을 극복하는 한가지 대안으로 널리 받아들여지게 된 것이다.

자바는 객체지향언어로서 하나의 프로그램은 다수의 객체, 즉 클래스들로 구성된다. 자바에서 각 클래스들은 클래스 파일이라는 이름으로 저장된다.

이들 클래스 파일들은 그 자체로, 또는 변형된 형태로 메모리에 적재되어 자바 프로그램 실행 시 사용되어진다. 임베디드 시스템은 메모리가 매우 제한적이므로 클래스 파일을 메모리에 적재할 때 그것을 메모리를 적게 사용할 수 있는 형태로 변형할 필요가 있다.

자바의 클래스 파일은 헤더, 상수풀, 클래스, 필드, 메소드 등 다섯 가지의 항목들로 구성되어져 있다 [2]. 일반적인 데스크톱 자바 환경에서 클래스 파일을 분석한 연구에 따르면 상수풀의 크기가 전체 파일의 60% 이상을 차지하는 것으로 조사되었다 [3].

즉 클래스 파일을 메모리에 적재할 때 상수풀이 가장 많은 메모리를 차지하게 된다는 것이다. 따라서 임베디드 시스템과 같이 제한된 메모리를 갖는 시스템에서 자바 프로그램을 실행하기 위해서는 클래스 파일 중 가장 많은 부분을 차지하는 상수풀의 크기를 최소화시켜 메모리에 적재할 필요가 있다.

본 연구에서는 임베디드 자바 시스템을 위한 클래

스 파일들의 상수풀에 대해 조사해보고, 상수풀 내의 각 항목들에 대한 분석을 통해 각 항목들이 메모리 사용에 미치는 영향을 알아보고자 한다. 즉 상수풀 내에 어떤 상수들이 있으며, 그것들의 평균적인 크기는 얼마나 되며, 메모리 사용을 줄이기 위해 이들 상수들 중 어떤 항목을 제외시킬 수 있는지 등에 대해 고찰해 보았다.

임베디드 시스템을 위한 자바 환경으로는 J2ME, Waba, Perc JVM, Jbed, simpleRTJ 등을 들 수 있으며, 본 연구에서는 이들 중 썬 마이크로 시스템사의 J2ME에서 정의되어있는 CLDC 환경과 [4], RTJ Computing 사에서 개발한 simpleRTJ [5] 에서 사용되고 있는 클래스 파일에 대해 각각 조사해 보았다.

실험 결과 이 두 가지 환경의 클래스들은 거의 동일한 결과 값을 보여주고 있으며, 따라서 본 연구의 결과는 임의의 임베디드 자바 시스템에도 적용될 수 있을 것으로 기대된다.

본 논문의 구성은 다음과 같다. 2절에서는 분석 대상으로 선택한 CLDC 핵심 클래스 파일들과 simpleRTJ 클래스 파일들에 대해 클래스 종류 및 그들의 계층구조에 대해 설명한다. 3절에서는 실제로 상수풀을 분석한 결과에 대해 소개하며, 4절에서는 분석 결과에 대한 고찰 및 결론에 대해 설명한다.

2 분석 대상 클래스 파일

J2ME/CLDC 의 경우 조사된 클래스는 네 개의 패키지에 나뉘어 배치되어 있는데 [6], java.lang 패키지에 38개, java.io 에 18개, java.util 에 13개, javax.microedition.io 에 10개 등 79개의 클래스와 인터페이스가 대상이 된다. 이들 중 인터페이스와 특수 클래스 등 17개를 제외한 총 62개를 대상 클래스로 하였다. 이들 중 전체 클래스의 거의 절반에 해당되는 29개의 클래스가 예외(exception) 및 오류(error)의 처리를 위해 사용되어지는 것들이다.

simpleRTJ 의 경우 클래스들은 두 개의 패키지에 나뉘어 배치되어 있는데 [7], java.lang 에 41개의 클래스와 2개의 인터페이스가 있으며, java.util 에 2개의 클래스와 1개의 인터페이스가 있다. 본 연구에서는 인터페이스를 제외한 총 43개의 클래스들을 분석 대상으로 했다. simpleRTJ 의 경우도 CLDC 의 경우와 마찬가지로 전체 클래스의 절반 이상인 28개의 클래스가 예외 및 오류처리를 위해 사용되어지는 것들이었다. 일반 클래스와 예외 및 오류 클래스들을 구별한 것은 3절에서 보이겠지만 이들의 특성에 큰 차이가 존재하고 있기 때문이다.

따라서 조사대상에 해당되는 전체 클래스 숫자는 총 105개이다.

3 상수풀 항목에 대한 통계

3.1 상수풀의 크기

서론에서 밝힌 바와 같이 데스크톱 환경을 위한 자바 클래스들의 경우 클래스 파일의 60% 이상을 상수풀이 차지한다. 임베디드 환경을 위한 자바 클래스 파일에서는 그 비율이 어떻게 되는지를 조사해 보자.

조사 결과 CLDC 클래스들의 경우 클래스 파일의 크기는 평균 1,168바이트였다. 이것을 클래스별로 구분하여 보면 일반 클래스는 1,986바이트, 예외/오류 클래스는 237바이트였다. 마찬가지로 simpleRTJ 클래스의 경우 전체 평균은 902바이트, 일반 클래스는 1,959바이트, 예외/오류 클래스는 335바이트로 나타났다.

다음으로는 상수풀의 크기를 조사해보았다. CLDC 클래스의 경우 상수풀의 크기는 평균 527바이트(표준편차는 532, 이하 동일 표기)였다. 이것을 클래스별로 구분하여 보면 일반 클래스는 860바이트(540), 예외/오류 클래스는 148바이트(63)였다. 마찬가지로 simpleRTJ 클래스의 경우 전체 평균은 414바이트(399), 일반 클래스는 805바이트(466), 예외/오류 클래스는 204바이트(52)로 나타났다.

클래스 파일의 크기와 상수풀의 크기를 각각 알게 되었으므로 이들의 비율을 알 수 있다. CLDC 클래스의 경우 상수풀의 크기는 전체 파일 크기 대비 45% 이며, 이것을 클래스별로 구분하여 보면 일반 클래스는 43%, 예외/오류 클래스는 62% 이다. 또한 simpleRTJ 클래스의 경우 전체적으로는 46% 이며, 클래스별로는 일반 클래스가 41%, 예외/오류 클래스가 61% 이다.

이 결과에 따르면 *임베디드 자바 시스템을 위한 클래스 파일에서 상수풀이 차지하는 비율은 전체적으로 46%* 정도라는 것을 알 수 있다. 이 값은 데스크톱 자바 시스템을 위한 클래스 파일에서의 비율인 60% 에 비해 다소 낮은 수준임을 알 수 있다. 다만 예외/오류 클래스들의 경우는 클래스 파일 자체가

표 1 세부 항목별 통계

	CLDC (ea)	simple-RTJ(ea)	All (%)
Utf8	26	22	56
Integer	1	1	3
Float	0	0	0
Long	0	0	0
Double	0	0	0
Class	5	3	9
String	1	0	1
NameAnd-Type	8	5	15
Fieldref	1	1	2
Methodref	7	5	14
Interface-Methodref	0	0	0

워낙 작기 때문에 그 비율은 62% 에 이르렀다.

3.2 상수의 개수

이번에는 상수풀에 들어있는 상수의 개수에 대해 분석해 보자. CLDC 클래스의 경우 상수의 개수는 평균 50개(표준편차는 52, 이하 동일 표기)였다. 이것을 클래스별로 구분하여 보면 일반 클래스는 83개(54), 예외/오류 클래스는 13개(4)였다. 마찬가지로 simpleRTJ 에서는 전체 평균이 37개(37), 일반 클래스는 76(40), 예외/오류 클래스는 16(4)으로 나타났다.

이것을 정리하면 임베디드 자바 시스템을 위한 클래스 파일에서 상수풀 내의 *상수들의 개수는 전체 평균 44개* 정도이며, 클래스별로는 일반 클래스는 80개, 예외/오류 클래스는 15개 정도의 상수를 갖는다는 것으로 요약된다.

3.3 세부항목별 통계

여기서는 상수풀 내의 각 항목들을 종류별로 나누어서 분류해보았다 (표 1). 표 내부의 값은 각 클래스들에 대한 평균값을 나타낸 것이다.

이 표에서 알 수 있듯이 상수풀에서 가장 많은 몫을 차지하는 것은 CONSTANT_Utf8 으로서 56% 정도를 점유한다. 이것은 데스크톱 환경의 자바 클래스 파일의 해당 값인 59% 와 비슷한 수준이다. 그 외에도 CONSTANT_NameAndType 이 15% (데스크톱 자바 클래스 파일의 해당 값은 13%, 이하 동일 표기), CONSTANT_Methodref 가 14% (10%), CONSTANT_Class 가 9% (9%), CONSTANT_Fieldref 가 2% (4%) 등을 차지하며, 나머지 상수는 거의 없는 수준이다. 세부항목별 통계에서는 데스크톱 환경의 자바 클래스 파일의 해당 값과 각각 비교해 보았을 때 큰 차이가 없는 것을 발견할 수 있었다.

3.4 기능항목별 통계

마지막으로 상수풀 내의 각 항목들을 기능별로 나누어 조사해보았다 (표 2).

즉 하나의 클래스에는 평균 44개의 상수들이 상수풀에 놓여져 있는데, 그들 중에서 바이트코드 실행 시 참조되어지는 상수들의 개수는 3개(6%) 정도이며, 34개(78%)의 형식 및 링크 정보들이 있고, 기타 디버깅 등 직접적인 필요가 없는 상수들이 7개(16%) 있다는 것이다.

데스크톱 환경의 자바 클래스 파일에 대해 조사한 결과 값은 상수가 8%, 형식 및 링크 정보가 60%, 기타 정보가 32% 로 나와있다 [2]. 따라서 임베디드 시스템의 경우와 비교해볼 때 상수는 비슷한 수준을 보이고 있고, 형식 및 링크 정보는 임베디드 시스템이 18% 정도 더 많으며, 기타 정보는 임베디드 시스템이 16% 정도 적은 수준을 나타내는 것을 알 수 있다.

즉 임베디드 시스템을 위한 클래스 파일에는 실제 바이트코드가 참조하는 상수들은 그리 많지 않으며, 대부분(78%)은 형식 및 링크 정보를 위한 상수임을 알 수 있다. 따라서 만일 안전성을 희생하여 형식 확인을 하지 않고, 또한 동적 클래스 적재를 하지 않는 환경을 가정하면 78% 이상의 상수를 없앨 수

표 2 기능 항목별 통계

	CLDC (ea)	simple-RTJ(ea)	All (%)
일반상수	3.6	1.7	6
형식 및 링크	41.4	27.5	78
기타	4.8	7.7	16

있다는 것이다. 또한 표2의 기타 정보도 실제 실행에는 필요하지 않으므로, 실제 소용되는 상수는 상수풀의 6% 정도로 족하다는 것이다.

4 결 론

본 논문에서 우리는 자바 클래스 파일의 크기 중 가장 큰 부분을 차지하고 있는 상수풀에 대해 고찰해 보았다. 임베디드 시스템을 위한 자바 클래스 파일인 경우 상수풀은 클래스 파일 크기의 46% 정도를 차지하는 것으로 조사되었으며, 이것은 470바이트 정도의 크기에 해당된다.

상수풀 내에 있는 11개의 개별 항목별로 상수들의 사용도를 조사해 보았으며, 하나의 클래스당 평균 44개 정도의 상수들이 있음을 알 수 있었다. 이들 중 실제로 바이트코드 실행에 사용되어지는 상수는 6% 정도인 3개에 지나지 않았으며, 78% 에 해당되는 34개가 형식 및 링크 정보를 위해 사용되었다.

따라서 실행 시 형식 확인을 생략하고, 또한 동적 클래스 적재를 하지 않는 환경이라면 거의 대부분의 상수풀 항목들을 사용하지 않을 수 있으므로 매우 큰 폭의 메모리 절감이 가능해질 수 있다는 것이다. 임베디드 시스템의 경우 메모리 자원이 제한적이므로 이와 같은 메모리 절감은 굉장히 중요하다고 볼 수 있다.

본 연구는 임베디드 자바 시스템의 클래스 파일에서 상수풀이 이상과 같은 특징을 가지는 것에 착안하여 메모리 사용 면에서 보다 효율적인 임베디드 자바 가상기계를 개발하는데 적용 될 수 있을 것이다.

참고문헌

[1] S. Helal, “Pervasive Java,” IEEE Pervasive Computing, Jan-Mar 2002, pp.82-85

[2] 양희재, 자바가상기계, 한국학술정보, 2001년 3월, ISBN 89-5520-342-4

[3] D. Antonioli and M. Pilz, “Analysis of the Java Class File Format,” TR, Dept of Computer Sci., Univ of Zurich, Apr 1998

[4] Sun Microsystems, Connected, Limited Device Configuration Specification, Version 1.0a, May 2000

[5] RTJ Computing, simpleRTJ: A Small Footprint Java VM for Embedded and Consumer Devices, http://www.rtjcom.com

[6] 양희재, "내장형 자바 시스템을 위한 J2ME/CLDC 클래스 파일의 분석", 대한전자공학회 컴퓨터/반도체 소사이어티 추계학술대회 논문집, v.25, n.2, pp.29-32, 2002년 11월

[7] 양희재, "simpleRTJ 클래스 파일의 형식 분석," 한국해양정보통신학회 2002년도 추계종합학술대회, v.6, n.2, pp.373-377, 2002년 11월