J2ME CDC 규격의 임베디드 자바플랫폼 개발

원희선° 김영호 김선자
한국전자통신연구원
{hswon°, yhkim, sunjakim}@etri.re.kr

Development of the Embedded Java Platform supporting J2ME CDC specification

Heesun Won° Youngho Kim, Sunja Kim
Electronics and Telecommunications Research Institute

요 약

임베디드 자바플랫폼을 정의하는 J2ME는 컨피규레이션에 따라 크게 CLDC와 CDC로 구분된다. CLDC 자바 플랫폼은 저사양 휴대 단말 등에 탑재되어 널리 상용화되고 있으며,CLDC와 비교하여 CDC 자바 플랫폼은 자바2와 호환 가능한 완전한 JVM이 포함되고 고기능의 폭넓은 자바 API의 지원이 가능하므로 홈서버, 디지털 TV, 텔레매틱스 분야 및 고사양 모바일 단말 등에서 제공될 신규 서비스를 위한 자바 플랫폼으로 지목되고 있다. 본 논문에서는 클린룸으로 구현한 JVM과 GNU 프로젝트인 Classpath를 기반으로 구현한 PP 규격의 클래스 라이브러리를 통합한 CDC 자바 플랫폼 개발에 대해 기술한다.

1. 서 론

임베디드 환경을 위한 자바 기술인 J2ME(Java 2 Micro Edition)[1]는 타겟 시스템의 메모리, CPU 및 응용 어플리케이션에 적합한 환경 구성을 위해 컨피규레이션과 프로파일을 정의하고 있으며, 컨피규레이션에 따라 크게 CLDC (Connected Limited Device Configuration) 와 CDC (Connected Device Configuration) 플랫폼으로 구분된다. CLDC 플랫폼은 휴대폰 등에 탑재되어 널리 사용되고 있으나 제한적 기능으로 인하여, 확장된 리소스를 갖는 시스템에서는 폭넓은 서비스를 제공할 수 있는 CDC 플랫폼이 요구되고 있다. 특히 OSGI와 ETSI 등의 국제 표준 단체에서 주관하는 홈서버, 텔레매틱스, 디지털 TV 관련 시스템 미들웨어들은 CDC 플랫폼을 기반으로 하므로 이에 대한 핵심 기술을 습득하고 상용화의 경우에 라이센스 부담을 최소화하기 위해 CDC 플랫폼에 대한 연구와 개발은 매우 중요하다.

본 논문에서는 CDC 플랫폼을 임베디드 환경에 적합하도록 구현한 내용을 기술한다. 2장에서는 J2ME 플랫폼에 대한 개요를 설명하고 3장에서는 구현 내용을 JVM(Java Virtual Machine)과 자바 API로 나누어 설명한다. 4장에서는 개발한 자바플랫폼을 검증하기 위해 개발된 자바API 테스트 킷에 대해 설명하고 5장에서는 결론을 맺는다.

2. J2ME 자바플랫폼

2.1 컨피규레이션과 프로파일

J2ME 구조는 다양한 컨피규레이션, 프로파일 및 선택 가능한 패키지들로 구성되며, 이들을 조합하여 타겟 디바이스와 시장 조건에 가장 적합한 자바 실행 환경을 구축할 있다.

컨피규레이션은 JVM과 최소 집합의 클래스 라이브러리들로 구성되며, 네트워크 연결성과 메모리 풋프린트 등의 비슷한 특성을 가지는 특정 영역의 디바이스들을 위한 기본 기능을 정의한다. 현재는 CLDC와 CDC등 두 개의 J2ME 컨피규레이션이 존재한다.

프로파일은 산업별로 요구되는 특정 디바이스에 필요한 상위 API 집합을 정의하며, 컨피규레이션과 결합하여 구체적인 자바 실행 환경을 정의한다. 주로 CLDC와 결합되어 핸드폰이나 PDA 등에 탑재되고 있는 MIDP (Mobile Information Device Profile)가 널리 알려진 프로파일의 예이며, CDC 플랫폼을 위해서는 FP(Foundation Profile), PBP(Personal Basis Profile), PP(Personal Profile) 등이 정의되어 있다.

2.2 CLDC와 CDC

CLDC와 비교하여 CDC는 보다 확장된 시스템 리소스를 갖는 광범위한 임베디드 디바이스를 타겟으로 하는 규격이다. CLDC VM에는 사용자 정의 클래스로더, 쓰레드 그룹 및 데몬 쓰레드, 클래스 인스턴스의 finalization, 비동기 예외 등의 기능이 제거되어 있는 반면, CDC 플랫폼은 표준 JVM 규격에 정의된 완전한 VM을 요구하며 네이티브 메소드 호출을 위한 인터페이스도 포함한다. 따라서 J2SE 및 J2EE(Java 2 Enterprise Edition)와도 완벽히 호환되므로 모바일 클라이언트로부터 서버까지 일관된 자바 기술을 이용할 수 있게 한다.

CLDC 라이브러리는 J2SE(Java 2 Standard Edition) 표준 라이브러리 중의 일부 클래스와 CLDC에만 해당하는 클래스 등의 두 가지 범주로 나누어지는데, CDC는 CLDC의 위 두 가지 범주의 모든 클래스 및 CLDC에서 제외된 J2SE의 많은 클래스들도 포함하고 있으므로 더욱 다양한 API를 사용할 수 있다.

2.3 프로파일: MIDP, FP, PBP, PP

MIDP(Mobile Information Device Profile)는 CLDC 이상의 컨피규레이션과 결합될 수 있는 프로파일로써 핸드폰과 같은 모바일 기기를 위한 기본적인 사용자 인터페이스와 네트워크 보안 기능등을 포함하고 있다. 최신 버전인 MIDP 2.0에는 멀티미디어와 게임 기능 등이 확장되어 있다.

FP(Foundation Profile), PBP(Personal Basis Profile), PP(Personal Profile) 등은 CDC 기술을 위한 프로파일로써 각각 서로 독립적이 아닌 포함 관계에 있다. 즉 FP 는 J2SE의 기본 클래스 라이브러리와 CLDC 라이브러리를 포함하나 GUI 기능은 지원하지 않는다. PBP는 FP에 정의된 API 외에 GUI를 위한 awt 패키지와 beans, rmi 등의 패키지를 포함한다. PP는 자바2 이전의 퍼스널 자바 환경의 지원을 목적으로 하며 PBP에 정의된 모든 API와 완전한 awt 패키지 및 애플릿등의 기능을 포함한다.

3. 구현

그림1은 자바플랫폼의 전체 구조를 보여준다. 자바플랫폼의 구현은 크게 JVM(Java Virtual Machine)과 자바 API로 구분할 수 있다. 본 연구에서는 CDC 플랫폼을 위한 표준 규격의 JVM을 네이티브 메소드 호출 인터페이스를 포함하여 클린룸으로 구현하였다. 자바 API를 위해서는 오픈소스 프로젝트인 Classpath 버전 0.05를 기반으로 PP 규격의 클래스 라이브러리를 개발하였으며 JVM과 통합하여 CDC/PP 규격의 자바플랫폼을 완성하였다. 개발 환경은 리눅스 상에서 GTK+1.2 그래픽 패키지를 기반으로 한다.

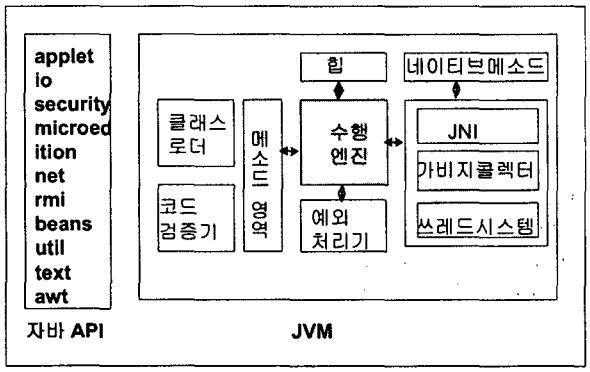

그림 1 자바 플랫폼의 구조

3.1 JVM 구현

본 연구에서는 CDC 플랫폼을 포함하여 J2SE 및 J2EE에도 사용 가능한 완전한 자바2 VM[2]을 클린룸으로 구현하였다.

JVM의 기본 동작은 로칼 파일이나 네트워크으로부터 읽어 들인 클래스파일을 해석하여 클래스 정보를 메소드 영역과 힙 등에 저장하고 저장된 정보를 기반으로 순서에 따라 일련의 메소드들을 수행시키는 것이다. 본 구현에서는 JNI(Java Native Interface)[3]와 동적 라이브러리 링킹을 지원하므로 프로그래머가 네이티브 메소드를 자유롭게 정의하여 사용할 수 있으며 코드 검증기 구현을 통해 안정성 및 보안 기능을 강화하였다. 특히 자원 제약이 많은 임베디드 시스템 환경에서 우수한 성능을 보장하도록 쓰레드 시스템, 자바 메모리 시스템 및 가비지 콜렉터 등을 설계, 구현하였다.

가) 클래스 로더

클래스 파일 형식으로 컴파일된 프로그램 정보는 클래스로더를 통해 로딩되면서 VM의 클래스 객체에 저장된다. 클래스로더는 기본적으로 시스템 클래스로더가 사용되며 개발자는 시스템 클래스로더의 하부 기능을 이용하는 사용자 정의 클래스로더를 정의하여 사용 할 수 있다. 클래스 로더의 수행 절차는 그림2와 같다.

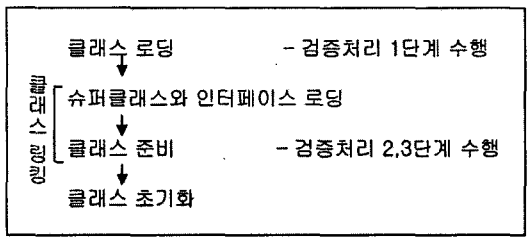

그림 2 클래스 로더의 수행 절차

그림 2의 클래스 로딩 단계에서는 클래스파일의 정보를 클래스 객체에 저장하고, 효율적인 클래스 관리를 위해 해쉬 테이블에 클래스 인덱스를 추가하며, 파일 형식을 검증하는 검증처리 1단계를 수행한다. 클래스 준비 단계에서는 로딩된 클래스의 메소드 테이블과 인터페이스 메소드 테이블을 만들고 VM 실행 옵션에 따라 클래스 타입과 상속관계 및 메소드 정보 등의 검증을 위한 검증처리 2,3 단계[2]를 수행한다. 클래스 초기화 단계에서는 클래스의 정적 변수들을 초기화한다.

나) 바이트코드 검증기와 레졸루션

검증 단계 4를 수행하는 바이트코드 검증기는 메소드를 가상적으로 실행시켜 메소드 코드의 안정적 수행을 검증한다. 본 구현에서는 검증기에서 메소드 수행에 필요한 모든 레졸루션을 수행하였는데 수행엔진에서 메소드 실행에 직접적으로 필요한 레졸루션만을 수행하는 방식과 비교하여 객체 동기화를 위한 오버헤드를 상당히 줄일 수 있어서 성능을 개선시킬 수 있었다.

다) 수행 엔진

JVM의 수행엔진으로는 주로 인터프리터와 JIT (Just-In-Time) 컴파일러가 사용되고 있다. 인터프리터는 JIT 컴파일러와 비교하여 성능이 떨어지는 단점이 있으나 실행시에 메모리 자원을 적게 요구하고 포팅이 쉬우며 코드가 단순하여 안정성이 우수하다는 등의 임베디드 시스템에 유

리한 여러가지 장점을 가지고 있다. 본 연구에서는 인터프리터를 구현하였으며 성능을 높이기 위해 토큰 쓰레딩을 이용한 인스트럭션 처리 방식[4]을 적용하였다.

라) 쓰레드 시스템

구현한 자바 쓰레드 시스템은 POSIX Pthread를 기반으로 하여 상이한 시스템으로의 포팅이 용이하도록 자바 계층, 네이티브 계층, 플랫폼 추상 계층 등의 3 계층으로 구분하였다. 또한 쓰레드-캐쉬를 사용하여 쓰레드 생성을 빠르게 하였다.

자바 언어는 기본적으로 멀티쓰레딩을 지원하므로 단일 쓰레드가 수행되고 있는 동안에도 빈번한 객체 동기화를 수행하게 된다. 이에 따른 상당한 오버헤드를 줄이고 다수의 쓰레드가 수행 중인 경우에도 동기화가 빠르게 수행되어 성능을 향상시킬 수 있도록 Thin lock과 Fat lock 등의 두 가지의 락 구조체를 정의하고 쓰레드들간의 경쟁 유무에 따라 적합한 락 객체를 사용하도록 구현하였다[5].

마) 메모리 시스템 및 가비지 콜렉터

시뮬레이션을 통해 생명 주기가 짧은 작은 객체 할당이 빈번이 요구되는 점과 특정 몇 개의 크기가 주로 사용되는 점을 파악하고 이러한 결과를 기반으로 프레그멘테이션을 최소화하고 메모리 분할과 통합에 대한 오버헤드를 줄이도록 자바 메모리 시스템을 설계하였다. 그림 3은 구현된 메모리 시스템의 구조를 나타낸다.

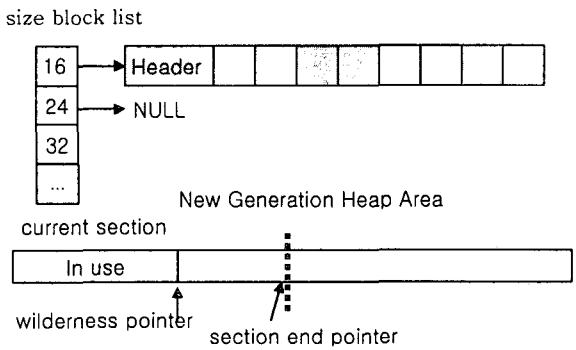

그림 3 자바 메모리 시스템 구조

메모리 시스템은 크게 두 세대(generation)로 구분되며 같은 크기의 블록들을 모은 블록 클러스터들의 리스트를 관리한다. 가비지 콜렉션은Generational 방식을 기본으로 mark/copying 기법을 사용하였다.

3.2 자바 API 구현

GNU Classpath는 J2SE 규격의 자바 API들을 개발하는 오픈소스 프로젝트이다. 본 연구에서는 라이센스 부담이 없는 자바 기술을 확보하기 위해 Classpath 버전0.05를 기반으로 VM과 통합하였으며 통합 테스트 수행 중에 수 많은 오류들을 발견하고 정정하여 그림 1의 자바 API 패키지들이 모두 포함되는 PP 규격의 클래스 라이브러리를 개발하였다.

Classpath에서는 버전 0.6 이후부터 자바 그래픽 패키지인 awt의 구현을 GTK+2.0 API를 사용하여 진행하고 있는 반면 본 연구에서는 자원 제약이 심한 임베디드 환경에 적합하도록 GTK+1.2를 기반으로 awt 구현을 완료하였다. 또한 한글 등의 두 바이트 문자 처리를 위한 문자 변환 코드 등을 추가하였으며, PP 규격에 속하지 않은 상당수의 불필요한 API들이 Classpath에 포함되어 있으므로 이들을 제거하는 다운사이징 작업을 수행하였다.

4. 자바 API 검증 테스트 킷

개발된 자바 플랫폼의 디버깅 및 안정화를 위해서 본 연구에서는 PP에 정의된 API들을 검증하는 테스트 킷을 개발하였다. 테스트 킷은 2,000여개의 테스트 프로그램들로 구성되며 하나의 테스트 프로그램에는 여러 개의 관련된 자바 API를 검증하기 위한 코드들이 포함되어 있다. 테스트 프로그램은 테스트 도구를 개발하는 GNU 프로젝트인 Mauve의 프레임워크를 기반으로 개발하였으므로 Mauve 환경을 이용하여 테스트를 수행할 수 있다.

5. 결론 및 향후 과제

본 연구의 목표는 디지털 TV, 홈서버 및 텔레매틱스 등과 관련된 고사양 임베디드 시스템의 자바 실행 환경을 구축하기 위한 것으로써 이에 적합한 CDC-PP 규격의 자바 플랫폼을 개발하였다. 개발된 플랫폼은 다양한 자원 제약이 있는 임베디드 환경에 적합하도록 개발한 JVM과 GNU Classpath를 기반으로 개발된 PP 규격의 자바 API를 통합한 것이며, 본 연구에서 개발된 테스트 킷을 이용하여 안정성을 검증하였다.

향후 연구 과제로는 다양한 임베디드 환경에 최적화 시킬 수 있는VM 핵심 모듈들의 적응적 기법, 인터프리터를 보완하기 위한 JIT 컴파일러와 AOTC 기술, 다양한 프로파일 지원을 위한 자바 API 확장과 경량화 기술 및 실시간 자바 기술에 대한 연구가 지속적으로 필요하다.

참고문헌

[1] Sun Microsystems, http://java.sun.com/j2me

[2] Tim Lindholm and Frank Yellin, "The Java Virtual Machine Specification", Addison-Wesley, 1999

[3] S.Liang,"The Java Native Interface, Programmer's Guide and Specification", Addison-Wesley, 1999.

[4] A.Beatty, K.Casey,D.Gregg and A.Nisbet, "An Optimized Java Interpreter for Connected Devices and Embedded Systems," ACM SAC, Mar. 2003

[5] D.F Bacon, "Thin Locks:Featherweight Synchro -nization for Java,"SIGPLAN, 1998.