제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

자바 가상 머신의 개발 및 활용

원희선*, 배유석*, 이지현**, 문경덕*
*한국전자통신연구원
** 충남대학교, 컴퓨터 과학과
e-mail : hswon@etri.re.kr

Development of Java Virtual Machine and its Use

Hee-Sun Won*, Yu-Suk Bae*, Ji-Hyun Lee*, Kyung-Duk Moon*
*Electronics and Telecommunications Research Institute
**Dept. of Computer Science, Chungnam National University

요 약

자바가상머신은 제한적인 자원을 갖는 내장형 시스템에서 기업용 서버에 이르기까지 다양한 플랫폼에서 사용되고 있으므로, 각 플랫폼과 응용분야에 최적화된 자바가상머신을 만드는 것은 중요하다. 본 논문에서는 자바가상머신 스펙 1.2 를 지원하도록 구현한 자바가상머신의 전체적인 구조와 주요 부분의 구현 내용을 설명하고, 다양한 플랫폼과 응용 분야에 따라 최적화된 자바가상머신을 생성하기 위하여 자바가상머신 내부의 핵심 데이터 구조와 모듈을 재구성하는 연구 방향에 대해서도 소개한다.

1. 서론

최근 휴대 전화 및 인터넷 서비스를 제공하는 다양한 내장형 장치들이 개발되고 있으며, 이와 같은 장치들은 기존의 PC 와는 다른 운영 환경을 제공한다. 이들 장치를 운용하기 위한 언어로써, 어떠한 플랫폼에도 의존적이지 않고 동작할 수 있는 자바를 많이 채택하고 있다. 자바 프로그램은 자바가상머신에 의해서 실행되며, 따라서 성능을 향상시키기 위해 자바가상머신을 효율적으로 구현하는 것은 매우 중요하다.

본 연구에서는 자바가상머신 스펙 1.2 를 기준으로 자바가상머신을 구현하였으며, 시스템의 주요 제한 자원이 되는 저장 장치와 연산 능력을 최적화하기 위해 가상머신의 내부 데이터 구조 및 수행 모듈을 적응적으로 변화시킬 수 있도록 하였다.

2. 개발 목적

향후 차세대 지능형 정보가전과 통신 기기의 시장은 급속히 성장할 것으로 예상되고 있다. 자바가상머신은 이러한 내장형 장치들의 핵심 기술이 될 자바의 실행 플랫폼으로써 자바가상머신에 대한 관심과 개발 경쟁이 국내외 적으로 가속화되고 있다.

비용의 지불 없이 사용할 수 있는 공개 자바가상머신들은 일반적으로 호스트 시스템 용으로써 연구/교육용으로는 활용이 가능하나 제품에 사용하기에는 보완하거나 개선해야 될 사항들이 많으며, 특히 내장형 시스템으로 이식하기 위해서는 구현된 자바가상머신의 설계와 코드의 상세한 파악이 필요하다. 또한 대부분의 공개 자바가상머신이 채택하고 있는 GPL 라이센스의 경우, 이를 사용한 소프트웨어 역시 같은 권리 하에 공개되어야 하므로 상업적으로 이용하기에는 제약이 있게 된다. Sun Microsystems 사는 자사가 개발한 자바가상머신을 관련 업체에 판매하고 있으나 이를 이용할 경우 막대한 라이센스 비용을 외국에 지불하게 된다.

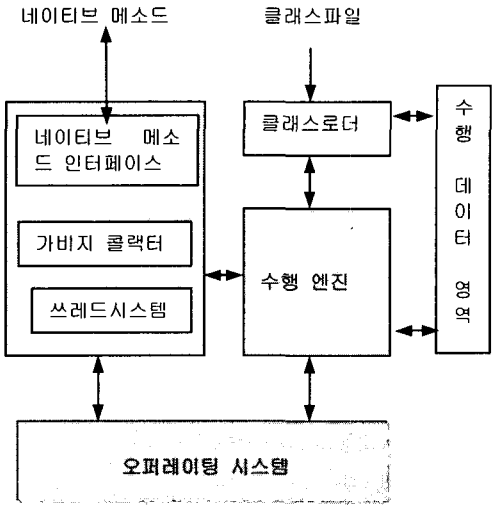
이러한 경우를 위해, 본 연구에서는 자바가상머신을 클린룸에서 구현하였으며, 기능 확장과 성능 개선에 대한 지속적인 연구 개발을 하여 다양한 스펙의 시스템에 최적화된 자바 실행 플랫폼의 개발을 지원하고자 한다.

3. 자바가상머신의 설계 및 구현

3.1. 자바가상머신의 구조

[그림 1]은 자바가상머신의 전체 구조를 보여준다. 자바가상머신은 수행중에 필요한 데이터를 저장하는 수행 데이터 영역, 자바 언어의 실행 코드인 클래스파일을 로드하여 사용이 용이한 정보로 가공한 후, 자바

가상머신의 실행 데이터 영역에 저장하는 클래스 로더와 바이트코드를 해석하여 자바가상머신을 동작시키는 핵심요소인 수행엔진등으로 나뉘며, 확장성과 이식성을 고려한 하부 구조에는 쓰레드 시스템, 메모리 재사용을 위한 가비지 콜렉터 및 자바이외의 언어로 구현된 네이티브 메소드를 사용하기 위한 네이티브 메소드 인터페이스(native method interface) 등이 있다.



[그림 1] 자바가상머신의 구조.

3.2. 자바가상머신의 구현

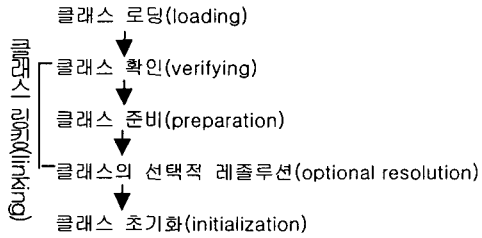
본 연구에서는 자바가상머신과 JDK1.1.8 수준의 자바 플랫폼을 위한 클래스라이브러리의 개발을 함께 진행하고 있으며, Linux 상에서 C 와 Java 언어로 개발하였다.

● 클래스로더

클래스에 관한 정보는 자바 언어가 컴파일된 클래스파일 형식으로 저장되어 있으며, 클래스로더를 통해 로딩되면서 자바가상머신에서 사용되는 클래스 구조체에 저장된다. 자바가상머신은 클래스 구조체를 기반으로 객체를 생성하고, 필요한 메소드들을 호출하여 수행한다. 클래스를 로딩하는 과정은 [그림 2]와 같다. 어느 단계까지 로딩하는지는 선택 사항하며, 해당 클래스의 필드나 메소드가 직접 쓰여질 때 해당 클래스가 사용 가능한 초기화 단계까지 수행되게 된다. 로딩된 클래스들은 클래스 관리를 위한 해쉬 테이블에서 관리된다.

● 레졸루션(resolution)

클래스파일 내에 정의되어 있는 컨스탄트 풀(constant pool)내의 심볼릭 링크를 실제 값으로 변환시켜주는 과정을 레졸루션이라고 한다. 본 구현에서는 가능한한 실제로 발생되는 참조 값에 대해서만 레졸루션이 수행되도록 하였으며, 레졸루션이 수행된 후, 컨스턴트 풀의 심볼릭 링크에 레졸루션된 결과 값을 바로 저장시켜서 다음 접근부터는 직접 사용이 가능하도록 설계하고 구현하였다. 특히, 필드의 경우에는 타입 클래스에 대한 정보를, 메소드의 경우에는 패러미터와 반환 타입 클래스들을 추출하고 이들 클래스에 대한 로딩되어 있지 않을 경우 이들에 대한 처리가 필요한데, 이러한 과정 역시 본 구현에서는 컨스턴트 풀을 로딩하면서 수행하고, 필드와 메소드의 .접근 플래그(access flag)에 새로운 플래그를 추가하여 이들의 상태를 표시하도록 하였다.



[그림 2] 클래스 로딩 단계.

● 수행 엔진

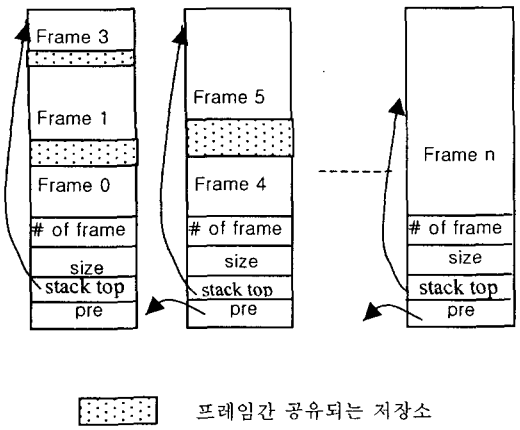
자바 바이트코드를 매번 해석하여 실행시키는 인터프리터 방식으로 구현하였다. 인터프리터 방식은 작고 이식성이 높은 장점이 있는 반면, 속도가 느린 단점이 있으므로, 해당 플랫폼의 기계어 변환 방식 등을 이용하는 다양한 성능 개선 방식을 적용할 계획이다.

● 자바 스택과 프레임 구조

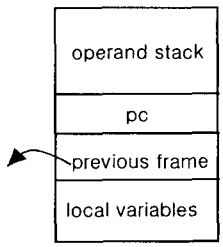
가상머신이 자바 스택을 직접 관리하도록 하였으며, 스택의 크기를 지정할 수 있고, 실행 중에 확장시킬 수도 있다. 메소드 호출시마다 스택에 생성되는 자바 프레임에는 지역변수, 피연산자 스택, 프레임 관리 정보등이 저장되어 있다. [그림 3(a)]와 [그림 3(b)]는 각각 간략화된 자바 스택과 프레임의 구조를 보여준다.

스택의 크기는 쓰레드 생성시에 정해지며, 프레임의 크기는 메소드에서 사용되는 지역변수와 오퍼런드 스택의 크기에 따라 가변적이다. 메소드를 호출하는 경우에, 호출한 메소드의 오퍼런드 스택에서 패러미터를 저장한 부분은 호출된 메소드의 지역변수 저장소로 재사용된다.

네이티브 메소드가 호출될 경우, 시스템에서 제공하는 네이티브 스택 상에 프레임이 생성되는데, 본 구현에서는 네이티브 프레임을 정의하고, 네이티브 메소드가 호출될 때, 동시에 자바 스택 상에도 네이티브 프레임을 생성하였다. 이렇게 함으로써, 메소드 수행 중에 오류나 예외가 발생하여 프레임을 꺼내거나(pop) 추적하는 과정이 필요할 경우에, 네이티브 메소드와 자바 메소드를 구분하지 않고 효율적으로 일관된 처리를 할 수 있다.



[그림 3a] 자바 스택 구조



[그림 3b] 자바 프레임 구조

● 쓰레드 시스템

자바가상머신이 멀티쓰레딩을 지원하기 위해서는 쓰레드들 사이의 스케쥴링 방법을 제공해야 한다. 본 구현에서는 기본적으로 쓰레드 스케줄러를 운영체제가 지원하는 것을 이용하도록 하였으며, Linux 의 pthread 인터페이스를 기반으로 하였다. 즉, 자바 쓰레드는 기반 OS 인 Linux 시스템의 쓰레드에 매핑되어 OS 에 의해 스케쥴링된다.

쓰레드를 구현하기 위한 자바가상머신 내부의 데이터구조는 자바의 쓰레드 클래스를 위한 자바 레벨의 쓰레드 데이터 구조와 기반 OS 에 의해 관리되는 OS 레벨 쓰레드 데이터 구조로 구분된다. 이와 같이 OS 에 영향을 받는 필드들을 자바 쓰레드 데이터 구조와 분리하고 자바 쓰레드 클래스의 각 메소드를 가능한한 네이티브 메소드로 구현함으로써 플랫폼 특성의 반영을 용이하게 하고 성능을 향상시키도록 하였다.

● 네이티브 메소드 인터페이스(Native Method Interface)

Sun 사에서 정의한 JNI(Java Native Interface) 명세에 포함된 함수들을 구현하였으며, 모든 JNI 함수는 JNIEnv 구조를 통해서 고정된 오프셋을 따라서 접근된다[7].

● 수행 데이터 영역(Runtime Data Area)

수행 데이터 영역은 한 개의 메소드 영역과 한 개의 힙(heap)으로 구성되며, 이들 영역은 자바가상머신 내의 모든 쓰레드에 의해 공유된다. 이 영역을 관리하기 위한 가비지 콜렉터(garbage collector)로써 three-color 모델을 적용한 Mark-Sweep 알고리즘을 구현한다.

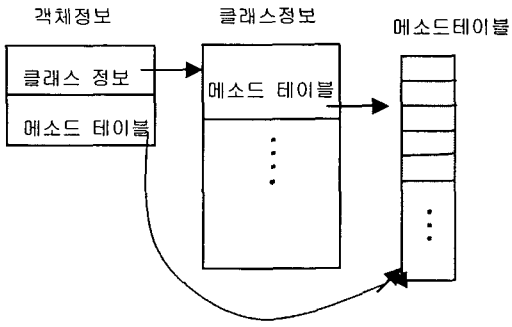
4. 적응적 자바가상머신에 대한 연구

본 절에서는 플랫폼과 응용 분야 등에 따라 제한자원이 될 수 있는 메모리와 연산 능력을 최대한 활용할 수 있도록, 자바가상머신의 내부 데이터 구조 및 모듈 중에서 적응적으로 변화 가능한 부분에 대해 알아본다.

4.1. 클래스와 객체의 데이터 구조

자바는 객체 기반 프로그래밍 언어로써 클래스와 객체의 표현 방식은 자바가상머신의 수행 속도 및 메모리 관리등에 많은 영향을 끼치게 된다.

연산 능력이 제한적인 상황에서는 가능한한 메모리 간접 접근을 최소화하도록 하여야 한다. 본 구현에서는 클래스마다 메소드 테이블을 생성하여 메소드 호출 과정을 빠르게 하였는데, 연산능력이 제한적인 경우에는 객체에도 메소드 테이블에 대한 포인터를 포함시켜 객체에서 참조되는 메소드 테이블로의 접근 속도를 높일 수 있다. [그림 4]는 이와 같은 경우에 객체, 클래스, 메소드 테이블의 관계를 나타낸다.



[그림 4] 클래스, 객체와 메소드테이블의 관계

위의 구조를 사용하게 되면, 프로그램의 종류에 따라 객체가 많이 생성되는 경우 메모리 요구량이 비례적으로 많아지지만, 내장형 장치의 경우, 프로그램의 복잡도가 크지 않고, 생성하는 객체수가 많지 않은 프로그램은 이러한 구조를 통하여 속도를 향상시키는 것이 이득으로 판단되고, 메모리 제약이 적은 데스크탑 시스템의 경우에도 이러한 설계가 속도 향상에 도움이 될 것으로 판단된다.

4.2. 자바 스택과 프레임 구조

현재 쓰레드 마다 스택을 할당하여 관리하도록 구

현되어 있으나, 필요한 스택의 크기를 가늠하기 어렵고 메모리 자원에 제약이 있는 시스템의 경우 미리 정해진 크기의 스택을 할당하여 사용하는 것은 비효율적이다. 따라서, 메소드가 호출 될 때마다 프레임에 필요한 메모리를 할당 받아 리스트로 연결된 스택 구조를 사용하도록 하여 메모리의 효율을 높인다.

4.3. 멀티쓰레드 환경에서의 클래스 관리

멀티쓰레드 환경에서는 클래스의 일관성을 유지하기 위해 클래스 로딩(레졸루션)이 수행될 때, 로딩된 클래스를 관리하는 해쉬 테이블과 클래스 등을 락킹한 후, 로딩(레졸루션)하려는 클래스가 이미 로딩(레졸루션)된 클래스인가 또는 다른 쓰레드에 의해 로딩(레졸루션)이 진행 중인가등에 대한 검사를 수행한다. 이러한 과정은 자바가상머신의 성능에 상당한 오버헤드를 초래하므로, 응용프로그램에서 실행될 쓰레드의 개수나 클래스의 동시 접근 빈도 등의 정보를 미리 파악하여 위와 관련된 클래스 로딩이나 레졸루션과 관련된 모듈에서 이 과정이 효율에 큰 영향을 미치지 않는 부분의 락킹과 검사 과정을 줄일 수 있다.

5. 결론

본 연구에서 개발 중인 자바가상머신은 자바 클래스 라이브러리와의 통합 테스트 및 실험 단계에 있다.

제 4 절에서 제안한 적응적 자바가상머신의 개발을 위해서는, 플랫폼 및 응용프로그램 등을 시스템 리소스에 따라 적절히 분류할 수 있어야 하고, 다양한 환경에서의 실험을 통하여 자바가상머신의 생성을 위한 최적의 파라미터를 추출해야 한다. 이 결과를 기반으로 수행 속도와 메모리 효율성 측면에서 다양한 시스템에 적응적인 자바가상머신을 할 수 있도록 지속적인 연구가 필요하다.

자바가상머신의 성능을 개선시키기 위하여 현재 수행엔진으로 사용되는 인터프리터를 자바 바이트 코드를 네이티브 코드로 변환하는 JIT(Just-In-Time) 컴파일러 등으로 대치하기 위한 연구가 필요하다.

실시간 자바 스펙을 구현하기 위한 실시간 쓰레드 스케쥴링과 실시간 가비지 콜렉터에 대한 연구가 함께 진행 중이다. 이를 지원하기 위해 현재 구현된 쓰레드 시스템에 비동기 이벤트에 대한 처리 모듈을 보완하여야 하고, 이를 지원할 수 있는 실시간 OS 에 대한 조사, 분석도 필요하다.

참고문헌

[1] 하영국, 임신영, 함호상, "정보가전 및 내장형 장치를 위한 Java 가술", ETRI 전자통신분석, vol.16, no.2, 2001, pp.31-39.
[2] Sun Microsystems, http:// java.sun.com
[3] Bill Venners, "Inside the Java Virtual Machine", McGraw Hill, 1999.
[4] Tim Lindholm and Frank Yellin, "The Java Virtual Machine Specification", Addison-Wesley, 1999.
[5] James Gosling, Bill Joy and Guy Steele, "The Java Language Specification", Addison-Wesley, 1996.
[6] http://www.kaffe.org
[7] Sheng Liang, "The Java Native Interface, Programmers's Guide and specification", Addison-Wesley, 1999.
[8] Patric Chan, Rosanna Lee and Douglas Kramer, "The Java Class Libraries", Addison-Wesley, 1998.
[9] http://java.sun.com/j2se/1.3/docs/api/index.html