자바가상기계에서의 동적 언어의 +
지원을 위한 데이터 표현

o
박 관 민*, 정 민 수*, 이 준 석**, 윤 성 순**, 윤 기 송**
*경남대학교 컴퓨터공학과
**한국전자통신연구원

The data representation to support dynamic languages on the Java virtual machine

Kwan-Min Park*, Min-Soo Jung*, Jun-Seok Lee**, Seong-soon Yoon**, Ki-Song Yoon
*Dept. of Computer Engineering, Kyungnam University
**Electronics and Telecommunications Research Institute

요 약

본 논문에서는 동적 언어의 데이터가 자바 가상 기계에 적합한 수행을 할 수 있도록 자바가상기계에 알맞도록 동적 언어에 대한 새로운 데이터 표현 방법을 제시한다. 그리고 자바의 다형성을 이용하여 새로운 클래스 라이브러리를 자바가상기계에 추가한다. 이러한 자바 가상 기계의 수행을 추적함으로써 동적 언어의 데이터 표현에 대한 유효성을 검증한다

1. 서 론

현재 자바가상기계의 주된 연구 방향은 소형화 및 속도 개선에 있다. 그로 인해 다양한 프로그래밍 언어를 지원을 하기위해 새로운 요소를 추가한다는 것은 잘못된 것이다. 하지만 자바가상기계에 자바가 아닌 다른 프로그래밍 언어의 원활한 수행을 위해서는 자바가상기계에 다른 프로그래밍 언어에 대한 자바가상기계 코드 세트(set)을 지정하고 확장된 라이브러리(library)를 통해 바이트코드(byre code)에 접근한다면 어떠한 프로그래밍 언어에 대해서도 수행가능할 것이다.

본 논문에서는 동적 언어에 대해 자바가상기계가 갖는 데이터와 연산의 능률적인 표현에 관한 문제점들을 제기하고, 이러한 문제를 해결하기 위해서 동적 데이터의 효과적인 표현과 그에 따른 요소들을 추가한다. 자바가상기계를 확장하여 클래스들에 대해서 효율성을 검증해 본다.

본 논문의 구성은 다음과 같다. 2 장에서는 데이터 표현의 문제를 파악하고 새로운 개념들을 소개, 문제 해결 방안, 그리고 자바가상기계에 새로운 개념의 추가를 제시한다.
3 장에서는 추가된 개념을 도입한 가상기계의 수행을 추적해 보고, 마지막으로 4 장에서 결론과 향후 연구 방안을 제시한다.

2. 데이터 표현의 문제와 해결 방안 제시

자바가상기계는 하나의 기계워드가 32 비트인 스택을 기반으로 동작하고 데이터의 타입(type)에 따라 일정한 크기의 데이터를 유지한다. 반면에 동적 언어의 데이터의 크기는 일정하게 표현되지 않고 수행에 따라 다양한 크기로 나타난다.

동적 언어의 데이터를 자바가상기계에 맞는 정적 타입(static type)의 데이터 표현으로의 변환은 버그(bug)를 수반하게 된다. 이러한 문제를 해결하기 위해서 동적 언어의 데이터를 일정한 크기로 균일하게 표현해야 하며 이를 위해 동적 언어의 데이터를 자바가상기계의 한 워드에 일치하도록 데이터를 표현한다.

또한, 균일한 데이터로 작성된 동적 언어의 데이터들은 새로운 클래스의 생성을 용이하게 한다. 동적 언어의 데이터가 문자 또는 불린(boolean)과 같은 자바가상기계의 한 기계워드보다 작은 값이라면 자바가상기계의 한 워드에 맞도록 채워지고 워드의 일부 비트는 데이터 값의 타입을 표현하는데 사용된다. 이러한 표현을 직접 서술자(immediate descriptor)라고 한다.

동적 프로그래밍 언어의 데이터가 배정도 부동소수점

+ 본 연구는 한국전자통신연구원의 연구 개발 지원금에 의해 작성되었습니다

수와 같이 자바가상기계의 워드보다 큰 경우에는 실제 데이터 값은 메모리에 저장하고 저장된 메모리의 위치를 가리키는 주소에 대한 값을 가진다. 이런 표현을 boxed 표현이라고 한다. 자바가상기계는 클래스 계층에 의해 다형성을 표현할 수 있다. 즉, 동적 프로그래밍 언어에서 서로 다른 타입의 객체는 자바 객체 클래스의 서브클래스(sub-class)로 표현할 수 있다. 만약 동적 언어의 30 비트 정수를 표현한다면 자바가상기계는 Java Int object 의 서브 클래스로 표현 가능하다. 또한 서로 다른 클래스의 객체를 균일하게 표현하기위해 boxed 표현을 이용하여 객체는 그 클래스와 정적변수의 설명을 메모리 영역에 대한 포인터에 의해 표현되는 것이다. 따라서 균일한 데이터 표현이 가능하면 동적 프로그래밍 언어의 다형성의 효과적인 수행이 가능하다.

앞에서 언급했듯이 직접 서술자의 일부 비트는 값의 타입을 설명하며 boxed 값을 나타내는 포인터와 구별하는 데 이용된다.

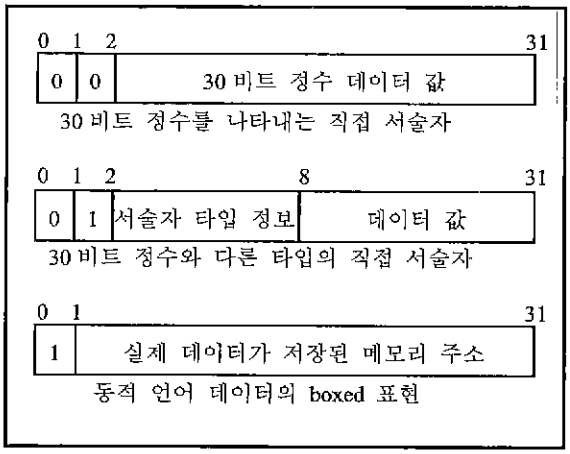

그림 1 직접 서술자와 boxed 데이터의 구별

그림 1 에서 직접 서술자에서 최초 비트 즉, 가장 낮은 순서 비트의 값이 0(zero)이면 직접 서술자를 나타내고 그 다음에 오는 비트는 30 비트 정수와 다른 여러 타입의 직접 서술자를 구분하는 데 사용된다.

만약 그 값이 0 이면 30 비트 정수를 나타내는 직접 서술자를 나타내고 그렇지 않은 값이면 다음 6 비트는 다른 타입의 서술자에 대한 타입 정보를 지원하는데 사용되며 나머지 24 비트는 실제 데이터에 대해 제공된다. 이렇게 직접 서술자의 경우 30 비트 정수에 대해서 태그(tag) 비트값을 00 으로 만드는 것에 의해 기존의 사칙연산을 원활한 수행이 가능할 것이다. 다시말해서, 단순 시프트(shift)로써 연산을 완벽하게 수행하기위해서인 것이다.

2.1 자바가상기계의 직접 서술자의 추가

직접 서술자와 boxed 표현을 이용한 가상기계는 동적 프로그래밍 언어에 대한 뒤떨어진 호환성을 보완, 변경하는 것이 가능하게 될 것이다. 먼저 boxed 자바 객체에 대한 모든 포인터들은 하나의 포인터를 갖도록 만들고 새로운 클래스로 ImmediateDescriptor 클래스를 생성하여 추가할 것이다. ImmediateDescriptor 클래스는 Object 클래스의 부클래스로 정확하게 31 비트를 가진다. 그리고 그 객체는 최소 비트가 0 인 기계워드로 표현되고 이는 보조기억장소(backing storage)를 요구하지 않고 boxed 객체와 구별된다. 자바가상기계에 이러한 서술자 클래스를 추가하면 동적 프로그램밍 언어에 한정된 직접 서술자이므로 다른 프로그램과의 충돌은 없을 것이다. 그리고 boxed 값으로 표현된 경우 메모리에 값에 대한 포인터는 자바객체에 대한 method lookup 과 유사하게 수행되지만 기존 자바가상기계는 method lookup 을 수행할 때 method table 을 통해 검증을 수행함으로써 boxed 로 표현된 값과 구별될 것이다. 또한, boxed 값에 대한 보안과 검증을 위한 추가도 필요하게 될 것이다.

3. 동적 데이터 표현의 수행

동적 언어의 데이터를 자바가상기계에서 수행하기 위해서는 먼저 동적 언어를 컴파일(compile)한 후 바이터 코드 생성기(bytecode generator)를 통해서 자바가상기계에서 수행 가능한 클래스를 생성하여야 한다. 이러한 클래스들을 생성에 직접 서술자와 boxed 데이터 표현을 적용하여 클래스를 생성하여 자바가상기계의 새로운 클래스 라이브러리로 추가한다. 자바가상기계의 구조는 클래스 로더 서브시스템(class loader subsystem),과 런타임 데이터 영역(runtime data areas) 그리고 실행엔진(excution engine)으로 구성되어 지고 클래스 로더 서브시스템(class loader subsystem)의 의해 클래스 파일을 런타임 데이터 영역에 적재한다.

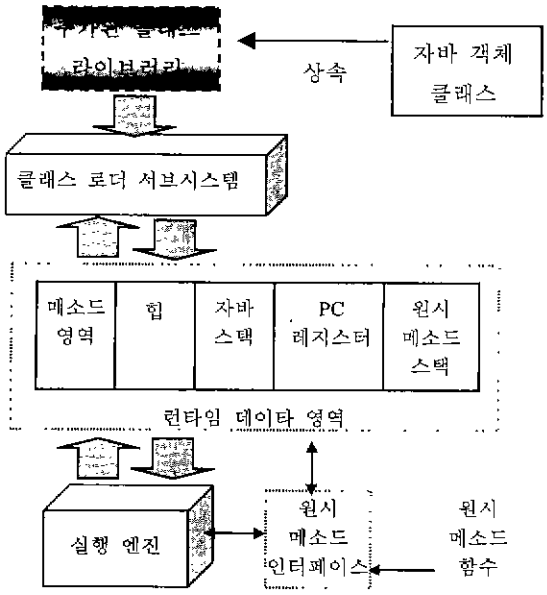

그림 2 자바가상기계에 클래스 라이브러리 추가

추가된 ImmediateDescriptor 클래스 파일은 데이터의 크기가 32 비트이므로 4 개의 8 비트 바이트를 읽음으로써 구성되고 자바 클래스 파일에 타입, 이름, 개수(count)의 정보를 순차적으로 담게 된다.

이러한 클래스 파일을 그림 2 와 같이 클래스 로더 서브시스템(class loader subsystem)을 통해 런타임 데이터 영역(runtime data areas)으로 ImmediateDescriptor 클래스를 적재한다. ImmediateDescriptor 클래스 파일의 가장 낮은 순서 비트의 값의 다음에 오는 비트 값으로 타입에 대한 정보를 통해서 메쏘드 영역에서 타입의 정보를 저장된다. 30 비트 정수를 갖는 ImmediateDescriptor 클래스의 경우 직접서술자의 낮은 순서의 2 비트를 통해서 정수 타입임을 구분하고 자바 객체 클래스로부터 상속되어지도록 클래스 파일을 정의한다.

ImmediateDescriptor 클래스는 타입을 나타내는 태그 비트의 정보에 따라 자바 객체 클래스의 상속을 통해서 새로운 타입의 객체들을 생성하고 그 상태 비트들의 정보를 통해 부모 클래스를 지정한다.

즉, ImmediateDescriptor 의 태그 비트의 값이 "00"이면 한 워드보다 작은 정수 데이터를 나타내게 되고 java Int class 를 부모 클래스로 갖는 서브클래스가 된다. ImmediateDescriptor 클래스의 객체는 메쏘드에 대한 포인터를 사용하지 않고 객체에 대한 메쏘드 테이블을 구성하며 이 테이블은 기존의 자바 객체 테이블과 구분되어야 할 것이다.

ImmediateDescriptor 클래스가 아닌 boxed 표현에 의한 데이터를 생각해 보자. ImmediateDescriptor 클래스의 객체가 아닌 자바 객체 클래스의 서브 클래스로 정의된 것은 boxed 표현의 객체임을 보증하게 된다.

boxed 객체는 표준 자바의 메쏘드 룩업(method lookup)을 사용할 수 있다. boxed 표현을 이용한 객체의 표현은 객체의 클래스와 정적 변수의 설명을 가지는 메모리 영역의 포인터를 가지게 된다. 따라서 일종의 레퍼런스 타입이라고 할 수 있다.

클래스 로더 서브 시스템은 boxed 로 표현된 객체의 인터페이스를 인식하여 메모리에 저장된 실제 데이터 값에 대한 클래스를 메쏘드 룩업에 의해 타입에 대한 정보를 찾고 그것을 런타임 데이터 영역의 메소드 영역에 저장하고 힙에 저장되는 객체는 객체의 인스턴스 데이터에 접근하기 위하여 객체의 클래스를 가르키는 포인터를 가지게 됨으로써 자바가상기계에 적재될 수 있다.

4. 결론 및 향후 연구 방향

본 논문에서 제시된 동적 언어의 데이터를 바이트코드로 인코딩(incoding)하여 새로운 클래스 라이브러리를 추가함으로써 효과적으로 동적 언어의 데이터 표현이 가능하고 다른 언어를 지원하도록 하는 가상기계의 진정한 목표를 이루게 해 줄 것이다. 하지만 자바가상기계의 크기는 더 크게 되는 문제점을 안고 있고 이를 소형화하는 기술이 요구된다. 그리고, 이러한 데이터 표현을 능률적으로 사용하기위해서 새로운 명령어 세트가 추가적으로 요구되며 이에 대한 검증이 필요하다.

참 고 문 헌

1. Peter Lee(editor). "Topics in Advanced Language Implementation.", MIT Press,(1991)
2. Olin Shivers "Supporting dynamic languages on the Java virtual machine", Dynamics Objects Workshop(1996)
3. A. Taivalsaari, "Implementation a Java Virtual Machine in the java programming Language ",SUN Lab, (1997)
4. B. Venners, Inside Java Virtual Machine, McGraw-Hill, (1997)
5. J. Gosling,,. "Java Language Specification", Addison-Wesley(1997)
6. T. Lindholm and F. Yellin, "The Java Virtual Machine Specification" ADDISON-WESLEY (1997)
7. http://wilson.ai.mit.edu/java-vm?37/ MIT AI Lab Home Page
8. http://java.sun.com/, Sun Microsystem, Java Home Page