재목적 기술을 이용한 가상기계의 탑재에 관한 연구

고광만, 유재민°
상지대학교 컴퓨터정보공학부
{kkman, comike°}@sangji.ac.kr

A Study on the Adapting of the Virtual Machine using Retargetable Techniques

Kwangman Ko, Jaemin Yoo°
School of Computer and Information, SangJi University

요 약

본 논문에서는 컴파일러 개발 시에 적용되었던 재목적 기술을 응용하여 다양한 플랫폼에 가상기계를 보다 원활히 탑재하기 위한 가상기계의 자동화 탑재 기술을 제안하고 이를 구현한다. 이를 위해, 가상기계를 플랫폼 독립적인 가상기계 핵심(core) 부분과 플랫폼 의존적인 부분으로 재구성한 후 다음과 같은 세가지 부분을 설계하고 구현한다. 첫째, 플랫폼 의존적인 부분을 정형화된 방법으로 기술할 수 있는 플랫폼 디스크립션을 설계한다. 둘째, 설계된 플랫폼 디스크립션을 입력으로 받아 최적의 플랫폼 정보를 생성할 수 있는 탑재 정보 생성기를 구현한다. 마지막으로 탑재 정보 생성기의 출력과 가상기계의 핵심 부분을 결합하는 가상기계 생성기를 개발한다.

1. 서 론

최근 인터넷 및 무선 통신 기술이 급속도로 발전되고 응용 분야가 확대되면서 응용 소프트웨어를 프로세서나 운영체제와 같은 플랫폼에 의존하지 않고 독립적으로 실행할 수 있는 기술에 대한 연구와 이를 지원할 수 있는 실행시간 환경 개발에 관한 연구가 국내외적으로 지속적으로 진행되고 있다. 휴대폰의 모바일 플랫폼, PDA, 디지털 TV, 셋톱박스, DMB 등과 같은 임베디드 시스템에 가상기계가 탑재되고 있지만 다양한 임베디드 시스템에 가상기계를 효율적으로 탑재할 수 있는 자동화 기술은 그 필요성에도 불구하고 아직은 국내외적으로 이에 대한 연구와 기술이 미흡한 실정이다.

임베디드 시스템과 같은 다양한 플랫폼에 가상기계를 탑재하기 위해서는 가상기계 개발 시에 특정 플랫폼의 특성을 고려하여 가상기계를 설계하고 구현해야 한다. 즉, 가상기계를 플랫폼 독립적인 부분과 플랫폼 의존적인 부분으로 구분하여 개발해야 하는데 현재까지 개발된 가상기계는 이러한 요소를 고려하지 않고 개발되고 있으며 최근에 이를 보완할 수 연구가 시작되고 있다. 따라서 가상기계를 특정 플랫폼에 탑재하기 위해서는 개발된 가상기계 소스를 재작성하거나 처음부터 플랫폼 특성에 의존적인 가상기계를 다시 개발해야 하는 단점을 가지고 있다. 특히, 앞으로 가상기계를 임베디드 시스템, 유비쿼터스 컴퓨팅과 같은 다양한 환경에 탑재하려는 수요는 여러 분야에서 폭발적으로 증가될 것으로 판단되므로 정형화된 방법으로 보다 쉽게 개발된 가상기계를 탑재할 수 있는 기초 연구 및 이를 지원할 수 있는 다양한 연구가 반드시 필요하다. 또한 이러한 가상기계를 다양한 플랫폼에 정형화된 기법을 도입하여 탑재할 수 있는 자동화 탑재 기술은 국내외적인 수요에 비해 연구 및 개발 자원이 부족한 관계로 향후에 국제 경쟁력 확보에 중요한 역할을 할 수 있는 원천 기술로서 그 개발 필요성이 매우 중요한다. 현재 가상기계의 개발 및 탑재 기술은 국내에서는 휴대폰에만 한정되어 있고 그나마 일부는 선진국에서 로열티를 내고 사용하는 입장이다. 앞으로 성장이 예상되는 임베디드 시스템, 유비쿼터스 컴퓨팅 시장에서 선진국의 기술 종속성을 피하고 독립적인 기술을 확보하기 위해서는 장기적인 국가과제로써 추진하고 개발해야 하는 중요한 기술이다.

본 연구의 최종 목표는 "가상기계의 필요성과 그 수요가 폭발적으로 증가하고 있는 상황에서 휴대폰의 모바일 플랫폼, PDA, 디지털 TV, 셋톱박스, DMB 등과 같은 임베디드 시스템, 유비쿼터스 컴퓨팅 환경에 가상기계를 보다 용이하게 탑재할 수 있도록 가상기계의 자동화 탑재 모델을 제안하고 이를 구현"하고자 한다. 이를 위해 다음과 같은 3가지 세부 목표를 달성한다. 첫째, 가상기계를 플랫폼 의존적인 부분과 플랫폼 독립적인 부분으로 구분한 후 플랫폼 의존적인 정보를 정형화된 기법으로 기술하는 플랫폼 디스크립션(platform description)을 설계한다. 둘째, 탑재 정보 생성기는 플랫폼 디스크립션을 입력으로 받아 가상기계의 코어에서 참조할 수 있는 플랫폼에 대한 최적의 정보를 생성한다. 셋째, 가상기계 자동화 생성기는 탑재 정보 생성기에 의해 생성된 플랫폼 정보와 가상기계의 코어 부분을 결합하여 가상기계를 생성한다. 본 연구를 통해 플랫폼에 가상기계를 탑재할 경우 가상기계의 코어 부분의 수정 없이 플랫폼 디스크립

션의 변경을 통해 다양한 시스템에 가상기계를 보다 용이하게 탑재할 수 있는 특성이 있다.

2. 관련연구

2.1 재목적 기술에 관한 연구

재목적 기술은 컴파일러 후단부의 핵심인 목적코드 생성 시스템을 자동 생성하기 위한 방법으로서 컴파일러 후단부에서 다양한 목적기계에 대해 원시 프로그램의 중간 코드를 목적기계 코드로 변환하는 과정을 정형화된 방법을 통하여 자동적으로 구성한다. 이러한 재목적 기술을 적용하여 목적코드 생성 과정을 목적기계 의존적인 부분과 목적기계 독립적인 부분으로 나누어 정형화하는데 목적을 두고 있으며 템플리트 매칭 또는 테이블을 이용한 방법에 의해 목적기계 코드를 생성한다. 목적기계 의존적인 부분에서는 목적기계 표현 테이블을 입력으로 받아 목적기계 정보 및 중간 코드에 대한 목적기계 코드 생성 정보를 저장하고 있는 테이블을 생성한다. 이러한 기술은 컴파일러 개발 시에 목적기계가 변경되는 경우 목적기계 정보 및 중간 코드에 대한 목적기계 코드 생성 정보를 기술하고 있는 목적기계 표현 테이블(Machine Description Table)을 재구성함으로서 목적기계 코드 생성기를 보다 손쉽게 개발할 수 있는 장점을 가지고 있다.

2.2 가상기계에 관한 연구

가상기계란 프로세서, 운영체제 등이 바뀌더라도 응용프로그램을 변경하지 않고 사용할 수 있는 기술로 특히, 임베디드 시스템을 위한 가상기계 기술은 모바일 디바이스와 디지털 TV 등에 탑재할 수 있는 핵심기술로 다운로드 솔루션에서는 꼭 필요한 소프트웨어 기술이다. 또한, 응용프로그램의 개발 방법 및 실행 방법은 크게 네이티브 어플리케이션과 가상기계 어플리케이션으로 나눌 수 있으며 전자는 이제까지 사용하던 방법으로 실행 속도면에서는 탁월한 장점을 갖는다. 그러나, 플랫폼이 바뀌면 모든 응용프로그램을 변경해야 할뿐만 아니라 심지어는 사용할 수 없게 된다. 이러한 단점을 극복하기 위해서 가상기계를 탑재하여 응용프로그램을 실행시켜주는 가상기계 솔루션이 등장하였으며 특히, 임베디드 시스템에서는 프로세서 및 운영체제의 다양성과 잦은 변경으로 인하여 가상기계를 이용하는 것이 적합하고 더욱이 다운로드 솔루션에서는 가상기계 어플리케이션이 타당한 방법이다. 현재까지 수많은 가상기계가 개발되어 사용되고 있으며 대표적으로 JVM, KVM, Waba VM 등이 있으며 응용 목적에 따라 소규모 장치에서 활용될 수 있는 가상기계도 개발되고 있다.

2.3 가상기계 탑재에 관한 연구

가상기계 구현은 가상기계의 핵심 동작을 구현하는 부분과 특정 플랫폼과 인터페이스 부분을 담당하는 어댑터(adapter) 기능을 구현하는 부분으로 크게 구분된다. 따라서 가상기계의 핵심 부분은 플랫폼에 독립적인 부분으로서 특정 플랫폼에 탑재하더라도 코드의 수정 없이 구현된다. 어댑터는 가상기계가 탑재되는 플랫폼의 특성에 의존하여 구현되는 부분으로서 가상기계의 핵심 부분이 플랫폼에 독립적으로 작동을 할 수 있도록 플랫폼의 API를 이용해서 구현된다. 가상기계를 다른 플랫폼에 탑재할 때마다 새로 설계 하고 만드는 작업은 상당히 비효율적인 방법이며 가상기계를 만들 때는 이식성을 충분이 고려해서 설계 및 제작 되어야 한다[18][21].

가상기계 코어 부분은 실행 파일 포맷을 로딩하는 로더와 실질적인 실행을 담당하는 인터프리터로 구성되어 있다. 가상기계를 플랫폼에 탑재하기 위한 어댑터는 플랫폼과 가상기계 코어간의 인터페이스 역할을 하며 세부적인 구조는 첫째, vmCore에서 플랫폼 독립적이기 위해 별로 요구하는 자료 형을 맞추어 주는 정의 부분, 둘째, vmCore의 입출력과 실행을 위한 인터페이스를 제공하는 주 어댑터 부분, 셋째, 운영체제와 가상기계간에 실질적인 인터페이스를 담당하는 네이티브 코드 부분으로 구성되어 있으며 네이티브 코드 부분의 구현이 가장 큰 비중을 차지한다. 이러한 네이티브 함수는 네이티브 함수 테이블(native function table)을 통해 가상 함수와 매핑될 수 있으며 가상 함수를 추가하고 네이티브 함수 테이블과 네이티브 코드부를 재구성함으로써 가상기계가 지원하는 기능을 계속해서 확장시킬 수 있다. 또한 가상기계의 속도가 최우선 된다면 거의 모든 API들을 네이티브 함수로 구현함으로써 크기는 커지지만 상당히 빠른 실행 속도를 얻을 수도 있다.

가상기계의 로더에서 파일을 메모리에 적재하는 중에 가상함수를 만나게 되면 파일에 있는 코드 대신 네이티브 코드를 저장하고 있는 네이티브 함수의 포인터를 적재하게 되며 가상 함수와 네이티브 함수의 매핑 관계는 네이티브 함수 테이블에 명시 되어있다. 네이티브 함수 테이블을 통해 가상함수와 1:1 매핑하는 함수를 네이티브 함수라 부르며 네이티브 함수에서 실질적인 시스템 호출이 일어나게 되는 것으로 가상기계의 네이티브 코드 실행을 위한 부분은 네이티브 함수와 네이티브 함수 테이블로 나누어진다. 네이티브 함수는 가상기계 상의 언어에서 가상 함수를 대신 하는 가상기계 내부에 있는 함수로서 시스템 함수 호출뿐만 아니라 다른 나머지 함수들도 가상 함수를 통해 네이티브 함수를 호출하는 형태로 구현될 수 가있으며 네이티브 함수에 의존도가 높아질수록 가상기계의 속도는 증가하지만 부피는 커지게 된다. 그러므로 반드시 필요한 시스템 호출 함수만 네이티브 함수로 구현하는 것이 보편적이며 이를 통하여 가상기계가 특정 플랫폼에 탑재 될 수 있다. 네이티브 코드로 구현해야 할 부분들은 프로그램이 수행되기 위해서 시스템에 종속적일 수밖에 없는 부분으로서 기본적인 입출력, Network, GUI, 디바이스 요청 등과 같은 부분을 수행해야 한다.

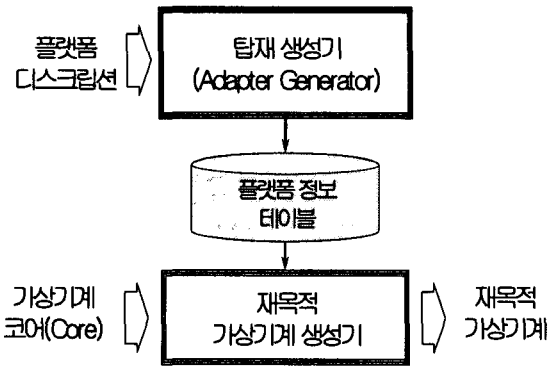
3. 가상기계 탑재 생성기

3.1 개발 모델 설계

특정 플랫폼에 가상기계를 탑재하기 위해서는 탑재할 플랫폼에 관한 기본 지식과 자료가 필요하며 해당 플랫

품에서 제공하는 API를 이용하여 프로그램을 작성할 수 있는 능력을 필요로 한다. 가상기계에서 제공하는 API들 중에서 기본적인 연산에 관련된 것들은 가상기계 코어(VM Core)의 인터프리터에서 수행하지만 입출력, GUI, 통신에 관한 부분은 해당 플랫폼의 API를 이용해서 작업을 수행한다. 따라서 가상기계를 특정 플랫폼에 탑재하기 위해서는 플랫폼의 다양한 특성인 자료형, 사용가능한 자원, 소프트웨어적으로 지원하는 함수, 파일 관리 기법 등을 충분히 숙지하고 있어야 한다.

본 연구에서는 [그림 1]과 같이 재목적 기법을 적용하여 가상기계 개발을 플랫폼 독립적인 부분과 플랫폼 의존적인 부분으로 구분한 후 플랫폼 의존적인 부분을 정형화된 방법으로 기술한 후 이를 위한 탑재 생성기를 개발한다. 또한 가상기계 생성기는 개발된 가상기계 핵심 부분과 탑재 생성기 정보를 참조하여 특정 플랫폼의 탑재에 적합한 가상기계를 자동 생성한다.



[그림 1] 가상기계의 재목적 탑재 모델

3.2 플랫폼 디스크립션 및 탑재 생성기

플랫폼 디스크립션은 가상기계를 탑재하려는 플랫폼 정보를 본 연구에서 제안하는 정형화된 방법에 따라 기술된다. 플랫폼 디스크립션의 설계를 위해 컴파일러 자동화 도구인 ACK에서 목적기계 및 코드 생성 정보를 정형화된 방법으로 기술하였던 기존 연구를 기반으로 한다.플랫폼 디스크립션에 기술되는 정보는 목적기계 정보, 스택 정보, 코드 생성 정보로 크게 3부분으로 나누어 기술하였다.

탑재 정보 생성기는 플랫폼 디스크립션을 입력으로 받아 플랫폼 정보 테이블을 생성하는 부분으로 플랫폼 디스크립션에 대한 어휘 분석, 구문 분석 단계 등을 거쳐 최적의 플랫폼 정보를 생성한다. 어휘 분석과 구문 분석의 결과는 추상화 구문 트리에 표현하였으며 추상화 구문 트리를 순회하면서 최적의 테이블 구조를 생성한다.

4. 결론 및 향후 연구

본 논문에서는 컴파일러 개발 시에 적용되었던 재목적 기술을 응용하여 다양한 플랫폼에 가상기계를 보다 원활히 탑재하기 위한 가상기계의 자동화 탑재 기술을 제안하고 이를 구현하였다. 이를 위해, 첫째, 플랫폼 의존적인 부분을 정형화된 방법으로 기술할 수 있는 플랫폼 디스크립션을 컴파일러 자동화 도구인 ACK의 목적기계 표현 테이블을 기반으로 설계하였다. 둘째, 설계된 플랫폼 디스크립션을 입력으로 받아 최적의 플랫폼 정보를 생성할 수 있는 탑재 생성기를 구현했다. 생성된 정보를 실질적인 가상기계 생성을 위해 필요한 정보를 최적의 구조를 갖도록 설계하고 구현하였다. 마지막으로 탑재 정보 생성기의 출력과 가상기계의 핵심 부분을 결합하는 가상기계 생성기를 개발하였다. 현재까지 설계된 모델을 기반으로 다양한 플랫폼에 탑재될 수 있도록 개발 연구를 진행하고 있다. 향후에는 실질적인 상용 플랫폼에 탑재할 수 있도록 보완할 예정이다.

5. 참고문헌

[1]R. G. G. Cattell, "Automatic Derivation of Code Generators from Machine Descriptions," ACM TOPLAS, Vol. 2, No. 2, pp.173-190, Apr., 1980.

[2]Andrew S. Tanenbaum, Hans van Staveren and Johan W. Stevenson, "A Practical Tool Kit for Making Portable Compilers," CACM, Vol. 26, No. 9, Sep., 1983.

[31]Mahadevan Ganapathi, Charles N.Fisher and John L. Hennessy, "Retargetable Compiler Code Generation," ACM Computing Surveys, Vol.14, No.4, 1982.

[4]Wen-mei W. Hwu, "Java Bytecode to Native Code Translation: The Caffeine Prototype and Preliminary Results", The Proceeding of the 29th Annual International Symposium on Microarchitecture, Dec., 1996.

[5]James S. Miller, The Common Language Infrastructure Annotated Standard, Addison Wesley, 2003.

[6]Joshua Engel, Programming for the Java Virtual Machine, Addison-Wesley, 2000.

[7]Tim Lindholm, Frank Yellin, The Java Virtual Machine Specification, 2/E, Addison Wesley, 1999.

[8]Jon Meyer, Troy Downing, Java Virtual Machine, O'Reilly & Associates, 1997.