유비쿼터스 환경을 위한 컴포넌트 기반 자바가상머신의 설계

윤승환° 이승룡
경희대학교 컴퓨터공학과
{nataxia°, sylee}@oslab.khu.ac.kr

Design of Java Virtual Machine Based on Component Model for Ubiquitous Environment

Seungwhan Youn°, Sungyoung Lee
Dept. of Computer Engineering, Kyung Hee University

요 약

다양한 환경과 급변하는 시장의 적시성 요구사항(time-to-market)을 반영하기 위해서는 재구성 가능한 컴포넌트 기법을 통한 개발이 소프트웨어 공학 측면에서 많이 연구되어 왔다. 하지만 이러한 컴포넌트 기법은 여러 장점에도 불구하고, 성능 저하라는 측면 하나만으로 시스템 및 플랫폼 구축분야에서 소외당해 온 것이 사실이다. 그러나 최근 유비쿼터스 및 내장형 시스템이 대두됨에 따라 이러한 판도에 많은 변화가 예상되고 있다. 이는 현재 활발히 연구·개발되고 있는 유비쿼터스 및 내장형 시스템 기반의 어플리케이션은 획일화 되지 않은 환경 하에 있는 물리적 환경의 특성상 일정한 플랫폼의 형태에서 벗어나 매우 다양한 환경에 융통성 있게 적용되어야 하고, 이에 상위 프로그램에 투명성을 제공할 수 있는 플랫폼이 필요로 하게 되었다. 따라서 본 논문에서는 내장형 시스템 및 유비쿼터스 환경에 적합한 미들웨어 플랫폼을 구성하기 위해, 재구성과 융통성을 제공하기 위한 컴포넌트기반 소프트웨어 개발 방법을 적용한 새로운 자바 가상머신 설계기법을 제안한다.

1. 서 론

최근 임베디드 시스템 및 유비쿼터스 시스템은 그 확장성 및 범용성을 지원하기 위해 기본적으로 미들웨어 플랫폼을 기반으로 하는 연구가 활발히 진행되고 있다. 이는 유비쿼터스 및 미들웨어 환경이 하드웨어 또는 소프트웨어적으로 매우 다양한 계층에 적용되기 때문이다. 또한 '무어의 법칙'에 따른 하드웨어의 발전과 이에 필적하는 소프트웨어 개발 시간의 단축은 과거의 단편적인 개발 구조에서 벗어나 보다 효율적인 방안을 모색하지 않을 수 없게 되었다. 따라서 개발자에게 매우 열악한 환경에서부터 범용 시스템에 이르기 까지 매우 다양화된 시스템에 구애받지 않고 거의 동일한 개발 및 실행환경을 지원할 수 있는 미들웨어 플랫폼의 필요성이 대두되고 있다.

자바는 자바 가상머신이라는 미들웨어를 통해 여러 플랫폼에서 동일한 동작을 하는 바이너리 포맷을 지원하는 언어이다. 따라서 자바는 단순히 하나의 언어라기보다는 하나의 플랫폼의 형태라 할 수 있다. 이러한 자바의 특성상 자바를 구동하여 주는 가상머신은 매우 중요하며, 근래에 와서는 이동통신 및 매우 다양한 플랫폼에서 유사한 형태의 미들웨어 가상머신들이 선보일 만큼 보편화된 기술이라고 할 수 있다.

자바 플랫폼은 자바 어플리케이션을 지원하는 API군, 이를 규정하는 프로파일과, 시스템과 자바 어플리케이션의 독립성을 확보하는 가상머신으로 나눌 수 있다. 현재 기존의 자바 가상머신은 주로 PC 및 대형 프레임 워크를 위주로 사용되어 왔으며, 휴대형 단말기 및 PDA등의 발전으로 인해 점차 소형화된 기기에서도 그 사용이 증가되고 있는 추세이다. 그러나 현재 자바를 지원하는 플랫폼을 구축하는 것은 또 하나의 새로운 운영체제를 구축하는 것과 마찬가지로 매우 어렵고 힘든 일이다.

따라서 본 논문에서는 현재까지의 일관적인 자바 가상머신 및 자바 플랫폼의 개발을 컴포넌트 기반으로 확장하여 보다 다양한 시스템에 응용시킬 수 있도록 기존의 컴포넌트 모델을 적용하여 전체를 관리하는 프레임워크와 각 세부적인 컴포넌트의 구성을 가진 자바 가상머신 구조를 제안하고자 한다.

일반적인 컴포넌트 모델은 자바 가상머신과 같은 시스템에 밀접한 프로그램을 구현하는데 많은 문제점을 야기 시킨다. 그 중 프로그램의 성능저하에 대한 부분은 아직도 많은 논란이 되고 있는 부분이다. 본 논문에서 적용하는 PBO모델은 실제 매우 소규모의 하드웨어에서 사용하는 모델로 실시간 프로그램에 응용이 될 정도로 수행 능력이 뛰어나다. 그러나 PBO는 매우 단순한 형태의 컴포넌트 모델을 가지고 있고, 간단한 동작 위주의 로보틱스에 응용되어 왔기 때문에, 복잡한 컴퓨팅을 구현하는데는 무리가 있다. 따라서 본 논문에서는 기존의 PBO를 기본 모델로 자바 가상머신의 설계에 맞도록 몇몇 부분을 수정 보완하여 적용하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로서 자바 가상머신과 컴포넌트 모델인 PBO모델에 대해 소개하고, 3장에서는 본 논문에서 제안하는 컴포넌트 기반 자바 가상머신의 설계에 대해 설명을 하며, 4장에서는 결론을 맺으며 향후 연구에 대해 언급 한다

2. 관련연구

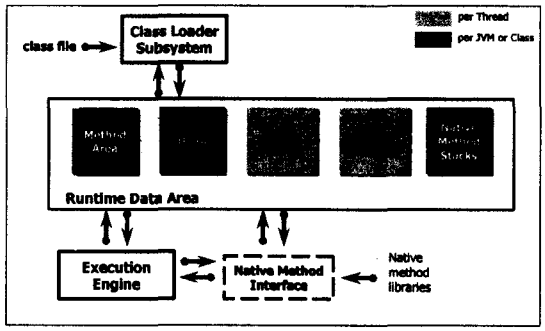
2.1 자바 가상머신

자바 플랫폼은 자바 어플리케이션을 지원하는 API군, 이를 규정하는 프로파일과, 시스템과 자바 어플리케이션의 독립성을 확보하는 가상머신으로 나눌 수 있다. 이 중 자바 가상머신은 어플리케이션을 구동하여 주는 미들웨어로서, 클래스를 로딩하고, 메모리를 관리 하며, 결과적으로 자바 프로그램을 실행하여 주는 직접적인 역할을 한다[1,2]. 따라서 자바 가상머신은 하드웨어적인 기능(CPU적인 기능)과 소프트 웨어 운영의 기능(메모리 관리, 쓰레드 관리 등)을 동시에 가지고 있어, 컴퓨터

※ 이 논문은 2003년도 한국과학재단 목적기초연구(과제번호: R01- 000-00357-0)의 연구비에 의하여 연구되었음.

시스템의 핵심을 응축한 기술이라 볼 수 있다.

기본적인 자바 가상머신의 구조는 아래 [그림 1]과 같다.

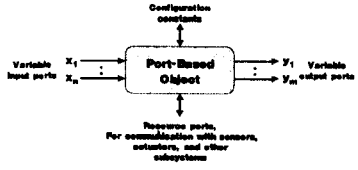


[그림 1] 자바 가상머신의 기본구조

현재 자바 가상머신은 운영체제와 매우 밀접하게 연관 되어 있다는 특성에 의해 효율성 위주의 구성으로 되어 있었다. 그러나 앞으로 보다 다양하고 복잡한 환경을 필요로 하는 유비쿼터스 환경에 적용하려면 이와 같은 일률적인 구성으로는 유동적인 플랫폼 구성이 불가능 하다. 결국 이는 컴포넌트 기법을 통하여 이를 극복할 수밖에 없다. 그러나 일반적인 소프트웨어 공학에서 사용하는 컴포넌트 모델은 플랫폼 구축에 사용할 수 있을 정도의 효율을 기대하기 힘들기 때문에, 실시간 로보틱스에서 사용되고 있는 매우 가볍고 간단한 PBO모델을 그 기본 모델로 적용하였다.

2.2 PBO(Port-Based Object)를 이용한 컴포넌트 기법

PBO[3,4]는 Carnegie Mellon Univ의 Advanced Manipulators Laboratory에서 개발하였으며, 도메인 컴포넌트의 개발에 기반하고 있다. PBO 모델의 기본은 독립적인 태스크이다. 이 태스크는 다른 컴포넌트와의 통신을 허용하지 않으며, 느슨하게 결합되어 재사용하기 쉽다. PBO의 설계 목적은 컴포넌트 사이의 동기화와 통신의 최소화이다. PBO의 데이터 흐름은 입출력 포트를 통하여 규정한다. 입력 포트를 통해 데이터 생산자에 대한 지식 없이도 가장 최근의 정보를 읽을 수 있으며, 다른 컴포넌트를 위한 유효한 정보를 만들기 원할 때 출력 포트에 저장할 수 있다. PBO에서는 컴포넌트를 더욱 유연성 있고 재사용할 수 있게 만들기 위해 매개변수 인터페이스(parameterization interface)가 제공되며, 이를 통해 서로 다른 행위가 단일 컴포넌트에 의해 구현이 가능하다. [그림 2]은 PBO를 나타낸다.



[그림 2] Port-Based Object

PBO로 정의된 객체들은 PBO Framework를 통해서 작업을 수행하게 된다. 즉, PBO Framework는 PBO 객체들의 작업수행을 제어하며 컴포넌트의 재구성을 위한 메카니즘을 제공한다.

이러한 PBO 모델을 적용하여 자바 가상머신을 설계할 경우, 가상머신의 컴포넌트화가 기능적 컴포넌트형 모델로 설계할 수 있다. 이러한 PBO의 특성은 어떤 컴포넌트의 구성 시에 관련된 컴포넌트의 접합이 함수의 형태를 띄어 접근하며, 이는 미리 설계되어 있는 포인터형 인터페이스를 통하게 된다. 이때 PBO 모델에 의하면, 각 컴포넌트들은 포트를 기반으로 자료를 주고받게 되어있는데, 이러한 부분은 마치 마이크로 커널의 통신형태와 비슷하여 자바 가상머신의 속도 저하의 요인이 될 수 있다. 따라서 이를 자바의 Runtime Data Area를 공유할 수 있도록 보완하였다. 또한 PBO는 실시간 하드웨어 조작을 위한 목적으로 설계되었기 때문에 가상머신 컴포넌트를 원활히 지원할 수 있도록 여러 부분을 보완하여 적용하였다.

2.3 유비쿼터스 환경

유비쿼터스는 라틴어에서 유래된 말로 "언제, 어디서나" "동시에 존재 한다"라는 뜻으로, 물이나 공기처럼 도처에 편재한 자연 상태를 의미한다. 이는 '언제, 어디서나, 어떤 기기를 이용해서라도 정보를 교환할 수 있는 미래의 컴퓨팅 환경을 한마디로 압축 한 것이다[5]. 실세계의 물리적 사물 및 환경의 전반에 걸쳐 사용자에게 컴퓨터의 겉모습을 드러내지 않고 융화된 모습, 공상 과학 영화에서 나오는 자동화되고 편리한 세상을 유비쿼터스 컴퓨팅 환경을 통해 구현할 수 있도록 고안되었다.

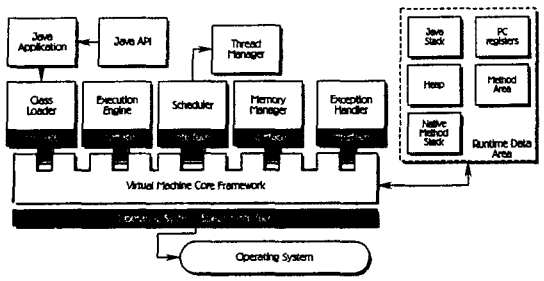
이러한 유비쿼터스 환경은 현재의 데스크탑 컴퓨터 위주의 환경과는 매우 다른 특징을 지닌다. 우선 물리적인 부분에서는 임베디드 시스템과 같이 일상 가전이나 제품 내에 컴퓨터가 삽입되기도 하고, 먼지정도의 극소 크기의 컴퓨터 센서로 구현되기도 한다. 따라서 이를 지원하는 소프트 웨어는 네트워크나 저장공간, 심지어는 전력 공급까지도 매우 다양한 형태의 물리적 제약을 유동적으로 극복해야만 한다. 어플리케이션 입장에서의 환경적인 역할을 하는 운영체제를 보면, 현재 유비쿼터스와 환경이 유사한 임베디드 업계의 운영체제처럼 그 제품의 성격에 따라 각기 다른 운영체제를 사용하고 있다. 따라서 앞으로 유비쿼터스 컴퓨팅 환경을 지원하기 위해서는 프로그래머가 여러 운영체제 및 물리적 환경에 맞는 프로그램을 여러 버전으로 만들거나, 운영체제 및 물리적 환경을 한 곳으로 잇는 미들웨어 기반의 어플리케이션으로 제작하는 방법이 있다.

3. 자바 가상머신의 설계

3.1 가상머신의 전체 구조

자바 가상머신은 간단히 보면, 자바 프로그램을 읽어서 실행하는 역할을 한다. 이를 위해서 자바 가상머신은 우선 자바 클래스 파일을 읽고 이를 해독하는 작업을 수행한다. 또한 클래스 정보를 토대로 이를 실행해 주며, 이 과정에 쓰레드 관리, 메모리 관리, 예외 관리와 같은 여러 작업들을 관리해 준다. 이는 자바 언어 자체의 성격에 의한 부분이 많은 영향을 미친다. 이러한 자바 가상머신을 각 수행 역할에 맞춰 각각의 수행 컴포넌트로 설계 하였다. 이들 컴포넌트는 매우 열악한 환경과 다양한 환경에서의 적용을 위해 실시간 로보틱스에서 응용되는 PBO 모델을 기반으로 설계 하였다.(그림 3 참조)

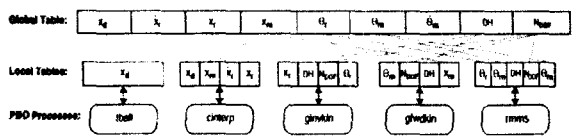
이러한 자바 가상머신의 구조는 가상머신이 다양한 플랫폼에 수정 및 컴파일 과정을 거치지 않고, 컴포넌트 재조립의 형식으로 간단히 포팅 가능하게 한다.



[그림 3] 컴포넌트 기반의 자바 가상머신의 구조

3.2 컴포넌트 모델의 적용

위의 [그림 3]에서와 같이 자바 가상머신의 전체구성은 클래스를 로딩하는 로더, 실행엔진, 쓰레드 스케쥴러, 메모리 관리자, 예외관리자와 같은 수행별 컴포넌트와 이를 통합하는 프레임워크의 조합으로 이루어져 있다. 이러한 각 컴포넌트들은 기본적으로 PBO모델을 기반으로 하고 있다. 그러나 일반적인 PBO에서 사용하는 포트 기반의 컴포넌트 자원 공유방식은 위에서도 잠시 언급한 바와 같이 수행성능을 악화시키는 주 원인이 된다. 이러한 PBO 공유 데이터 방식의 수정은 일반적인 PBO가 하드웨어 모듈의 컴포넌트화를 위해 만들어진 개념이기에 복잡한 데이터의 연동이 사실상 필요 없었기 때문이었다. PBO는 하드웨어에서 출력하는 신호 데이터나 입력 신호 데이터를 단순한 데이터 버퍼 형태로 저장하고 이를 관리한다. 이러한 방식은 많은 정보량을 공유하기에 효율이 떨어질 뿐 아니라, 다중 쓰레드에 의한 자원공유를 지원해야 하는 자바 가상머신에게는 매우 불합리한 방식이다.



[그림 4] 일반적인 PBO의 자원 공유 방식

따라서 이를 개선하여 자바에서 사용하는 저장 공간(Runtime Data Area)을 공유 할 수 있도록 고려된 가상머신 프레임워크가 각 컴포넌트들을 연결하면 관리하게 하였다. 이 프레임 워크에서는 각 컴포넌트들이 자원을 자유롭게 사용할 수 있도록 하며, 클래스 로더, 실행엔진, 스케쥴러, 메모리 관리자 등의 컴포넌트들은 이를 통해 하나의 실시간 하드웨어 모듈과 같이 자원에 접근하고, 다중 쓰레드 내에서 각 수행을 보장받는다.

3.3 유비쿼터스 환경을 위한 고려

자바 플랫폼은 크게 컨피규레이션과 프로파일로 나눌 수 있는데, 이는 가상머신과 API군의 구성을 말한다. 일반적으로 핸드폰과 같은 경량 하드웨어는 KVM을 기반으로 CDC/CLDC 프로파일을 제공한다. 이러한 자바의 전략은 하드웨어 및 컴퓨팅 환경에 맞춰 보다 효율적인 자바 프로그램이 가능하게 하기 위해서 이다. 그러나 모든 어플리케이션을 지원하면서 모든 환경에서 동작하는 플랫폼이 존재할 수 없기 때문에, 환경에 맞는 가상머신과 API를 각각 구축해야만 한다.

결국 유비쿼터스 환경에서 보다 효율적인 플랫폼 구축을 위해서는 가상머신의 컴포넌트화 뿐만 아니라 가상머신과 연계된 API군의 지원도 가능해야만 한다. 이미 자바에서는 다양한 환경에 보다 쉽게 API를 구축할 수 있도록 JNI라는 것을 규정하고 있어, 이를 이용하여 API군을 쉽게 구축할 수 있다. 이미 GNU에서는 이를 이용하여 오픈 자바 클래스 프로젝트를 추진하고 있다[6]. 그러나 근본적으로 가상머신과 API의 의존관계를 명확하게 규정하고 있지 않아, 구동상의 정확성을 확보할 수 없다. 이에 각 자바 가상머신 컴포넌트는 지원 API의 의존성을 명확히 명시하여 이러한 문제점을 근본적으로 해결할 수 있다.

4. 결론 및 향후 연구

자바 가상머신은 자바 플랫폼을 서로 다른 환경에서도 같은 코드로 동일한 작업을 수행시켜 줄 수 있도록 만들어주는 시스템 통합 미들웨어 이다. 이러한 미들웨어는 현재 난립하고 있는 임베디드 시장 및 앞으로 더욱 다양해질 유비쿼터스 시장에 시스템 투명성(transparency)라는 측면에서 매우 중요한 위치를 지니게 될 것이다. 따라서 이러한 미들웨어를 적시에 공급하기 위해서는 기존의 '일률적인 개발 후 각 플랫폼마다 포팅'이라는 개념이 아닌 컴포넌트 개발과 재조립을 통한 플랫폼 구성기술이 매우 효율적이다.

본 논문에서 제안하는 컴포넌트 기반의 자바 가상머신은 실시간 로보틱스에 응용되어온 PBO모델을 수정 보완하여, 자바 가상머신이라는 미들웨어 플랫폼 구성에 응용하였다. 이를 위해 일반적인 자바 가상머신의 구조를 보다 기능적인 면에서 정리하고 이를 컴포넌트로 설계하여 기존의 컴포넌트 모델이 가진 장점을 시스템 프로그램에서 충분히 활용할 수 있도록 하였다. 또한 기존 임베디드 업계에서 관행처럼 기피하던 컴포넌트를 미들웨어 플랫폼 구축에 과감히 적용하였으며, 이를 통해 위에 언급된 개발 및 유지보수의 효율성을 극대화 하고자 하였다. 그리고 현재 활발히 진행되고 있는 유비쿼터스 환경에서 매우 다양한 환경에서 통일된 개발환경을 제공할 수 있도록 고려하였다. 물론 그 구조가 아직은 초기 연구 단계로 여러 프로그램을 지원하기에 미흡하며, 수행성능 면에 있어서 다른 가상머신과 비교 할 수 있는 정도에 미치지 못한다. 앞으로 이에 대한 많은 연구가 필요하다.

참고문헌

[1] Tim Lindholm and Frank Yellin. *The Java Virtual Machine Specification. Addison- Wesley.*

[2] Kaffe : *http://www.kaffe.org.*

[3] David B. Stewart. Designing Software Component for Real-Time Applications. *2001 Embedded Systems Conference San Francisco, CA*, April 2001.

[4] David B.Stewart, Richard A.Volpe, and Pradeep K.Khosla. Design of Dynamically Reconfigurable Real-Time Software Using Port-Based Objects. *IEEE Transactions on Software Engineering, VOL. 23, No. 12*, pages 759-776, December 1997.

[5] Jochen Burkhardt, Dr. Horst Henn, Stefan Hepper, Klaus Rintdorff, Thomas, "Pervasive Computing" Wesley, 2002

[6] GNU Classpath : http://www.gnu.org/software/classpath/

[7] Bill Venners, *Inside The Java 2 Virtual Machine, McGraw-Hill.*