스마트 카드 메모리 관리를 통한 JCVM 성능 향상

김민정°, 조증보, 이상용, 정민수
경남대학교 컴퓨터공학과

A study on Performance improvement of the JCVM through the Smart Card Memory Management

MinJung Kim, JeungBo Cho, SangYong RLee, MinSoo Jung
Dept. of Computer Engineering, Kyungnam Univ.

요 약

자바는 스마트 카드 상의 다중 애플리케이션 기능을 지원하기 위한 가장 유용한 프로그래밍 언어이다. JCVM(Java Card Virtual Machine)은 자바 언어로 작성된 프로그램들을 스마트 카드 상에서 동작 가능하게 한다. 현재 스마트 카드는 작은 프로세서를 가지고 있으며 이런 제한적인 환경에서의 JCVM 성능 향상은 매우 중요한 이슈 중의 하나이다. 그리고 기존의 스마트 카드는 쓰기 속도가 느린 EEPROM에 객체를 생성하여 사용함으로 JCVM의 성능 저하를 가져왔다.
본 논문에서는 스마트 카드 메모리 관리, 즉, EEPROM에서 RAM으로 객체를 이동시킴으로써 JCVM 성능을 보다 향상시키는 알고리즘에 관해 제안하고자 한다.

1. 서론

자바 카드 기술(Java Card Technology)이란 자바 플랫폼을 스마트 카드 상에서 동작 가능하도록 적용시킨 것이다. 그러나 현재 스마트 카드는 카드의 한정된 자원 제약으로 인하여 자바 언어 특징을 부분적으로만 지원해준다. 예를 들면, 자바 카드에서는 모든 원시 타입(Primitive type)이 지원되지 않으며, 클래스 동적 다운로드 또한 허용되지 않는다.

이와 같이, 스마트 카드는 카드 상에서 동작 할 수 있는 애플리케이션의 크기를 엄격히 제한함으로써 사용 가능한 메모리의 양을 제한한다.

따라서, 애플리케이션 설치 시 최소한의 공간 할당을 위해서 각 애플리케이션 특성에 따라 메모리를 설정하여 시스템에 보다 많은 애플리케이션을 설치하는 것이 바람직하다. 애플릿 개발에 있어 자바 카드의 작은 메모리는 가장 심각한 제약사항이다.

자바 카드는 몇 가지 면에서 데스크 탑 컴퓨터와 다르다. 일반적 데스크 탑 컴퓨터에서의 표준 자바 플랫폼은 객체들을 RAM에 생성하는 반면, 자바 카드 플랫폼에서는 RAM 메모리 자원 제약과 컴퓨터 전원의 영향으로 인해 객체들은 EEPROM에 생성하게 된다.[3][12]

JCVM은 프로그램 실행 시, EEPROM의 객체들에

대한 다중 액세스 권한을 갖게 된다. 그러나 EEPROM에서의 쓰기 연산(Write operation)은 RAM에서의 쓰기 연산(Write operation)에 비해 약 100,000배 가량 느리다는 단점을 가지고 있다. 이를 해결하고자 본 논문에서는 EEPROM에서 RAM으로 객체를 이동시킴으로써 보다 빠른 액세스 시간을 지원하는 알고리즘을 제안하고자 한다.

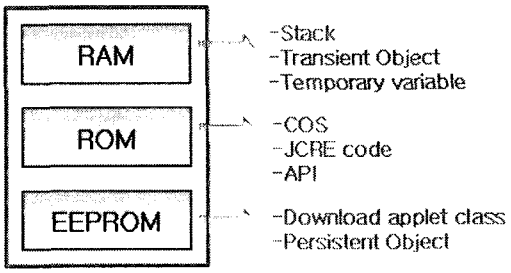
본 논문은 구성은 다음과 같다. 2장에서는 관련 연구와 스마트 카드 메모리 타입들의 비교에 관한 내용을, 3장에서는 EEPROM에서 RAM으로 객체를 이동시키기 위한 논리적 방법에 대해 설명한다. 그리고 4장에서는 성능 테스트 결과를 보여주며, 마지막으로, 결론은 5장에서 다루고자 한다.

2. 관련연구

2.1 스마트 카드 메모리

JCVM은 프로그램 실행 시, 바이트 코드와 정보, 프로그램 객체에 대한 설명, 메소드 매개변수, 리턴 값, 지역변수, 그리고 계산 결과의 중간 생성물을 저장하기 위한 메모리를 필요로 한다.

스마트 카드 메모리는 일반적으로 ROM, RAM, 그리고 EEPROM으로 구성되어 있다. 오늘날, 자바 카드는 보통 16Kbyte의 RAM과 128Kbyte의 ROM과 64Kbyte의 EEPROM으로 구성된다.



[그림 1] 자바 카드 메모리

■ RAM
- 자바 카드 런타임 스택과 비영구적 객체(transient objects)를 위해 사용됨.
- 전원 차단 시 저장된 내용을 모두 잃기 때문에 카드의 한 세션(Session) 동안 저장/변경되는 데이터들을 비영구적으로 저장.

■ ROM
- Chip Operating System(COS), JCRE code(VM, API class, interpreter), 하나 혹은 여러 개의 애플리케이션으로 구성.
- 단지 읽기 연산(Read operation)만 가능.
- 생산 당시 이미 칩(Chip)에 "Hard-Writed" 되고 생산 초부터 OS routine외에 다양한 루틴을 포함.

■ EEPROM
- 카드 사후 발급 후 적재(Load)되는 애플리케이션을 위해 사용.
- Longer-Lived 데이터, 영속 객체(Persistent Object), Static Field, 클래스 값 저장.
- RAM보다 쓰기 속도가 약 100,000배 가량 느림.
- 메모리 전원 차단 시 데이터들이 저장되므로 ROM과 RAM에 비해 좀더 복잡한 구조로 구성.

2.2 메모리 타입 비교 (RAM, EEPROM, ROM)

애플리케이션의 애플릿 인스턴스와 관련된 영속 객체들은 카드의 한 세션 동안만 존재하므로 카드의 비휘발성(non-volatile) 저장소인 EEPROM에 저장하게 된다. EEPROM은 RAM과 동일한 읽기/쓰기 연산을 제공하지만 가장 중요한 차이점은 EEPROM에서는 물리적으로 쓰기 횟수가 제한되고, EEPROM cell의 쓰기 연산 속도가 일반적으로 RAM보다 100,000배 이상 더 느리다는 점이다.

[표 1] 스마트 카드의 메모리 타입 비교

Type	Write Number	Writing time	Cell size with
RAM	unlimited	70 ns	1700 □m²
EEPROM	10,000 □ 1,000,000	3-10 ms	400 □m²
ROM	1	100ms	-

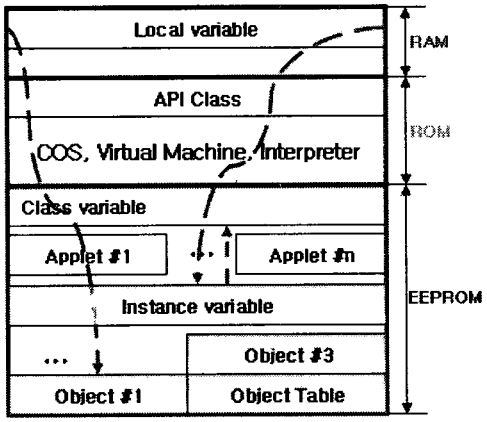
표 1에서 보면 RAM cell은 EEPROM cell에 비해 약 4배 정도의 공간을 더 필요로 한다. 이는 스마트 카드에서 RAM을 거의 가지고 있지 않는 이유라 할 수 있겠다.

3. EEPROM에서 ROM으로 객체를 이동시키는 새로운 JCVM 설계

3.1 일반적 자바 카드에서의 객체 처리 방식

일반적인 자바 환경은 비영속 프로그래밍 환경으로 설계되지만 자바 카드 환경에서는 비영속 객체와 영속 객체에 대한 접근을 모두 지원해야만 한다. 하나의 원시 타입 배열 데이터는 RAM에 할당될 수 있다. 그리고 영속적 저장소에서 특별한 static 메소드를 사용해 객체를 새로운 바이트 코드들에 할당하는 것은 단지 EEPROM과 RAM 데이터의 객체 헤더(Header)에 할당하기 위해 사용된다.

단, 비영속 배열의 내용은 지워진다는 것에 유의해야 한다. 따라서 비영속 배열의 배열 참조를 EEPROM에 저장함으로써 각 세션 시작 시마다 비영속 배열의 참조를 재할당(reallocate) 할 필요가 없다.



[그림 2] SUN의 JCVM 실행 환경의 메모리 구조

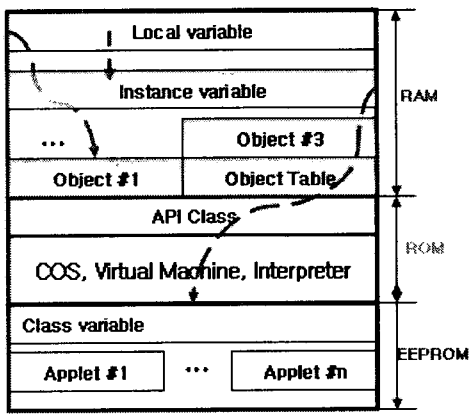
그림 2에서처럼 기존의 JCVM은 객체를 순서대로 EEPROM에 생성하고 생성된 객체의 참조 위치는 RAM에 저장한다. 그리고 EEPROM에 저장된 객체는 다양한 정보(객체 변경 정보, 클래스 주소 참조)를 포함하고 있다. 여기서 생성된 객체의 인스턴스 변수는 EEPROM에서 클래스 변수를 나타낸다. 그리고 클래스 변수에는 RAM에 저장된 객체의 참조 정보가 저장되게 된다.

3.2 RAM을 기반으로 객체를 구성한 새로운 JCVM

기존 스마트 카드의 영속 메모리의 계속적인 사용은 EEPROM의 비용 증가와 액세스 속도 감소를 야기 시킨다. 따라서 본 논문은 이러한 단점을 극복하기 위한 방안으로 새로운 JCVM을 제시하고자 한다. 객체를 영속 메모리(예)EEPROM)에 저장하는 대신 비영속 메모리(예)RAM)에 저장함으로써 비용 감소와 액세스 속도를 높여주는 것이다. 즉, 객체 생성 시 마다 EEPROM으로 저장되는 정보를 쓰기 속도가 약 100,000배 정도 빠른 RAM으로 저장하여 비용 감소와 액세스 속도 향상을 꾀하고자 하는 것이다.

기존의 자바 카드는 새로운 객체 생성 시 객체에 대한 정보(객체 크기, 객체 변경정보, 참조 클래스 주소)를 EEPROM에 저장하고 EEPROM에 대한 위치를 RAM의 스택 공간에 저장하는 방식이었다. 따라서 갑작스런 전원 차단 시 RAM 내의 정보들이 모두 사라지게 되고 생성된 객체에 대한 위치 정보와 그 외 중요 연산 정보들을 모두 잃어버리게 되는 문제가 발생하게 된다.

이에 따라 본 논문에서는 전원 차단 시 영속 메모리에서 발생하는 객체 정보 손실을 해결하기 위해 객체 생성 시 마다 객체는 RAM에 저장하고, 그 RAM 내의 각 객체 정보들은 EEPROM에 저장하는 방식을 사용함으로써 전원 차단 시 발생하는 문제를 해결할 수 있도록 하였다.



[그림 3] JCVM 실행 환경의 새로운 메모리 구조

4. 평가

본 논문의 평가를 위해 자바 카드에 대한 다양한 샘플들을 분석하고 바이트 코드 명령을 검사하였다. "PutField" 명령은 EEPROM에 객체 필드를 쓰는 것을 의미한다. 자바 카드의 각 샘플들은 아래 표 2와 같이 다양한 "PutField" 명령을 가진다.

[표 2] "PutField" 명령의 Frequency와 reduced time

Sample file	PutField	Total Instruction	Reduced Time
HelloWorld	1	62	3-10 ms
Wallet	3	243	9-30 ms
JavaPurse	33	1473	99-330 ms
average	15	796	45-150 ms

표 2는 본 논문에서 제안한 알고리즘을 사용함으로써 각 샘플들이 "PutField" 명령은 평균 15, reduced time은 평균적으로 150 ms 정도 향상되었음을 나타내는 결과표이다.

5. 결론

스마트 카드 시장은 최근 폭발적으로 증가하기 시작했다. 그러나 많은 부분에서 스마트 카드는 충분한 기능을 제공하지 못하므로 앞으로의 스마트 카드 활용은 경제, 상업, 법률, 그리고 정치 등 여러 분야와 연계되어 발전해야 나가야 할 것이다.

본 논문에서는 EEPROM에서 RAM으로 객체를 이동함으로써 보다 향상된 JCVM에 대해 제안하였다. 또한 이 알고리즘을 사용한 JCVM을 실험하여 그 결과 수백 ms까지 속도가 향상됨을 보였다. 따라서 본 논문에서 제시한 JCVM은 기존의 JCVM보다 더 빠른 필드를 쓸 수 있다.

또한 본 논문에서는 RAM 내의 각 객체 정보들을 EEPROM에 저장하는 방식을 사용함으로써 전원 차단 시 발생하는 문제를 해결할 수 있도록 하였다.

[참고문헌]

[1] E. Giguere, Java 2 Micro Edition, (2000)

[2] Sun Microsystems, Inc. http://java.sun.com/, Sun Microsystems, Java Home Page

[3] Lindholm, T. and Yellin, F., The Java™ Virtual Machine Specification Second Edition, Addison-Wesley.

[4] B. Venners, *Inside the Java Virtual Machine*, McGraw-Hill, (1998)

[5] Java Card Forum, URL : http://www.javacardforum.org.

[6] W.Rankl,. W.Effing,. : Smart Card Handbook" Second Edition, John Wiley & Sons, (2001).

[7] Z. Chen, Java Card™ Technology for Smart Cards: Architecture and Programmer' s Guide, ADDISONWESLEY (2000).

[8] U.Hansmann., M.Nicklous.,T.schack., F.Seliger., :Smart Card Applicatioin Development Using Java, Springer (1999)

[9] Sun Microsystems, Inc.2002. *Java Card Development Tool Kit.* Sun Microsystems, Inc.

[10] Sun Microsystems, Inc.2002. *Java Card™ 2.2 Virtual Machine Specification, Sun Microsystems,* Inc. URL: http://java.sun.com/products/javacard

[11] Sun Microsystems, Inc.2002. *Java CardTM 2.2 Application Programming Interface Specification.* Sun Microsystems, Inc. URL: http://java.sun.com/products/javacard

[12] Sun Microsystems, Inc.2002. *Java CardTM 2.2 Runtime Environment (JCRE) Specification.* Sun Microsystems, Inc. URL: http://java.sun.com/products/javacard

[13] E. Vetillard, : Tools for Integrating the Java Card™ API into Jini™ Connection Technology: javaone conf. (2000)

[14] X. Leroy., : On-Card Bytecode Verification for Java Card E-smart 2001, LNCS 2140, (2001) 150 -164