Hard Real-time System을 위한 효율적인 KVM의 설계

최인범°, 정명조, 조문행, 이철훈
충남대학교 컴퓨터공학과
(ibchoi°, mjjung, mhcho, clee)@cnu.ac.kr

A Design of Effective KVM for Hard Real-time System

In-Bum Choi°, Myoung-Jo Jung, Moon-Haeng Cho, Cheol-Hoon Lee
Dept. of Computer Engineering, Chungnam National Univ.

요 약

임베디드 시스템은 연성 실시간 시스템과 경성 실시간 시스템의 두 가지 종류로 나뉜다. 이러한 두 가지 종류의 임베디드 시스템 중에서, 응용 프로그램이 지정된 시간 안에 동작하여야 시스템의 붕괴를 막을 수 있는 경성 실시간 시스템에 JVM(Java Virtual Machine)환경을 사용하기 위해서는 JVM 내부 동작과 관련하여 여러 가지 고려하여야 할 부분이 많다.

본 논문에서는 위에서 언급한 바와 같이 경성 실시간 시스템에 적합한 JVM 환경을 구현하기 위하여 Sun' s KVM 을 기반으로 경성 실시간 시스템에 적합하도록 우선순위(Priority) 정책 및 가비지 콜렉션(Garbage Collection) 기법을 적용하였다.

1. 서론

군에서 사용하는 최첨단 무기체계나 인공위성 등의 과학실험용 임베디드 장비들에 탑재되는 실시간 시스템은 동작의 정확성 및 빠른 응답속도를 보장해야 한다. 이러한 요구사항을 만족시키기 위하여 해당 임베디드 시스템들은 시간 결정성을 고려하여 구현된 경성 실시간 시스템을 탑재하며, 이에 적합한 미들웨어 및 응용 프로그램이 탑재되어 수행되어야 한다.

지금까지의 경성 실시간 시스템은 간단한 응용 프로그램의 수행만 관리하여 왔으나, 시스템이 복잡해지고, 발생하는 예외상황이 다양해지고 있으며, 시스템 간의 상호 연동 등 여러 가지 새로운 상황을 고려해야 하기 때문에 점차 중간단계의 관리 모듈인 미들웨어를 탑재하는 추세이다.

다양한 미들웨어 플랫폼 중 자바는 플랫폼 독립성, 보안성, 네트워크 이동성, 실행코드의 재사용성, 작은 실행 파일 크기 등의 장점을 가지고 있기 때문에 임베디드 시스템에서 보편적인 미들웨어 플랫폼으로 채택, 사용되고 있다. 경성 실시간 시스템에서 자바 플랫폼을 채택하게 되면, 자바의 장점을 활용하여 경성 실시간 운영체제 시스템의 동작을 좀 더 원활하게 할 수 있게 된다.

본 논문은 이러한 경성 실시간 시스템들에 맞게 자바 플랫폼을 설계하여, 좀 더 원활한 동작을 지원할 수 있도록 한다.

본 논문에서는 2 장에서 관련 연구로 연성 실시간 시스템과 경성 실시간 시스템의 비교, KVM 의 쓰레드 관리 정책 및 KVM 에서 사용하는 기본 가비지 콜렉션에 대한 내용을 살펴보고, 3 장에서 경성 실시간 시스템을 위한 두 가지 KVM 의 응용 프로그램 관리 정책에 대한 설계를, 4 장에서 결론 및 향후 연구 과제를 기술한다.

* 본 연구는 산업자원부의 지역전략산업 석,박사 연구인력 양성사업의 지원으로 수행된 것임

2. 관련 연구

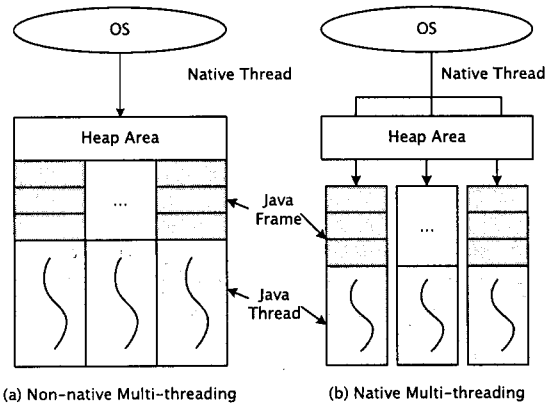
2.1 연성, 경성 실시간 운영체제 비교

임베디드 시스템은 시간 결정성의 보장관계에 따라 크게 두 가지로 분류된다. 첫째로 경성 실시간 시스템(Hard Real-time System)은 종료시한을 조금이라도 넘겨 작업을 마치면 그 결과가 아무리 정확하여도 시스템에 치명적인 영향을 미치는 시스템이다. 일반적으로 이를 이용한 실시간 운영체제의 태스크 스케줄링은 종료시한을 기반으로, 종료시한이 가장 적게 남은 태스크를 먼저 스케줄링하는 EDF 등이 있다. 둘째로 연성 실시간 시스템(Soft Real-time System)은 종료시한을 넘겨 작업을 마친 결과라 할지라도 그 결과에 의미있는 시스템이다. 일반적으로 이를 이용한 실시간 운영체제의 태스크 스케줄링은 종료시한을 기반으로 하지 않고 각 태스크의 중요도에 따라 우선순위를 부여하여 우선순위를 기반으로 스케줄링하는 실시간 운영체제가 이에 해당된다.

2.2 KVM의 쓰레드 관리 정책

KVM 에서는 멀티 쓰레딩(Multi-threading) 방식을 지원한다. KVM 에서 멀티 쓰레드를 지원하기 위해서

는 [그림 1]과 같은 두 가지 방법이 고려되며, 기본적으로는 [그림 1]의 (a)와 같이 Non-native 방식이 사용된다.

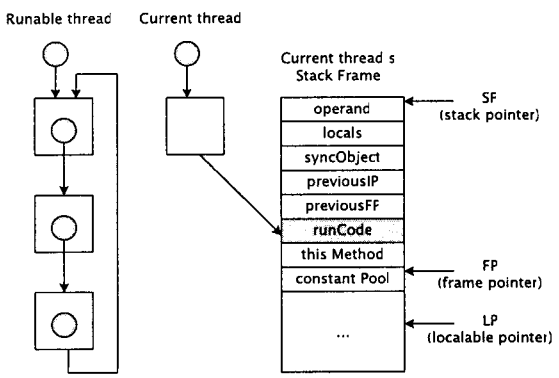


[그림 1] KVM의 Multi-threading 방식

Non-native 멀티 쓰레딩 방식은 운영체제로부터 하나의 phygical 쓰레드를 받아서 사용하기 때문에, KVM 설계 시 KVM 내부적으로 동기화 Mutex 가 필요없고, 간단한 카운터를 이용하여 자바객체들을 모니터할 수 있다. 또한, Low 레벨 단계에서 운영체제에 이식할 때 동기화를 고려하지 않기 때문에 구현이 용이하다. 하지만 I/O, 인터프리터 동작 시 오버헤드가 커서 성능이 나빠지는 단점이 있다.

2.3 KVM의 쓰레드 스케줄링 방식

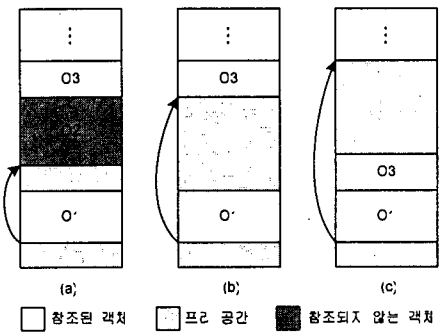
쓰레드들간의 문맥교환(context switching)시 [그림 2]와 같이 KVM 은 내부적으로 실행 큐(runable thread)로부터 다음에 수행할 쓰레드를 결정해야 한다. 다음 수행할 쓰레드를 결정하기 위하여 KVM 에서는 기본적으로 우선순위 기반의 쓰레드 스케줄링 방식을 제공하며, 동일한 우선순위에서는 RR(Round-Robin) 방식을 사용한다.



[그림 2] KVM의 쓰레드 스케줄링 방식

2.4 KVM의 기본 가비지 콜렉션

KVM 에서 지원하는 기본적인 가비지 콜렉션(Garbage Collection)은 마크-회수(압축) 방법이다. [그림 3]의 (a)와 (b)에서 보는 바와 같이 새로운 객체나 배열을 위한 메모리 할당 시 더 이상 힙에 메모리를 할당하지 못할 경우 마크-회수 방법을 이용하여 참조되지 않는 오브젝트의 Mark bit 을 셋팅한 후 이를 수거하여 프리 리스트에 추가하게 된다. 이렇게 추가된 프리 리스트에서 할당할 적당한 메모리 공간을 받아 사용하게 되는데, 만약 프리 리스트에 적당한 크기의 빈 공간이 없을 경우에는 [그림 3]의 (c)에서 보는 바와 같이 압축을 실행하여 빈 메모리 공간을 모아 전체적인 큰 하나의 공간으로 만들어 할당하게 된다. 이 방법은 간단하고 오버헤드가 적은 장점이 있으나, 시간이 지남에 따라 힙에 단편화 현상을 발생시켜 메모리의 효율성을 점점 감소시키는 단점이 있다.



[그림 3] 마크-회수(압축) 알고리즘

3. Hard Real-time System을 위한 효율적인 KVM의 설계

본 논문에서는 2 장에서 살펴본 내용을 토대로 경성 실시간 시스템을 위한 효율적인 KVM 을 설계하였다. 경성 실시간 시스템은 종료시한을 조금이라도 넘겨 작업을 마치면 그 결과가 아무리 정확하여도 시스템에 치명적인 영향을 미치기 때문에 KVM 의 설계 시, 응용 프로그램의 수행 속도에 큰 영향을 끼치는 쓰레드 스케줄링과 가비지 콜렉션의 두 가지 측면에서 좀 더 많은 고려를 해야 한다.

3.1 경성 실시간 시스템을 위한 KVM 의 쓰레드 스케줄링 기법 설계

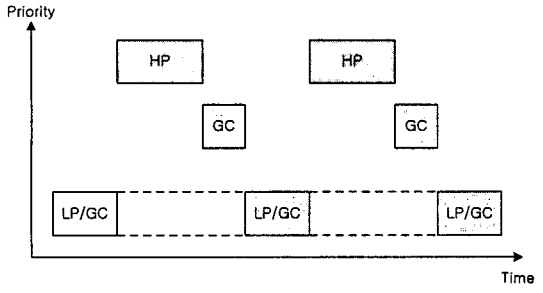
연성 실시간 시스템에서 KVM 은 2 장에서 살펴본 바와 같이 우선순위 기반의 쓰레드 스케줄링 방식을 제공하며, 동일한 우선순위에서는 RR 방식을 사용한다.

KVM 을 통하여 수행되는 대부분의 자바 애플리케이션들은 상호 연관성이 적고, 독립적으로 수행이 가능하기 때문에, 기존의 KVM 과 비교하여 경성 실시간 시스템을 위한 KVM 의 설계에서도 기본적인 자바 응용 프로그램의 수행은 크게 달라지지 않는다.

연성 실시간 시스템과 비교하여 경성 실시간 시스템에서는 쓰레드의 생성, 수행, 종료가 적절한 시간 안에 모두 동작을 수행하여 원하는 결과를 낼 수 있어야 한다. 하지만 기존 KVM 의 스케줄링 정책만으로는 경성 실시간 시스템의 조건을 만족하기 어렵다.

본 논문에서는 이와 같은 경성 실시간 시스템의 조건을 만족하기 위하여, 새로운 쓰레드 스케줄링 방법을 제안하였다.

[그림 4]에서 보는 바와 같이 기존 KVM 에서 제공하는 5 단계의 우선순위를 경성 실시간 시스템에 맞게 간단히 3 단계의 우선순위로 바꾸어, 가장 긴급한 쓰레드(HP)를 우선순위 1 로, 새로 제안할 가비지 콜렉션(GC)을 우선순위 2 로, 기존의 쓰레드나 기본 가비지 콜렉션(LP/GC)은 우선순위 3 으로 한다.



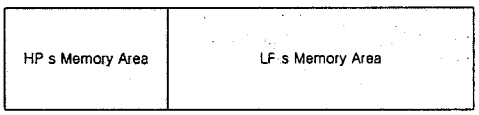
[그림 4] 경성 실시간 시스템을 위한 스케줄링 정책

이와 같이 설계하게 되면 우선순위가 낮은 기존의 쓰레드가 수행하는 중간에도 긴급한 쓰레드가 발생할 경우 이를 우선 처리할 수 있으며, 우선순위 단계를 간소화 함으로써 복잡한 우선순위 단계를 처리하기 위한 오버헤드를 줄일 수 있다.

3.2 경성 실시간 시스템을 위한 KVM 의 가비지 콜렉션 기법 설계

본 논문에서는 [그림 4]에서 보는 바와 같이 HP 와 LP 를 위한 가비지 콜렉션 기법을 분리, 적용함으로써 가비지 콜렉션을 수행하는데 발생하는 오버헤드를 줄일 수 있고, Out-of-memory 상황을 피할 수 있다.

본 논문에서 설계한 두 가지 가비지 콜렉션을 적용하기 위해서는 [그림 5]와 같이 힙의 메모리 영역을 HP 를 위한 영역과 LP 를 위한 영역의 두 부분으로 나누어야 한다.



[그림 5] KVM 의 Heap 메모리의 구조

LP 쓰레드의 수행을 위한 메모리 할당은 LP 메모리 영역에 하며, 기존의 가비지 콜렉션 기법 (마크-회수-압축)을 통하여 메모리를 수거한다. 이 방법을 통하여 LP 쓰레드를 수행시키는 중 HP 쓰레드가 발생하게 되면, HP 메모리 영역에 필요한 메모리를 할당하여 HP 쓰레드를 수행하게 되며, [그림 4]에서와 같이 HP 쓰레드의 수행이 종료되면 바로 새로운 가비지 콜렉션을 수행하여, HP 메모리 영역의 메모리를 수거하게 된다.

새로운 가비지 콜렉션은 HP 쓰레드가 종료되면 바로 실행되며, HP 메모리 영역을 모두 초기화 한다.

HP 쓰레드는 긴급한 쓰레드로, 수행을 위한 메모리 공간의 할당 시 메모리 공간에 여유가 없어 메모리를 할당 하지 못하는 상황(Out-of-memory)이 발생하면 안된다. 따라서 새로운 가비지 콜렉션은 기본 가비지 콜렉션 (마크-회수-압축) 알고리즘의 마지막에 HP 메모리 영역을 모두 초기화하는 루틴을 삽입하여, 후에 또 다른 HP 쓰레드가 발생하여 메모리 영역을 할당할 경우 Out-of-memory 상황이 발생하지 않도록 한다.

본 논문에서 설계한 두 가지 가비지 콜렉션은 기존의 KVM에서 사용하는 가비지 콜렉션과 비교하여 알고리즘 상의 차이점은 존재하지 않는다. 그러나 관리하는 메모리 영역이 기존 KVM의 가비지 콜렉션보다 작기 때문에 상대적으로 전체적인 가비지 콜렉션의 수행 시간은 빨라지게 된다.

4. 결론 및 향후 연구과제

본 논문에서는 경성 실시간 시스템을 위한 KVM을 설계하였다. 기존의 우선순위 정책을 5 단계에서 3 단계로 축소, 적용하고, 힙의 메모리 영역을 HP와 LP를 위한 공간으로 분리하여 두 가지 가비지 콜렉션을 적용하였다. 우선순위 단계의 축소를 통하여, 복잡한 우선순위를 처리하기 위한 오버헤드를 줄이고, 메모리 공간의 분리를 통하여 가비지 콜렉션의 오버헤드와 Out-of-memory상황을 피할 수 있게 되었다.

KVM 의 가비지 콜렉션 기법은 매우 다양하다. 본 논문에서는 기본적인 마크-회수-압축 방법을 응용하여 설계하였으나, 이 방법은 수행이 빠르고 오버헤드가 적은 반면 힙의 단편화 현상을 발생시키는 단점이 있다. 따라서 향후 좀 더 효율성이 좋은 가비지 콜렉션 기법을 연구, 적용하여 경성 실시간 시스템을 위한 KVM 의 설계에 적용하여야 한다.

5. 참고문헌

[1] Sven Robertz, " *Applying Priorities to Memory Allocation*", International Symposium on Memory Management, Proceedings of the 3rd international symposium on Memory management, 2003, pp.108-118

[2] Sun Microsystems, " *JSR-30 J2ME Connected, Limited Device Configuration*"

[3] Bill Venners, " *Inside the Java Virtual Machine*",1999

[4] Tim Lindholm, " *The Java™ Virtual Machine Specification Second Edition*", 1999