자바가상머신 쓰레드 모델 분석 및 설계

유용선°, 박윤미, 류현수, 이철훈
충남대학교 컴퓨터공학과
{ysryu°, ympark, hsryu, chlee}@ce.cnu.ac.kr

The Analysis and Design of Thread Model for Java Virtual Machine

YongSun Ryu°, YoonMi Park, HyunSoo Ryu, Cheol-Hoon Lee
Dept. of Computer Engineering, Chungnam National Univ.

요 약

최근들어 인터넷의 발달과 더불어 PDA, 핸드폰과 같은 모바일 디바이스와 다양한 정보가전용 기기들에 네트워크 기반의 자바기술이 적용되고 있으며, 이러한 자바 기술을 사용함으로써 플랫폼 독립성, 이식성, 보안성, 이동성 등의 장점을 얻을 수 있다. 그러나, 자바로 작성된 응용프로그램은 C, C++로 작성된 응용프로그램 보다 수행속도가 느리다는 단점이 있다. 이러한 문제점을 해결하기 위해서는 자바가상머신의 성능향상이 필수적이다. 지금까지 메모리 관리를 위한 가비지 컬렉션, 소프트웨어나 하드웨어를 이용한 바이트 코드 변환, 인라인캐시(inline-cache)를 사용한 접근 속도 향상 등 많은 부분에서 활발한 연구가 진행되고 있다. 본 논문에서는 모바일 플랫폼에서 동작하는 KVM(kilo-virtual machine)의 성능향상을 위한 쓰레드 구조를 분석하고 설계한다.

1. 서 론

C나 C++로 작성된 기존의 임베디드 애플리케이션은 새로운 환경이나 플랫폼에서 상호 연동되지 않는 문제점들이 있다. 그래서, 개발자들은 매번 새로운 환경에 맞게 수정해야 하는 번거로움과 유지보수 측면에서의 비용낭비가 발생한다. 따라서, 이와 같은 문제점을 해결하기 위해서 플랫폼 독립성, 이식성, 보안성, 이동성 등의 장점을 갖는 자바언어를 고안하게 되었다. 그러나, 자바로 작성된 응용프로그램이 기존의 C, C++로 작성된 응용프로그램 보다 수행속도가 느리다는 단점이 있고, 최근에는 이러한 문제점을 해결하기 위해, 메모리 관리를 위한 가비지 컬렉션, 소프트웨어나 하드웨어를 이용한 바이트 코드 변환, 인라인캐시(inline-cache)를 사용한 접근 속도 향상 등 많은 부분에서 자바가상머신의 성능향상을 위한 연구가 활발히 진행되고 있다.

본 논문에서는 다양한 플랫폼을 지원하는 자바가상머신 중에서 CLDC 명세서(Specification)를 기반으로 한 KVM(kilo-virtual machine)의 쓰레드 모델을 설계한다.

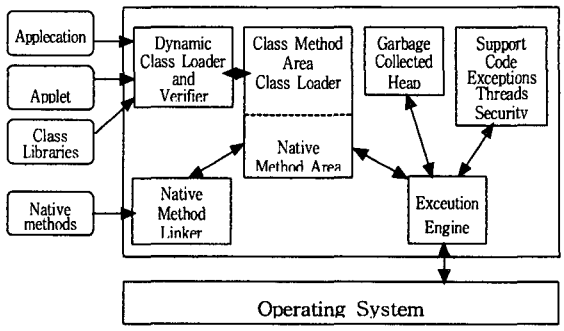
본 논문의 2장에서는 관련연구로 자바가상머신의 구성요소(components)들을 살펴보고, 3장에서는 KVM상의 쓰레드 모델을 분석 및 설계하며, 마지막으로 4장에서 결론 및 향후과제에 관해 기술한다.

2. 관련연구

2.1 자바가상머신(Java Virtual Machine)

자바의 가장 큰 장점은 "write-once run-everywhere"이다. 자바로 구현된 응용프로그램들이 각각의 플랫폼에 맞게 재작성 되거나, 다시 컴파일하지 않아도 다른 시스템에서 실행되는 것을 말하며, 궁극적으로 각각의 시스템에 따른 자바가상머신이 존재하기 때문에 가능한 것이다.

자바가상머신은 명령어 집합, 레지스터 집합, 스택, 가비지를 수행하는 힙 메모리와 같은 실행영역과 클래스 로더 시스템, 실행엔진 등으로 구성되며, 컴파일된 자바 바이너리 코드와 실제로 프로그램의 명령어를 실행하는 마이크로프로세서 또는 하드웨어 플랫폼간에 인터페이스 역할을 담당한다.



그림[1] 자바가상머신 내부 구조

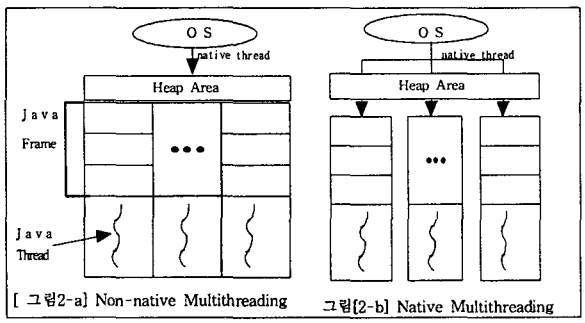
* 본 논문은 한국과학재단이 지정한지역협력연구센터(RRC)인 충남대학교소프트웨어 연구센터의 지원으로 수행된 과제의 결과입니다.

3. KVM 쓰레드 모델 분석 / 설계

본 장에서는 KVM(kilo virthual machine)에서 동작하는 쓰레드의 구조를 동기화, Mutex, 스케줄링 기법등을 고려하여 설계한다.

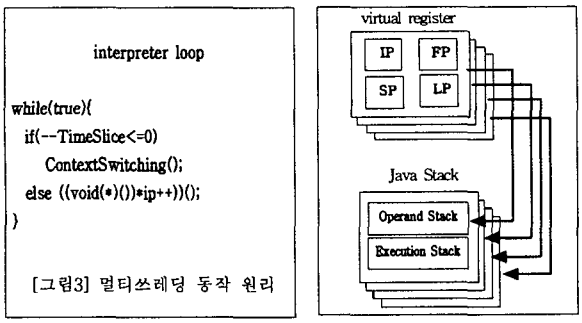
3. 1 멀티 쓰레드 지원 방식(Multi-threding)

자바가상머신에서 멀티 쓰레드를 지원하기 위해서는 [그림2]와 같은 두 가지 방법이 고려되며, KVM에서는 플랫폼에 독립적인 non-native 방식을 적용한다.



[그림2-a] Non-native Multithreading 그림[2-b] Native Multithreading

Non-native 멀티 쓰레드 방식은 운영체제로부터 하나의 phygical 쓰레드를 받아서 사용하기 때문에, VM을 설계시 VM 내부적으로 동기화 Mutex가 필요없고, 간단한 카운터를 이용하여 자바객체들을 모니터할 수 있다. 또한, 로우 레벨 단계에서 운영체제와 포팅할 때 동기화를 고려하지 않기 때문에 구현이 용이하다. 하지만 I/O, 인터프리터 동작시 오버헤드가 커서 성능이 나빠지는 단점이 있다.



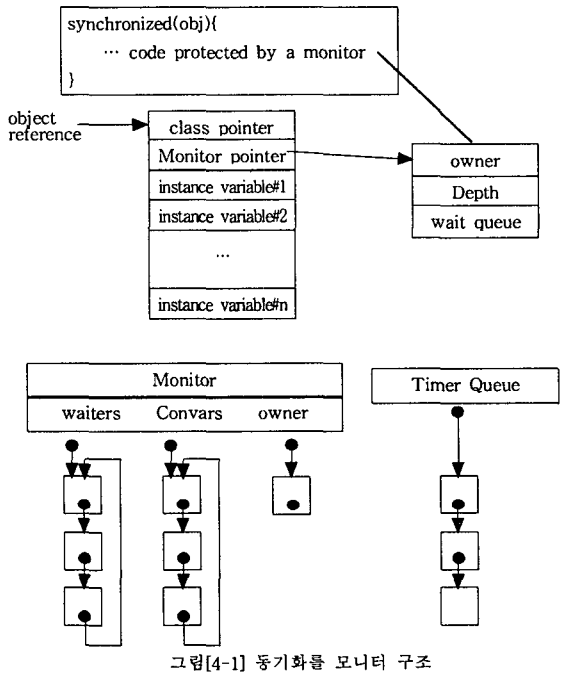
[그림3] 멀티쓰레딩 동작 원리

그림[3]은 KVM에서 동작하는 멀티 쓰레드들간의 문맥교환(context switching)을 나타낸다. 현재 수행하는 쓰레드의 레지스터 값(IP, FP, SP, LP 등)을 저장하고, 다음에 수행될 쓰레드의 레지스터 값으로 셋팅함으로써, 새로운 자바스택으로 문맥교환이 일어난다.

3. 2 동기화 지원(Synchronization)

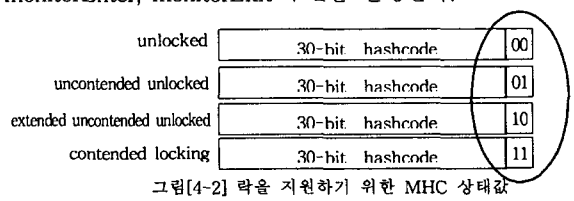
모든 객체는 자신과 관련된 모니터를 갖는다. 따라서, 자바가상머신은 모니터가 하나의 쓰레드에 의해서만 소유될 수 있도록 보장해 주어야 하며, 이를 통해서 프로그래머는 여러 가지 동기화 제어를 구현할 수 있다.

자바가상머신은 모니터와 관련된 monitorEnter와 monitorExit라는 2개의 명령어를 제공한다. 쓰레드가 monitorEnter 명령어를 실행하면, 해당 쓰레드는 스택 최상위 객체에 대한 모니터를 얻는다. 만약 이미 다른 쓰레드에서 해당 객체에 관한 모니터를 얻은 상태라면 모니터를 놓을 때까지 대기하게 되고, 같은 객체에 대해 monitorExit 명령어를 실행하면 모니터는 놓아진다.



그림[4-1] 동기화를 모니터 구조

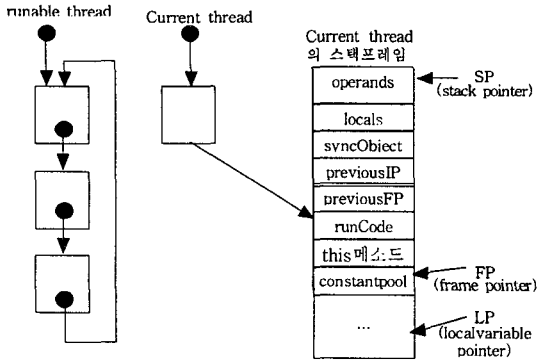
한편, 락(lock)은 한 시점에 오직 하나의 쓰레드만이 동작하도록 보장해주며, 각 객체의 MHC(MonitorOrHashCode) 필드의 최하위 2bit값을 가져와(fetch) 그 정보에 따라 monitorEnter, monitorExit 루틴을 실행한다.



그림[4-2] 락을 지원하기 위한 MHC 상태값

3. 3 스케줄링 방식(Scheduling)

쓰레드들간의 문맥교환(context switching)시 자바가상머신은 내부적으로 실행큐(runable thread)로부터 다음에 수행할 쓰레드를 결정해야 한다. 본 논문에서는 기본적으로 우선순위 기반의 스케줄링 방식을 제공하며, 동일한 우선순위에서는 RR(Round-Robin)방식을 사용한다.



3. 4 타이머(Timer) 큐

자바 쓰레드는 타이머 인터럽트나 디버거 코드를 만나면 특정 시간만큼 실행이 지연(suspended) 될 수 있다. 타이머 큐는 이러한 쓰레드들을 효율적으로 관리하기 위한 구조체로, 쓰레드를 타이머 큐에 등록하거나, 특정시간 후 쓰레드를 깨워주고, 쓰레드를 큐에서 제거하는 간단한 동작들이 정의되어야 한다.

```
struct TimerQueue {
    THREAD nextAliveThread;
    THREAD nextThread;  //특정시간 suspended 되는 runnable 쓰레드
    JAVATHREAD javaThread;
    long    timeslice;
    STACK stack;  // 쓰레드의 execution stack
  //쓰레드가 실행큐에 있는 동안, 레지스터의 상태를 저장하기 위한 환경변수들
    BYTE* ipStore;     FRAME fpStore;
    cell* spStore;     cell* nativeLp;
    MONITOR monitor;  //모니터 큐
    short monitor_depth;
    THREAD nextAlarmThread;
    long    wakeupTime[2];
   //알람에서 깨어난후의, 쓰레드의 callback fuunction 값
    void (*wakeupCall)(THREAD);
}
```

그림[5-1] 타이머 큐의 데이터 구조(data structure)

```
// 큐에서 대기할 시간을 셋팅하는 부분
wakeupTime = CurrentTime_md();
ll_inc(wakeupTime, delta);        /* wakeUp += delta */
SET_ULONG(thread->wakeupTime, wakeupTime);

//쓰레드의 callback function을 저장
thread->wakeupCall = wakeupCall;

//Timer 큐에 쓰레드를 등록시키는 부분
q = TimerQueue;
prev_q = NULL;
while (q != NULL) {
    ulong64 qtime = GET_ULONG(q->wakeupTime);
    if (ll_compare_ge(qtime, wakeupTime)) {
        break;
    }
    prev_q = q;
    q = q->nextAlarmThread;
  }
  if (prev_q != NULL) {
      prev_q->nextAlarmThread = thread;
      thread->nextAlarmThread = q;
  } else {
      thread->nextAlarmThread = TimerQueue;
      TimerQueue = thread;
  }
```

그림[5-2] 타이머 큐에 쓰레드를 등록하는 registerAlarm() 코드 일부분

4. 결론 및 향후과제

본 논문에서의 쓰레드 구조는 JVM 명세서, KVM 명세서 그리고 CLDC 1.04 Source code를 기반으로 분석 및 설계되었다. Non-native 쓰레드 방식을 사용함으로써, 플랫폼 독립적인 장점을 갖지만, I/O 처리시 효율성이 떨어지며, 비동기적인 I/O 처리를 위해서 native 쓰레드 풀(pool)의 구현이 필요하다.

최근들어 실시간 시스템에 동작하는 자바가상머신의 연구가 진행되고 있다. 이러한 시스템은 반응시간과 지연시간이 모두 예측가능 해야 하며, 태스크들이 정확한 시간내에 수행되어야 하는 제약조건을 갖는다. 따라서, 이러한 시스템에 자바가상머신이 수행되기 위해 모든 태스크들의 반응시간, 실행시간을 예측할 수 있는 Task Schduling방식과 여러 개의 태스크들이 동시에 임계영역(critical section)에 접근시 코드를 보호하기 위한 Task 동기화가 제공되어야 한다. 향후과제로 실시간 시스템에 수행될 자바가상머신에 관한 연구가 이루어져야 한다.

참고문헌

[1] Timothy Lindholm and Frank Yellin. The Java Virtual Machine Spedification, The Java Series, Addison-Wesley, 1996

[2] Sun Microsystems, Inc., Java™ CLDC1.04 source code

[3] Divid Dice, Implementing Fast Java™ Monitors with Relaxed-Lock, Proceedings of the Java™ Virtual Machine Reaserch and Technology Symposium(JVM01')

[4] Ken Arnold and James Gosling. The Java Programming Language, Sun Miicrosystems, Inc.

[5] B. Lewis, D. Berg, The Thread Primer : A Guide to Multi-threaded Programming, Sunsoft Press, Prentice Hall, 1996