연성 실시간을 보장하는 자바 M:N 쓰레드 맵핑 모델

양영록° 손봉기 김명준
충북대학교 컴퓨터 과학과
marine95@korea.com° bkson77@hanmail.net mjkim@cbucc.chungbuk.ac.kr

A Java M:N Thread Mapping Model for Guaranteeing Soft Real-Time

YoungRok Yang° BongKi Sohn MyungJun Kim
Department of Computer Science, Chungbuk National University

요 약

사용자 쓰레드와 시스템 쓰레드간의 1:1맵핑 모델은 병렬성을 지원하는 장점이 있고, M:N 맵핑은 병렬성 지원과 빠른 문맥교환의 장점이 있다. 리눅스 자바 가상 머신에서는 1:1 맵핑 모델만을 지원한다. 연성 실시간을 보장하기 위해서는 쓰레드간의 문맥교환을 최소화하여 성능 향상시킬 필요가 있다. 이 논문에서는 자바 어플리케이션 레벨에서 경량 프로세스(Light Weight Process, LWP) 개념을 도입하여 리눅스 자바 가상 머신에서 M:N 맵핑을 지원하는 자바 쓰레드 모델을 제안한다. 제안한 모델은 그린 쓰레드(Green Thread)의 빠른 문맥교환과 네이티브 쓰레드(Native Thread)의 병렬성 지원 장점을 혼합한 것으로 빠른 처리속도와 자바 플랫폼의 독립성을 그대로 유지할 수 있다. 또한, MTR-LS 알고리즘을 경량 프로세스 스케쥴링에 채택함으로서, 자바 응용프로그램의 연성 실시간을 보장한다. 1:1 및 M:1 맵핑 모델과의 성능 비교를 통해 제안한 모델이 좋은 성능과 연성 실시간을 보장한다는 것을 보인다.

1. 서 론

자바는 플랫폼 독립성, 높은 보안 수준, 멀티 쓰레드, 가비지 콜렉터와 같은 장점을 가진 언어로서 인터넷 응용 분야, 내장형 시스템 응용프로그램 등의 여러 분야에 사용되고 있다. 그러나 자바 응용프로그램 레벨에서 실시간을 지원하기에는 많은 제한점을 가지고 있다. 예를 들어, 가비지 콜렉터 메모리 관리로 발생하는 예측할 수 없는 지연시간, 물리적인 메모리 접근 불가, 표준화된 실시간 API의 부재 등과 같은 특징들은 자바가 실시간 응용프로그램에 적합하지 않음을 보여 주고 있다.

이러한 자바의 언어적 특성에도 불구하고, 자바 응용프로그램 레벨에서 연성 실시간을 지원하기 위한 많은 연구들이 이루어지고 있다. RTSJ(Real-Time Specification for Java)은 실시간을 지원하기 위한 자바 기술 문서로서 실시간 응용프로그램을 위해 여러 가지 API를 제공한다[1]. Nilsen에 의한 자바 플랫폼 상에서 RMS(Rate Monotonic Scheduling) 분석, Static Cycle Scheduling, Real-Time 가비지 콜렉터 등의 연구가 진행되고 있으나 하드웨어 의존적이어서 현실적으로 적용하기 어렵다[2].

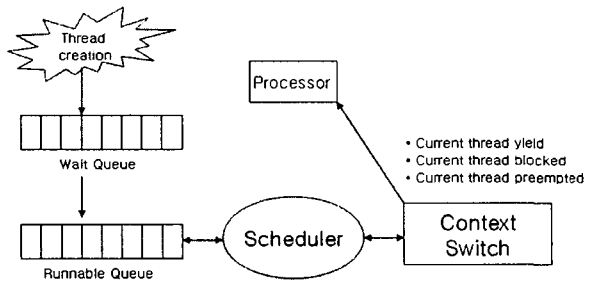
QJVM은 그린 쓰레드 라이브러리를 사용한 자바 가상 머신으로서 쓰레드 생성 및 자원 관리, 스케쥴링, 동기화등 모든 것이 사용자 레벨에서 이루어진다[3]. QJVM에서는 연성 실시간을 만족하는 MTR-LS(Move-to-Rear List Scheduling) 알고리즘을 이용하여 M:1 맵핑 모델만을 고려하였다. M:1 맵핑 모델은 문맥교환이 커널과 무관하게 사용자 레벨에서 이루어지므로 오버헤드 비용이 매우 낮은 반면, 입·출력 작업시 하나의 쓰레드가 커널로 진입하는 순간 다른 모든 쓰레드가 기다리는(Pending) 상태가 발생함에 따라 응답시간이 느려질 수 있다. 또한 다수의 사용자 쓰레드와 하나의 커널 쓰레드 가 맵핑되므로 커널 레벨에서 병렬성(Parallelism)을 지원하지 못하고 단지 사용자 레벨에서의 동시성(Concurrency)만 제공한다. 최근 버전의 자바 가상 머신과 리눅스에서는 병렬성을 제공하기 위해 커널 레벨에서 네이티브 쓰레드 맵핑 모델을 채택하고 있다. 이 방식은 사용자 쓰레드와 커널 쓰레드가 1:1 방식으로 맵핑되어 있으므로 커널 레벨에서 병렬성을 제공한다. 그러나 사용자 레벨에서 문맥교환이 일어나면 커널로 진입하기 때문에 오버헤드가 많이 발생한다. 연성 실시간을 만족하기 위해 하나의 응용프로그램 레벨에서 자바가 지원하는 다수의 쓰레드 스케쥴링시 자원을 공정(Fairness)하게 할당하는 알고리즘이 필요하다[4][5][6]. 따라서 연성 실시간을 보장하기 위해 빠른 문맥교환과 병렬성을 지원하는 맵핑 모델이 필요하다.

이 논문에서는 빠른 문맥교환과 병렬성을 지원함으로서 연성 실시간을 보장하는 자바 M:N 쓰레드 모델을 제안한다. 제안한 모델에서는 다수의 그린 쓰레드를 사용자가 명시적으로 하나의 경량 프로세스로 맵핑할 수 있다.

이 논문의 구성은 다음과 같다. 2장에서는 자바 가상 머신에서 구현된 MTR-LS 알고리즘에 대해 살펴본다. 3장에서는 제안하는 자바 M:N 맵핑 모델의 장점에 대해서 기술한다. 4장에서는 리눅스 자바 가상 머신 상에서 제안한 모델과 1:1 및 M:1 모델과의 성능 실험을 통해 알아보고, 5장에서는 결론을 맺는다.

2. MTR-LS 알고리즘

MTR-LS는 CPU 자원을 응용프로그램 레벨에서 필요한 만큼 예약할 수 있으며, 사용자로부터 입력받은 서비스 타임인 CPU 퀀텀을 기준으로 스케쥴링 하는 알고리즘이다. 모든 쓰레드는 퀀텀 값에 의해 우선순위가 할당되며, 서비스 퀀텀값 만큼 CPU를 선점할 수 있다. 따라서 MTR-LS는 연성 실시간에 보장된 서비스를 제공할 수 있다. 자바 가상 머신에서 구현된 MTR-LS 알고리즘의 쓰레드 흐름은 [그림 1]과 같고, 각 쓰레드 상태에서의 특징은 다음과 같다.



[그림 1] 쓰레드 흐름도

㉠ Thread Creation
1. 쓰레드 자료구조에 퀀텀 항목을 추가한다.
2. 퀀텀값은 사용자가 명시적으로 입력한다.
3. 퀀텀값에 따라 큐에서 우선순위 결정된다.

㉡ Wait Queue
1. 조건 변수가 참이 될 때까지 기다린다.
2. 정렬은 수행하지 않는다.

㉢ Runnable Queue
1. 우선순위 정렬시 링크드 리스트보다 시간복잡도가 낮은 힙 자료구조를 사용한다.
2. 퀀텀 값에 따라 적절한 우선순위 할당한다.
3. 쓰레드가 가지고 있는 퀀텀값 만큼 CPU 자원을 MTR- LS 방식으로 서비스한다.

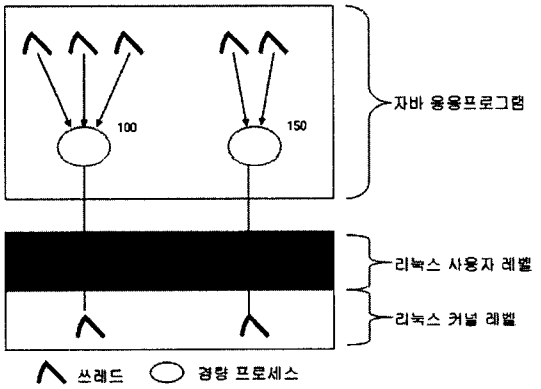
㉣ Scheduler
1. Runnable Queue에서 쓰레드 패치시 퀀텀값 체크한다.
2. 퀀텀 값이 0보다 크다면 CPU를 할당하고, 그렇지 않으면 새로운 퀀텀값을 할당하고 큐의 끝으로 이동한다.

㉤ Context Switch
1. 퀀텀값 주기적으로 계산
   ■first in-line function
   ■second in-line function

제안한 모델에서 연성 실시간을 보장하는 알고리즘을 적용하기 위해서는 MTR-LS에 대한 몇 가지 변경이 필요하다. 첫째 그린 쓰레드와 경량 프로세스를 동시에 생성하고, 퀀텀값은 경량 프로세스에 할당한다. 둘째 그린 쓰레드는 협력형으로 스케쥴링되고, 경량 프로세스는 MTR-LS 선점형 알고리즘에 의해 스케쥴링 된다.

3, 자바 응용프로그램 레벨에서의 M:N 쓰레드 맵핑 모델

자바 응용프로그램 레벨에서의 사용자 쓰레드는 운영체제의 시스템 쓰레드로 맵핑이 이루어져야, 실제로 CPU를 점유해서 연산을 수행한다. 유닉스나 윈도우 NT, Win95, 리눅스 등이 쓰레드를 지원하는 방식이 서로 다르기 때문에, 운영체제마다 시스템 쓰레드와 사용자 쓰레드 맵핑 모델이 다를 수 있다. 운영체제에서의 쓰레드는 사용자 쓰레드와 커널 쓰레드로 구분이 되어진다. 사용자 쓰레드는 사용자 라이브러리를 이용해서 만든 쓰레드이고, 커널 쓰레드는 운영체제에서 실제 스케쥴링 대상이 되는 단위이다. 쓰레드 맵핑 모델에는 1:1, M:1, M:N이 있지만, M:N 맵핑 모델이 빠른 문맥교환과 병렬성을 요하는 문제에서 성능이 가장 좋다. 리눅스 시스템에서는 1:1모델만을 지원하기 때문에, 응용 프로그램 레벨에서 M:N 맵핑 모델을 지원함으로서 성능을 향상시킬 수 있다. [그림 2]는 제안한 모델의 구조로서, 자바 응용프로그램 레벨에서 경량 프로세스 개념을 적용한다.



[그림 2] M:N 쓰레드 맵핑 모델

하나 또는 다수의 그린 쓰레드가 경량 프로세스로 맵핑될 수 있으며, 맵핑은 사용자에 의해 명시적으로 이루어진다. 경량 프로세스에 대한 우선순위는 사용자가 입력한 타임 퀀텀값에 따라 적절하게 할당된다. 경량 프로세스에 할당된 타임 퀀텀값은 그린 쓰레드들이 공유하고 빠른 문맥교환과 함께 우선순위에 따라 변경된다. 여러 개의 경량 프로세스는 리눅스 사용자 레벨에서의 네이티브 쓰레드와 1:1 맵핑 모델을 따르기 때문에 커널 레벨에서 병렬 처리가 가능하다. 경량 프로세스는 MTR-LS 방식의 선점형 스케쥴링을 따르고, 우선순위는 [그림 2]와 같이 생성시 사용자가 입력한 퀀텀 값에 의해 적절하게 할당된다. 이러한 모델은 사용자 관점에서 퀀텀값을 입력함으로서 어플리케이션 관점에서 CPU자원을 관리하고, 스케쥴링 가능성을 예측할 수 있는 장점이 있다.

4. 구현 및 실험

이 장에서는 제안한 모델을 구현함에 있어 고려할 사항에 대해 알아보고, 1:1 맵핑 모델과 M:1 맵핑 모델과의 성능을 실험을 통해 비교한다.

우선 그린 쓰레드로 사용할 Runnable 배열 객체는 그린 쓰레드를 생성하고 저장하는 객체로 협력형 방식으로 스케쥴링 된다. Runnable 객체는 환영구조로 구현되며, 다수의 쓰레드에 의한 데이터 공유문제는 synchronized 해결한다[7]. 그린 쓰레드를 생성하고 저장하는 Runnable 객체에 대한 코드는 다음과 같다.

```
private Runnable[] task =
{
        new Runnable() { public void run() {}},
        new Runnable() { public void run() {}},
        new Runnable() { public void run() {}}
}
```

다수의 Runnable 객체 배열을 쓰레드 생성자에 넘겨줌으로서 경량 프로세스를 생성한다. 그린 쓰레드는 경량 프로세스의 모든 데이터 영역을 공유하며 실행된다. 경량 프로세스 생성 코드는 다음과 같다.

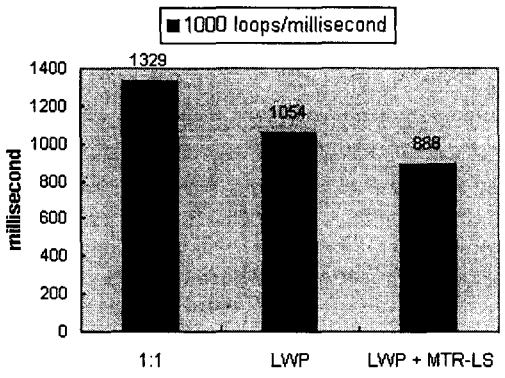
```
new Thread(new Runnable[] {new My_runnable_Object(),
                new My_other_runnable_Object()});
```

경량 프로세스 기술은 여러 가지 항목들로 구성된다. 퀀텀 항목은 MTR-LS 알고리즘을 적용할 때 사용된다.

```
public class Thread
{
        Object value;
        String id;
        long quantum;
        int priority;
        Node next;
        public Node(Object o) {
                value = o;
                next = null
        }
}
```

리눅스 자바 가상 머신 상에서 M:N 쓰레드 맵핑 모델과 1:1, M:1 맵핑 모델과의 비교 실험은 펜티엄-III 600Mhz, 리눅스 Paran 7.3, JDK2.2 glibc(Native Version)의 시스템 환경에서 수행하였다. 실험은 똑같은 쓰레드 개수 9개를 생성하여 1000번 루핑하는 동안 각 쓰레드가 종료할 때까지의 시간을 측정하였다. 1:1은 스케쥴링 정책을 사용하지 않은 것이고, LWP는 경량 프로세스 3개(경량 프로세스 당 그린 쓰레드 3개)를 생성하여 MTR-LS 알고리즘을 적용하지 않은 것이다. LWP+MTR-LS은 경량 프로세스를 생성하여 MTR-LS 알고리즘을 적용하여 실험한 것이다.

실험을 통해 자바 응용프로그램 레벨에서 M:N 맵핑 모델과 MTR-LS 알고리즘을 적용한 모델이 다른 두 모델보다 성능이 좋다는 것을 알 수 있다.



[그림 3] 실험 결과

5. 결론

이 논문에서는 연성 실시간을 보장하는 자바 응용프로그램 레벨에서의 M:N 쓰레드 맵핑 모델을 제안하였다. 제안한 모델은 그린 쓰레드와 경량 프로세스를 자바 응용프로그램 레벨에서 생성하고, M:N 맵핑 모델에 의해 맵핑이 이루어진다. 또한, M:N 맵핑 모델에 의해 생성된 경량 프로세스에 대해 MTR-LS 알고리즘을 적용하여 자바 응용프로그램이 연성 실시간을 보장하게 한다. 실험을 통해 제안한 모델이 1:1 및 M:1 모델보다 좋은 성능을 보임을 알 수 있었다.

6 참고 문헌

[1] G. Bollella, et al., The Real-Time Specification for Java, *Addison-Wesley*, June 2000.

[2] K. Nilsen, Java for Real-Time, *In the journal of Real-Time System*, pp.197-205, February 1996.

[3] James C. Pang, Gholamali C. Shoja, Eric G. Manning, Providing Soft Real-time QoS Guarantees for Java Threads, *Concurrency and Computation: Practice and Experience*, pp.521-538, March-April 2003.

[4] P. Goyal, H.M. Vin, H. Cheng, Start-time Fair Queueing: A Scheduling Algorithm for Integrated Service Packet Switching Networks, *In Proceedings of ACM SIGCOMM'96*, pp.157-168, August 1996.

[5] A. Demers, S. Keshav, S. Shenker, Analysis and Simulation of a Fair Queueing Algorithm, *In Proceedings of ACM SIGCOMM*, pp. 1-12, September 1989.

[6] J. Bruno, E. Gabber, B. Ozden, A. Silberschatz, Move-To-Rear List Scheduling: A New Scheduling Algorithm for Providing QoS Guarantees, *Proceedings of the Conference on Multimedia*, pp. 63-73, November 1997.

[7] Allen I. Holub, Taming Java Threads, *Apress*, Jun 2000.