멀티미디어 지원을 위한 메모리 관리 기법 설계 및 구현

고 영웅[O], 아 재용, 홍 철호, 유 혁
고려대학교 컴퓨터학과

{yuko[O], jyah, chhong}@os.korea.ac.kr, hxy@os.korea.ac.kr

Design and Implementation of Memory Management Mechanism for Multimedia Applications

Young-woong Ko[O], Jae -young Ah, Chul-ho Hong, Chuck Yoo
Dept. of Computer Science and Engineering, Korea University

요 약

가상 메모리 시스템은 다중 프로그래밍의 정도를 높이고 효율적으로 주기억 장치를 관리하는 장점을 제공해 주지만 예측할 수 없는 지연을 발생시키는 문제점을 가지고 있다. 따라서 가상 메모리 시스템은 시간 제약이 엄격한 경성 실시간 시스템에서 사용하기에 부적합하며, 시간 제약이 완화된 멀티미디어 분야에서도 태스크의 제한 시간 실패율을 높이는 원인을 제공함으로써 멀티미디어 서비스 품질을 저하시키고 있다. 본 논문에서는 동적으로 태스크가 유입되는 범용 시스템 환경에서 가상 메모리 시스템이 멀티미디어 태스크에 미치는 영향을 분석하고, 멀티미디어 태스크의 제한 시간 실패율을 최소화할 수 있는 메모리 관리 방법을 제시하였다. 본 논문에서는 동적으로 유입되는 태스크의 페이지 폴트를 제한된 비율로 유지시키는 기법을 사용하며, 이를 통해서 부하를 분산시키고, 결과적으로 멀티미디어 응용이 원활히 수행될 수 있도록 하였다.

1. 서론

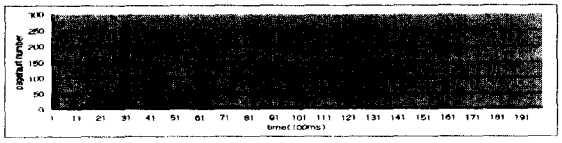
리눅스(Linux)와 같은 범용 운영체제는 다중 프로그램을 지원하기 위해서 가상 메모리 기법을 사용하고 있으며, 프로그램 시작 시에 디스크에 저장된 프로그램 이미지가 메모리에 모두 적재되지 않고, 필요할 때마다 한 페이지씩 물리 메모리로 적재되는 요구 페이징(demanding page) 기법을 사용하고 있다. 이와 같은 가상 메모리 시스템은 프로그램이 수행하면서 참조하는 주소 영역이 물리 메모리에 존재하지 않을 때, 페이지 폴트 핸들러 (page fault handler)가 동작하여 필요한 물리 메모리를 할당하고, 필요한 경우 디스크에 있는 프로그램 코드 또는 데이터를 메모리로 가져온다. 따라서 가상 메모리 시스템을 사용하는 경우, 메모리 접근 명령을 사용한 프로그램 코드가 물리 메모리를 할당 받고, 디스크 입출력을 처리하는 작업을 병행할 수 있으므로 예측할 수 없는 지연 시간이 추가될 수 있는 것이다. 이러한 문제점으로 인하여, 가상 메모리 시스템은 엄격한 시간 제약을 요구하는 경성 실시간 시스템에서는 사용할 수 없으며, 시간 제약이 완화된 연성 실시간 시스템에 있어서도 서비스 품질에 영향을 주는 요인으로 작용하게 된다.

본 연구는 한국과학재단의 특정기초 연구과제 연구비 지원에 의한 결과임 (과제번호 97-01-00-09-01-3)

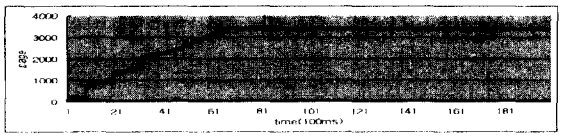
2. 가상 메모리 시스템의 문제점 및 특성

[그림 1]은 리눅스 운영체제에서 웹 브라우저 (netscape)를 수행시킬 때 시간의 경과에 따른 페이지 폴트 횟수를 보여주고 있으며 [그림 2]는 페이지 폴트 처리가 진행되면서 프로그램이 사용하는 물리 메모리의 양이 점차 증가함을 보여주고 있다. 그림에서 알 수 있듯이 프로그램이 수행된 초반에 대부분의 메모리 참조가 발생했으며, 웹 브라우저의 경우 대략적으로 3300 개의 물리 프레임이 집중적으로 요구되었음을 알 수 있다.

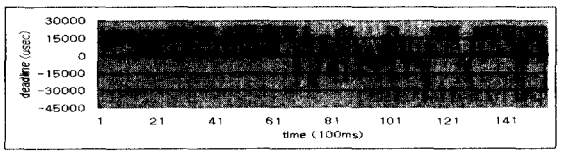
이처럼 짧은 시간 동안 수많은 페이지 폴트와 이에 따른 디스크 입출력이 발생되며, 결과적으로 페이지 폴트가 발생하는 동안 대부분의 응용 프로그램이 정상적으로 동작하기 힘들게 됨을 예측할 수 있다. [그림 3]은 동영상 플레이어가 수행 중인 중간에 웹 브라우저를 수행시킨 결과를 보이고 있으며, 재한시간을 놓치는 주기가 갑자기 증가하는 것을 알 수 있다.



[그림 1] Netscape 프로그램의 페이지 폴트

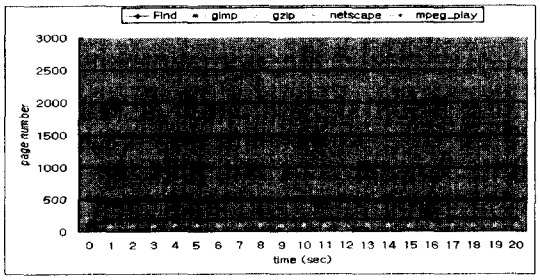


[그림 2] 시간에 따른 Netscape 의 메모리 사용량



[그림 3] 동적 태스크 유입에 따른 동영상 플레이어의 제한 시간 실패 현상

가상 메모리 시스템에서 페이지 폴트의 발생 빈도는 태스크가 로딩(loading)되는 초기 단계에 집중적으로 발생되며, 태스크가 수행되는 중간 단계에서는 페이지 폴트가 거의 발생되지 않는 경향을 가지고 있다. 이러한 특성을 보이는 이유는 태스크가 수행되는 초기 단계에서 대부분의 프로그램 코드와 데이터를 참조하기 때문이다. 아래 [그림 4]는 리눅스에서 수행시킨 몇 가지 응용 프로그램들의 메모리 사용 경향을 보이고 있다.



[그림 4] 범용 프로그램의 메모리 적재 상태

[그림 4]에서 알 수 있듯이 프로그램의 크기가 큰 Netscape 와 Gimp 는 각각 7 초 및 4 초의 시간동안 집중적으로 메모리를 요구하고 있지만, 프로그램 크기가 작은 find, gzip 그리고 mpeg_play(Berkeley MPEG-1)는 수백 밀리 세컨드 내에 대부분의 메모리 요구가 완료되고 있다.

3. 알고리즘

앞에서 언급한 바와 같이, 가상 메모리 시스템의 문제점은 태스크가 유입될 때 발생되는 페이지 폴트 작업이 시스템 자원을 급격히 소모함으로 시스템에 일시적인 과부하를 초래하는 것이다. 이와 같이 시스템 자원에 대한 요구가 일시적으로 높아졌을 때, 수용할 수 있는 범위까지만 처리를 하고 나머지 요구를 우선 순위에 따라서 지연시킴으로써, 제한 시간을 가지고 수행되는 중요한 작업들이 원활히 수행될 수 있는 방법이 제공되어야 한다.

본 논문에서는 비율에 기반한 페이지 폴트 처리 방식을 제시하고 있으며, 이것은 자원 예약 기법 (resource reservation)[1]에 바탕을 두고 있다. 자원 예약 기법을 활용하기 위해서 현재 시스템의 자원 사용 현황에 대해서 실시간 통계를 가지고 있어야 하며, 페이지 폴트가 소모하는 시스템 자원에 대해서도 미리 측정된 결과를 가지고 있어야 한다. 즉 페이지 폴트에 따른 프로세서 자원 소모율과 같은 정보를 시스템에서 유지하고 있어야 하며, 이러한 정보를 이용해서 현재 가용한 시스템 자원의 범위 내에서 페이지 폴트를 처리 할 수 있게 된다.

본 논문의 알고리즘은 시스템 자원에 대한 모니터링 및 자원 예약을 해주기 위해서 리눅스 커널[2]을 수정하여 사용하고 있으며, 멀티미디어 응용을 효율적으로 지원하는 기능을 추가하고 있으므로 LMX(Linux *Multimedia Extension*) 라 부른다. LMX 는 태스크를 크게 실시간 태스크(real-time task)와 범용 태스크(general task)로 분류하며, 커널은 실시간 태스크가 사용하는 시스템 자원을 계속적으로 모니터링하며, 실시간 태스크가 사용하지 않는 자원은 범용 태스크가 사용할 수 있도록 관리한다. LMX 의 모니터링 모듈에서 유지하는 값은 각 실시간 태스크의 자원 사용량, 전체 범용 태스크의 자원 사용량, 유휴 자원량 등이다.

새로운 실시간 태스크가 유입될 때, LMX 의 승인 제어부(admission control module)는 각 실시간 태스크의 자원 사용량과 유입된 태스크의 자원 사용량의 합계가 지정된 시스템 자원 한계를 초과하지 않는 경우 실시간 태스크를 수행시키게 된다. 이때 시스템 자원 전체를 실시간 태스크에게 할당하지 않고, 일부는 범용 태스크가 수행 될 수 있도록 예약하는 방식을 사용하고 있다. 범용 태스크는 유휴 자원량이 남아 있는 경우에 수행이 허가되며, 남아 있는 유휴 자원량의 범위 내에서 페이지 폴트 작업이 수행될 수 있도록 한다.

페이지 폴트 처리의 과부하 상황을 해결하기 위해서 단위 시간당 발생되는 페이지 폴트 발생 빈도를 일정한 비율로 유지시키는 과정은 다음과 같이 두 단계로 나뉘어 진다.

- 페이지 폴트 처리를 위한 자원 할당 단계
- 주어진 자원 한도 내에서 페이지 폴트 처리 단계

페이지 폴트 제어를 위해서 자원을 할당하는 방법은 프로세스가 적재되는 과정에서 페이지 폴트 처리 주기(fault_period) 및 한계 값(fault_limit)을 할당 해준다.

[2] 본 논문에서 수정하여 사용한 리눅스 커널을 **LMX(*Linux* Multimedia Extension)** 라고 부름 : 커널 버전 2.4

여기서 페이지 폴트 처리 주기는 페이지 폴트가 주어진 한계 값 내에서 수행되고 있는지 점검하는 시간 구간이며, 한계 값은 페이지 폴트 처리 주기에 프로세스가 사용할 수 있는 페이지 폴트 처리 상한 값을 의미한다.
태스크가 할당 받은 시스템 자원을 R(T) 라고 하고, 단위 페이지 폴트가 소비하는 시스템 자원을 R(P)라 할 때 다음과 같은 식이 성립된다.

$$R(T) = \frac{fault\_limit}{fault\_period} \times R(P)$$

따라서 태스크에게 주어진 가용 자원이 20 % (CPU)이고, 단위 페이지 폴트가 소비하는 자원이 0.001 % (CPU) 라고 하였을 때, 100 밀리 세컨드 주기마다 처리할 수 있는 페이지 폴트 한계는 200 개가 된다.

비율에 기반한 페이지 폴트 처리 단계에서는 각 태스크의 페이지 폴트 처리 주기 동안에 fault_limit 를 초과하는 요청이 들어올 경우 태스크의 우선 순위를 낮추고 있다. 따라서 페이지 폴트 처리 주기 내에서 페이지 폴트가 주어진 한계 이상으로 빈번하게 발생될수록 태스크의 우선 순위는 점점 저하되어 프로세서를 사용할 기회가 점점 줄어들게 된다. 다음 [그림 5]는 페이지 폴트 처리 알고리즘을 보이고 있다.

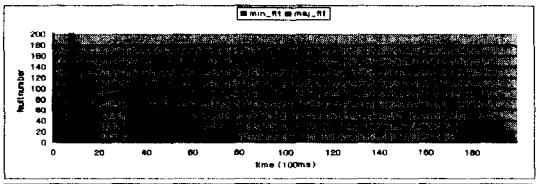
```
if(한 주기가 지났으면 ){
    새로운 주기값 지정;
    fault_count=0;
 }else{
    fault_count++;
    if(fault_count > fault_limit) {
       태스크의 우선순위 저하
       재스케줄링 요청
    }
 }
```

[그림 5] 페이지 폴트 처리 알고리즘

4. 구현 및 실험 결과

이번 장에서는 본 논문에서 제시하는 비율 기반 페이지 폴트 처리 방식(LMX)과 기존의 리눅스(Linux) 시스템을 상호 비교한다. 실험은 멀티미디어 응용 및 수식 계산 프로그램 그리고 범용 프로그램(웹 브라우저, 그림 툴 등)을 동적으로 유입 시키면서 페이지 폴트가 효율적으로 분산되고 있는지 여부와 멀티미디어 응용이 제한시간을 잘 지키면서 수행되는지를 살펴본다. 실험에 사용된 동영상 플레이어는 Berkeley MPEG-1 이며, 단위시간당 30 프레임을 디스플레이하는 데이터를 사용하였다.

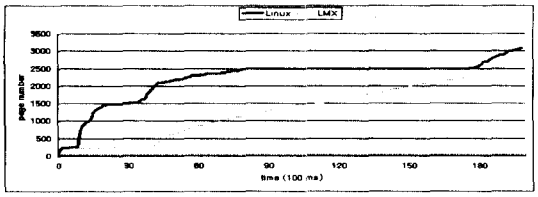
[그림 6]의 상위 부분은 기존의 리눅스에서 실험한 결과이며 하위 부분은 LMX 에서 실험한 결과이다. 리눅스에서 초기에 과도하게 발생하는 페이지 폴트 비율이 LMX 에서 완만한 수준으로 유지되고 있음을 알 수 있으며, 이때 수행중인 동영상 플레이어의 제한 시간 실패율은 10 퍼센트 정도 줄일 수 있었다. 또한 제한 시간 실패 프레임이 평균적으로 리눅스에서 15 밀리 세컨드 지연되었지만, LMX 에서는 7 밀리 세컨드로 지연시간이 낮아졌다.



	frame number	deadline miss	miss rate(%)	average delay
LINUX	500	161	32.2	15054
LMX	500	110	22	6954

[그림 6] netscape 동적 유입 시 페이지 폴트

[그림 7] 은 Netscape 가 시간에 따라서 사용하고 있는 물리 메모리 양의 변화를 보여주고 있다. LMX 는 점진적으로 메모리 양을 늘려주고 있는데 비해서 기존의 리눅스는 초반에 과도한 물리 메모리 할당을 해주고 있다.



[그림 7] Linux 와 LMX 메모리 요구량 변화도

5. 결론 및 향후 계획

범용 시스템 환경에서 멀티미디어 응용을 수행하고자 할 때 동적으로 유입되는 태스크의 영향이 상당히 크며, 대부분의 시스템 부하가 페이지 폴트를 처리하는 부분에서 발생되고 있음을 알 수 있다. 따라서 멀티미디어 응용과 같은 실시간 태스크를 일시적인 과부하로부터 보호하기 위한 방법으로 가상 메모리 시스템의 자원 사용률에 제한을 가하는 비율 기반의 페이지 폴트 처리 방식이 유용하게 사용될 수 있을 것이다. 향후 물리 메모리에 적재된 페이지가 메모리 부족에 의해서 페이지 교체되거나 스와핑(swapping) 되는 문제점을 메모리 자원 예약 메커니즘을 적용해서 해결하는 연구를 진행할 계획이다.

참 고 문 헌

[1] C. W. Mercer, S. Savage, H Tokuda, "Processor Capacity reserves: Operating System Support for Multimedia Applications", Proceedings of the IEEE international Conference on Multimedia Computing and Systems, Boston, MA, pp. 90-99, May 1994.