주문형 멀티미디어 서버의 마감시간보장을 위한 2단계 디스크 스케줄링 기법

김정원*, 전봉기*, 윤홍원*,**

A Two-step Disk Scheduling Scheme for Deadline Guarantee of Multimedia on Demand Server

Jeong-won Kim*, Jun Bong-Gi*, Hong-Won Yun*,**

요 약

기존의 Best-effort 응용을 위한 디스크 스케줄링 기법들은 멀티미디어 객체의 실시간성을 만족하지 못하며, 실시간 응용을 위한 디스크 스케줄링 기법들은 시스템의 처리율을 만족시키지 못한다. 따라서, 본 논문에서는 범용 운영체제에서 멀티미디어 객체의 주기적인 연성 실시간 요구와 비실시간 서비스를 동시에 만족시키는 2 단계 디스크 스케줄링 기법을 제안한다. 제안 하는 기법은 실시간 요구와 비실시간 요구에 적절한 가중치를 부여하는 라운드로빈 기법에 기초하고 있다. 리눅스 커널에서의 실험 결과 실시간 태스크와 비실시간 태스크 사이의 공정성이 보장됨을 확인하였다.

ABSTRACT

The previous disk scheduling schemes for best-effort applications do not guarantee the real-time requirement of multimedia objects and the real-time disk scheduling schemes do not satisfy throughput of multimedia server. So, this paper propose a two-step disk scheduling scheme to satisfy the requirement of best-effort as well as soft real-time applications. This scheme is based on the round robin algorithm that imposes different weights on the best-effort task and the real-time one. The experiment results on the Linux kernel have shown that both best-effort tasks and real-time tasks could get fair service.

키워드

Disk Scheduling, MOD, Multimedia, real-time

Ⅰ. 서 론

이동식 헤드를 장착한 디스크의 발명 이래로 I/O 성능을 향상시키기 위해서 효율적인 디스크 스케줄링 알고리즘에 대한 많은 연구들이 있었다 [1,2,3,4]. 이러한 연구들은 크게 세 가지 범주로 분류될 수 있다. 첫째, best-effort응용을 위한 알고리즘들로서 디스크 헤드의 탐색의 최적화를 위한 알고리즘들이다. 둘째, 실시간 요구의 마감시간을 보장하기 위한 알고리즘들이다. 마지막으로 이 두 가지 목적을 동시에 만족하고자 하는 알고리즘들이다.

한편, MOD(Multimedia on Demand)서버에서 멀티미디어 객체는 주기적인 연성 실시간 적인 특성을 가진다. 따라서, 멀티미디어 객체의 서비스를 위해서는 정해진 주기내에 일정한 대역폭을 보

* 신라대학교 컴퓨터정보공학부
** 교신저자
접수일자 : 2003. 5. 14

장해야 한다. 그러나, Best-effort 응용을 위한 디스크 스케줄링의 목적은 평균 응답시간을 최소화시키는 것이 목적이기 때문에 멀티미디어 객체의 주기적인 연성 실시간 특성을 만족시키지 못하므로 빈번한 마감시간 실패를 초래한다. 또한, 실시간 응용을 위한 디스크 스케줄링의 경우 실시간 시스템을 목적으로 설계되었다. 따라서, 리눅스와 같은 범용 운영체제에서는 적합하지 못한 디스크 스케줄링 기법이다. 또한, 경성 실시간을 고려하여 만들어진 디스크 스케줄링 기법이기 때문에 멀티미디어 서비스와 같은 연성 실시간 서비스의 경우 시스템의 처리율을 저하시킨다. 두 가지를 동시에 만족하는 연구들은 기존 디스크 스케줄링 알고리즘의 변형에 그치고 있다.

본 논문에서는 MOD 서버의 주기적인 디스크 대역폭 요구를 만족할 수 있고, 디스크 헤드의 탐색시간도 최적화할 수 있는 새로운 2 단계 디스크 스케줄링 알고리즘을 제안한다. 2장에서는 관련 연구를 살펴보고 3장에서는 2단계 디스크 스케줄링 알고리즘을 제안하고 분석하고 4장에서는 실험 결과를 설명하고 5장에서는 결론을 맺는다.

II. 관련 연구

이 장에서는 기존의 기법들과 본 연구의 기법을 비교 및 분석한다.

2.1 탐색 시간의 최적화를 위한 기존 연구들

Best-effort 서비스에 최적화된 디스크 스케줄링 알고리즘들이다. Best-effort 응용은 문서 편집기, 컴파일러, http 서버, ftp 등이 있는데 디스크 헤드의 탐색시간을 최적화하기 위해서 FCFS, SSTF, SCAN, LOOK, STF, ASAFT 등이 있다. 이러한 알고리즘들은 대화형 best-effort 응용을 위해서 평균 응답 시간을 최소화하는데 그 목적이 있다. 이러한 연구들은 MOD응용의 실시간 요구를 만족할 수 없다.

2.2 마감시간 보장을 위한 기존 연구들

실시간 요구, 즉 마감시간을 요구하는 응용에 최적화된 디스크 스케줄링 알고리즘들이다. 이러한 응용에는 MOD, VOD 등이 있는데 이 알고리즘에는 단순한 EDF, PSCAN, ED-SCAN, FD-SCAN, SCAN-EDF 등이 있다. 이러한 알고리즘들의 목적은 마감시간 이전에 디스크 블록을 서비스 하는 것이다. 다음은 대표적인 알고리즘들이다.

2.3 탐색 시간의 최적화와 마감시간 보장을 동시에 만족하기 위한 기존 연구들

[3,4]에서는 비실시간 요구를 마감시간을 가지는 실시간 요구로 전환하여 시스템에 존재하는 모든 요구를 실시간 요구로 처리하고 있다. 이 방식은 저장 시스템의 효율성을 저하시킨다. [5]에서는 각 요구들을 우선순위로 분류하여 우선순위가 높은 요구를 먼저 서비스하는 방식이다. 그러나, 이 방식은 우선순위의 분류가 어렵고, 우선순위가 낮은 요구들의 기아 현상을 초래할 수 있다. 한편, 최근 Cello 디스크 스케줄링은 다양한 응용을 지원하는 범용 2 레벨 디스크 스케줄링 기법[3]으로서 다양한 응용간의 독립성을 유지하며 각 응용에 최적화된 스케줄링 기법을 사용함으로써 차세대 운영체제를 위한 디스크 스케줄링을 제시하고 있다.

III. 2단계 디스크 스케줄링

3.1 2단계 디스크 스케줄링의 구조

본 논문에서 제시하는 디스크 스케줄링 기법은 2 단계 큐를 사용하여 위에서 제시한 고려 사항을 만족하고자 한다. 2 단계 스케줄링은 그림 1에서 보듯이 태스크 의존적 레벨과 태스크 독립적인 레벨로 구성된다.

그림 1에서 태스크 의존적 레벨(비주기, 주기, 대기 큐)에서는 디스크 요구들을 태스크 필터(Task filter)를 통해 실시간 클래스와 비실시간 클래스로 구분하여 큐에 입력한다. 실시간 클래스에 속하는 각 태스크의 요구는 라운드 당 보장 받은 블록 수만큼 주기 큐에 입력되고, 보장 받은 대

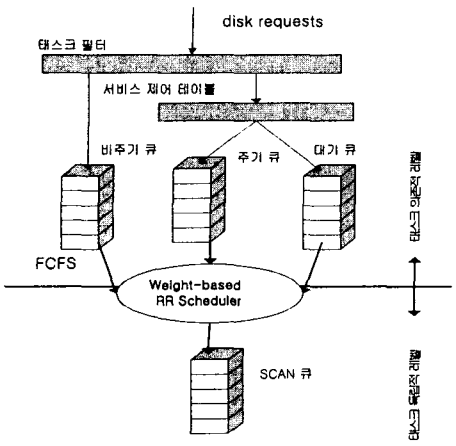

그림 4. 2단계 디스크 스케줄링 구조
Fig 1. The architecture of two-step disk scheduling

역폭을 초과하면 대기 큐로 보내진다. 따라서, 태스크 의존적 레벨에서는 실시간 클래스의 라운드당 블록 요구를 보장하고 각 태스크간 대역폭 할당의 공정성을 제공하게 된다.

태스크 독립적 레벨에서는 태스크의 종류에 관계없이 여러 요구들의 서비스 순서를 결정하게 된다. 이 레벨에서 스케줄링의 목적은 디스크 탐색 시간을 최소화하는 것이다. 이 태스크 독립적 레벨과 의존적 레벨 사이에는 가중치 기반의 라운드로빈 스케줄러(Weight-based round robin scheduler)가 있다. 이 스케줄러는 MOD 서버의 라운드 R을 기준으로 R*Wr (실시간 클래스의 가중치) 만큼 실시간 클래스를 서비스하고, R*Wnr (비실시간 클래스의 가중치) 만큼 비실시간 클래스를 서비스하게 된다. 만약 실시간 클래스 요구들의 총 소요 시간이 할당치 보다 크면 다음 요구들은 대기 큐로 보내진다. 이 대기 큐에 소속된 요구들은 현재 라운드에서 디스크가 휴지 상태일 경우 또는 다음 라운드 시작 시 다시 스케줄러에 의해 SCAN 큐로 이동하게 된다.

그림 2는 2단계 디스크 스케줄링의 예를 보이고 있다. 현재 MOD 저장 서버에 세 개의 실시간 태스크가 그림 2 의 태스크 테이블과 같이 존재한다고 가정하자. 테이블에서 r_i는 태스크의 주기 당 요구 블록수이고, b'_i는 현재 요구한 블록 번호이다. 태스크 0은 주기 당 3 블록, 태스크 1은 4블록, 그리고 태스크 2는 2블록을 요구한다. 그리고 이 시점에서 2개의 비실시간 요구가 비주기 큐에 있다. 실시간 태스크들은 프로세스 수준의 스케줄링 순서에 의해 주기 큐에 그림 2와 같이 입력된다. 이때 태스크 2의 요구가 다시 입력되면 현재 태스크 2의 라운드 당 요구 블록 수는 2이므로 대기 큐로 보내지고 그 다음의 태스크 1의 요구가 주기 큐로 입력되게 된다. 따라서 태스크 1은 라운드당 대역폭을 보장 받게 되고, 태스크 2는 현재 라운드에 초과하는 대역폭을 다른 태스크에게 양보하게 된다. 결국 실시간 태스크 간의 공정한 디스크 자원의 사용을 보장하게 된다.

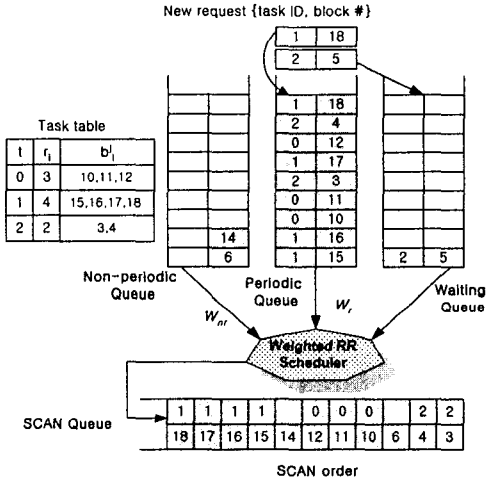

그림 5. 2단계 스케줄링의 예
Fig. 2 Example of two-step scheduling

3.2 알고리즘의 세부 구성

3.2 절에서는 2단계 디스크 스케줄링의 세부 요소들에 대한 설명을 제시한다.

3.2.1 태스크 필터 및 서비스 제어 테이블

태스크 필터는 각 요구의 클래스를 구분하여 실시간 요구는 주기 큐로 비실시간 요구는 비주기 큐로 입력한다. 그리고, 실시간 요구의 경우 표 1 의 서비스 제어 테이블을 참조하여 라운드 당 보장된 대역폭을 초과할 경우 대기 큐로 입력하여 디스크가 휴지 상태에 있거나 다음 라운드의 시작

표 1. 디스크 서비스 제어 테이블
Table. 1 Disk service control table

기호	정의
R	MOD 서버의 라운드 길이(초)
D_{band}	통계적인 디스크 총 대역폭
W_r	실시간 클래스의 가중치
W_{nr}	비실시간 클래스의 가중치
NT	MOD 저장 서버에 활동중인 총 태스크의 수
r_i	태스크 i 의 라운드당 요구 블록 수
$_ij$	태스크 i의 요구 블록 번호 j
SB_i	태스크 i가 현재주기에서 서비스 받은 블록의 수
WBi	태스크 i 가 대기 큐에서 대기중인 블록의 수
t_{seek}	디스크의 최대 탐색 시간
t_{rot}	디스크의 최대 회전 지연 시간
t^{zone}_{trans}	디스크 존의 전송률
B	파일 시스템의 블록 크기
λ_r	임의의 라운드에서 실시간 클래스의 총 서비스 시간
λ_{nr}	임의의 라운드에서 비실시간 클래스의 총 서비스 시간

시간에 서비스 받도록 한다.

3.2.2 허용 제어 및 대역폭 보장

MOD 서버는 새로운 세션이 생성될 때마다 커널의 블록 디바이스 루틴에게 대역폭 보장을 요구한다. 이때, 블록 수로 변환된 세션의 주기 당 요구 대역폭이 인자(Parameter)로 전달된다. 이 디바이스 루틴은 표 1 의 Dband 즉 총 디스크 대역폭을 기준으로 허용 여부를 결정한다. 이 총 대역폭은 결정적(deterministic) 또는 통계적(stochastic)으로 결정될 수 있는데, MOD 서버가 연성 실시간 응용이므로 통계적 방법으로 Dband 를 결정한다. 이 대역폭 결정은 현재 평균 대역폭으로 설정되는데 정확한 대역폭의 결정으로 향후 연구 과제이다. 아래의 식에서 X 가 비실시간 가중치에 대한 실시간 가중치의 비율이라면 실시간 클래스에는 Dband * p, 비실시간 클래스에게는 Dband * q 만큼의 대역폭이 할당된다.

$Let\ X = \frac{w_r}{w_{nr}}$, (비실시간 가중치에 대한 실시간 가중치의 비율)

$q = \frac{1}{X+1}$, (전체 디스크 대역폭에 대한 실시간 요구의 비율)

$p = \frac{X}{X+1}$, (전체 디스크 대역폭에 대한 비실시간 요구의 비율)

Then, 실시간 요구에 대한 대역폭 = $p * D_{band}$
비실시간 요구에 대한 대역폭 = $q * D_{band}$

실시간 태스크는 세션이 시작될 때 측정된 디스크 대역폭을 기준으로 허용 여부가 결정된다. 새로운 실시간 태스크를 포함한 전체 실시간 요구들에 대한 서비스 시간(r)이 실시간 클래스에 할

```
void request_filter ()
{
  if (request belongs to best-effort requests)
    push in non-periodic queue;
  if (request belongs to multimedia requests) {
    if ( ri == SBi) {
      push in watiing queue;
      WBi = WBi + 1;
    } else {
          push in periodic queue;
          SBi = SBi + 1; }
  }
}
```

그림 3 요구 필터 알고리즘
Fig. 3 Request filter algorithm

당된 시간($R * W_r$)내에 서비스 가능한지 검사한다. 총 서비스 시간은 다음 수식 (1) 에 의해 계산된다[6].

$$\lambda_r = 2 \times t_{seek} + \sum_{i=1}^{NT} \frac{t_{rot}}{2} + \sum_{i=1}^{NT} \frac{B \times r_i}{t^{zone}_{trans}} \quad (1)$$

디스크의 서비스 시간은 수식 (1)과 같이 탐색시간, 회전지연시간, 그리고 전송시간으로 결정된다. 현재 MOD에서는 대역폭의 보장을 위해 탐색

시간은 최악의 경우를 고려하는데 디스크 헤드가 모든 실린더를 한번 왕복하는데 소요되는 시간($2 \times T_{seek}$)을 사용한다. 회전 지연의 경우 평균 회전 지연 시간 ($T_{rot} / 2$)를 사용한다. 그리고, 전송 시간은 수식 1 에서 보듯이 각 파일이 저장된 존의 전송률을 채택하고 있다. 이것은 MOD 서버에 있어서 각 세션의 정확한 디스크 대역폭 사용량을 예측하여 서버의 처리율을 높일 수 있다.

```
weight_filter ()
{
  unit_npq = Wnr * Dband;
  unit_pq = Wr * Dband;
  while ( one period ) {
    process unit_npq in non-periodic queue;
    process unit_nq in periodic queue;
  }
  process waitig queue;
  // 서비스 제어 블록을 업데이트
  for (i <= NT) {
    SBi = WBi;
    WBi = 0;
  }
}
```

그림 4. 가중치 필터 알고리즘
Fig. 4 Weight filter algorithm

3.2.3 비주기 큐, 주기 큐, 대기 큐

태스크 필터에 의해 분류된 디스크 요구는 비주기 큐 또는 주기 큐로 삽입된다. 비주기 큐는 best-effort서비스를 위한 큐로서 일정한 대역폭이 할당되어 서비스 된다. 이 비주기 큐의 요구들은 공정한 서비스의 보장과 기아 현상을 방지하기 위해서 FCFS로 정렬되어서 라운드 내에 서비스 가능할 경우만 SCAN 큐로 입력된다.

주기 큐는 멀티미디어 서비스 요구들을 저장하는 큐이다. 태스크 필터로부터 분리된 멀티미디어 서비스 요구들을 디스크 서비스 테이블을 참조하여 주기내 할당된 대역폭에서 사용하지 않은 대역폭이 있으면 주기 큐에 삽입한다.

멀티미디어 태스크의 요구 중에서 주기 내의 할당된 대역폭을 초과한 요구들에 대해 대기 큐에 삽입하여 다음 주기에 처리되도록 한다. 새로운 주기가 시작될 때 대기 큐의 요구들을 한번에 처리하고 디스크 서비스 테이블을 갱신한다. 그림 3은 앞서 설명한 일련의 처리 과정을 알고리즘화 한 것이다.

3.2.4 가중치 기반 라운드 로빈 스케줄러

분류된 요구들을 SCAN 큐로 보내기 위해서는 각 큐에서 요구들을 추출하여 2단계 큐로 보내는 알고리즘이 필요하다. 그림 4 는 가중치 기반의 라운드 로빈 필터에서 각 큐의 요구를 처리하는 알고리즘이다. 이 알고리즘은 우선 각 큐에서 추출할 요구 블록의 개수를 계산한다. 그리고 한 주기 동안 반복적으로 각 큐에서 블록을 추출하여 SCAN 큐로 보내게 된다. 한 주기가 끝나게 되면 대기 큐에 있는 요구 블록들을 한번에 처리하고 서비스 테이블을 갱신한 다음 주기를 다시 시작하게 된다.

IV. 분석

본 장에서는 2단계 디스크 스케줄링을 분석하고자 한다. 분석 대상은 대역폭 보장, 공정성이다.

4.1 대역폭 보장(bandwidth guarantees)

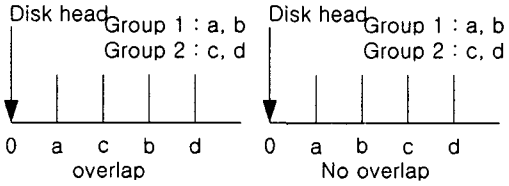

그림 5 중복된 요구와 중복되지 않은 요구의 예
Fig. 5 Examples of overlaped and non-overlaped request

2단계 큐잉 시스템 내에서 각 태스크가 주기 당 서비스 받은 시간이 실제 요구 시간보다 같거나 클 경우 모든 요구들이 주기 당 서비스 받은 시간들의 합이 모든 태스크의 요구 블록들의 서비스 시간보다 크면 대역폭이 보장된다[7]. 서비스 시

간은 서비스 블록의 수와 동일하므로 이것은 수식 (2)로 표현된다.

$$If\ (SB_i \geq r_i)\quad then\quad \sum_{i=1}^{NT} SB_i \geq \sum_{i=1}^{NT} r_i \qquad (2)$$

저장 서버의 요구를 실시간 클래스와 비실시간 클래스로 분류하여 다단계 큐잉 시스템으로 입력한 후 하나의 독립된 비주기 큐로 합병할 경우 대역폭의 보장은 디스크 헤드의 탐색 시간에 좌우된다. 본 논문에서는 최악의 경우를 고려하므로 회전 지연 대기 시간은 전체 서비스 시간에 큰 영향을 미치지 않는다. 이 식의 증명을 위해 두 개의 그룹이 그림 5과 같이 각각 *a,b* 와 *c,d*블록을 요구한다고 가정하자. 각 그룹의 요구가 합병되면 요구가 중복되는 경우와 그렇지 않은 경우가 발생되는데, 서로 다른 클래스의 요구가 하나의 큐로 합병될 때 발생되는 모든 경우를 포함한다.

중복될 경우 그룹 1과 그룹의 2의 탐색 시간은 다음과 같다.

$$S1 = S_{0a} + S_{ab},\ S2 = S_{0c} + S_{cd} \qquad (3)$$

비주기 큐로 합병되었을 경우 탐색 거리는 다음과 같다.

$$S_{merged} = S_{0a} + S_{ac} + S_{cb} + S_{bd} \qquad (4)$$

수식 (2)를 만족시키기 위해서는 $S1+S2 \geq S_{merged}$ 임을 보이면 된다.

$$S_{0a}+S_{ab}+S_{0c}+S_{cd} \geq S_{0a}+S_{ac}+S_{cb}+S_{bd} \qquad (5)$$

그런데, $S_{0c} \geq S_{ac}$, $S_{ab} \geq S_{ac}$, 그리고 $S_{cd} \geq S_{bd}$ 이므로 위의 수식 (5)는 만족된다.

중복되지 않을 경우 그룹 1과 그룹의 2의 탐색 시간은 다음과 같다.

$$S1 = S_{0a} + S_{ab},\ S2 = S_{0c} + S_{cd} \qquad (6)$$

비주기 큐로 합병되었을 경우 탐색 거리는 다음과 같다.

$$S_{merged} = S_{0a} + S_{ab} + S_{bc} + S_{cd} \qquad (7)$$

수식 (2)를 만족시키기 위해서는 $S1+S2 \geq S_{merged}$ 임을 보이면 된다.

$$S_{0a}+S_{ab}+S_{0c}+S_{cd} \geq S_{0a}+S_{ab}+S_{bc}+S_{cd} \qquad (8)$$

그런데, $S_{0c} \geq S_{bc}$ 이므로 위의 수식 (8)은 만족된다.

4.2 공정성(Fairness)

네트워크의 라우터(router)에서 입력되는 패킷을 다중 큐로 입력하여 각 큐에 가중치를 부여하여 각 패킷이 속한 흐름(flow)에게 공정한 서비스를 제공하는 연구들이 있다[8,9]. 본 연구의 2단계 디스크 스케줄링에서 각 큐에 입력되는 요구들이 공정하게 디스크 대역폭을 사용하는 문제는 일종의 weighted fair queuing 문제[9]와 동일하다. 따라서, 각 큐에 입력된 요구들은 가중치에 따라 공정하게 디스크 대역폭을 사용함을 확인할 수 있다.

V. 성능 평가

2 단계 디스크 스케줄링의 성능을 평가하기 위해서 MOD의 단일 저장 서버상[10]에서 두 가지 종류의 실험을 진행하였다.

그림 6은 다양한 실시간 클래스의 부하에서 비실시간 프로그램이 100 MB의 파일을 읽는데 소요되는 시간을 나타내고 있다. 이 실험에서 비실시간 클래스의 가중치 q 는 0.95, 0.5, 0.05 이다. 가중치 q가 0.95 라는 것은 디스크 대역폭의 95%를 비실시간 클래스에 할당하는 것이므로, 그림 6에서 실시간 클래스의 부하에 관계없이 일정한 경과 시간을 나타내고 있다. 이 경우에서 경과 시간의 점진적인 증가는 CPU와 디스크 대역폭을 실시간 클래스에 의해 일정량 할당되기 때문이다.

반면, q가 0.05, 즉 비실시간 클래스에 디스크 대역폭이 5%만 할당될 때 실시간 클래스의 수가 증가할수록 급격히 증가하는 것을 볼 수 있다.

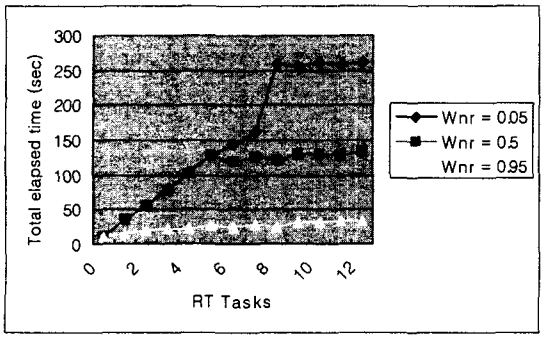

그림 6. 실시간 요구수에 대한 비실시간 요구의 읽기시간
Fig. 6 NRT read time vs. RT load

그림 7은 실시간 태스크의 라운드 당 소요시간에 대한 비실시간 클래스의 영향을 나타내고 있다. 가중치 p가 0.95, 0.05일 때, 즉, 디스크의 대역폭의 95%, 5% 가 실시간 클래스에 할당 되어 있을 때 그림 8에서 보듯이 마감시간 실패회수가 거의 일정한 것을 확인할 수 있다. 이것은 실시간 클래스의 디스크 자원에 대한 QoS가 보장되고 있음을 증명한다.

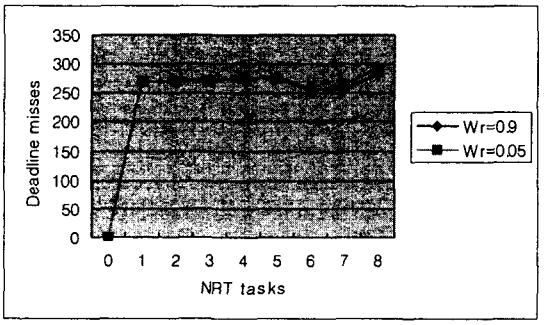

그림 7. 실시간 태스크의 마감시간 실패 회수
Fig. 7 Deadline misses of RT vs. NRT load

VI. 결 론

본 논문에서는 실시간 요구와 비실시간 요구가 동시에 존재하는 MOD서버를 위한 새로운 2단계 디스크 스케줄링 기법을 제안하였다. 제안된 큐잉 시스템 모델은 각 요구를 분류하고 각 큐에 가중치를 할당하여 서비스의 공정성과 대역폭을 보장하는 태스크 의존적 레벨과 요구의 종류에 관계없이 디스크 헤드의 탐색 최적화를 목적으로 하는 태스크 독립적 레벨로 구성된다. MOD서버의 실시간 요구는 주기 당 대역폭을 사용자 수준의 QoS를 명시하도록 하여 대역폭을 보장 받는다.

성능 평가를 위해 리눅스 커널 버전 2.x의 블록 디바이스 드라이브 수준에서 2 단계 큐잉 시스템을 설계 및 구현하였다. 다양한 가중치에 따라 실험이 진행되었는데 각 클래스의 요구는 QoS에 따라 디스크 대역폭을 보장 받고, 공정하게 디스크 대역폭을 사용하였다. 또한, 기존 단일 큐잉 시스템의 MOD보다 적은 마감시간 실패율을 보였다. 2단계 큐잉 시스템은 태스크 의존적 레벨의 각 큐에 가중치를 할당하여 Round Robin으로 서비스하는데 대역폭 할당면에서 태스크사이의 공정성이 보장된다. 따라서, 제안된 2단계 디스크 스케줄링은 SMOD서버의 마감시간을 보장과 처리율 향상을 동시에 만족함을 확인할 수 있었다.

참고 문헌

[1] A.L.Reddy and J.Wylle, "Disk Scheduling in a Multimedia I/O System,"In Proceedings of the First International Conference on Multimedia", pp 225-223, 1993.

[2] M.S.Chen, D.D.Kandlur, and P.S.Yu, "Optimization of the Group Sweeping Scheduling with Heterogeneous Multimedia Streams," In Proceedings of the First ACM International Conference on Multimedia, pp 235-241, 1993.

[3] J.S Prashant and M.V. Harick, "Cello: A Disk Scheduling Framework for Next Generation Operating Systems,"SIGMETRICS, pp.44-55, 1998.

[4] J. P. Lehoczky and S. Ramos-Thuel. "An optimal algorithm for scheduling soft -aperiodic tasks in fixed priority preemptive systems," In Proceedings of Real Time Systems Symposium, pp.110-123,

1992.
[5] D.P.Anderson, Y.Osawa, and R. Govindan, "A File system for Continuous Media," ACM Trans, on Computer Systems, Vol.10 No.4, Nov, pp 311-337, 1992.
[6] C. Ruemmler and J. Wilkes. An Introduction to Disk Drive Modeling. IEEE Computer, pp.89-99, 1994.
[7] W. Ravi, A.L.Narasimha Reddy, "Providing QOS guarantees for disk I/O," ACM multimedia systems 8:57-68 (2000), 2000.
[8] M. Shreedhar, G. Varghese, "Efficient Fair Queuing using Deficit Round Robin," ACM SIGCOMM, pp.231-242, 1995.
[9] D. Stiliadis, A. Varma, "Efficient Fair Queuing Algorithms for ATM and Packet Networks," ACM SIGMETRICS, 1996.
[10] 김정원, "Linux 상에서 확장 가능한 VOD 시스템의 설계 및 구현", 한국멀티미디어학회 논문지, 2권3호, pp265-276, 1999.

저자 소개

김정원(Kim Jeong-Won)

1995년 부산대학교 전자계산학과 (학사)
1997년 부산대학교 대학원 전자계산학과 (석사)
2000년 부산대학교 대학원 전자계산학과 (박사)
2000년~2001년 기술신용보증기금 기술평가역(차장)
2002년~현재 신라대학교 컴퓨터정보공학부 전임강사
※ 관심분야 : 내장형시스템, 멀티미디어, 운영체제

전봉기(Jun Bong-Gi)

1991년 부산대학교 컴퓨터공학과 (학사).
1993년 부산대학교 대학원 컴퓨터공학과 (석사).
1993년~1998년 : 한국통신 연구소 전임연구원.
2003년 8월 : 부산대학교 대학원 컴퓨터공학과(박사).
2003년 9월~현재 : 신라대학교 컴퓨터정보공학부 전임강사
※ 관심분야 : 공간 데이터베이스, 이동체 데이터베이스

윤홍원(Yun Hong-won)

1986년 부산대학교 계산통계학과 졸업(학사)
1990년 한국외국어대학교 경영정보대학원 전자계산학과(석사)
1998년 부산대학교 대학원 전자계산학과(박사)
1996년~현재 신라대학교 컴퓨터정보공학부 부교수
※ 관심분야 : 데이터베이스 시스템, 시간 데이터베이스