연속매체 상영을 위한 실시간 디스크 프리팻칭 기법

임 성 채†

요 약

연속매체(Continuous Media : CM)와 같은 데이터를 온라인으로 상영해야 하는 멀티미디어 시스템에서는 연속매체의 시간제약성을 만족시켜 줄 수 있는 실시간 디스크 스케줄링 기법이 요구되며 이를 통해 상영중인 CM 스트림(stream)의 끊김현상(hiccup)을 막을 수 있다. 이런 디스크 스케줄링을 위해 주기성을 가지는 프리팻칭(prefetching) 기법이 널리 쓰이고 있으며, 이는 연속매체가 상당기간 동안 계속 상영된다는 특성에 기반한 것이다. 본 논문에서도 효과적인 디스크 프리팻칭을 위해, 정시도착채널이란 실시간 디스크 채널을 이용한 스케줄링 기법을 제안한다. 이런 디스크 채널의 생성을 위해 bulk-SCAN 기법이 사용되며 유연한 채널 할당을 위해 실시간 알고리즘인 EDF(earliest-deadline-first) 알고리즘이 이용된다. 제안한 기법은 기존 방법에서와 같이 끊김현상 없는 상영을 제공함은 물론이고, I/O 처리율과 서비스 응답시간 면에서 우수성을 가진다. 논문에서는 이런 성능상의 장점을 시뮬레이션을 통해 보인다

A Real-Time Disk Prefetch Scheme for Continuous Media Playback

Sung Chae Lim†

ABSTRACT

To play back CM (Continuous Media) in online mode, the multimedia system is required to have a real-time disk scheduling scheme that can efficiently fulfill the strict temporal constraints of serviced CM streams to prevent hiccups. In general, such disk scheduling is performed based on the concept of periodic prefetching since a CM stream has a rather long playback time. In this paper, we also propose a periodic prefetching scheme that runs by using real-time disk channels, called *on-time delivery channels*. Since the channels are generated from the bulk-SCAN algorithm and they can be allocated in a very flexible manner based on the EDF (earliest-deadline-first) algorithm, the proposed scheme provides a better performance in terms of I/O throughput and the average response time, as well as hiccup-free playback of concurrent CM streams. To show that the proposed scheme outperforms other methods, we give some simulation results.

키워드 : 실시간 디스크 스케줄링(Real-time Disk Scheduling), 연속매체(Continuous Media), 멀티미디어(Multimedia), 스트리밍 서비스(Streaming Service)

1. 서 론

컴퓨터 통신기술과 컴퓨터 계산능력의 발전으로 인해 고속 통신망을 통한 멀티미디어 데이터의 원격 전송이 가능해졌다[1, 2]. 원격지 사용자로부터 멀티미디어 데이터 전송 요청을 처리하는 멀티미디어 정보 시스템(Multimedia Information System : MMIS)은 주문형 비디오(VOD), 주문형 뉴스(NOD), 원격 교육, 홈 쇼핑, 대화형 TV, 그리고 병원 진료 등과 같이 매우 다양한 응용 범위를 가진다. 이러한 MMIS에서는 동영상, 음성, 그리고 음악과 같이 자료 표현에 있어 시간제약이 강한 연속매체(CM)를 다소 긴 기간 동안 검색해야 하는 요구사항을 가진다[2-5].

CM의 시간 제약성이란 CM 끊김현상이 생기지 않도록 상영기간 동안 일정 크기의 속도로 데이터를 사용할 수 있어야 함을 의미한다. 이런 데이터 사용 속도를 *상영대역폭*(playback rate)이라 부른다. MMIS는 이런 시간제약성을 만족시키기 위해 사용되는 데이터 보다 항상 앞서서 데이터를 버퍼로 읽어 들여야 한다. 이처럼 상영대역폭에 앞선 데이터 읽기를 read-ahead라고 하며 read-ahead가 상영 기간 동안 항상 지켜져야 한다[6]. 동일한 디스크에서 여러 개의 CM 스트림을 서비스하고자 할 때 각 스트림 별로 read-ahead를 충족시키기 위한 방법들이 연구되었다[3, 6-12]. 이 중에서 라운드로빈 방식에 의한 CM 상영 기법이 널리 쓰이고 있다[13-16].

라운드로빈 방식의 CM 상영기법은 고정된 시간 길이의 주기를 정해두고 매 주기마다 서비스 하는 스트림들에 라운드로빈 방식으로 디스크 대역폭을 할당하는 기법이다. 이때 한 주기에 대해 각 스트림이 할당받는 디스크 대역폭은 스트림의 상영대역폭 크기에 비례하여 정해진다. 할당된 디스크 대역폭을 이용하여 읽은 CM 데이터는 다음 주기 동안의 상영에 사용되며, 이와 동시에 같은 크기의 데이터가 읽혀진다. 즉, 일정한 크기의 시간 간격으로 정해진 크기의 데이터를 프리팻칭 함으로써 read-ahead 조건을 지킨다. 서비스되는 모든 스트림에 대해 동일한 주기를 가지고 라운드로빈

† 정 회 원 : 코리아와이즈넛 연구소 수석연구원
논문접수 : 2004년 5월 17일, 심사완료 : 2004년 11월 11일

의 디스크 대역폭 할당이 행해짐으로써, 비교적 간단하게 여러 개의 스트림을 끊김현상 없이 상영할 수 있다. 하지만 변화하는 사용자 요청 상황에 따라 디스크 대역폭을 유연성 있게 사용하기 어렵고 모든 스트림이 동일한 프리팻칭 주기를 가짐에 따라 디스크 대역폭의 낭비가 심하다는 문제가 있다.

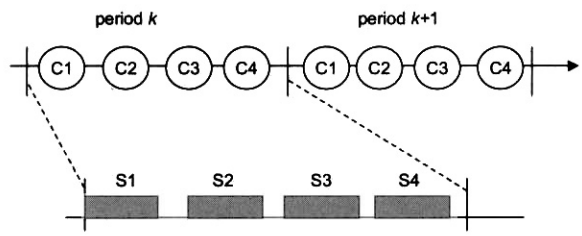
본 논문에서는 이런 기존 라운드 로빈 방식의 문제점들을 해결하기 위해 새로운 CM 상영기법을 제안한다. 제안하는 기법은 각 스트림 별로 서로 다른 크기의 프리팻칭 주기를 허용하고, 이에 따라 적절한 크기의 데이터를 bulk-SCAN에 의해 읽어 들인다. 시스템의 상황 변화에 따라 잔여분의 디스크 대역폭을 효과적으로 이용할 수 있음으로 해서 기존 방법에 비해 좋은 성능을 보인다. 본 논문의 구성은 다음과 같다. 2장에서는 기존 CM 상영 기법인 라운드 로빈 기법에 대해 살펴보고, 3장에서는 제안하는 기법의 기본 개념을 기술한다. 4장에서는 bulk-SCAN에 의한 디스크 채널 생성과 승인 통제를 위한 함수를 기술하고, 5장에서는 제안하는 CM 상영 서버인 S^{EDF*}의 알고리즘을 기술한다. 6장에서는 제안된 기법의 성능실험의 결과를 보이고 7장에서 결론을 맺는다.

2. 관련 연구

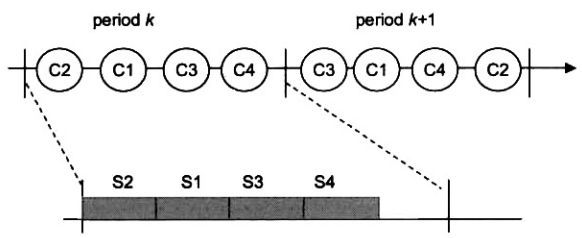
디스크 검색 효율을 높이기 위해 CM 객체를 정해진 크기로 클러스터링하여 저장하는 것이 보통이다. 클러스터링된 각각의 부 객체를 데이터 세그먼트라 부를 때, CM 객체는 이런 데이터 세그먼트들의 연속체이다. 데이터 세그먼트들은 모두 동일한 크기를 가지며, 하나의 데이터 세그먼트를 읽을 수 있는 디스크 대역폭의 할당을 *채널*(channel)이라고 한다. 일반적으로 데이터 세그먼트의 크기는 디스크 트랙을 기본 단위로 삼는다. 이는 CM 데이터가 디스크 공간에서 연속적으로 읽혀지는 특성을 이용하여 디스크 탐색시간을 크게 줄이기 위한 고려이다[13]. 본 논문에서도 모든 데이터 세그먼트는 동일 실린더에 위치한 하나 이상의 트랙으로 이루어 짐을 가정한다. 그리고 [17]에서와 같이 디스크 드라이브에 트랙 하나를 버퍼링 할 수 있는 내부 버퍼를 둘 경우 트랙 전체를 읽을 때는 회전 지연 시간을 없앨 수 있다. 본 논문에서는 이런 점을 고려하여 채널 시간으로 실제 데이터 전송에 걸린 시간(transfer time)과 탐색시간(seek-time) 만을 고려한다.

라운드로빈 방식은 시스템에서 정한 시간주기 내에 생성 가능한 채널을 각 스트림에 일정하게 할당하는 방식이다. 이 때 각 스트림은 매 주기마다 할당된 채널 수 만큼의 데이터 세그먼트를 프리팻칭하여 read-ahead를 지켜나간다. 기존의 라운드로빈 방식은 채널 생성 방식에 따라 크게 Fixed-Stretch 방식과 Sweep 방식으로 나뉜다. Fixed-Stretch 방식은 매 주기마다 일정한 스트림 순서로 해당 세그먼트를 읽어 들이고, Sweep 방식은 elevator-SCAN 방식을 이용하며 읽혀지는 스트림들의 순서는 일정하지 않다. Fixed-Stretch 방식은 각 채널 시간에 전체 디스크 이동에 필요한 탐색시간이 포함되므로 인해 Sweep 방식에 비해 디스크 사용효율이 떨어진다. 하지만 Fixed-Stretch 방식은 각 스트림에 대해 일정한 시간간격으로 프리팻칭을 할 수 있으므로 버퍼 요구량은 상대적으로 작다[16].

두 기법의 채널 생성에 대한 예를 (그림 1)에서 보인다. 원안에 표시된 기호 C_i는 스트림 S_i가 사용한 채널을 표시한다. 한 주기에는 네 개의 채널이 생성되며 편의상 동수의 스트림이 각각 한 개의 채널을 사용하도록 했다. 각 채널의 실제 생성 시간(데이터 전송시간 + 탐색 시간)은 아래 쪽의 빗금친 네모로 표시한다. 그림에서 보이는 바와 같이 Fixed-Stretch 방식은 각 채널에 잔여시간이 발생하고, Sweep 방식은 각 주기의 끝에 잔여시간이 생긴다. 라운드로빈 기법은 이런 잔여분의 시간 동안 디스크를 유휴상태로 둠으로써 일정한 크기의 데이터만이 프리팻칭 되도록 하여 버퍼 요구량을 일정 수준으로 유지한다.



(a) Fixed-Stretch Scheme



(b) Sweep Scheme

(그림 1) 라운드로빈 방식의 채널생성 예

라운드로빈 방식에서 주기를 L_p로, 데이터 세그먼트의 크기를 S_{ds}로 표시하면 각 채널을 통해 S_{ds}/L_p (bits/sec) 크기의 디스크 대역폭을 얻을 수 있다. 이 크기를 B_{ch}로 표시할 때 B_i 를 상영 대역폭으로 하는 스트림 S_i를 상영승인(admission)하기 위해서는 주기당 $\lceil B_i/B_{ch} \rceil$ 의 채널을 해당 스트림에 할당할 수 있어야 한다. 이 때 B_i의 크기가 B_{ch}의 정수 배가 아닌 경우 디스크 낭비가 초래될 수 있다.

상영승인이 된 후에 실제 상영 시작은 할당 받은 채널을 이용하여 한번의 주기 동안 사용될 데이터가 버퍼로 읽혀진 시점 이후에 가능하다. 이렇게 상영 시작 전의 프리팻칭을 특별히 *초기프리팻칭*이라고 부른다. Fixed-Stretch 방식의 경우 초기 데이터 프리팻칭이 수행된 후 곧바로 상영이 시작될 수 있으나, Sweep 방식은 elevator-SCAN 방식의 특성상 초기 프리팻칭이 끝난 후에도 다음 주기가 시작되기 전까지는 상영을 시작할 수 없다. 이처럼 두 방식간에 다소 차이는 있지만 상영 승인이 된 후에 상영 개시까지의 시간은 L_p에 비례한다[16].

3. 제안하는 CM 상영 기법의 기본 개념

CM의 read-ahead 요구를 지키기 위해 기존 라운드로빈 방식에서 사용한 주기적인 프리팻칭 기법은 여러 개의 CM 스트림을 서비스할 때 매우 유용하다. 논문에서 제안하는 방법도 이런 주기적인 프리팻칭 기법을 이용한 방식이다. 하지만 제안하는 기법은 기존 라운드로빈 방식과는 달리 각 스트림 별로 서로 다른 프리팻칭 주기를 줄 수 있고, 스트림들에 채널을 할당하는 방식 역시 단순한 라운드로빈 방식이 아닌 보다 유연한 채널 할당 방식을 채용한다.

시스템의 시간을 일정한 간격 T로 나누고 나뉜 각 시점을 기본 간격점이라고 부르자. 임의의 스트림 S_i의 프리팻칭 주기를 항상 T의 정수 배 크기로 하고 이 정수값을 p_i로 표시한다. 이제 각 스트림의 상영 시작 시간을 기본 간격점 중 하나와 같도록 두면, 모든 스트림의 프리팻칭 마감시간은 기본 간격점 상에만 존재한다. 예를 들어 s번째 기본 간격점 $s\times T$에서 상영 시작된스트림 S_i는 $T\times p_i$ 크기의 프리팻칭 주기를 가지며, $s+n\times p_i$ 번째 ($n=1, 2,\cdots$) 기본 간격점이 프리팻칭 마감시간이 된다.

여러 개의 데이터 세그먼트들을 읽을 때 디스크 이동 거리를 작게 하여 탐색시간을 줄이기 위해서는 디스크 암이 한쪽 방향으로만 움직이며 데이터를 읽어야 한다. 이를 위해 bulk-SCAN이 유용하게 쓰인다. bulk-SCAN이란 디스크 스케줄링을 시작하기 전에 읽어들일 여러 개의 데이터 세그먼트들을 미리 정해주고 한번의 SCAN으로, 즉, 디스크 실린더의 한쪽 끝에서 다른 끝으로 데이터를 읽어들이는 것을 말한다. 이 때 하나의 bulk-SCAN에 의해 생기는 채널들을 동일한 *채널군*(channel group)에 있다고 한다. 동일 채널군에 속한 채널의 전체 개수는 채널군의 크기라 한다. 논문에서는 이런 bulk-SCAN을 이용하여 채널을 생성하며, 아래의 상영승인규칙에 따라 상영 승인된 스트림 집합의 채널 요구량을 CM 서버에서 생성 가능한 채널 수보다 항상 같거나 작게 만든다.

[상영승인규칙]

S_1에서 S_j까지 j개의 스트림이 상영 승인된 상태에서 $(j+1)$번째 스트림을 상영 승인하기 위해서는 다음 부등식이 만족되어야 한다. 여기서 k는 T 시간 내에 항상 생성 가능한 채널군의 크기이며 B_i는 스트림 S_i의 상영대역폭을, LCM은 최소공배수를 나타낸다.

$$k \times \mathrm{LCM}(p_1, p_2, \cdots, p_{j+1}) \geq \sum_{i=1}^{j+1}(m_i \times n_i), \text{ where}$$

$$m_i = \mathrm{LCM}(p_1, p_2, \cdots, p_{j+1})/p_i, n_i = \lceil B_i \times T \times p_i / S_{ds} \rceil .$$

식에서 n_i는 스트림 S_i의 *주기당 상영채널수*라고 부르며 스트림의 프리팻칭 주기를 정하는 p_i에 따라 결정된다. 이처럼 스트림 S_i에 따라 주어지는 정수쌍 (p_i, n_i)를 *채널할당패턴*(Channel Allocation Pattern : CAP)이라 부르고, S_i가 CAP에 따라 할당받는 디스크 대역폭의 크기는 $S_{ds}/(p_i\times T) \times n_i$ (Mbps)와 같다. LCM($p_1, p_2, \cdots, p_{i+1}$)은 p_i들의 최소공배수이며 이 값은 스트림 S_1, S_2, $\cdots$, S_{j+1}들의 프리팻칭 주기들이 일치하는 최소 길이와 같다. 이를 *최소공통주기*라 부르겠다.

예를 들어 S_{ds} = 0.6 Mbits, T = 0.5 초, 그리고 k = 4라고 가정하자. 그리고 상영 승인된 스트림 집합 {S_1, S_2, S_3, S_4}의 상영대역폭을 B_1 = 1.5 Mbps, B_2 = 0.6 Mbps, B_3 = 1.6 Mbps, B_4 = 0.8 Mbps라고 하자. 스트림 S_i의 CAP를 C_i라고 하면 C_1 = (4, 5), C_2 = (2, 1), C_3 = (3, 4), C_4 = (3, 2)로 두어 상영승인 규칙을 만족시킨다. 주어진 스트림 집합에 대한 CAP 생성 방식은 다음 장에서 기술한다. 이들 CAP 집합은 최소공통주기 12 안에 45개의 채널만을 요구하므로 이 기간 동안의 최대가용 채널수 48을 넘지 않는다. 이런 스트림 집합이 서비스 될 때, 시스템에는 최소공통주기 12마다 3개의 채널이 *자유채널*(free channel)로 남는다.

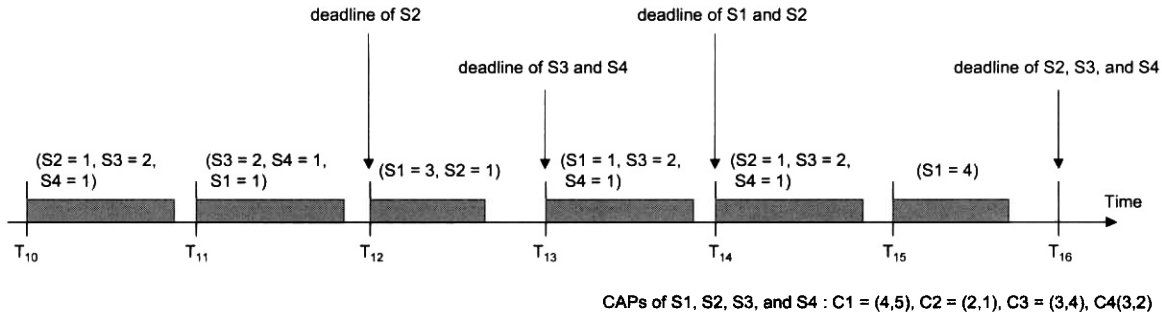
이런 상영승인 방법과 기존 라운드로빈 방식의 상영승인 방법과 비교해 보자. 기존 라운드 로빈 방식에서는 모든 스트림들이 L_p를 프리팻칭 주기로 하기에 디스크 대역폭 할당의 granularity가 고정되어, 다양한 상영대역폭이 요구될 때는 디스크 대역폭의 낭비가 심해진다. 예를 들어 L_p = 0.5 초라면, 채널 하나의 대역폭은 1.2 Mbps가 된다. 이제 앞선 예의 스트림 집합을 상연승인 한다면 S_1, S_2, S_4에 각각 두 개, 한 개, 한 개의 채널을 할당할 수 있으며, S_3는 상영 승인을 받을 수 없다. 이것은 상영대역폭과 할당되는 디스크 대역폭과의 차이에서 오는 단편화(fragmentation) 현상에 의해 스트림의 동시 상영수가 작아짐을 의미한다. 이런 단편화 현상을 줄이기 위해서 L_p를 크게 설정할 수도 있으나 이런 경우 상영시작까지의 시간이 길어지는 문제가 생긴다. 따라서 라운드로빈 방식에서는 각 스트림에 다양한 상영대역폭이 요구되는 경우 효과적인 디스크 대역폭의 할당이 어렵다.

이에 반해 제안하는 방식은 작은 크기의 T를 기반으로 하여 스트림의 프리팻칭 주기를 각 스트림의 상영 대역폭에 맞게 T의 배수 길이로 적절히 결정함으로써 단편화 현상을 크게 줄이면서도 초기 상영 지연이 커짐을 막을 수 있다. 이를 위해 본 논문에서는 주기성 작업의 실시간 스케줄링 알고리즘으로 널리 쓰이는 EDF 알고리즘을 이용한다.

bulk-SCAN에 의해 생성되는 채널 하나는 스케줄링 작업 시간의 단위로 프리팻칭 주기는 주기성 작업의 발생 주기로 간주하면, 각 스트림에 채널을 할당하는 것을 주기성 작업의 CPU 스케줄링으로 생각할 수 있다. EDF 알고리즘의 경우 CPU 점유율이 100%일 때까지 주기성 작업을 스케줄링 할 수 있으므로 상영승인규칙을 만족하는 모든 스트림 집합들에 채널을 할당할 수 있다[18, 19].

(그림 2)는 앞서 사용한 스트림 집합 {S_1, S_2, S_3, S_4}에 대해서 이런 채널 할당 방법을 적용한 예를 보인다. 기본 간격점을 T_j로 표시하였고 모든 스트림들이 T_{10}에서 시작하도록 했다. k를 4라고 가정하였으며 스트림들의 CAP는 그림 하단부에 나타냈다. 그림에서 빗금친 사각형은 각 bulk-SCAN의 실제 수행시간을 나타내며, 그 위 괄호안의 내용은 스트림 별로 할당된 채널 수를 표시한다.

T_{10}시점에서 가장 가까운 프리팻칭 마감시간을 가지는 스트림은 S_2이며, S_2의 주기당 상영채널수 1만큼의 채널을 S_2에 먼저 할당한다. 남은 3개의 채널은 스트림 S_3, S_4가 모두



(그림 2) 네 개의 스트림들에 대한 EDF 방식의 채널 할당 예

같은 프리팻칭 마감시간을 가지므로 이 두 개의 스트림에 고르게 분배한다. T_{11}에서는 S_3와 S_4의 상영채널수 만큼을 모두 할당한 후에 S_1에 채널을 할당하여, S_1, S_3, S_4에 각각 1개, 2개, 1개의 채널이 할당된다. T_{12}에서도 이런 방식에 따라 가장 빠른 프리팻칭 마감시간을 가지는 스트림들 S_1과 S_2에 각각 3개와 1개의 채널을 할당한다.

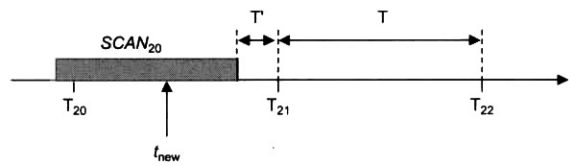
(그림 2)에서는 각 bulk-SCAN의 시작 시간을 기본 간격점에 일치시켰다. 하지만 제안한 프리팻칭 주기의 결정 방식에 따르면 프리팻칭 구간들이 서로 중첩(overlapping)될 수 있기 때문에 하나의 채널군 생성을 끝낸 후, 다음 기본간격점이 될 때까지 디스크를 유휴상태로 둘 필요는 없다. 이것은 라운드로빈 방식에서 한 주기 내에 필요한 채널을 생성하고 나면 다음 주기가 시작될 때까지는 새로운 채널 요구량이 없기 때문에 디스크를 유휴상태로 두어야 하는 것과는 차이가 있다. 즉, 제안한 방식은 채널 요구량이 있다면 bulk-SCAN을 연달아 수행할 수 있다. 아래는 제안하는 채널 생성 방식이다.

[채널생성규칙]

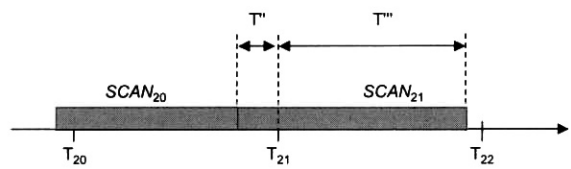
시간구간 (T_{j-1}, T_j) $(j = 1, 2, \cdots)$에서 시작된 bulk-SCAN의 수행 마감시간은 기본 간격점 T_{j+1}이 되며, 이 구간에 수행된 bulk-SCAN을 $SCAN_j$로 표시한다. $SCAN_j$가 기본 간격점 T_j 전에 종료된 경우 T_j까지 $SCAN_{j+1}$를 수행하지 않는다. 이런 경우가 아니라면 $SCAN_{j+1}$을 곧바로 수행할 수 있다.

각 채널군이 차지할 수 있는 최대 디스크 시간을 최대 가용스캔시간이라 부르고, 각 채널군의 종료시점과 마감시간(즉, 다음 기본간격점)까지의 시간차를 잔여 디스크 시간이라 하자. 채널생성규칙에 따라 j번째 채널군의 최대 가용스캔시간은 $j-1$번째 채널군에서 생긴 잔여 디스크 시간에 T를 합한 크기와 같다. 이런 잔여시간이 계속 더해지면 bulk-SCAN의 최대 가용스캔시간이 $2T$까지 커질 수도 있다.

(그림 3)은 채널생성규칙에 따른 bulk-SCAN 방식의 예이며, 앞의 (그림 2)의 스트림 집합을 고려한 것은 아니다. (그림 3)(a)에서 $SCAN_{20}$을 하고 남는 잔여분의 시간 T'는 여유분의 디스크 시간이다. 여유분의 채널 시간 T'를 새로운 스트림에 모두 할당하여 줌으로써 디스크 이용률을 높이는 동시에 스트림 상영을 앞당긴다. 이런 채널 사용을 (그림 3)(b)에서 보인다. 그림에서 편의상 S_8이 사용한 시간을 T''로 나머지 스트림에 할당된 시간을 T'''로 표시했다. 물론 bulk-SCAN으로 채널을 생성하므로 S_8에 할당된 채널이 그림과 같이 앞쪽에만 생기는 것은 아니지만 편의상 T''를 앞쪽에 표시했다.



(a) playback request of S_8 at time t_{new}



(b) initial prefetch for S_8 through $SCAN_{21}$

(그림 3) 제안하는 채널 생성 방법 및 그 사용 예

제안한 방식은 프리팻칭 마감시간이 기본간격점에만 있도록 하여 각 채널군의 최대 가용스캔시간 크기를 항상 T보다 크거나 같도록 두었다. T 시간 동안 생성 가능한 채널 수를 k로 둔다면, bulk-SCAN의 특성상 T 이상의 시간동안에는 항상 k와 같거나 많은 수의 채널 생성이 가능하다. 따라서 n개의 채널군에는 최소 $n \times k$개 이상의 채널을 만들 수 있고, 이런 최소 채널수의 범위 내에서만 스트림의 상영 승인을 허용하여 끊김현상을 없앤다. 역으로 생각해 보면 bulk-SCAN과 EDF 알고리즘을 적용하기 위해 기본간격점을 설정했다고 할 수 있다.

4. 상영 서버를 위한 주요 함수

4.1 Bulk-SCAN을 이용한 OTDC 생성

bulk-SCAN 기법에 의해 k개의 채널을 생성하는데 걸리는 시간 $T_{scan}(k)$은 아래의 식 (1) 과 같다. 식에서 R_t는 데이터 전송 속도를, $seek_time(d_i)$는 스캔방식에 따라 i번째 데이터 세그먼트를 읽기 위해 이동한 실린더 거리(즉, 실린더 수) d_i에 대한 탐색시간을 나타내는 함수이다. 이 때 d_{k+1}은

데이터 검색이 끝난 후에 디스크 암을 실린더의 어느 한쪽 끝에 위치시키기 위한 이동거리이고, d_i들의 합은 디스크 실린더의 전체 수 N_{cyl}과 같다.

$$T_{scan}(k) = k \times (S_{ds}/R_t) + \sum_{i=1}^{k+1} seek\_time(d_i),$$

$$\text{where } \sum_{i=1}^{k+1} d_i = N_{cyl} \quad (1)$$

식 (1)에서 S_{ds}/R_t는 고정된 값이므로 임의의 정수 k에 대한 $T_{scan}()$ 값은 d_i들에 따라 변하게 된다. 따라서 예상되는 $T_{scan}(k)$ 최대 크기는 탐색시간들 합의 상한값에 좌우되고, 이 상한값은 탐색시간 함수 *seek_time*()의 특성을 이용하여 얻을 수 있다. [17, 20]에 따르면 스캔 방식에 따라 여러 개의 데이터 세그먼트를 읽어 들일 때 탐색 시간의 합이 가장 커지는 경우는 읽고자 하는 데이터 세그먼트들이 실린더 상에 고르게 분포되어 있을 때이다. 이것은 함수 *seek_time*() 이 디스크 이동 거리에 대해 단조증가이고 위로 볼록한 (convex) 모양이란 성질에 의해 증명된다. 따라서 아래의 식 (2)와 같이 얻을 수 있다.

$$T_{scan}(k) \le n \times (S_{ds}/R_t) + (k+1) \times seek\_time(\lceil N_{cyl}/(k+1) \rceil) \quad (2)$$

본 논문에서는 채널그룹 내의 채널수가 동적으로 정해지며 각 채널들은 항상 마감시간을 지킬 수 있다. 논문에서 제안하는 디스크 채널을 OTDC(On-Time Delivery Channel)로 부르기로 한다. 아래 (그림 4)는 식 (2)를 이용하여 주어진 최대 가용스캔시간 내에 생성 가능한 최대 채널수를 구하는 함수 *Calc_Max_OTDC()*를 보인다. 이 함수는 $SCAN_j$를 시작하기 전에 불려지며, 이 함수에 의해 최대 채널 수가 정해지고 이 값의 범위 내에서 스트림들에 채널을 할당한다. 아래 그림에서 매개변수 *deadline*은 T_{j+1}을 값으로 가진다.

```
function Calc_Max_OTDC(current_time, deadline)
    Time = deadline - current_time; /* available time */
    Choose the largest integer k satisfying the inequality below :
        k×(S_ds/R_t)+(k+1)×seek_time(⌈N_cyl/(k+1)⌉)×Time ;
    Return k ;
end.
```

(그림 4) OTDC의 개수를 계산하는 *Calc_Max_OTDC*()

4.2 CAP 할당에 따른 승인 통제 기법

N_{otdc}를 T 시간 내에 생성할 수 있는 최대 OTDC 수라고 하자. 주어진 채널생성규칙에 따라 하나의 OTDC를 점유하는 경우 최소 S_{ds}/T의 디스크 대역폭을 얻을 수 있다. 이 값을 OTDC 대역폭 크기 B_{otdc}라 할 때, $B_{otdc} \times N_{otdc}$는 전체 사용 가능한 디스크 대역폭 크기이다. 이 크기 안에서 스트림들에 상영 승인된다. 즉, 이미 서비스 되고 있는 스트림들의 디스크 대역폭 점유 비율을 R_{used}로 표시할 때, 가용대역폭인 $N_{otdc} \times B_{otdc} \times (1-R_{used})$의 범위에서 새로운 스트림의 상영이 승인된다.

아래 (그림 5)는 B_i를 상영대역폭으로 하는 임의의 스트림에 상연승인이 되었을 때, CAP를 정하는 함수 *Determine_CAP*()를 보인다. 물론 B_i의 크기는 현재 가용 대역폭보다 작아야 하며, CAP는 스트림이 B_i 보다 작지 않은 범위에서 가장 작은 디스크 대역폭을 점유하도록 정한다. 함수의 매개변수 *MAX_PI*는 p_i의 최대값이며, 디스크 대역폭 점유 비율 R_{used}는 0에서 1까지의 범위를 가진다.

```
function Determine_CAP(B_i, MAX_PI)
    For (p = 2 to MAX_PI ) Do
        free_otdc ← p × N_otdc × (1-R_used) ;
        Choose the least integer n_p such that
                n_p ≤ free_otdc and (n_p × B_otdc)/p ≥ B_i ;
    EndFor
    Solve integer p so that (n_p × B_otdc/p - B_i) has the smallest value ;
    R_used ← R_used + n_p/(p × N_otdc) ;
    Return (p, n_p) as the CAP for this stream ;
end.
```

(그림 5) CAP를 정하는 함수 *Determine_CAP*()

상영서버에 의해 l개의 스트림들이 승인된 경우 이 스트림 집합에 의한 디스크 점유 비율 R_{used}는 다음의 식 (3)에서 구할 수 있다.

$$R_{used} = \frac{\sum_{i=1}^{l}(n_i \times B_{otdc}/p_i)}{Total\ Bandwidth} = \frac{\sum_{i=1}^{l}\left(\frac{n_i}{p_1} B_{otdc}\right)}{N_{otdc} \times B_{otcd}} = \frac{1}{N_{otdc}} \sum_{i=1}^{l} \frac{n_i}{p_i} \quad (3)$$

5. 제안하는 CM 상영 서버 S^{EDF^*}

5.1 서버 S^{EDF^*}의 메모리 요구사항

제안하는 상영서버 S^{EDF^*}은 각 스트림의 데이터 프리팻칭을 위한 버퍼링 메모리를 요구하고, 스트림 별로 요구되는 버퍼링 메모리의 크기는 read-ahead된 데이터 크기와 같다. CAP(p_i, n_i)를 가지는 스트림 S_i에 대한 $n_i \times S_{ds}$의 크기를 D_i (Mbits)로 표시할 때 S_i가 read-ahead하는 데이터는 최대 $2D_i$의 크기를 가진다. 이 크기는 프리팻칭된 데이터가 하나도 소모되지 않은 상태에서 다음 주기 동안 사용할 데이터가 모두 프리팻칭되는 것을 고려한 크기이다. 따라서 l개의 스트림들을 서비스할 때 필요한 최대 버퍼 메모리의 크기는 $2\sum_{i=1}^{l} D_i$이다.

하지만 이런 최대 버퍼 크기는 각 스트림에 독립적으로 메모리를 할당하는 경우 필요한 크기이고, 공유된 메모리 공간에서 버퍼를 할당하고 반환하는 경우에는 독립된 버퍼 할당에 비해 절반 크기의 메모리 만이 요구된다[16, 21]. 이런 버퍼 공유 기법을 이용하는 경우 전체 버퍼링 메모리의 크기는 D_i의 합으로 둘 수 있다. 다음 식 (4)는 l개의 스트림

들에 대해 서버 S^{EDF*}가 필요로 하는 최대 버퍼링 메모리 크기를 보이는 식이다.

$$\sum_{i=1}^{l} D_i = \sum_{i=1}^{l}(S_{ds}\times n_i) = S_{ds}\sum_{i=1}^{l}\left(\frac{p_i}{B_{otdc}}\cdot\frac{n_i\times B_{otdc}}{p_i}\right)$$
$$= T\sum_{i=1}^{l}\left(p_i\cdot\frac{n_i\times B_{otdc}}{p_i}\right) \le T\times MAX\_PI\sum_{i=1}^{l}\frac{n_i\times B_{otdc}}{p_i} \quad (4)$$

식 (4)에서 스트림 S_i에 대한 디스크 대역폭 크기 $(n_i\times B_{otdc})/p_i$의 합은 유효 디스크 대역폭(즉, $B_{otdc}\times N_{otdc}$) 보다 클 수 없다. 이 때 유효 디스크 대역폭은 특정 디스크에 대해 고정된 값을 가지므로 서버 S^{EDF*}에서 필요로 하는 버퍼 메모리의 크기는 $MAX_PI\times T$에 따라 정해짐을 알 수 있다. 따라서 주어진 버퍼 메모리 크기에 맞게 이 두 값을 정한다. T의 크기는 bulk-SCAN시에 드는 전체 시간 중에 탐색시간의 합이 차지하는 비율을 고려하여 정한다. 이 비율값은 T의 크기가 2초 보다 큰 범위에서는 거의 일정한 값을 가지므로, T의 크기를 2보다 큰 값으로 정하면 탐색시간으로 인한 성능저하는 생기지 않는다.

또, MAX_PI는 T와 함께 디스크 대역폭의 단편화 비율을 결정한다. 제안된 디스크 대역폭 할당 방식의 경우 승인된 각 스트림에 대해 최대 $S_{ds}/(T\times MAX\_PI)$ (Mbps) 크기의 디스크 대역폭의 단편화가 생길 수 있다. 따라서 상영 승인된 스트림 수가 N이라고 하면 최대 $(N\times S_{ds})/(T\times MAX\_PI)$ 크기의 디스크 낭비가 있을 수 있다. 이를 고려하여 디스크 단편화 현상으로 인한 낭비가 전체 유효 디스크 대역폭의 5%내에 있도록 MAX_PI를 결정한다.

5.2 서버 S^{EDF*}의 동작 알고리즘

서버 S^{EDF*}는 요청큐에 있는 상영요청을 FIFO 형태로 승인하는데, 앞에서 기술한 함수 *Determine_CAP*()이 사용된다. 상영승인된 스트림은 초기 프리팻칭이 끝나면 상영을 시작할 수 있다. 상영승인 되었으나 아직 초기 프리팻칭 단계에 있는 스트림들의 집합을 S_{new}로, 이미 상영이 시작된 스트림들은 S_{play}로 표시한다. S_{play}의 스트림들에 대해 최소 비율의 OTDC만을 이용하여 프리팻칭 마감시간을 지킨다. 즉, N_{otdc}개의 OTDC중에서 S_{play}의 스트림들이 점유한 비율의 OTDC만을 할당하며 스트림들의 프리팻칭 마감시간을 만족시킬 수 있다[12]. 이 때 S_{play}의 채널 점유비율은 앞의 식 (3)을 이용하여 구한다. 그리고 할당되고 남는 OTDC들은 S_{new}의 스트림들에 할당한다. S_{new}에 할당 가능한 OTDC를 이용하여 스트림 데이터를 위한 실시간성 I/O 요청과 함께 비실시간성 I/O 요청을 함께 처리하는 방법을 지원할 수도 있다. 이에 관련된 사항은 본 논문에서 기술하진 않지만 관련 연구로서 [22, 23]이 있다.

스트림들에 EDF 방식을 사용하기 위해서는 스트림 별로 프리팻칭 마감시간과 마감시간까지 할당해야 할 OTDC 수를 유지해야 한다. 이를 위해 각 스트림에 두 정수 변수 *deadline*과 *otdc*를 사용하며, 스트림 S_i의 두 변수를 $S_i.deadline$과 $S_i.otdc$로 표현한다. $S_i.deadline = d$이고 $S_i.otdc = m$이면, S_i의 프리팻칭 마감시간은 T_d이고 이때까지 할당해야 할 OTDC 수는 m임을 나타낸다. OTDC를 할당 받은 스트림에 대해서는 할당된 수만큼 *otdc*의 값을 감소시킨다. 이런 방식에 따라 T_d를 프리팻칭 마감시간으로 하는 스트림의 경우 $SCAN_{d-1}$이 끝나는 시점에서는 항상 *otdc* 값이 영이 된다. (그림 6)은 서버 S^{EDF*}의 동작 알고리즘을 보인다.

initialization : $S_{new} \leftarrow S_{play} \leftarrow WQ \leftarrow$ **null** ;
$R_{res} \leftarrow 0.0$, *Serial* $\leftarrow 1$, and *Clock* (0 ;
(1) While [$WQ \neq$ **null**] Do
(2) Get stream S_i from WQ. Let its playback rate be B_i ;
(3) If [$B_{otdc} \times N_{otdc} \times (1-R_{used}) \geq B_i$] Then
(4) $(p_i, n_i) \leftarrow Determin\_CAP(B_i, MAX\_PI)$; /* CAP */
(5) Set S_i (*deadline*, *otdc*) to (p_i + *Serial*, n_i) and put it into S_{new} ;
(6) Else put S_i back to WQ and go to (8) ; /* insufficient disk bandwidth */
(7) EndWhile
(8) Calculate R_p such that $R_p = \frac{1}{Notdc}\times\sum_{S_j}(n_j/p_j)$, where $S_j \in S_{play}$;
(9) $N_p \leftarrow R_p \times N_{otdc}$; /* the minimum number of OTDCs for S_{play} */
(10) Allocate N_p OTDCs to the streams in S_{play} based on the EDF policy, and derease *otdc* of them appropriately ;
(11) $Max\_OTDC \leftarrow Calc\_Max\_OTDC(Clock, (Serial + 1)\times T)$;
(12) $Remaining \leftarrow Max\_OTDC - N_p$;
(13) Allocate *Remaining* OTDCs to the streams in S_{new} according to the EDF policy, and decrease *otdc* of them appropriately ;
(14) Based on OTDC allocation in (10) and (13), we create a disk-scheduling plan. Then do a bulk-SCAN.
(15) Advance the bulk-SCAN serial number, i.e., *Serial*++ ;
(16) For each $S_j \in S_{new}$ such that $Sj.otdc = 0$ /* Initial prefetch is completed. */
(17) Reset S_j (*deadline*, *otdc*) to (*Serial* + p_j, n_j), and move it to S_{play} ;
(18) For each $S_j \in S_{play}$ such that $Sj.deadline$ = *Serial* /* end of the current period */
(19) If [playback of S_j is finished] Then remove S_j from S_{play} and update R_{used} ;
(20) Else set S_j(*deadline*, *otdc*) to (*Serial* + p_j, n_j) ;
(21) If [*Clock* < (*Serial*-1)×T]] Then wait until time (*Serial* -1)×T ;

(그림 6) 서버 S^{EDF*}의 동작 알고리즘

6. 성능 평가

본 장에서는 성능평가를 위해 모의실험을 수행한다. 모의실험에서는 성능 평가의 척도로 서비스 응답시간을 사용하며, 서비스 응답시간은 상영요청의 발생시점에서 상영개시 시점까지를 의미한다. 모의실험에서 상영이 요청되는 CM의 상영대역폭과 상영기간은 각각[20 Kbits/sec, 500 Kbits/sec]과 [2분, 4분]사이에서 균등한(uniform) 분포를 가진다.

모의실험에서는 <표 1>에 기술한 디스크모델(HP 97560 [17])을 이용하였고 사용 가능한 버퍼 메모리의 크기를 60 MBytes로 두었다. 사용자로부터의 상영 요청은 포아송분포에 따라 발생하도록 했으며 상영 요청되는 CM은 전체 CM 객체중에서 무작위(random)로 선택된다. 그리고 데이터 세그먼트는 하나의 트랙으로 구성된다.

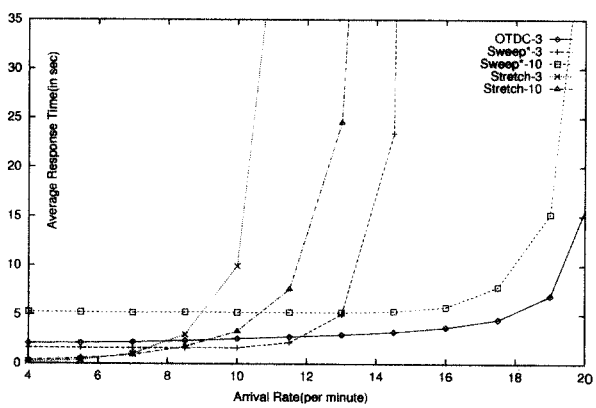
〈표 1〉 사용되는 디스크 드라이브 모델 및 관련 기호

기 호	기호의 의미	기본값
S_{ds}	세그먼트의 크기	36KBytes
R_t	데이터 전송속도	2.46MB/sec
N_{cyl}	전체 실린더 수	1962
seek time(d)	탐색 지연 시간 (msec)	$3.24+0.4\sqrt{d}$, if d <383 $8.0+0.008d$, otherwise
C_d	디스크 용량	1.3 GBytes

(그림 7)은 제안된 OTDC 이용 방식과 기존의 두 기법인 Fixed-Stretch와 Sweep 방식과의 성능 비교를 보인다. 이 두 방식을 성능비교의 대상으로 삼은 것은 이 두 방식이 상영하는 스트림들을 위한 I/O 읽기요청에 대한 마감시간을 늘 준수하기 때문이다. 제안하는 방식 또한 이런 조건을 만족하기에 이들 방식과 성능비교를 수행한다. 그림에서 그래프의 이름 표시에 있는 하이픈 다음에 있는 숫자는 라운드로빈 방식의 경우 L_p의 크기를, 제안한 OTDC 사용 기법의 경우 T의 크기를 나타내며, 단위는 초이다.

(그림 7)의 Sweep* 방식은 각 주기의 뒤쪽에 발생하는 잔여 디스크 시간을 스트림의 초기 프리팻칭에 사용하여 Sweep 방식의 응답시간을 향상시킨 기법으로서, 일반적인 Sweep 기법을 개선한 방식이다.

Sweep*를 사용하는 경우 시스템에 적당한 작업부하가 있을 때 원래의 Sweep 방식에 비해 L_p 크기만큼 서비스 응답 시간이 단축된다. 즉, 일반적인 Sweep-10은 (그림 7)과 같은 실험의 경우 분당 서비스 요청이 16을 넘지 않는 범위에서도 15초 정도의 응답시간을 보일 것이다. 그림의 OTDC-3 방식은 *MAX_PI*를 10으로 두었으며, T를 3초로 한 것은 비교되는 라운드로빈 방식의 L_p의 최소 크기에 맞춘 것이다.

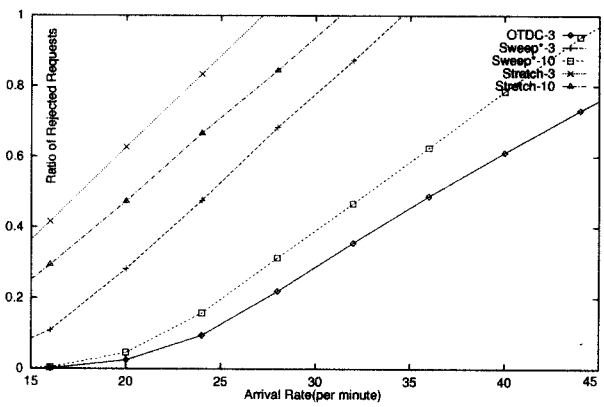


(그림 7) 상영 요청에 따른 평균 서비스 응답시간

Fixed-Stretch 방식은 상영 요청이 적은 경우 상당히 좋은 서비스 응답 시간을 보인 반면 과도한 디스크 대역폭의 낭비로 인해 과부하 시점이 매우 빠르며, 이는 서비스 처리율(throughput)이 상대적으로 매우 낮음을 의미한다. 제안한 방식은 Fixed-Stretch 방식에 비해 분당 9개 이상의 상영요청이 있는 경우 평균응답시간이 더 좋아지며, 처리율의 경우 2배에서 3배 나은 성능을 보인다. 또한 Sweep*-10과 비교해서는 제안한 방식이 처리율이 우수함에도 Sweep*-10에 대해 50% 정도의 낮은 평균 서비스 응답시간을 보인다.

(그림 7)의 실험 결과는 FIFO 방식에 따라 시스템에 들어온 상영요청을 모두 처리할 때의 결과이다. 이는 시스템 과부하의 발생 시점 전까지의 서비스 응답시간 변화를 알아보는데 의미가 있다. 하지만 시스템에 과부하가 생길 때 들어오는 상영 요청을 거절해 버리는 방식이 있을 수 있다.

(그림 8)은 상영 요청율에 따른 서비스 거절율의 변화를 나타낸 실험 결과이다. 수직축의 값은 전체 상영 요청 수에 대한 거절된 상영 요청 수의 비율로서, 그 크기가 클 수록 성능이 좋지 않음을 표시한다. 그림에서 알 수 있듯이 Fixed-Stretch 방식에서 거절율이 가장 높다. 이에 반해 제안한 방식은 거절율이 가장 작은 Sweep*-10에 비해서도 그 거절율이 30% 정도 낮고, Stretch-10에 비교할 때는 3배 정도 낮은 거절율을 보인다. Sweep*-10의 경우 제안한 방식에 비해 평균 응답시간이 2배나 긴 것을 감안한다면 제안한 방식이 매우 뛰어난 특성을 가지고 있음을 알 수 있다.



(그림 8) 과부하 상태에서의 서비스 거절율

7. 결 론

논문에서 제안한 기법은 주기적인 프리팻칭 기법을 사용함에 있어 디스크 채널 할당 방식에 유연성을 주기 위해 각 CM 스트림이 서로 다른 길이의 프리팻칭 주기를 사용할 수 있도록 했고, 각 주기 내에서의 채널 할당을 시스템의 디스크 사용 상황에 맞추어 동적으로 변화될 수 있게 했다. 이런 방식을 통해 CM 상영 요청 변화에 따라 생기는 잔여분의 디스크 대역폭을 적절히 이용할 수 있으며, CM의 상영 대역폭 크기에 근사한 값으로 디스크 대역폭을 점유할 수 있도록 하여 디스크 낭비를 최소로 한다.

이를 위해 시스템 시간을 기본 간격으로 나누고 이 간격 내에서 bulk-SCAN을 이용하여 일정 수 이상의 채널을 생성할 수 있도록 했다. 이 때 생성되는 채널은 EDF 알고리즘에 따라 프리팻칭 주기의 마감시간이 가까운 순서로 스트림에 할당된다. 제안된 방식은 기존의 라운드로빈 방식에 비해 디스크 이용률과 서비스 응답시간의 측면에서 우수한 성능

을 보임으로써, 다양한 형태의 MMIS 시스템의 CM 상영 스케줄링 기법으로 이용될 수 있다.

참 고 문 헌

[1] R. Baier, C. Gran, A. Scheller and A. Zisowsky, "Multimedia Middleware for the Future Home," In *Proc. of the Intl. Workshop on Multimedia Middleware*, pp.123-129, October, 2001.

[2] Wei Tsang Ooi, Peter Pletcher and Lawrence A. Rowe, "Indiva : Middleware for Managing Distributed Media Environment, Http://www.openmash.org/resources/pubs/2002/164, 2002.

[3] R. K. Abbott and H. Garcia-Molina, "Scheduling I/O Requests with Deadlines : A Performance Evaluation," In *Proc. of the Real-Time Systems Symposium*, pp.113-125, 1990.

[4] R. Wijayaratne and N. Reddy, "Integrated QoS Management for Disk I/O," In *Proc. of the IEEE Multimedia Systems*, pp.487-492, June, 1999.

[5] J. Aerts, J. Korst and S. Egner, "Random Duplicate Storage Strategies for Load Balancing in Multimedia Servers," Technical report, NL-MS 20.314, 2000.

[6] David P. Anderson, Yoshitomo Osawa and Ramesh Govindan, "A File System for Continuous Media," *ACM Trans. on Computer Systems*, pp.311-377, November, 1992.

[7] Steven Berson, Richard Muntz, Shahram Ghandeharizadeh and Xiangyu Ju, "Staggered Striping in Multimedia Information Systems," In *ACM SIGMOD*, pp.79-90, 1994.

[8] Asit Dan and Dinkar Sitaram. "An Online Video Placement Policy based on Bandwidth to Space Ratio(BSR)," In *ACM SIGMOD*, pp.376-385, 1995.

[9] Doron Rotem and J. Leon Zhao, "Buffer Management for Video Database Systems," In *Proc. of the IEEE Intl. Conference on Data Engineering*, pp.439-447, 1995.

[10] Yen-Jen Oyang, Meng-Huang Lee, Chun-Hung Wen and Chih-Yuan Cheng, "Design of Multimedia Storage Systems for On-Demand Playback", In *Proc. of the IEEE Intl. Conference on Data Engineering*, pp.457-465, 1995.

[11] P. Venkat Rangan and Harrick M. Vin, "Efficient Storage Techniques for Digital Continuous Multimedia", *IEEE Trans. on Knowledge and Data Engineering*, Vol.5, No.4, pp.567-573, 1993.

[12] Sungchae Lim and Myoung-Ho Kim, "Real-time Disk Scanning for Timely Retrieval of Continuous Media Objects," *Information and Software Technology*, Vol.45, No.9, pp.547-558, June, 2003.

[13] Huang-Jen Chen and Thomas D. C. Little, "Storage allocation policies for time-dependent multimedia data," *IEEE Trans. on Knowledge and Data Engineering*, Vol.8, No.5, pp.855-864, 1996.

[14] Raymod T. Ng and Jinhai Yang, "Maximizing Buffer and Disk Utilization for News On-Demand," In *Proc. of the Intl. Conference on Very Large Databases*, pp.451-462, 1994.

[15] Ahmed K. Elmagarmid and Haitao Jiang, *Video Database Systems : Issues, Projects and Applications*, Kluwer Academic Pub., March, 1997.

[16] Edward Y. Chang and Hector Garcia-Molina, "Effective Memory Use in a Media Server," In *Proc. of the Intl. Conference on Very Large Databases*, pp.496-505, 1997.

[17] C. Ruemmler and J. Wilkes, "An Introduction to Disk Modeling" *IEEE Computer*, Vol.27, No.3, pp.17-28, March, 1994.

[18] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of the ACM*, Vol.20, No.1, pp.46-61, 1973.

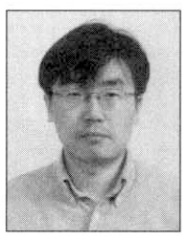
[19] Houssine Chetto and Maryline Chetto, "Some Results of the Earliest Deadline Scheduling Algorithm," *IEEE Trans. on Software Engineering*, Vol.15, No.10, pp.1261-1269, 1989.

[20] Yen-Jen Qyang, "A Tight Upper Bound of the Lumped Disk Seek Time for the SCAN Disk Scheduling Policy," *Information Processing Letters*, Vol.54, No.6, pp.323-329, 1997.

[21] Edward Chang and Yi-Yen Chen, Minimizing Memory Requirements in Media Servers, Technical report, Stanford Technical Report SIDL-WP-1990-0050, Oct., 1996.

[22] E. Balafoutis, M. Paterkakis and P. Triantafillou, "Clustered Scheduling Algorithms for Mixed-Media Disk Workloads in a Multimedia Server," *Cluster Computing Journal*, Vol.6, No.1, pp.75-86, 2003.

[23] Ibrahim Kamel, T. Niranjan and Shahram Ghandeharizedah, "A Novel Deadline Driven Disk Scheduling Algorithms for Multi-Priority Multimedia Objects," In *Proc. of the Intl. Conference on Data Engineering*, pp. 349-358, 2000.



임 성 채

e-mail : savvyguy@daum.net

1992년 서울대학교 컴퓨터공학과(학사)

1994년 한국과학기술원 대학원 전산학과 (이학석사)

2004년 한국과학기술원 대학원 전산학과 (공학박사)

1999년 충남대학교 전산학과 강사

2000년~2000년 서울정보시스템 기술부 부장

2000년~현재 코리아와이즈넛 연구소 수석연구원 웹검색 엔진 및 Enterprise 검색엔진 개발

관심분야 : Web IR, 멀티미디어 시스템, 메인메모리 DBMS, 분산시스템