디스크 배열-기반 주문형 비디오 서버에서의 탐색 시간 단축을 위한 효율적인 주기적 요청 묶음 기법

(An Efficient Periodic-Request-Grouping Technique for Reduced Seek Time in Disk Array-based Video-on-Demand Server)

김 운 석 † 김 지 홍 †† 민 상 렬 †† 노 삼 혁 †††

(Woon seok Kim) (Ji hong Kim) (Sang Lyul Min) (Sam Hyuk Noh)

요 약 주문형 비디오 서버에서 동시에 서비스 될 수 있는 사용자의 수는 서버의 단위 시간당 처리량에 의해 결정되는데, 서버의 처리량은 대부분 디스크의 처리량에 의해 제약을 받는다. 따라서, 디스크의 처리량은 서버 설계에 있어서 매우 중요한 요소가 되며, 대부분의 서버의 경우, 이를 높이기 위해 디스크 배열을 이용한다. 본 논문에서는 디스크 배열에서 디스크들의 탐색 시간을 줄여 전반적으로 디스크의 처리량을 증대시킬 수 있는 효율적인 주기적 요청 묶음 기법을 제안한다. 제안된 기법에서는 디스크 내 인접 영역에 저장된 데이타를 요청하는 주기적 요청들을 하나의 묶음으로 만들고, 각 묶음들이 동일 디스크를 순차적으로 접근할 수 있도록 좌대칭 혹은 우대칭 형식으로 정렬하여 각 디스크에서 요청들이 겪게되는 탐색 시간을 줄인다. 이는 단일 데이타 스트림에 의해 요구되는 디스크 대역폭을 줄일 수 있으므로, 서버는 주어진 시스템 자원 하에서 보다 많은 사용자들을 서비스 할 수 있다. 모의 실험 결과, 제안된 기법은 기존의 기법들에 비해 보다 많은 사용자들을 서비스 할 수 있음이 확인되었다. 일반적인 192KB의 블록 크기를 이용할 경우, 일반 디스크 배열에 비해 평균 20%의 사용자 대기 시간 감소와 평균 8%의 동시 이용자 증가를 보였다. 또한, 본 논문에서는 비디오 스트림들에 대한 사용자 선호도 변화에 맞추어 묶음들을 재구성하는 적응 기법을 제시한다.

Abstract In Video-on-Demand (VoD) servers, disk throughput is an important system design parameter because it is directly related to the number of user requests that can be served simultaneously. In this paper, we propose an efficient periodic request grouping scheme for disk array-based VoD servers that reduces the disk seek time, thus improving the disk throughput of VoD disk arrays. To reduce the disk seek time, the proposed scheme groups the periodic requests that access data blocks stored in adjacent regions into one, and arranges these groups in a pre-determined order (e.g., in left symmetric or right-symmetric fashion). Our simulation result shows that the proposed scheme reduces the average disk bandwidth required by a single video stream and can serve more user requests than existing schemes. For a data block size of 192KB, the number of simultaneously served user requests is increased by 8% while the average waiting time for a user request is decreased by 20%. We also propose an adaptation technique that conforms the proposed scheme to the user preference changes for video streams.

1. 서 론

† 비 회 원 : 서울대학교 컴퓨터공학과
wskim@archi.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학부 교수
jihong@davinci.snu.ac.kr
symin@dandelion.snu.ac.kr
††† 종신회원 : 홍익대학교 정보컴퓨터공학부 교수
noh@cs.hongik.ac.kr
논문접수 : 2000년 10월 19일
심사완료 : 2001년 8월 27일

통신 기술의 발전과 디지털 압축 기술의 발전은 서버로부터 원거리에 위치한 사용자들에게 멀티미디어 서비스의 제공을 가능하게 하였다. 이러한 서비스의 예로는 원격 강의, 화상 회의, 그리고 주문형 비디오 및 오디오를 들 수 있는데, 이들에 의해 제공되는 멀티미디어 데이타들은 일반 데이타들에 비해 그 크기가 크고 실시간 전송을 요구한다는 특성을 가진다. 예를 들어, MPEG-1 압축 기법을 이용하여 생성된 100분 길이의 비디오 파

일의 경우, 대략 그 크기는 1.125 GB 정도가 되며 초당 192 KB의 주기적 데이타 전송을 요구한다. 이러한 데이타를 DRAM과 같은 칩에 저장하기는 어렵기 때문에 전용 제생기를 제외한 대부분의 시스템들에서는 디스크와 같은 보조 기억 장치를 이용한다. 특히, 수 십 편의 비디오 파일을 서비스하는 주문형 비디오 서버의 경우에 있어서는 단일 디스크(1~9 GB) 만으로는 부족하므로 다수의 디스크들로 구성된 디스크 배열(100~500 GB)을 이용한다.

이러한 대용량 저장 장치를 가진 주문형 비디오 서버에 대해 사용자들은 일반적으로 네트워크를 통해 다양한 데이타에 대한 서비스를 요청하고 이를 제공받는다. 서버는 이러한 사용자들에 대해 서비스가 종료될 때까지 처음 보장한 서비스의 질로 지속적인 서비스를 제공할 수 있어야 한다. 즉, 서비스 제공이 허용된 사용자에 대해 사용자측에서 데이타 부족으로 인한 불편[1]과 데이타 과잉 공급으로 인한 버퍼 오버플로우가 발생하지 않도록 적절한 전송률[2]로 데이타를 전송해 주어야 한다. 사용자측으로의 데이타 전송이 요구되는 시점에 있어 서버는 데이타 전송에 차질이 없도록 전송될 데이타를 메모리와 같이 접근 시간이 짧은 버퍼에 미리 준비해 두어야 한다. 즉, 서비스를 받고 있는 사용자들에 대해 항상 버퍼, 네트워크 대역폭, 그리고 디스크 대역폭과 같은 시스템 자원들이 가용할 수 있도록 서비스 제공이 시작되는 순간에 이들에게 할당해 주어야 한다. 그러나, 이러한 시스템 자원들은 물리적으로 한정된 것이므로 서버가 이를 어떻게 활용하느냐에 따라 그 효율성이 결정된다. 본 논문에서는 네트워크 대역폭과 버퍼는 충분하다고 가정하며 디스크 대역폭의 효율적인 이용에 초점을 두도록 하겠다.

디스크의 성능은 요청된 입출력 요구들을 어떻게 스케쥴링 하느냐에 많은 영향을 받는다. 특히, 동기화가 요구되는 디스크 배열의 경우 특정 디스크에 대한 디스크 입출력 요구 스케쥴링 결과가 다른 디스크의 스케쥴링 결과에도 영향을 미칠 수 있기 때문에 더욱 그러하다[1]. 단일 디스크 입출력 요구를 처리하는데 요구되는 접근 지연 시간(disk access time)은, 크게 탐색 시간(seek time), 회전 지연 시간(rotational delay), 그리고 데이타 전송 시간(transfer time)으로 나뉠 수 있는데 이중 탐색 시간이 스케쥴링에 의해 가장 크게 변화하는 요소이다. 표 1과 같은 사양을 가지는 디스크들을 가정하여 보자. 이들로부터 각각 한 트랙과 두 트랙 크기에 해당하는 데이타를 읽어들인다고 할 때 이에 소요되는 평균 접근 지연 시간과 이중 탐색 시간이 차지하는 비율을 계산하여 보면 표 2와 같다.

표 1 디스크 사양

	Viking	HP-97560
저장 용량 (Giga Byte)	2.2	1
최대 디스크 탐색 시간 (msec)	15	25
평균 디스크 탐색 시간 (msec)	8	12.5
최소 디스크 탐색 시간 (msec)	1	2
평균 회전 지연 시간 (msec)	4.17	7
데이타 전송률 (mbps)	59.3	21.2

표 2 디스크 접근 시간 내 탐색 시간 비율

	Viking		HP-97560	
	한 트랙 (63 KB)	두 트랙 (126 KB)	한 트랙 (38 KB)	두 트랙 (76 KB)
전체 접근 시간(msec)	20.47	28.77	33.5	47.5
평균 탐색 시간(msec)	8	8	12.5	12.5
비율 (%)	39.08	27.8	37.31	26.32

표 1에서와 같은 성능을 보이는 디스크를 이용하여 각 사용자들에게 매 1초마다 63 KB의 데이타를 전송하는 서버를 가정하여 보자. 각 63 KB 크기의 데이타에 대한 평균 디스크 접근 시간이 20.47 msec이라 할 때 만일 아무런 스케쥴링 정책 없이 선입선출 방식으로 사용자 요구를 처리한다면 초당 평균 48명의 사용자들에게 데이타를 전송할 수 있다. 하지만, 이중 탐색 시간으로 소요되는 40%정도의 시간을 효율적인 스케쥴링을 통해 줄이게 된다면 48개 이상의 데이타 블록을 읽어들일 수 있게 된다. 이는 주문형 비디오 서버에 있어서 동시 지원 가능한 사용자 수가 증가 될 수 있음을 의미하며 더 나아가 고객 당 요구되는 서비스 비용이 감소되어 서버의 성능이 가격적으로도 우수해 짐을 의미하는 것이다.

그러나, 디스크 처리량을 증대시켜 효율성만을 증대시킬 경우에는 멀티미디어 데이타가 가지는 시간 제약성을 만족시키기 어렵다. 이러한 문제를 해결하기 위해 기존의 연구들에서는 일반 디스크 스케쥴링 방법을 새로운 멀티미디어 환경에 적응시키거나 새로운 실시간 스케쥴링 방법을 제안하였다[2]. 그러나, 대부분의 연구들이 개개의 디스크에 대한 효율성과 시간 제약성을 해결

1) 비디오 파일에 대한 요구일 경우비디오 화면의 멈춤 혹은 더딤이 발생하는 경우를 말한다.

2) 일반적으로 사용자측의 데이타 소모율을 의미한다.

하여 디스크 배열 전반에 걸친 성능 향상을 유도했을 뿐 인접 디스크들과의 연관 관계는 고려하지 않았다. 따라서, 본 논문에서는 디스크 배열을 통해 주기적 디스크 입출력 요구들을 수행하는 주문형 비디오 서버에 있어서 디스크 배열의 특성과 인접 디스크들 간의 성능상의 연관 관계를 고려하여 사용자 요구들을 처리하는 효율적인 사용자 요구 묶음 기법을 제시한다.

본 논문에서 제안하는 기법은 주기적 디스크 입출력 요구를 유발하는 사용자 요구들에 대해 서로 참조하는 위치가 인접한 것들이 하나의 묶음을 형성토록 한다. 또한, 각 묶음들이 다른 인접 묶음들의 스케쥴링 정책에 보완적인 형태가 될 수 있도록 왼쪽 혹은 오른쪽 대칭 형식으로 배치한다. 따라서, 단일 디스크 입출력 요구를 처리하는데 필요한 디스크 대역폭 크기를 줄이고 시스템이 주어진 자원 하에서 보다 많은 사용자들에 대해 서비스를 제공할 수 있도록 한다.

디스크 내 참조하는 위치가 인접한 사용자 요구들을 하나의 묶음으로 형성하여 디스크를 이용토록 한다는 것은 새로이 발생한 사용자 요구들에 대해 그들이 참조하고자 하는 디스크 내 위치에 따라 차별적으로 서비스를 제공하겠다는 것을 의미한다. 다수의 디스크들이 존재하고 다수의 디스크 이용 묶음들이 존재하는 경우를 가정하여 보자. 디스크 내 특정 지역을 접근하고자 하는 사용자 요구들이 집중적으로 발생하여 일부 묶음에서는 디스크 대역폭을 최대한 이용하고 있고, 일부 묶음에서는 할당된 디스크 대역폭 중 일부만을 이용하고 있다고 하자. 시스템 전반적으로는 디스크 대역폭이 남지만 특정 디스크 위치를 접근하고자 하는 새로운 사용자 요구들은 자신이 원하는 묶음에 포함되지 못하고 상당 시간 대기 상태로 있게 된다. 따라서, 본 논문에서는 사용자 접근 유형의 변화를 십분 고려하여 시스템이 적응해 갈 수 있는 방안을 제시한다.

주문형 비디오 서버를 가정한 모의 실험 결과 디스크 탐색 구간 최적화와 사용자 접근 유형 변화에 대한 적응 기법을 통해 스트림 당 요구되는 디스크 대역폭의 양이 줄어들어 동시 지원 가능한 사용자의 수가 늘어나고 평균 사용자 대기 시간이 줄어듦을 확인하였다. 평균 탐색 시간이 21%를 차지하는 192 KB의 데이타 블록을 사용하는 경우에 있어서도 사용자 요구 처리 비율이 평균 8% 향상되었으며 평균 사용자 대기 시간도 평균 20% 감소하였다. 또한, 동시 지원 가능한 사용자 요구의 수에 있어서는 96 KB의 데이타 블록에 대해 평균 9.7%의 증가가 있었다.

이후 본 논문의 구성은 다음과 같다. 먼저 2장에서는 디스크 배열 기반의 주문형 비디오 서버의 기본적인 형태를 설명하고 개선되어야 할 사항들을 살펴보며 3장에서는 본 논문에서 제시하는 효율적인 사용자 요구 묶음 기법에 대해 설명한다. 4장에서는 제시된 기법의 효율성 정도를 측정하기 위해 실시한 모의 실험 결과를 제시하고 이에 대해 분석한다. 그리고, 5장에서는 기존의 연구들을 본 논문에서 제시한 기법과 비교토록 한다. 끝으로 6장에서 결론과 함께 향후 연구 과제에 대해 논의토록 한다.

2. 시스템 모델

본 절에서는 주문형 비디오 서버의 기본적인 형태와 이에 사용되는 용어들에 대해 알아보도록 한다. 여기서 설명되어지는 모델은 [3]에서 제시된 Coarse-grained striping 모델에 기반하고 있다.

주문형 비디오 서버는 일반적으로 다수의 영화를 디스크 배열과 같은 저장 장치에 저장한 후 네트워크를 통해 이를 사용자들에게 서비스한다. 이 때 서비스 받을 수 있는 사용자들의 수는 크게 두 가지 요소에 의해 제한되는데 하나는 사용자와 서버간의 네트워크 대역폭이고 다른 하나는 서버 자체의 성능상의 한계이다. 본 논문에서는 네트워크 대역폭은 매우 충분한 것으로 가정하며 디스크 배열을 사용하는 주문형 비디오 서버의 성능 향상에 초점을 두도록 하겠다.

디스크-0	디스크-1	디스크-2	디스크-3	디스크-4
a[0]	a[1]	a[2]	a[3]	a[4]
a[5]	a[6]	a[7]	a[8]	a[9]
a[10]	a[11]	a[12]	a[13]	a[14]
a[15]	b[0]	b[1]	b[2]	b[3]
b[4]	b[5]	b[6]	b[7]	b[8]
b[9]	b[10]	b[11]	b[12]	b[13]
b[14]	b[15]	c[0]	c[1]	c[2]
c[3]	c[4]	c[5]	c[6]	c[7]
c[8]	c[9]	c[10]	c[11]	c[12]

그림 1 스트라이핑 형식으로 저장된 데이타

디스크 배열은 결함 허용성을 갖추고 처리량을 높이기 위해 일반적으로 RAID 시스템과 같이 스트라이핑 기법을 이용한다[4]. 스트라이핑 기법을 이용하는 디스크 배열에서는 파일들이 스트라이핑 유닛이라 불리는 특정 크기의 데이타 블록으로 나뉘어 다수의 디스크들에 고르게 저장된다. 본 논문에서는 순수히 스트라이핑

기법만을 이용하는 것을 가정하며, RAID-5 시스템과는 달리, 손상된 데이타 복구에 이용되는 패리티 정보는 기록하지 않는 것으로 한다. 그림 1은 5개의 디스크를 스트라이핑하여 이용하는 디스크 배열에 데이타들이 저장되는 예를 보여주고 있다. 멀티미디어 데이타, 특히 비디오 파일은 그 크기가 일반 데이타에 비해 상당히 크기 때문에 모든 디스크들에 걸쳐 저장되며 서버가 파일을 읽어들일 때에는 모든 디스크들이 고르게 번갈아 이용된다. 예를 들어, 그림 1과 같이 데이타들이 저장되어 있는 시스템에서 b 비디오 파일에 대한 사용자 요구가 발생하였고 b 비디오 파일의 첫 데이타 블록이 b[0]라면 서버는 디스크-1로 부터 첫 블록을 읽어들인 후, 일정 시간 이후 디스크-2로 부터 잇따르는 데이타 블록을 읽어들일 것이다. 이러한 작업은 b 비디오 파일의 마지막 블록을 읽어들이는 순간까지 지속될 것이다.

여기서, 디스크-1로부터 데이타 블록 b[0]을 읽어들인 후 디스크-2로부터 데이타 블록 b[1]을 읽어들여야만 하는 최대 시간 간격 즉, 한 사용자 요구에 의해 주기적인 디스크 접근이 발생하는 시간 간격을 주기라 할 때, 이는 다음과 같이 계산되어 진다.

$$\text{주기의 길이(sec)} = \frac{\text{디스크로 부터 읽어들인 블록의 크기}(bits)}{\text{사용자측 데이타 소모율}(bps)} \quad (1)$$

예를들어, 데이타 블록의 크기가 192 KB이고 사용자측에서 1.5 mbps의 데이타 소모율을 가지는 MPEG-1 압축기법을 이용하여 재생한다고 할 때 서버는 매초마다 디스크 배열로부터 192 KB의 데이타 블록을 읽어들인 후 이를 버퍼에 저장하고 네트워크를 통해 사용자측에 전달하여야 한다. 이때, 이 두 과정은 동시에 수행될 수 있으며 이를 지원하기 위해 데이타 블록 두개의 크기에 해당하는 버퍼가 할당된다.

한 사용자 요구에 의해 형성된 스트림을 처리하는 과정에서 주기 내 남는 시간 동안에는 다른 사용자 요구에 의해 형성된 스트림들을 처리할 수 있다. 위의 예에서, 단일 디스크로부터 한 데이타 블록을 읽어들이는데 필요한 시간이 최대 0.1초라면 서버는 1초 주기 동안 최소 10개의 스트림에 대해 데이타 블록을 디스크로부터 읽어들일 수 있다. 이와 같이, 동일 주기 동안에 동일한 디스크를 이용하는 스트림들의 집합을 서비스 그룹이라 한다. 한 서비스 그룹이 특정 디스크를 이용하고 있는 동안에 디스크 배열 내 다른 디스크들은 다른 서비스 그룹들에 의해 이용될 수 있으며 서버 시스템 내에는 디스크 개수 만큼의 서비스 그룹이 존재할 수 있다.

표 3 사용된 기호들과 그 의미

q	단일 디스크에 대한 입출력 요구의 수
b	데이타 블록의 크기
$f_{seek}(dist)$	거리(dist)에 대한 디스크 탐색 시간
$T_{rotation}$	평균 회전 지연 시간
$R_{transfer}$	데이타 전송률 (bps)
$R_{display}$	데이타 소모율 (bps)
T_{settle}	디스크 헤드 정착 지연 시간
$LS_{i,j}$	i 번째 서비스 그룹의 j 번째 요구되는 데이타 블록 트랙 번호
$LE_{i,j}$	i 번째 서비스 그룹의 j 번째 요구되는 데이타 블록을 처리한 이후 디스크 헤드가 위치한 트랙 번호

시스템에 포함된 디스크 배열 내 디스크의 수와 데이타 블록의 크기가 정해지면 시스템이 수용할 수 있는 사용자 요구의 수와 버퍼의 크기는 계산되어 질 수 있다. 우선 i번째 디스크에서 q개의 데이타 블록을 읽어들이는데 소요되는 시간은 표 3에서의 기호를 사용하여 다음과 같이 나타내어 질 수 있다.

$$\begin{aligned} T_{access} = {} & f_{seek}(|LE_{i-1,last} - LS_{i,0}|) \\ & + \sum_{j=0}^{q-2} f_{seek}(|LS_{i,j} - LE_{i,j+1}|) \\ & + q\left(T_{rotation} + \frac{b}{R_{transfer}} + T_{settle}\right) \end{aligned} \quad (2)$$

상기 식에서 f_{seek} 함수에 의해 표시되는 부분들은 사용자 요구들이 서비스되는 순서에 따라 달라지는 디스크 탐색 시간이다. 이중 $f_{seek}(|LE_{i-1,last} - LS_{i,0}|)$ 부분은 인접 디스크들과의 연관 관계를 가지는 부분으로, 이전 서비스 그룹에 의해 디스크가 이용되어진 후, 현 서비스 그룹이 디스크를 이용하고자 할 때, 최초 소요되는 탐색 시간이다. $\sum_{j=0}^{q-2} f_{seek}(|LS_{i,j} - LE_{i,j+1}|)$ 부분은 현 서비스 그룹이 자신이 포함한 사용자 요구들을 서비스함에 있어 요구되는 탐색 구간으로 운영체제가 채택하고 있는 디스크 스케쥴링 정책에 의해 결정되는 요소이다. 특히 이 부분은 일반적으로 CSCAN 이나 SCAN과 같은 스케쥴링 정책에 의해 최소화된다.

사용자들의 데이타 전송 요구는 매 주기마다 발생하므로 T_{access}가 한 주기의 길이를 넘으면 안 된다. 따라서, 주기 당 읽어들여야 하는 데이타 블록의 수를 나타내는 q는 다음 식을 만족하도록 설정되어야 한다.

$$\frac{b}{R_{display}} \geq T_{access} \quad (3)$$

위 식을 만족하는 q의 최대 값은 단일 디스크를 이용하여 동시 지원 가능한 사용자 요구의 수를 의미하므로, 만일 SCAN과 같은 디스크 스케쥴링 정책이 이용되는 디스크 배열이 D개의 디스크들로 구성되어 있다면 서버가 동시에 지원할 수 있는 최대 사용자 요구의 수는 q×D가 된다. 또한, 이에 대해 요구되는 버퍼의 크기는 2×b×q×D가 된다.

본 논문에서는 단위 주기 내에서 서비스 그룹이 소요하는 시간중 위에서 설명된 두 부분의 탐색 시간을 줄일 수 있는 방안을 제시하고자 한다. 이는 사용자 요구당 요구되는 메모리 크기를 변화시키지 않으면서 추가적인 스트림 지원이 가능토록 함으로써 전반적인 시스템 성능 향상을 유도한다. 우선, 시스템 내에서 사용자 요구가 처리되는 과정과 개개의 디스크 내에서 헤드의 움직임에 대해 살펴보도록 하자.

그림 2와 같이 8개의 디스크들로 구성된 디스크 배열과 8개의 서비스 그룹을 가지는 시스템을 가정해 보자. 그림 하단의 원통들은 디스크들을 의미하며 원통 내 구분된 영역들은 서로 상이한 비디오 파일들이 저장된 영역들을 의미한다. 즉, 단일 영역 내에는 단일 비디오 파일에 해당하는 데이타 블록들만이 존재한다. 본 예에서는 a~j 의 알파벳으로 표시된 비디오 파일들이 바깥쪽 트랙에서 안쪽 트랙 방향으로 순서대로 스트라이핑 방식으로 저장되어 있다. 네트워크를 통해 새로이 도착한 사용자 요구들은 일단 사용자 요구-큐(request-queue)에서 서비스 허가를 기다린다. 여기서, 서비스 허가는 서버가 현재 시스템 내 가용한 자원들[3]을 바탕으로 이루어져야 하는데, 새로운 사용자의 추가로 인해 현재 서비스를 받고있는 기존의 사용자들이 영향을 받지 않는 한도 내에서 이루어져야 한다. 이렇게 새로운 사용자에 대해 서비스 제공의 가능 여부를 확인하는 작업을 수용제어(admission control)라 한다. 사용자-요구-큐 내의 알파벳은 현재 대기하고 있는 사용자 요구들이 서비스 받고자 하는 비디오 파일명을 나타낸다. 사용자-요구-큐 내에서 대기하고 있는 사용자 요구들에 대해 서버는 각 서비스 그룹에 대해 수용제어 조건을 확인한 후, 만족되는 서비스 그룹이 존재하면 새로운 사용자 요구를 해당 서비스 그룹에 포함시킨다. 새로운 사용자 요구를 포함한 서비스 그룹은 곧바로 이에 대한 서비스 제공을 시작하는 것이 아니라 새로운 사용자 요구가 요구하는 비디오 파일의 첫 데이타 블록이 저장된 디스크를 이용하게 되었을 때 비로소 이에 대한 서비스를 시작하게 된다.

3) 메모리, 디스크 대역폭 및 네트워크 대역폭등 서버와 사용자 간의 데이타 전송에 필요한 모든 요소들을 말한다.

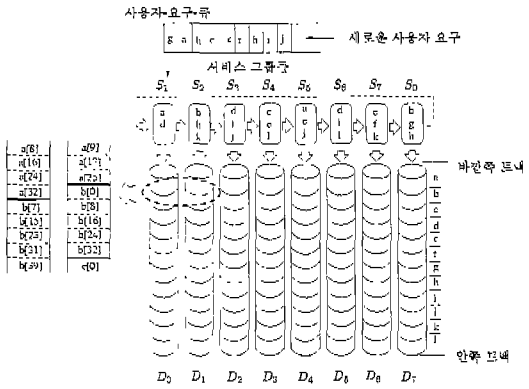


그림 2 사용자 요구 처리 과정

예를 들어, 주기의 길이 L_{period}가 1초이고 각 서비스 그룹이 1초 동안에 3개의 사용자 요구에 대해 디스크로부터 데이타 블록을 읽어들일 수 있다고 하자. 모든 서비스 그룹이 각각 3개의 사용자 요구를 서비스하고 있고 S_1만 두 개의 사용자 요구를 서비스하고 있다면, 대기하고 있던 사용자 요구들 중 가장 오래 대기한 사용자 요구가 서비스 그룹 S_1에 포함되어지게 된다. 그림 2의 예에서는 비디오 파일 g를 요구하는 새로운 사용자 요구가 일단 서비스 그룹 S_1에 포함되어 진 후, 디스크 대역폭과 버퍼를 예약한 상태로 기다리다가 서비스 그룹 S_1이 비디오 파일 g의 첫 데이타 블록인 g[0]를 포함 디스크를 이용하게 되었을 때부터 서비스 제공을 시작 받게 된다. 서비스를 제공받고 있는 사용자 요구들 중 사용자측에서 서비스 중단을 요청하거나 비디오 파일의 마지막 데이타 블록을 읽어들인 사용자 요구는 서비스 제공이 종료된 것을 의미한다. 따라서, 이러한 사용자 요구들은 소속된 서비스 그룹으로부터 제거되며 이때, 디스크 대역폭과 버퍼를 반납하게 되고 서버는 이를 새로운 사용자 요구를 서비스하는데 사용하게 된다.

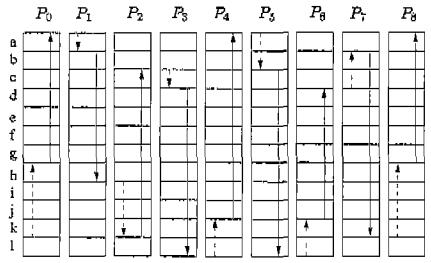
사용자 요구의 처리 과정을 개개의 디스크 입장에서 살펴보도록 하자. 디스크 D_2가 주기 P_0에 어떤 사용자에 대해 데이타 블록 b[0]를 서비스 하였다면 동일 디스크는 8주기 이후에 동일 사용자에 대해 데이타 블록 b[8]을 서비스하게 된다. 즉, 디스크 배열 내 디스크들이 각 사용자 요구에 대해 번갈아 서비스를 행하므로 각 디스크는 각 사용자 요구에 대해 8주기 단위로만 서비스를 행하면 된다. 비디오 파일들의 서비스 제공 시간이 대략 70~150분 정도인 점을 고려한다면 이러한 서비스 행위가 상당 시간 동안 지속되므로 사용자 요구의

스케줄링이 매우 중요함을 알 수 있다.

	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
D_0	gda	bgh	kfc	dil	jea	ccl	jid	bhk	gda
D_1	bhk	gda	bgh	kfc	dil	jea	ccl	jid	bhk
D_2	jid	bhk	gda	bgh	kfc	dil	jea	ccl	jid
D_3	ccl	jid	bhk	gda	bgh	kfc	dil	jea	ccl
D_4	jea	ccl	jid	bhk	gda	bgh	kfc	dil	jea
D_5	dil	jea	ccl	jid	bhk	gda	bgh	kfc	dil
D_6	kfc	dil	jea	ccl	jid	bhk	gda	bgh	kfc
D_7	bgh	kfc	dil	jea	ccl	jid	bhk	gda	bgh

그림 3 주기별 사용자 요구의 처리 과정

그림 3에서 비디오 g에 대한 사용자 요구가 서비스 그룹 S_1에 포함되었다고 했을 때 각 디스크의 주기별 사용자 요구 처리 유형을 살펴보면 그림 3과 같다. 각 디스크들이 SCAN 정책을 사용하여 사용자 요구들을 처리한다고 할 때 디스크 D_1의 디스크 탐색 구간을 살펴보면 그림 4와 같다.



서비스 그룹내 사용자 요구들을 처리하는데 소요되는 탐색 구간

최초 스케줄된 사용자 요구를 처리하는데 소요되는 탐색 구간

그림 4 SCAN 정책하에서 단일 디스크 내 주기별 탐색 구간

각 디스크들은 각 서비스 그룹을 서비스함에 있어 디스크 효율성을 높이기 위해 SCAN 혹은 CSCAN 형식으로 이들을 서비스한다. 또한, 전체 서비스 그룹들을 정해진 라운드-로빈(Round-robin) 형식으로 순차적으로 서비스하여야 한다. 각 데이타의 시간-제약성을 고려하지 않고 각 디스크 입장에서 가장 이상적으로 이들을 서비스하는 방법은 전체 서비스 그룹에 속한 사용자 요구들을 SCAN 형식으로 처리하여 탐색 시간을 줄이는 것이다. 즉, 그림 5와 같은 순서로 처리하는 것이 디스크 입장에서는 가장 효율적인 서비스 방법이다. 이와 같이 하기 위해서는 사용자 요구를 받아들여 서비스 그룹을 할당할 때에 차후 이상적으로 스케줄 될 수 있도록 하여야 한다. 그러나, 앞으로 발생할 사용자 요구에 대해서는 사전에 알기 어려우며 일단 서비스 그룹에 배정된 사용자 요구를 다른 서비스 그룹으로 이동시킨다는 것은 추가적인 시스템 자원의 할당 없이는 불가능하다.[4]

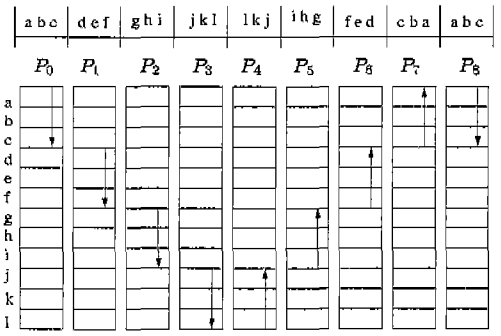


그림 5 이상적인 경우 단일 디스크 내 주기별 탐색 구간

따라서, 본 논문에서는 이러한 제약 조건들 하에서 디스크 배열이 사용자 요구들을 보다 효율적인 형태로 즉, 디스크 배열 내 개개 디스크의 디스크 탐색 구간을 단축시키는 방향으로 서비스하는 방법으로 서비스 영역이라는 개념을 도입하고, 시스템 처리량 낭비를 막기 위해 사용자 접근 유형을 이용하는 방안을 제시코자 한다.

3. 주기적 요청 묶음 기법

본 절에서는 본 논문에서 제시하는 서비스 영역 분할 및 할당 방법을 설명한다. 또한, 오프라인(off-line) 상태에서 비디오 파일들을 저장한 후 서비스 영역을 나누는 방법을 설명하며, 사용자 요구 접근 유형 변화에 따른 서비스 그룹간 작업 부하 불균형 해소 방안에 대해 살펴보도록 하겠다.

3.1 서비스 영역 분할 및 할당

이전 절에서 살펴보았듯이 각 디스크가 각 서비스 그룹을 서비스함에 있어 단일 주기 내에 처리해야 하는 사용자 요구들이 디스크 내 물리적으로 인접한 영역들에 저장된 데이타 블록들에 대해 접근을 요구한다면 주기 당 요구되는 탐색 시간은 줄어들게 된다. 또한, 각 서비스 그룹들 내 마지막으로 서비스되는 사용자 요구

4) 여분의 디스크 대역폭을 가진 서비스 그룹으로 이동할 경우 시간 제약성을 만족시키기 위해서는 이동되어진 서비스 그룹은 통해 데이타를 미리 읽어들인 뒤 정해진 시간에 전송해야 한다. 따라서, 추가적인 버퍼의 사용이 요구된다.

가 접근하고자 하는 위치와 잇따르는 서비스 그룹들내 최초 서비스되는 사용자 요구가 접근하고자 하는 위치가 서로 인접 하다면 각 서비스 그룹에 대해 매 주기마다 서비스를 시작할 때 소요되는 최초 탐색 시간은 줄어들게 된다. 즉, 단일 디스크 입장에서는 자신이 다수의 주기 동안 서비스해야 하는 모든 시스템 내 사용자 요구들을 SCAN 형식으로 처리하여 탐색 시간을 최소화하는 것이 가장 이상적인 스케줄 결과가 되는 것이다. 본 논문에서 제시하는 서비스 영역 분할 기법은 이와 같이 서버가 사용자 요구들을 처리할 수 있도록 다음과 같이 시스템을 설정한다.

- 각 서비스 그룹이 스트림들을 서비스하기 위해 단위 주기 내 소요하는 디스크 탐색 시간을 줄이기 위해, 전체 디스크를 P개의 서비스 영역으로 나눈 후 각 서비스 그룹이 이중 한 영역만을 전담토록 한다. 이때, P는 전체 서비스 영역들이 서비스 그룹들에 의해 이용되는 정도가 동일하도록 $\frac{\text{서비스 그룹의 수}}{2}$ 의 약수로 설정한다. (만일, CSCAN 형식으로 사용자 요구들이 처리될 수 있도록 할 경우에는 서비스 그룹의 수와 P는 동일할 수 있다. 또한, 홀수개의 디스크들을 포함하는 디스크 배열의 경우에는 이 둘의 중간 형태를 취해야 한다.) 그림 6(a)에서 각 디스크 내 작은 사각형들은 각 파일 단위로 나뉜 영역을 의미하며, 동일한 명암으로 표시된 부분은 모두 동일한 서비스 영역에 해당된다. 이 예에서 서비스 영역의 수는 4개이며, 각 서비스 영역에 대해 두 개의 서비스 그룹이 전담하게 된다. S_1와 A_i는 각각 I번째 서비스 그룹과 서비스 영역을 의미하는데, S_0과 S_7은 A_0을, S_1과 S_6은 A_1을, S_2와 S_5는 A_2를, S_3과 S_4는 A_3을 각각 전담하게 된다.

- 매 주기 시작시 각 서비스 그룹들이 최초로 스케줄된 사용자 요구를 처리하기위해 소요하는 탐색 시간을 줄이기 위해 분할된 영역들이 전체적으로 순환되도록 그림 6(b)에서와 같이 서비스 영역들을 배치한다. (디스크 헤드의 진행 방향이 바깥쪽인 경우에는 왼쪽 대칭 형식으로, 안쪽인 경우에는 오른쪽 대칭 방식으로 배치한다.)

위와 같이 설정된 시스템에서 사용자 요구의 처리와 운용은 다음과 같다. 서버에 요청된 사용자 요구들은, 사용자 요구 큐, 예약 리스트, 그리고, 서비스 리스트를 순차적으로 이동하며 처리된다. 사용자 요구 큐는 네트워크를 통해 전달된 사용자 요구들이 디스크 대역폭을 할당 받기 이전까지 대기하는 장소이며, 예약 리스트는 디스크 대역폭을 할당 받은 사용자 요구가 실제 서비스 제공을 받기 이전까지 지연되고 있는 동안 대기하는 장소이고, 서비스 리스트는 실제 서비스가 시작되어 디스크 배열에 대해 입출력 요구를 요청하는 스트림화된 사용자 요구들의 집합이다. 이에 대한 구체적인 알고리즘은 다음과 같다.

```
1. AdmissionControl()
2. {
3. 네트워크를 통해 전달된 사용자 요구들을 일단 사용자 요구 큐에 저장된다.
4.    새로운 사용자 요구(Newer)가 도착하면,
5.    {
6.       TargetGroup = IsSchedulable(Newer)
7.       if ( TargetGroup == NULL )
8.          then  Newer를 사용자 요구 큐에 넣는다.
9.          else  ServiceGroup[TargetGroup]이 관리하는 예약 리스트에
                  Newer를 넣는다
10.   }
11.   Available = GroupService()
12.   if ( Available == YES )
13.   {
14.      for ( 사용자 요구 큐 내의 모든 요소들에 대해 )
15.      {
16.         Older ← 사용자 요구 큐 내에서 스케줄 가능성이 검증되지 않은
                  사용자 요구들중 가장 오래된 사용자 요구
17.         TargetGroup = IsSchedulable(Older)
18.         if ( TargetGroup == NULL )
19.            then  continue
20.            else  Older를 ServiceGroup[TargetGroup]이 관리하는
                     예약 리스트에 넣는다.
21.      }
22.   }
23. }
24. IsSchedulable(Newer) {
25.    Gmin = NULL
26.    StartDisk = Newer가 요청하는 비디오 화일의 첫 블록이 저장된 디스크
27.    for ( i = 0 ; i < D ; i ++ )
28.    {
29.       i번째 서비스 그룹에 대해 스케줄 가능성을 조사한다.
30.          스케줄 가능 : 서비스 그룹 i가 Newer를 포함하여도 그룹내 포함
                           된 스트림들은 종료시한 이내에 서비스 받을 수
                           있다.
                         : 서비스 그룹 i가 Newer를 포함하여도 인접 서비스
                           그룹의 종료시한 만족에는 영향을 주지 않는다
          스케줄 불가능 : Newer가 서비스 그룹 i에 포함됨으로써 기존의
                           스트림들중 하나 이상의 스트림이 종료시한 이
                           내에 서비스 받지 못하게 되거나, 인접 서비스
                           그룹의 종료시한 만족에 영향을 준다.
31.      if ( 스케줄 가능 )
32.         if ( Gmin == NULL )
33.            then  Gmin = i
34.            else  if ( dist( i, StartDisk ) < dist( Gmin, StartDisk ) )
35.                     then  Gmin = i
36.   }
37.   return (Gmin)
38. }
39. dist( i, j )
40. {
41.    if ( i > j )
42.       then  return( D - i + j )
43.       else  return( j - i )
44. }
```

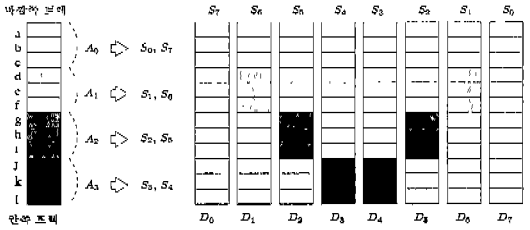
알고리즘에서, IsSchedulable() 함수는 새로운 사용자 요구에 대해 스케줄 가능성을 진단한 후, 그 결과를 알려준다. 만일 스케줄 가능하면, 수용 가능한 서비스 그룹의 번호를 알려주고, 그렇지 않으면, 널(NULL) 값을 돌려준다. GroupService() 함수는 각 서비스 그룹들에 속한 스트림들이 요구하는 디스크 입출력 요구를 수행하는 함수로서, 만일 화일의 마지막 블록을 읽게되는 스트림이 발생하면, 디스크 배열 내 일부 서비스 그룹이

가용함을 호출자에게 알려준다. 따라서, 가용 대역폭을 가진 서비스 그룹이 발생하는 경우엔 Available 변수는 YES라는 값을 가지게 된다.

기존의 서비스 그룹 개념에서는 각 서비스 그룹이 수용할 수 있는 사용자 요구에 대해 차별적인 요소가 없었지만 위의 설정 방식에서는 디스크의 물리적 접근 위치가 인접한 사용자 요구들이 동시에 같은 디스크를 접근할 수 있도록 어떤 비디오 파일을 요구하는 사용자 요구는 특정 서비스 그룹들에만 속할 수 있게 된다. 각 서비스 그룹들은 자신이 관할하고 있는 서비스 영역 내 속한 비디오 파일에 대한 사용자 요구만을 포함한다는 것은 자칫 서비스 그룹간에 작업 부하의 불균형을 가져올 수 있다. 이어지는 절들에서 이에 대한 해결 방안을 설명토록 하겠다.

3.2 비디오 파일의 저장 및 서비스 영역 설정

모든 서비스 영역들이 동일한 개수의 비디오 파일들을 포함토록 설정된 경우를 가정해 보자. 디스크 배열 내에 총 12개의 파일이 저장되어 있고 이중 2개의 비디오 파일에 대한 사용자 요구 발생이 전체 사용자 요구 발생의 절반 이상을 차지한다고 하면, 이 두 파일이 저장된 영역을 관장하는 서비스 그룹들은 다른 서비스 그룹들에 비해 상당히 작업 부하가 심할 것이다. 즉, 다른 서비스 그룹들이 가용 디스크 대역폭을 가짐에도 불구하고 이들을 포함하는 서비스 그룹들에 대한 사용자 요구들은 디스크 대역폭의 부족으로 인해 상당 시간 대기해야 하는 것이다. 이는 각 비디오 파일들이 가지는 선호도의 차이에서 비롯되는 것으로, 이를 바탕으로 하여 서비스 영역을 설정함으로써 해결될 수 있다.



(a) 분할 (b) 서비스 영역 할당

그림 6 디스크 영역의 분할 및 서비스 영역 할당

이에 대한 해결책으로 본 논문에서는 다음과 같이 서비스 영역을 설정토록 한다. 비디오 파일들에 대한 대략적인 선호도를 수집한 다음 선호도가 높은 비디오 파일들이 상호 인접하지 않도록 디스크 배열에 저장한다. 그 세부적인 알고리즘은 아래와 같다.

비디오 화일 선호도를 고려한 저장 알고리즘

```
1. binSize[P] : 각 덩어리에 할당된 비디오 화일들의 선호도 합을 유지하는
   변수
2. binList[P] : 각 덩어리에 할당된 비디오 파일들을 유지하기 위한 리스트
3. 전체 비디오 화일들을 선호도에 근거하여 정렬한 후 videoList에 저장한다
4. while ( videoList ≠ NULL ) {
     mvideo = videoList내 가장 인기 있는 비디오 파일
     NB = 덩어리들 중 현재 할당받은 비디오 화일들의 선호도 합이 가장 작
          은 것의 번호 (즉, binSize값이 가장 작은 bin의 번호)
     binSize[NB] = binSize[NB] + mvideo의 선호도
     mvideo를 videoList에서 제거한 후 binList[NB]에 추가한다.
   }
5. for ( k = 1 ; k = P ; k++) {
       while ( binList[k] ≠ NULL )
           binList[k]로부터 하나의 비디오 파일을 끄내어 videoList에 추가
           한다
   }
6. videoList내 순서대로 디스크 배열에 비디오 파일들을 저장한다.
```

이 알고리즘의 목적은 비디오 파일들을 P개의 덩어리로 나누고, 각 덩어리 내 포함된 비디오 파일들의 참조 빈도수의 합이 비슷하도록 한 후, 차례대로 디스크 배열에 저장되도록 하는 것이다. 이와 같이 분류된 덩어리들은 각 서비스 그룹들에 상응하게 되므로, 같은 영역 내에서는 참조 빈도의 차이가 있을 수 있지만, 서비스 그룹들 간의 작업 부하는 균형을 이루게 된다.

서비스 영역의 설정 시에는 각 서비스 영역 내 포함된 비디오 파일들의 선호도의 합이 다른 서비스 영역 내 포함된 비디오 파일들의 선호도의 합과 비슷하도록 설정한다.

이와 같이 설정할 경우 각 서비스 영역 내 포함된 비디오 파일들의 수는 다른 서비스 영역 내 포함된 비디오 파일들의 수와 다를 수 있다. 그러나, 각 서비스 영역 내 포함된 비디오 파일들의 선호도의 정도는 다른 서비스 영역들과 비슷하므로 이들을 관장하는 서비스 그룹들 간의 작업 부하의 불균형은 완화되게 된다.

여기서 또 하나 고려해야 할 사항은 사용자 요구의 접근 유형은 항상 일정하지 않다는 것이다. 즉, 시스템 내 저장된 비디오 파일들에 대한 사용자들의 선호도가 변화할 수도 있다는 것이다. 이를 해결하기 위해서는 서비스 영역이 이에 맞추어 변화해 주어야 함을 의미하는데 다음절에서 이에 대한 해결책을 알아보도록 하겠다.

3.3 사용자 요구 접근 유형 변화에 대한 적응 방안

일반적으로 사용자 요구의 접근 유형, 즉 비디오 파일들에 대한 선호도는 각 파일들에 대해 상당한 차이를 보일 수 있다[5,6]. 이는 특정 영화에 대한 장시간에 걸친 선호도 변화는 그 종류와 내용에 따라 다양한 편이지만 짧은 시간내의 여러 영화들에 대한 선호도 차이는 대부분 Zipf분포를 따르기 때문이다[5]. 이전 절에서 제시하였듯이 디스크 배열에 비디오 파일들이 저장되기

이전에 이들에 대한 선호도를 알 수 있다면 서비스 영역 할당으로 인한 서비스 그룹간의 작업 부하 불균형을 완화할 수는 있다. 하지만, 사용자 접근 유형이 기존의 예측과는 달리 변화하는 경우에는 작업 부하의 불균형을 막을 수 없다. 따라서, 서비스 영역은 사용자 요구의 접근 유형에 맞추어 동적으로 조절되어야 한다. 사용자 접근 유형의 변화는 비디오 파일의 선호도 변화를 의미하므로 각 서비스 영역의 범위를 조절하여 영역 내 비디오 파일들의 선호도의 합이 다른 서비스 영역 내 비디오 파일들의 선호도의 합과 비슷하도록 하여야 한다. 즉, 선호도가 높은 서비스 영역에 대해서는 보다 적은 비디오 파일들을 포함토록 이의 서비스 영역을 줄이고 반대로, 선호도가 낮은 서비스 영역에 대해서는 보다 많은 비디오 파일들을 포함토록 이의 서비스 영역을 늘여 서비스 영역간의 선호도 차이를 조절해 나가는 것이다.

본 논문에서는 이러한 작업을 다음과 같이 하도록 하여 이후 보여질 실험에 적용하였다.

- 일정 주기로 현재 서비스되고 있는 스트림들과 현재 서비스 대기중인 사용자 요구들이 요청하는 비디오 파일들에 근거하여 선호도에 대한 자료를 수집한다. 얻어진 자료를 바탕으로 각 비디오 파일들의 참조 빈도와 평균 참조 빈도를 구한다.
- 수집된 자료를 바탕으로 바깥쪽 트랙으로부터 안쪽 트랙 방향으로 비디오 파일을 하나씩 포함시키면서 평균 참조 빈도와 비슷해지면 이를 하나의 서비스 영역으로 설정한다. 이때, 서비스 영역간의 경계선상에 놓이는 비디오 파일들은 선택적으로 두 서비스 영역에 모두 포함되어질 수도 있다.

예를 들어, 그림 6과 같이 비디오 파일들이 저장되어 있고, 서비스 그룹이 구성되었다고 하자. 전체 서비스 그룹의 수는 8개이고, 같은 영역을 서비스하는 그룹의 수가 2개이므로, 서비스 영역은 4개로 나뉘어 진다. 일정 간격 내에 100%의 사용자 요구 중 25%의 사용자 요구가 비디오 a와 b에 집중되어 있다면, S_0와 S_7은 이 두 파일만을 서비스하도록 재조정되며, 나머지 비디오 화일들에 대해서는 다른 서비스 그룹들이 서비스하도록 재 설정한다.

위와 같이 새로이 서비스 영역이 설정되면 서비스 그룹들은 새로운 사용자 요구 수용에 있어서 새로이 설정된 영역에 포함된 비디오 파일들에 대한 사용자 요구만을 포함하게 된다. 물론, 기존의 서비스를 받던 사용자 요구들에 대해서는 그들에 대한 서비스가 종료될 때까지 계속 서비스하게 된다.

4. 실 험

디스크 배열을 서비스 영역 단위로 나누고 인접 서비스 그룹들간의 디스크 이용 관계를 개선함으로써 얻어지는 이득을 모의 실험을 통해 확인하였다. 본 실험에서는 다양한 사용자 요구 발생 환경, 데이타 블록의 크기, 분할된 영역의 수, 그리고 비디오 파일 참조 유형 변화에 대해 동시 지원 가능한 사용자 요구의 수를 기준으로 성능 개선 정도를 측정하였다.

4.1 실험 환경

서버는 용량이 1.5 GB인 디스크 100개로 구성하였으며 각 디스크 시뮬레이션 부분은 버클리 대학에서 제작한 raidSim RAID 시뮬레이터[7] 내 디스크 모듈을 이용하였다. 비디오 상영 시간은 75분에서 170분까지의 구간에서 평균 112분에 표준편차 20분인 정규 분포를 따르는 것으로 하였다. 따라서, 105개에서 120개 정도의 비디오 파일이 디스크 배열 전체에 저장될 수 있으며 본 실험에서는 112개의 비디오 파일이 저장된 것으로 하였다.

이와 같은 실험에서 특별히 주의해야 할 것은 통계 자료의 수집이다. 초기 실험 시작 후 일정 시간 간격 동안에는 모든 사용자 요구들이 서버로부터 서비스 제공을 허가 받게 되고 이들에 대한 서비스 제공이 끝나기 전까지는 새로운 사용자 요구에 대한 서비스 제공이 허가되지 못한다. 즉, 서비스 제공에 대한 수용 제어가 고르게 이루어지지 못하게 되어 서비스 제공의 허가와 거부가 시간대 별로 집중적으로 발생할 수 있다는 것이다. 이러한 현상은 난수에 의한 사용자 요구 발생이 시스템을 일관된 상태로 만들 때까지 여러 번 반복될 수 있다[6]. 이로 인한 통계 수집상의 오류를 피하기 위해 본 실험에서는 초기에 모든 디스크 대역폭을 임의의 사용자 요구들에게 할당하고 할당받은 사용자 요구들은 잔여 상영 시간이 0분에서 100분이 되도록 하여 시스템이 일관된 상태로 도달하는데 필요한 시간을 최소화하였다. 또한, 난수 발생으로 인한 효과가 발생할 때까지를 초기화 시간으로 설정하고 이 시간 동안에는 통계 자료를 수집하지 않도록 하였다. 따라서, 이후 보여질 실험 결과들에서는 난수 발생으로 인한 효과가 충분히 적용되어 시스템이 안정된 상태에서 수집된 통계 자료들만이 포함되었다. 모든 실험에서 시스템이 항상 고부하에 놓일 수 있도록 사용자 요구 발생 간격을 설정하였다. 또한, 실험에서 사용자 요구들의 비디오 파일에 대한 선호도는 균등 분포, Zipf 분포, 그리고, 두 분포가 번갈아 발생되는 경우들이 고려되었다.

4.2 실험 결과 및 분석

본 절에서는 제안된 서비스 영역 분할 기법에 대한 실 험 결과를 제시하고 이에 대해 분석한다. 또한, 사용자 접근 유형 변화에 따른 시스템 성능 변화와 이에 대한 적응 방안 적용 결과를 제시하고 분석한다.

4.2.1 서비스 영역 분할에 따른 성능 변화

그림 7은 분할된 영역의 수가 증가할수록 동시 지원되고 있는 스트림의 수가 증감함을 보여준다. 그래프에서 가로축은 분할된 영역의 수를 나타내는데 분할된 영역의 수가 1인 경우는 Ozden의 Coarse-grained disk striping 방식을 적용한 경우에 해당한다[3]. 그림 7은 초기화 시간을 2만초로, 실험 시간을 3만초로 하여 총 5만초 동안에 시스템에 의해 처리된 결과이다. 그래프에서 세개의 곡선은 서로 상이한 데이타 블록 크기 즉, 96 KB, 192 KB, 그리고 384 KB를 가지는 경우이며, 각각 주기의 길이는 0.5초, 1초, 그리고 2초가 된다. 사용자들의 비디오 파일 참조 분포는 균등 분포를 따르는 것으로 하였다. 실험 결과 분할된 영역의 수를 최대로 하였을 경우, 각각 9.7%, 5.4%, 그리고, 1.8%의 동시 지원 가능한 사용자 요구 수 증가를 보였다.

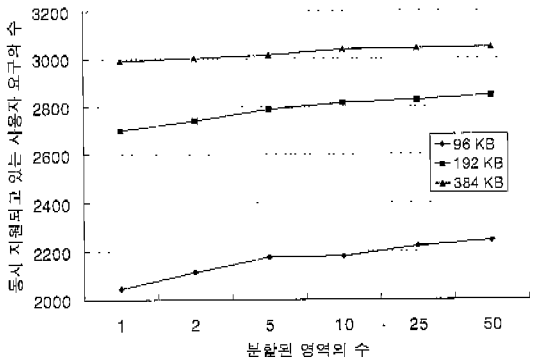


그림 7 서비스 영역 분할에 따른 성능 변화

그림 7의 그래프에서 데이타 블록의 크기가 증가 할수록 서비스된 스트림 수는 증가하였지만, 디스크 영역의 분할로 얻어지는 이득은 줄어들었다. 이는 데이타 블록의 크기가 증가할수록 디스크 탐색 시간에 비해 데이타 전송 시간이 상대적으로 크게 증가하여 탐색 시간 단축에 의해 얻어지는 시간이 추가적인 스트림 지원에 필요한 시간 보다 적기 때문이다.

위의 그래프에서 분할된 영역의 수가 1인 경우 즉, 영역을 분할하지 않은 경우들을 비교하여 보자. 주기의 길이가 1초와 2초인 경우에 사용자 요구당 요구되는 메모리의 양은 0.5초의 주기를 가지는 경우보다 각각 100% 그리고 300% 증가하였지만 이로 인해 얻은 수 있었던 이득은 각각 32% 그리고 46%에 불과하였다. 그러나, 서비스 영역 분할을 통해 디스크 탐색 구간을 최적화할 경우에는 사용자 요구당 요구되는 메모리 양을 증대시키지 않은 상태에서도 동시 지원 가능한 사용자 요구의 수가 대략 10% 증가하였다.

4.2.2 사용자 접근 유형별 성능 변화

사용자 접근 유형 즉, 비디오 파일에 대한 사용자 선호도에 따른 시스템의 성능 개선 정도를 측정하기 위해 Zipf 분포, 균등 분포, 그리고 혼합된 사용자 접근 유형에 대해 실험을 실시하였다. 그림 8과 9는 이 세가지 접근 유형에 대한 성능 개선 정도를 보여준다. 실험은 초기화 시간 1만초에 실험 시간 4만초로 하였으며 주기의 길이와 데이타 블록의 크기는 각각 1초와 192 KB로 하였다. Zipf분포, 균등 분포, 혼합된 사용자 접근 유형에 대해 각각 최대 5.6%, 8%, 5.2%의 스트림 수 증가가 있었다. Zipf 분포의 사용자 접근 유형에 대해서 보다

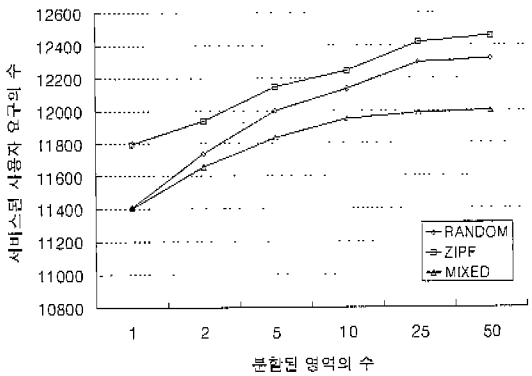


그림 8 사용자 접근 유형별 성능 변화(서버 용량)

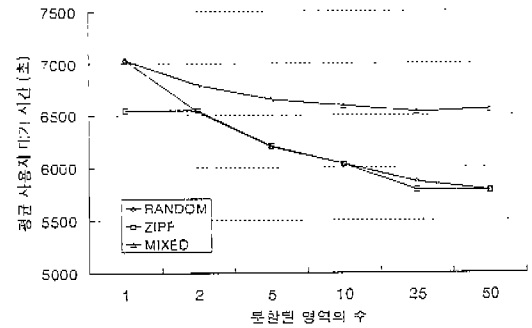


그림 9 사용자 접근 유형별 성능 변화(서비스 지연 시간)

균등 분포의 사용자 접근 유형에 있어서 성능 개선 정도가 더 높은 것은 Zipf 분포 자체가 상당한 지역성(locality)을 보이는 것에 기인한 것으로, 탐색 시간을 줄이기 위한 기법이 적용되지 않은 경우에 있어서도 탐색 시간이 일부 줄어들었음을 보여준다.

그림 10과 11은 상기 실험 내용 중 사용자 접근 유형이 각각 Zipf 분포와 균등 분포를 따를 때 시간대별 서비스되고 있는 스트림의 수를 보여준다. n-Zipf과 n-Random에서 n은 분할된 영역의 수를 의미한다. 사용자 접근 유형이 Zipf 분포를 따르는 경우 시스템으로부터 동시 지원되고 있는 스트림의 수는 평균 5% 증가하였으며 균등 분포의 경우 평균 5.8% 증가하였다.

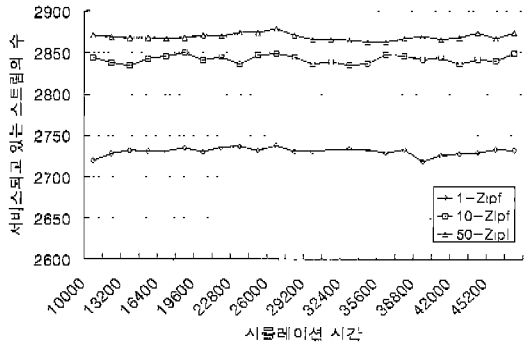


그림 10 Zipf 분포의 사용자 접근 유형에 대한 성능 변화

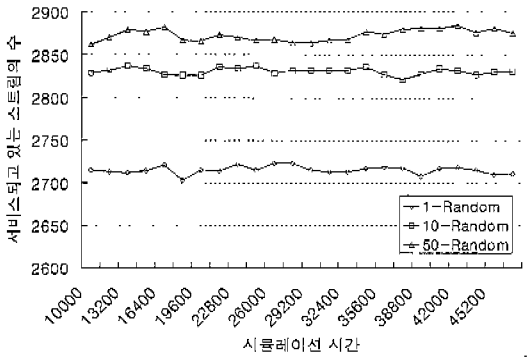


그림 11 균등 분포의 사용자 접근 유형에 대한 성능 변화

4.2.3 사용자 선호도 변화에 대한 시스템 적응성 측정

비디오 파일들에 대한 사용자 선호도의 변화에 따른 시스템 적응성을 보이기 위해 사용자 선호도를 시간 별로 변화시키면서 실험을 실시하였다. 그림 12은 초기화 시간 1만초와 실험 시간 4만초, 총 5만초 동안에 대해 8천초 단위로 사용자 접근 유형을 변화시킨 결과이다. 여기서 사용자들의 비디오 파일 선호도는 초기에 균등 분포로 시작하여 8천초 단위로 Zipf 분포와 번갈아 변화하도록 하였으며 적응 주기는 1천초로 하였다. 그래프에서 가로축은 시간을 의미하며 세로축은 해당 시간에 서비스되고 있는 스트림의 수를 나타낸다. 그림 12에서 10-FIX와 50-FIX는 사용자 접근 유형을 고려하지 않은 상태에서 디스크 배열 영역을 각각 10개와 50개로 분할한 상태의 그래프이다. 1-OZDEN으로 표시된 그래프는 기존의 Ozden 모델을 의미하는 것이며 10-DYM과 50-DYM은 분할된 영역의 수가 각각 10개와 50개인 경우에 있어서 사용자 접근 유형에 따라 서비스 영역을 동적으로 변화시킨 모델에 대한 그래프이다.

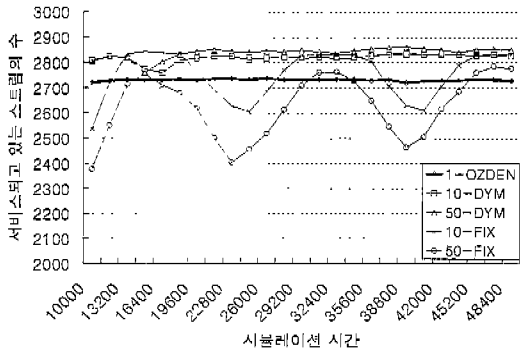


그림 12 사용자 접근 유형 변화에 따른 시스템 성능 변화

그림 12에서 고정된 크기의 분할 영역을 가지는 두 그래프는 사용자 접근 유형이 Zipf 분포를 보이는 구간에서 기존의 방식인 1-OZDEN 보다 적은 스트림들을 지원하고 있음을 알 수 있다. 이는 특정 비디오 파일에 대한 사용자 요구의 집중으로 서비스 그룹들 간의 작업 부하 불균형을 시스템이 극복하지 못한데서 오는 결과이다. 그림 12의 10-FIX와 50-FIX를 통해 알 수 있듯이, 분할된 영역의 수가 커질수록 시스템 성능은 사용자 접근 유형 변화에 더욱 민감해 진다. 분할된 영역의 수가 디스크의 수 만큼 늘어날 경우 사용자들의 선호도 변화에 대해 시스템이 적응해 나가는 과정에서 발생하는 오버헤드가 점차 더 심해질 수도 있다. 즉, 시스템의 서비스 영역이 새로이 변화하는 과정에서 변화하기 이전에 도함되어진 스트림들이 잔존하고 새로운 영역에 대한 사용자 요구들이 수용되는 경우에는 새로운 사용자 요구들에 대한 스케쥴 가능성은 적어지고 스케쥴 되

더라도 해당 서비스 그룹이 실제 서비스하는 영역이 넓어 상대적으로 적은 수의 사용자 요구들만이 스케쥴 되어 진다. 이는 변화 전에 포함되어진 사용자 요구들에 대해 서비스 제공이 모두 종료될 때까지 지속되는 것으로 이러한 경우, 보다 적은 수의 영역으로 분할하였을 때 보다 못한 성능을 보일 수도 있다.

5. 관련 연구

멀티미디어 서버의 설계에 있어서 중요한 쟁점은 얼마나 합리적인 가격으로 최대한의 사용자들에게 양질의 서비스를 제공할 수 있는냐 하는 것이다. 이를 위해 시스템 자원들의 구성, 주어진 자원들의 활용, 그리고 고객들에 대한 서비스 정책등이 효율적으로 이루어져야 한다. 예를 들면, 고가의 자원인 메모리와 이에 비해 저가인 디스크들을 어떠한 정책에 근거하여 어느 정도의 비율로 사용할 것이며 메모리와 디스크들의 효율적인 사용 방안을 결정하고 어느 정도의 서비스를 고객에게 보장하고자 하는지를 결정해야 한다. 이러한 목표들에 대한 기존의 연구들은 주로 버퍼 관리 방안, 디스크 활용 방안, 사용자 스케쥴링 등으로 나뉠수 있으며 본 절에서는 디스크 대역폭과 사용자 접근 유형에 대한 기존의 연구들을 중심으로 살펴보도록 하겠다.

Ozden, Rastogi 그리고 Silberchats는 시스템 구성과 이의 효율적인 이용에 있어서 주기-기반 방식을 바탕으로 디스크 배열과 버퍼의 효율적인 활용을 가격적인 면을 기준으로 분석하였다[3]. 이들은 사용자에게 주기적으로 전송되는 데이타 블록의 크기를 결정함에 있어 메모리 요구량의 증가로 인한 시스템 가격 상승과 지원 가능한 사용자 요구의 수 간의 이해 득실 관계를 설명하였다. 또한, 주어진 가격하에서 효율적으로 구성할 수 있는 메모리와 디스크 배열의 크기를 산출할 수 있는 방법과 이에 적합한 데이타 블록의 크기를 결정하는 방안을 제시하였다. 이들은 버퍼 크기에 의해 시스템 성능이 제한되는 상황에 대해 [8]에서 버퍼 관리를 위해 요구-페이징(Demand-paging) 기법을 이용할 것을 제안하였다. 이는 시스템 내 디스크 대역폭이 부족한 상황에서 사용자 요구들 간의 데이타 블록 공유를 확대함으로써 사용자 요구당 요구되는 디스크 대역폭을 줄인다. 또한, 버퍼가 부족한 상황에서는 기존의 버퍼를 공유하던 사용자 요구들에 대해 별도의 디스크 대역폭을 할당함으로써 버퍼 부족을 해결한다는 것이다. 이와 같이 이들은 저가의 디스크들과 주어진 메모리 자원을 효율적으로 이용하여 시스템에 요구되는 사용자 요구당 서비스 비용을 줄일 수 있는 방안들을 제시하였다.

이와는 달리, Chen 등은 [9]에서 디스크 스케쥴링이 버퍼 요구량에 미칠수 있는 영향을 분석하였고 이를 바탕으로 디스크 대역폭과 버퍼 요구량 간의 이해 득실을 조절할 수 있는 GSS(Grouped Sweeping Scheme) 정책을 제안하였다. 이는 사용자 요구들을 처리함에 있어 디스크 스케쥴링이 SCAN 방식을 지원할 경우 디스크 효율성은 좋아지지만 읽어들인 데이타 블록의 버퍼 내 잔류 시간이 증가되어 버퍼 요구량이 증대하고 선입 선출 방식으로 지원할 경우 버퍼 내 잔류 시간은 줄어들지만 디스크 효율성이 떨어진다는 점에 착안한 것이다. 이에 대해 Chen의 GSS 방식은 한 주기 내 서비스되는 사용자 요구들을 특정 개수 단위로 묶고 각 묶음들 내에서는 SCAN 형식의 디스크 스케쥴링을 행하고 각 묶음들 간에는 선입 선출 방식의 디스크 스케쥴링을 행함으로써 디스크 효율성과 버퍼 요구량 간의 이해 득실을 조율한다.

상기 연구들은 모두 디스크 대역폭과 요구되는 버퍼간의 이해 득실 관계를 조율하고자 하는데 근거하고 있다. 본 논문에서 제시한 사용자 요구 묶음 기법은 이 두 연구에 기반한 것으로, 디스크 배열 내 단위 디스크 입장에서의 효율적인 GSS 방식으로도 볼 수 있다. 그러나, 이들이 디스크 대역폭의 효율적인 활용을 위해 스케쥴링을 행함에 있어, 단위 디스크 관점으로 접근한 반면, 본 논문에서 제시한 기법은 디스크 배열의 특성을 이용하여 인접 디스크들 간의 스케쥴링 간섭을 고려하였다.

이외에도 디스크 배열에 있어서 디스크 대역폭의 효율적인 활용을 위한 많은 연구들이 있었다[10,11]. 이러한 연구들은 단위 사용자 요구를 디스크 배열이 처리함에 있어, 디스크 배열의 병렬성과 병행성을 어떻게 활용하느냐에 초점을 둔 것으로, 서버의 대화식 기능이나[10], 다양한 미디어 형식의 제공을 목적으로 하고 있다[11]. 또한, Shenoy 등은[12]에서 스트라이핑 기법을 이용하는 멀티미디어 서버의 최적 성능을 위한 스트라이핑 단위 설정 방안 등을 다룬바 있다.

본 논문에서 제시한 기법은 사용자 요구의 접근 유형을 충분히 고려하였는데, 사용자 요구의 접근 유형에 관한 기존의 연구는 크게, 접근 유형 분석과 이를 기반으로 한 사용자 요구의 스케쥴링으로 나눌 수 있다. Dan과 Griwodz는 각각 [6]와 [5]에서 실제 영화들에 대한 일반 대중의 선호도 분석을 통해 비디오 화일들에 대한 접근 유형을 분석하고, 다수의 영화에 대해 특정 시간 내 일반 대중의 선호도가 Zipf 분포를 따름을 설명하였다[5]. 또한, Dan은 이러한 사실에 근거한 사용자 요구 스케쥴링 방안을 제시하였다[6]. 이는 동일 비디오 파일을 요구하는 사용자 요구들을 멀티캐스팅(multicasting)함에

있어, 서비스를 기다리는 사용자 요구들이 장시간의 대기로 인해 서비스를 포기하는 정도를 낮추기 위해, 이들의 참을성과 사용자 요구의 발생 유형을 바탕으로 사용자 요구들에 대한 스케쥴링 방안을 결정한다는 것이다.

6. 결 론

멀티미디어 서버를 설계함에 있어 가장 중요한 것은 주어진 자원을 가지고 얼마나 많은 사용자들에게 지속적인 서비스를 제공할 수 있느냐 하는 것이다. 디스크 배열을 이용하는 서버의 경우, 지원 가능한 스트림의 수를 늘리기 위해서는 블록의 크기를 증가시켜 주기의 길이를 늘려야 한다. 하지만, 이와 같은 방법은 메모리 요구량 또한 현저히 증가시키게 되어 시스템 전반에 걸친 설계 비용이 증가하게 된다. 본 논문에서는 물리적으로 인접한 곳을 접근하는 사용자 요구들을 하나의 묶음으로 만들고, 인접한 사용자 요구 묶음들 간의 디스크 이용에 있어서의 상호 연관성을 고려하여 이들을 배치함으로써, 스트림당 요구되는 디스크 대역폭의 크기를 줄이고, 더 나아가 지원 가능한 스트림의 수를 늘릴 수 있는 방안을 제시하였다. 주문형 비디오 서버를 가정한 모의 실험 결과, 디스크 탐색 구간 최적화를 통해 스트림당 요구되는 디스크 대역폭의 양을 줄임으로써, 동시 지원 가능한 사용자의 수를 늘이고, 평균 사용자 대기 시간이 줄어들 수 있음을 확인하였다. 그 결과, 96 KB의 데이타 블록을 사용하는 경우에 있어 동시 지원 가능한 사용자 요구의 수가 평균 9.7% 향상되었다.

현재까지 제안된 멀티미디어 디스크 스케쥴링 방안들이 대부분 단일 디스크 입장에서 기존의 디스크 스케쥴링 방법들을 바탕으로 하여 진행되어 있지만, 본 논문에서는 멀티미디어 디스크 스케쥴링이 디스크 배열 전체를 고려하여 이루어졌을 때 그 효율성이 극대화 될 수 있음을 보여 주었다.

참 고 문 헌

[1] M. Holland. On-Line Data Reconstruction in Redundant Disk Arrays. PhD thesis. Computer Science, Carnegie Mellon University, 1994.

[2] D. Gemmell, H. Vin, D. Kandlur, P. Rangan, and L. Rowe. Multimedia Storage Servers: A Tutorial and Survey. IEEE Computer. 28(5):40--49. May 1995.

[3] B. Ozden, R. Rastogi, and A. Silberschatz. Disk Striping in Video Server Environments. In Proceedings of the IEEE International Conference on Multimedia Computing and Systems, pages 580--589, June 1996.

[4] D. A. Patterson, G. Gibson, and R. H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 109--116, January 1988.

[5] C. Griwodz, M. Bar, and L. C. Wolf. Long-term Movie Popularity Models in Video-on-Demand Systems or The Life of an on-Demand Movie. In Proceedings of the ACM Multimedia'97, pages 349--357, November 1997.

[6] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling Policies for an On-Demand Video Server with Batching. In Proceedings of the 2nd Annual ACM Multimedia Conference and Exposition, pages 15--23, October 1994.

[7] E. K. Lee. Performance Modeling and Analysis of Disk Arrays. PhD thesis, Computer Science, University of California at Berkeley, 1993.

[8] B. Ozden, R. Rastogi, and A. Silberschatz. Demand Paging for Movie-on-Demand Servers. In Proceedings of the IEEE International Conference on Multimedia Computing and Systems, pages 264--273, May 1995.

[9] M. S. Chen, D. D. Kandlur, and P. S. Yu. Optimization of the Grouped Sweeping Scheduling (GSS) with Heterogeneous Multimedia Server. In Proceedings of the ACM Multimedia'93, pages 235--242, August 1993.

[10] M. S. Chen, D. D. Kandlur, and P. S. Yu. Support for Fully Interactive Playout in a Disk-Array-Based Video Server. In Proceedings of the ACM Multimedia'94, pages 126--135, October 1994.

[11] S. Berson, S. Ghandeharizadeh, R. Muntz, and X. Ju. Staggered Striping in Multimedia Information Systems. In Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 79--89, June 1994.

[12] P. Shenoy and H. M. Vin. Efficient Support for Interactive Operations in Multi-resolution Video Servers. ACM Multimedia Systems Journal, 7(3): 241--253, May 1999.

김 운 석

1997년 홍익대학교 컴퓨터공학과 학사. 1999년 서울대학교 컴퓨터공학과 석사. 2000년 ~ 현재 서울대학교 컴퓨터공학과 박사과정 재학중. 관심분야는 실시간 시스템, 저전력 시스템, 멀티미디어 시스템임

김 지 홍
정보과학회논문지 : 시스템 및 이론
제 28 권 제 10 호 참조

민 상 렬
정보과학회논문지 : 시스템 및 이론
제 28 권 제 10 호 참조

노 삼 혁
정보과학회논문지 : 시스템 및 이론
제 28 권 제 10 호 참조