한국통신학회지 제 14 권 제 7 호 1996년 7월

《主 題》

VOD 시스템을 위한 버퍼링 기법

전용희 · 박정숙 · 박영덕*
(대구효성가톨릭대학교, 한국전자통신연구원*)

□차 례□

Ⅰ. 서 론

VOD 데이타와 같은 연속 매체(CM: Continuous Media)는 정해진 시간내에 연속하여 제시될 때에만 의미를 가진다. 이와 같이 VOD 시스템에서 요구되는 중요한 특성으로는 연속성(continuity)이 있다. 연속성이란 시간에 맞게 그리고 동시에 손실없이 비디오 정보들을 클라이언트에게 전송해주는 성질을 말한다. 이러한 성질을 만족시켜 시스템의 성능을 최대화하기 위해서는 일정량의 데이타 블럭을 미리 읽어둘(read-ahead 혹은 pre-fetch) 필요가 있으며, 데이타를 잠시 보관하기 위하여 버퍼링(buffering)이 요구된다. 특히 VOD 시스템에서는 디스크로부터 읽어올 때 발생하는 비동기적인 성질과 클라이언트의 주기적인 성질 사이의 차이를 해결해 주기 위하여 버퍼링이 필요하다. 그러므로 VOD 시스템을 위한 버퍼링 기법에 대한 연구도 필수적인 사항이다.

VOD 시스템에서의 데이타와 같은 연속 매체의 재생을 위하여 서버는 출력 장치에서 기아(starvation)를 방지할 수 있도록 디스크(즉, 저장소)로부터 충분한 데이타를 서버 버퍼에 공급하여야 한다. 즉, 버퍼된 데이타(즉, 누적 데이터 도착 - 누적 데이터 소비)의 양이 절대로 음수(negative)가 될 수 없음을 의미한다. 이런 성질을 가지는 알고리즘들을 work ahead-augmenting 혹은 buffer-conserving이라 한다. 버퍼 보존(buffer conservation)은 기아(starvation)를 방지하기 위한 필요 조건은 아니지만 충분 조건으로 사용할 수는 있다.

어떤 매체 스트림의 연속 검색은 데드라인을 가진 일련의 주기적인 태스크로 이루어져 있다. 여기서 태스크는 디스크로부터의 미디어 블럭 검색에 해당되며, 데드라인은 계획된 재생 시간에 해당된다. 검색은 버스티하기 때문에 미디어 블럭들은 검색이 재생을 앞서게 될 경우 버퍼될 필요가 있게 된다.

Ⅱ. 기본적인 버퍼링 모델

일반적인 버퍼링을 서버의 데이타 공급함수(R(t))와 클라이언트의 데이타 소비함수(C(t))의 차이인 버퍼함수(B(t))로 그림 1과 같이 모델할 수 있다[1].

$$B(t,t0) = R(t,t0)-C(t,t0) \quad (1)$$

B(t,t0): 버퍼함수(B(t)), 시간 t에서 버퍼에 저장되어 있는 양
R(t,t0): 공급함수(R(t)), 서버의 데이터 전송
C(t,t0): 소비함수(C(t)), 클라이언트의 데이터 소비

그림 1에서 버퍼크기는 Bmax와 Bmin에 의해 결정되며, 이때 Bmin이 0인 경우는 버퍼의 기근(starvation) 현상이 발생하지 않으므로 버퍼 크기를 Bmax만큼 정하고, Bmin이 0보다 작은 경우는 버퍼의 기근 현상이 발생하므로 B(t) 함수를 B'(t) =

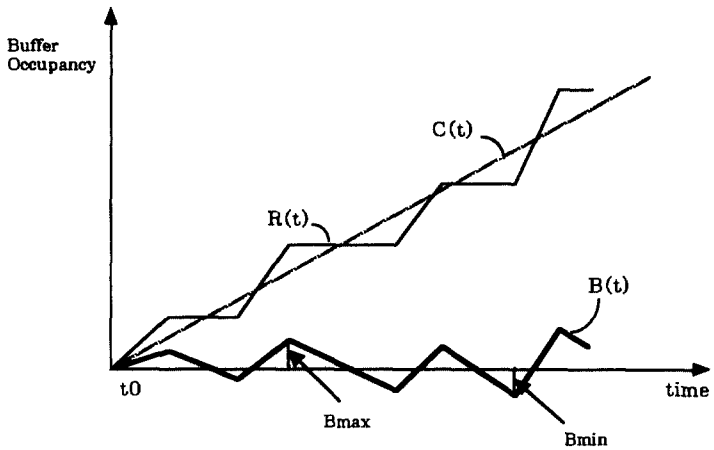


그림 1. 공급함수와 소비함수에 따른 버퍼함수의 변화

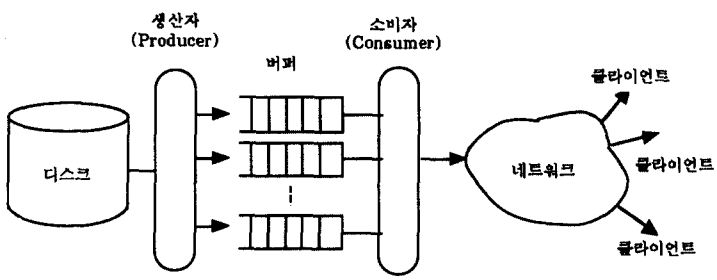


그림 2. VOD 서비스 시스템에서의 생산자-소비자 모델

B(t) + |Bmin| 함수로 보정하여 B'(t)의 최대치만큼의 버퍼크기를 정하면 버퍼의 기근이나 과잉없이 연속재생이 가능하다.

이러한 버퍼함수를 그림 2에서와 같이 서버에 있는 버퍼만을 고려하여 생산자-소비자 모델(Producer-Consumer Model)로 표현하였다[2]. 즉, 생산자가 요청된 스트림을 디스크로부터 읽어서 버퍼에 갖다 두면 소비자는 그 데이타를 망을 거쳐 그 데이타를 요청한 사용자에게 전달한다.

이때 망을 거치는 전송경로 상에서 비디오 정보는 디스크로부터 읽어오는 전달단위보다는 더 작은 패킷들로 전송되므로, 다운스트림(downstream) 버퍼가 거의 또는 전혀 없게 된다. SCSI 포트로부터 동시에 서비스되는 스트림의 수를 Nbus라 하고 U를 디스크로부터의 전달 단위라 하면, 필요한 버퍼크기는 NbusU가 된다. 그러나 비동기적인 데이타 검색 및 검색된 데이타들을 전달하는 ATM 망의 비동기적 통신과 동기적인 재생 사이의 차이를 평활화하기 위해 부가적인 버퍼 공간이 요구된다. 일반적으로는 스트림당 2U의 버퍼 공간이 요구되며 하나가 데이타를 읽어오고 있을 때 다른 하나는 데이타 소비를 위해 사용되며 그 역으로도 가능하다. 그래서 만일 동시에 서비스되는 스트림 수를 n이라 하면 전체 2nU의 버퍼 공간이 필요하게 된다.

III. 버퍼 요구사항

3.1 라운드 길이와 지연(latency)의 상호보완(trade-offs)

연속적인 스트림 검색 사이의 지연은 재생 시작 지연과 버퍼 요구사항에 대해 여러 의미를 가진다. 또

한 여러 스트림을 동시에 서비스할 경우는 매 주기동안 모든 스트림에 대해 일정량의 데이타를 읽어야 한다[3]. 일반적으로, 전송단위가 증가하면 재생 초기지연(클라이언트의 재생요청 시점부터 재생시작까지의 시간)과 버퍼요구량도 증가한다.

그림 3은 디스크 스케줄링별 라운드 길이와 서비스 시간(재생 지연) 사이의 관계를 나타내며, 그림 4는 디스크 스케줄링별 서비스 타이밍과 버퍼 요구량을 나타낸다.

라운드 로빈 알고리즘에서는, 스트림의 첫번째 요구로부터의 모든 블럭들이 검색된 후 재생이 곧 바로 시작될 수 있다. 스트림의 연속적인 요청의 검색 시간 사이의 최대 지연은 한 라운드의 길이에 의하여 제한된다. 한 라운드동안 데이타 소비를 만족하는 버퍼 공간만 있으면 충분하다.

Scan 알고리즘에서는, 한 스트림이 라운드의 시작부에서 서비스를 받고 다음 라운드의 마지막 부분에서 서비스를 받을 수 있다. 그러므로 재생은 라운드

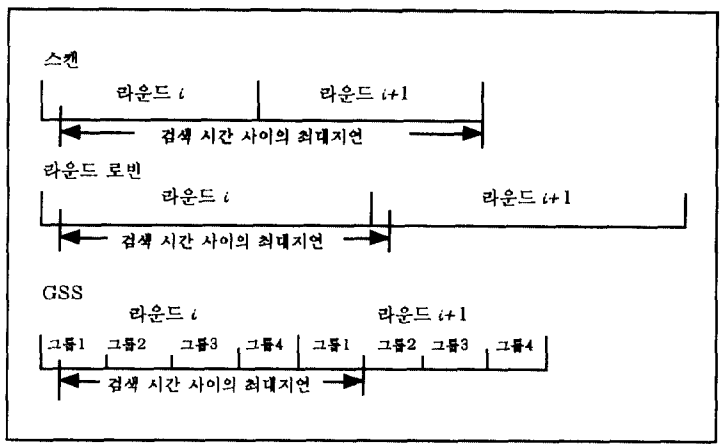


그림 3. 스케줄링별 라운드 길이와 서비스 시간 사이의 관계

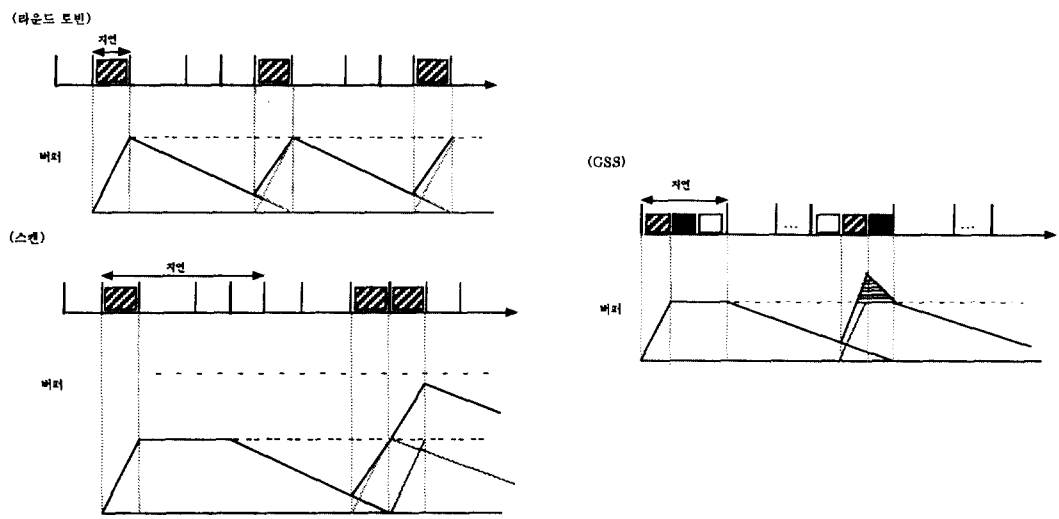


그림 4. 스케줄링별 서비스 타이밍과 버퍼요구량의 예시

의 마지막까지 대기하여야 한다. 거의 두개 라운드동안 소비를 만족하는 충분한 버퍼 공간이 필요하다. 그러나 Scan의 라운드가 라운드 로빈보다 짧기 때문에 그림 3과 같이 연속적인 스트림 검색 사이의 라운드 길이와 지연사이에 상호보완이 존재한다.

GSS에서는 재생이 각 그룹의 끝에서 시작되므로 지연은 각 그룹의 길이로 제한되며 필요한 버퍼크기도 한 그룹안에 있는 스트림 수에 대해서만 추가적으로 필요하므로 Scan에 비해 버퍼 요구량도 줄일 수 있다.

3.2 읽기(reading) 및 버퍼링 요구사항

한 라운드동안 하나의 스트림을 위하여 받는 데이타의 양은 적어도 그 스트림 재생에 의하여 소비될 양과 같아야 한다. 이것은 라운드 별로 기초하여 데이타 생산이 소비에 뒤쳐지지 않음을 의미한다. 즉, 버퍼된 데이타의 양이 절대로 음수(negative)가 될 수 없음을 의미한다. 이런 성질을 가지는 알고리즘들을 work ahead-augmenting 혹은 buffer-conserving이라 한다. 버퍼 보존은 기아를 방지하기 위한 필요 조건은 아니지만 충분 조건으로 사용할 수는 있다. 연속적인 매체 스트림의 재생을 보장하기 위하여 충분한 수의 블럭들이 라운드의 전체 기간동안 매 라운드 동안 모든 시청자를 위하여 검색되어야 한다. 이 수를 결정하기 위하여 서버는 한 라운드의 최대 기간을 알아야 한다. 라운드 길이는 모든 스트림에 대하여 검색되는 블럭들의 수에 의존한다. 라운드 길이를 최소화하기 위하여 모든 라운드동안 모든 스트림에 대하여 검색되는 블럭들의 수는 스트림의 소비율(consumption rate)에 비례해야 한다.

다음에는 버퍼링을 위한 요구사항에 대해서 기술한다. 서버는 수행될 다음 읽기를 위하여 충분한 자유 공간을 남겨두도록 버퍼를 관리해야 한다. 하나의 스트림당에 기초하면 가장 적절한 버퍼 모델은 선입 선출(first-in first-out) 큐이다. FIFO, 연속적인(contiguous) 화일 및 라운드 로빈 스케줄링을 사용하면 버퍼 크기는 최대한 요구되는 읽기 크기와 근사하게 된다. 이 경우에 각 스트림의 FIFO는 모든 라운드에서 단순히 "가득 채워진다(topped up)".

대조적으로, Scan 전략은 적어도 이중 버퍼 기법(double-buffered scheme)이 요구된다. 여기서 각 버퍼는 최대 읽기 크기이다. 가득 채우기 전략으로 모든 읽기 주기동안 한 스트림에 대한 읽는 양은 자유 버퍼 공간의 크기에 따라 변하게 된다.

- □ 두 번째 집단의 데이타 블럭들을 I/O 스케줄러로 보내는 동안 디스크로부터 병행해서 m 데이타 블럭들을 미리 가져온다(prefetch). 이 경우 적어도 2×m 데이타 블럭의 버퍼가 소요된다.
- □ 각 데이타 블럭이 소모되는 동안 추가적인 데이타 블럭들을 가져온다. 이 경우 적어도 m+1 데이타 블럭의 버퍼가 소요된다.

다음 두개의 절에서는 VOD 서버에서의 버퍼링 기법에 대해 크게 단일 디스크인 경우(독립적인 디스크들로 구성되는 디스크 배열도 포함)와 스트라이핑시킨 디스크 배열인 경우로 나누어 기술하였다.

IV. VOD 서버에서의 버퍼링 기법

앞에서 기술한 바와 같이 VOD 서버에서 스트림들을 서비스할 경우에는 주기를 정해야 한다. 이러한 주기 C는 버퍼에 있는 한 전송단위를 재생하는 시간을 말하며 전송단위 U와 재생률 R의 비로 식 (2)와 같이 나타낼 수 있다.

$$C = \frac{U}{R} \tag{2}$$

즉, 이 식은 매 주기마다 U 크기의 데이타가 각 스트림을 위해서 디스크로부터 읽혀져야 함을 의미한다. 그렇지 않을 경우는 버퍼에 기근 현상이 발생되고 재생이 불연속적으로 되어 비디오의 QoS 저하를 초래한다. (2)식에서도 나타나듯이 U가 커질수록 그리고 R이 낮을수록 C가 길어져 동시에 많은 스트림을 수용할 수 있지만 U가 커지면 버퍼크기도 커져야 하므로 비용이 많이 들게 된다.

4.1 이중 버퍼 기법(Twin buffer scheme)

가장 기본적인 버퍼 기법으로, 한 사이클에 비디오 내용을 내보내는 동안 다음 사이클에서 필요한 데이타를 미리 읽어두기 위해 버퍼를 이중으로 구성한다. 이는 앞서 기술한 생산자-소비자 모델에서와 같은 원리로, 소비자가 데이타를 연속적으로 소비할 수 있도록 하기 위해서 생산자가 소비자의 요청 이전에 미리 데이타를 버퍼에 갖다 놓아야 한다. 이때 만약 연속적인 메모리 세그먼트에 읽혀져야 한다면 한 스트림

에 할당되어야 하는 버퍼 구역은 현재 내보내는 데이타와 유입되는 전달단위 둘 다를 유지할 수 있을 정도로 커야 한다.

이중 버퍼 기법은 버퍼의 공유 여부에 따라 완전 분할(Complete partitioning)과 버퍼 공유(Buffer sharing)로 세분할 수 있다.

가. 완전 분할(Complete partitioning)

이 경우 수용할 수 있는 최대 스트림 수를 Nmax라 하면 버퍼 공간은 한 스트림에 하나씩 Nmax개의 독립된 구역으로 나누어진다. 각 스트림을 위한 버퍼 구역은 전송단위의 2배가 되어야 하므로 필요한 총 버퍼크기는 2NmaxU가 된다. 또한 독립된 Nd개의 디스크로 구성된 디스크 배열에서 데이타들이 읽혀진다면 필요한 버퍼 크기 Bpartitioned는 식 (3)과 같다.

$$Bpartitioned = 2NmaxNdU \quad (3)$$

나. 버퍼 공유(Buffer Sharing)

만약 데이타가 버퍼의 비연속적인 부분에 읽혀질 수 있다면 버퍼 공간을 여러 스트림들 사이에 공유되도록 만들 수 있다. 따라서 각 스트림은 매 C초마다 한 블럭을 소비하고 그 블럭 소비의 끝에서 해당하는 버퍼를 해제하여 다음 스트림을 위해 사용되도록 한다. 그러므로, 디스크 배열인 경우 한 사이클의 시작에서 모든 스트림 서비스에 필요한 NmaxNd개의 버퍼를 할당하는 것이 아니라 그 사이클에서는 첫번째 Nmax개의 블럭을 읽어올 버퍼만 할당하고 C초마다 스트림들에 의해 해제되는 Nmax개의 버퍼는 다음 Nmax개의 I/O를 위해 사용된다. 그러므로 필요한 총 버퍼 크기는 식 (4)와 같다.

$$Bshared = Nmax\ (Nd + 1)\ U \quad (4)$$

그러나, 연속적인 버퍼할당 기법이 버퍼로의 데이타 전송을 상당히 단순화시키고, I/O 보드 제어기상의 실시간 요구를 감소시키므로, 일반적으로는 연속적인 버퍼공간 할당방법이 선호된다.

4.2 GSS(Grouped sweeping schedule) 기법

이 기법에서는, 그룹수를 g라 할 때 한 서비스 사이클이 g개의 서브사이클로 나누어지고 각 서브사이클 동안에 디스크는 해당되는 그룹에 속하는 Nmax/g개의 요청을 서비스한다. 그러므로 이중버퍼는 한번에 Nmax/g개의 요청을 위해서만 사용된다. 따라서 필요한 총 버퍼크기는 (Nmax+Nmax/g)U가 된다.

이때 이중버퍼로 사용된 메모리는 각 서브사이클의 끝에서 다음 서브사이클을 위한 요청이 발행되기 전에 비워져야 하는데, 이러한 처리에 부가되는 오버헤드는 각 서브사이클에 대해 행해져야 한다. 이것은 U의 크기를 증가시켜 Scan 기법보다 더 큰 버퍼요구량을 발생시킬 수 있는 단점이 있다. 실험 결과에 의하면 GSS는 데이타율이 작을 때 각 서브사이클에서 서비스받을 수 있는 스트림수가 증가하므로 성능이 상당히 개선된다. 또한 GSS는 디스크 스트라이핑인 경우에 매우 유용한 성질을 가진다.

V. 스트라이핑된 디스크에서의 버퍼관리

디스크 스트라이핑 기법이 사용될 때는 스트림의 데이타율 요구조건을 만족시키기 위해 동일 스트림에 대한 몇개의 데이타 전달 단위가 다른 디스크들로부터 읽혀져야 한다. 이러한 액세스 모드에서는 모든 스트림들에 대한 데이타 요청이 동시에 발행되고 d-디스크 배열에서의 다른 디스크들에 대한 액세스는 시간적으로 정렬되며, 배열로부터 읽어오는 데이타량은 Ud가 된다. 각 스트림에 대해 이중 버퍼 기법을 사용하면 요구되는 총 버퍼 공간은 각 스트림에 대해 2Ud가 된다.

이러한 버퍼 요구량을 줄이기 위한 기법이 여러 논문[4]에서 제안되어 있으며 그 대표적인 것으로 오프셋 스케줄 기법(offset-schedules scheme)과 스플릿 스케줄 기법(split schedules)이 있다. 본 절에서는 스트라이핑되는 단위가 한 디스크의 전송단위와 같다고 가정한다.

5.1 오프셋 스케줄(Offset schedules)

이 기법은 그림 5와 같이 각 디스크는 한 사이클 동안에 필요한 모든 액세스 요청을 동시에 수신하지만 디스크 배열 내에 있는 서로 다른 디스크들에 대한 사이클은 서로서로에 대해 시간적으로 변위(오프셋)되어 있다.

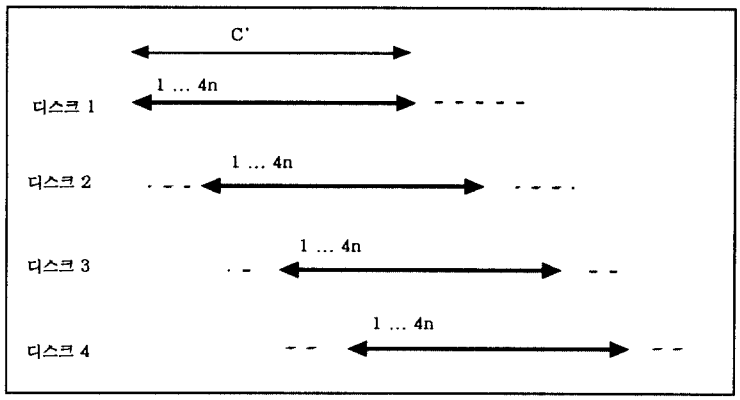


그림 5. 오프셋 스케줄 기법(4-디스크 배열)

만약 d-디스크 배열 내에 있는 디스크들이 1부터 d까지 번호매겨져 있고 C'가 사이클 길이라고 한다면, 이 경우에 디스크 1이 시간 t에서 새로운 서비스 사이클을 시작한다면 디스크 i는 시간 t+(i-1)C'/d에서 새로운 사이클을 시작할 것이다. 각 스트림에 대해 새로운 전송단위 U'가 매 C'/d 초마다 버퍼로 적재되며 이때 필요한 버퍼크기는 (d+1)U'가 된다. 앞에 소개된 기본적인 기법에 비해서 (d+1)U' 〈 2dU' 이기 때문에 상당한 양의 버퍼 요구량을 절감할 수 있다.

이 기법에서는 버퍼 관리가 그림 6과 같은 원리로 이루어진다. 환상 버퍼(circular buffer)에 있는 각 세그먼트의 크기는 디스크당 전달 단위인 U'와 같다. 그림 6에서 재생과정이 현재의 버퍼를 다 소비할 때까지 다른 디스크들 중 하나가 그 자신의 버퍼를 채운다. 그리고 나면 재생 과정이 그 버퍼로 이동하게 되고 비어진 버퍼는 디스크 1에 할당되어 디스크 1은 새로운 서비스 주기를 시작한다. 디스크 1은 재생 과정이 한 바퀴 돌아오기 전까지 그 스트림의 또다른 전송단위를 채워두어야 한다.

한 버퍼를 소비하고 나면 재생과정은 시계방향으로 있는 다음 버퍼로 이동하게 되는 반면, 디스크들은 반시계 방향으로 연속적인 버퍼에 할당된다. 이 기법에서의 최대 시작지연은 2C'가 된다.

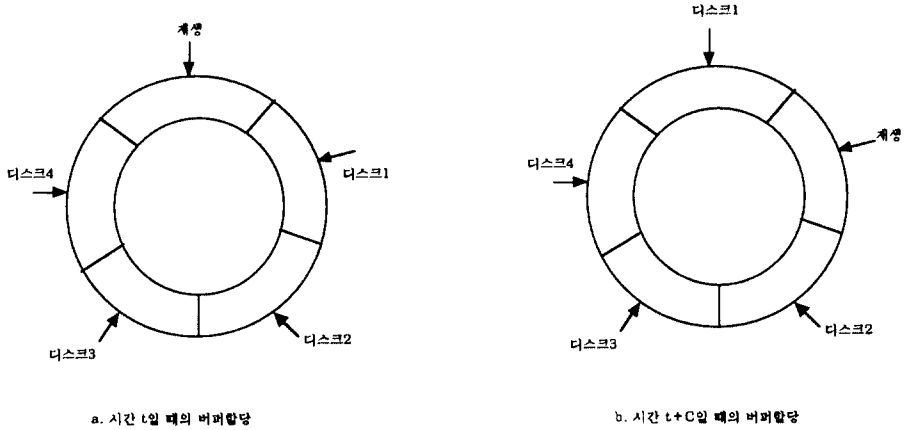


그림 6. 오프셋 스케줄의 버퍼관리(4-디스크 배열)

5.2 스플릿 스케줄(Split schedules)

스플릿 스케줄은 [4]에서 제안된 기법이다. 이 기법을 소개하기 전에 우선 GSS 기법을 스트라이핑된 디스크 배열에 적용시키는 방법부터 기술한다.

가. GSS 기법

전달단위 U를 가진 독립된 단일 디스크로부터 지원될 수 있는 스트림의 수가 n이면, GSS는 그림 7에서처럼 d-디스크 배열로부터 dn개의 스트림 수를 크기 n의 d개 그룹으로 그룹화하여 C=U/R 길이의 연속적인 사이클동안 라운드 로빈 형식으로 그룹들을 서비스함으로써 스트림당 버퍼 크기 (d+1)U를 가지고 지원할 수 있다.

GSS에서 이중 버퍼링은 현재 발생하는 요청에 대해서만 요구되고, 어느 때라도 dn개의 요청 중 n개만 요청을 서비스하면 되므로 전체 버퍼크기는 식 (5)와 같다.

$$nd \times dU + n \times dU \qquad (5)$$

또한 스트림당 버퍼크기는 (d+1)U이다. 시작지연은 버퍼까지 전달하는데는 최고 dC까지 걸리며 실제적인 재생까지는 (d+1)C가 걸릴 수 있다.

GSS에서의 버퍼요구량은 오프셋 스케줄 기법보다는 다소 높다. 그 이유는 오프셋 스케줄이 모든 nd개의 요청들을 동시에 서비스하여 탐색 효율성이 높으므로 오프셋의 전송단위인 U'가 U보다 작기 때문이다.

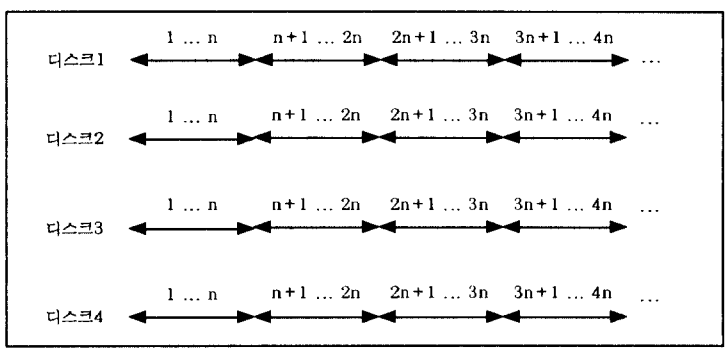


그림 7. GSS 기법(4-디스크 배열)

나. 스플릿 스케줄(Split schedules)

[4]에서는 디스크 스트라이핑시 스트림당 필요한 버퍼요구량을 (d+1)U에서 2U까지 줄일 수 있는 스플릿 스케줄 기법을 제안하였다. 그림 8에 그 원리가 기술되어 있다. 디스크 배열에 의해 지원가능한 전체 요청들을 n개 요청들의 집합들로 나눈다. 이때 n은 한 디스크로부터 지원될 수 있는 요청의 최대수와 같다. 또한 디스크 배열로부터는 nd개까지의 스트림이 지원될 수 있다. n개의 요청들을 지원하기 위해서는 주기를 C로, 전달단위를 U로 설정한다. 각 사이클에서 한 디스크는 다른 n개의 요청들을 가진 그룹을 서비스한다. 이때 스트라이핑 단위는 U와 같고 스케줄은 디스크별로 변위되어 있다.

스플릿 스케줄 기법의 최대 시작 지연은 (d+1)C로 GSS와 같다. 이 기법에서는 디스크 배열에 대해서도 단순히 이중 버퍼링 기법을 사용하여 지원할 수 있다. 이중 버퍼가 디스크 배열에 있는 모든 d개의 디스크에 접근가능하다고 가정하면, 한 스트림은 두 사이클동안, 주어진 버퍼 풀(pool)로부터 한 버퍼를 사용할 수 있다. 즉 한 사이클동안은 디스크 i로부터 데이타를 읽어오는데 사용하고 또다른 사이클 동안에는 그 데이타를 재생하는데 사용한다.

이 원리는 그림 9를 예로 들어 설명하면 다음과 같다. 디스크 i로부터 데이타를 읽어올 동안 재생은 버퍼 풀 i-1에서 발생한다. 그리고 버퍼 풀 i로부터 재생이 진행되는 동안에는 버퍼 풀 i+1로부터의 버퍼가

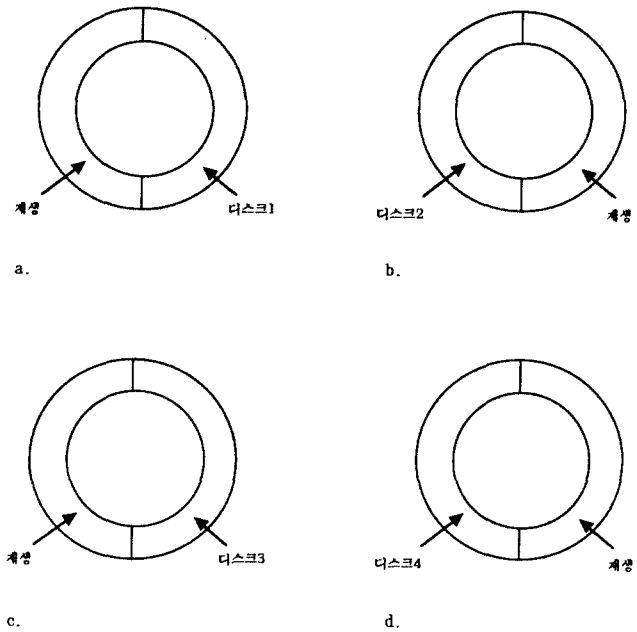


그림 8. 스플릿 스케줄(4-디스크 배열)

디스크 i+1로부터 데이타를 읽어오는데 사용된다. 어떤 때에도, 한 스트림은 그것에 할당된 두개의 버퍼만을 가지며 그 각각의 버퍼는 두개의 연속적인 버퍼 풀 중 하나에 속하는 것이다.

스플릿 스케줄은 독립적인 디스크의 경우보다 더 큰 버퍼공간을 요구하지 않는 스케줄 기법을 생성하기 위해 오프셋 스케줄과 GSS의 그룹화 기법을 결합시킨 것이다. 이 기법은 VOD 서버에서 큰 규모로 디스크 스트라이핑을 시킬 경우에 문제가 되는 장애 요인을 효율적으로 제거할 수 있다.

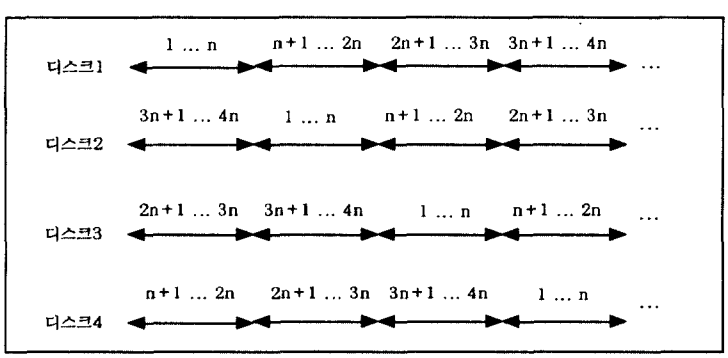
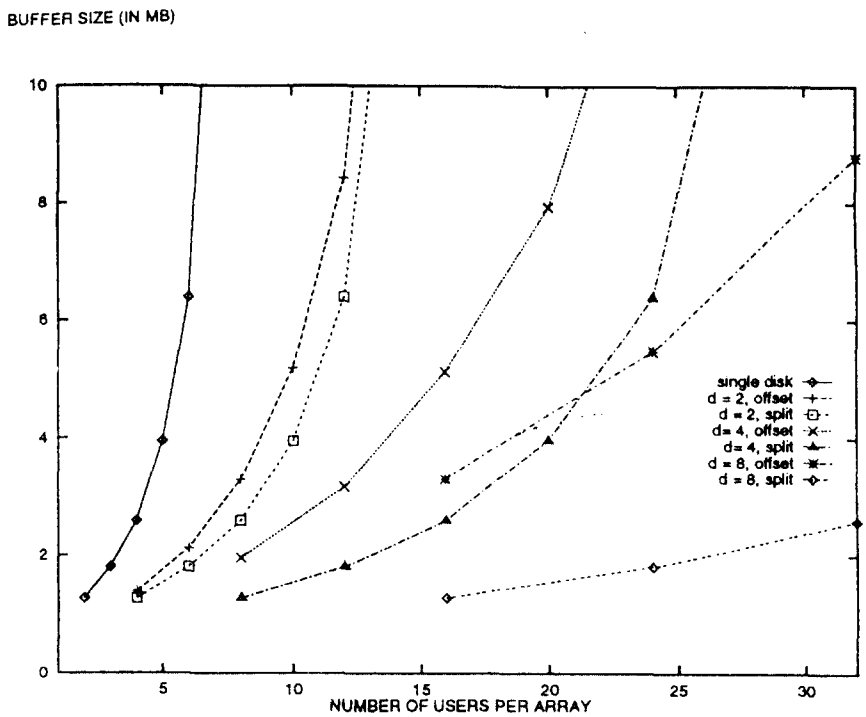


그림 9. 스플릿 스케줄의 버퍼 사용(4-디스크 배열)



(그림 10. 디스크 스트라이핑을 위한 버퍼 크기(R=3Mb/s))

그림 10은 오프셋 스케줄과 스플릿 스케줄에 대해 디스크 배열 크기에 따른 버퍼크기 요구량을 나타내고 있다. U'<U임에도 스플릿 스케줄에 대한 버퍼크기(2U)가 오프셋 스케줄에 필요한 버퍼크기((d+1)U')보다는 상당히 낮다는 것을 확인할 수 있다.[4]

VI. 결 론

VOD 시스템에서 요구되는 중요한 특성으로는 연속성(continuity)이 있다. 연속성이란 실시간적으로 손실없이 스트림 데이타들을 클라이언트에 전달하기 위한 필수적인 요구사항이다. 이러한 성질을 만족시키고 시스템의 성능을 최대화하기 위해서는 일정량의 데이타 블럭을 미리 읽어둘(read-ahead 또는 pre-fetch) 필요가 있으며, 이러한 데이타를 잠시 보관하기 위해서 버퍼링(buffering)이 요구된다. 특히 VOD 시스템에서는 디스크로부터 읽어올 때 발생하는 비동기적인 검색과 클라이언트의 주기적인 재생 사이의 차이를 해결해 주기 위해 버퍼는 망 대역폭과 함께 VOD 시스템의 주요한 자원으로 사용된다.

VOD 시스템에서는 크게 클라이언트와 서버 두 곳에서 버퍼가 필요하다. 버퍼를 위치시키는 방법에 따라 1) 서버와 클라이언트 양쪽에 버퍼를 두는 경우와 2) 한쪽에 버퍼를 두는 경우로 나눌 수 있다. 양쪽에 버퍼를 위치시킬 때 서버쪽의 버퍼는 서버의 전송률과 클라이언트의 소비율 차를 보정하기 위해 사용되며, 클라이언트쪽의 버퍼는 네트워크로 인해 발생하는 전송률의 오차를 보정하는 역할을 한다. 한쪽에 버퍼를 두는 경우는 주로 클라이언트쪽에 버퍼를 위치시키며, 이때 버퍼는 서버의 전송률과 클라이언트의 소비율 사이에서 네트워크 전송으로 인해 생긴 차이를 해결하는 역할을 한다. 이 경우에는 양쪽에 버퍼를 두는 것보다 클라이언트에서 더 큰 버퍼가 요구

된다.

요약하면, 단일 스트림을 효율적으로 서비스하는 문제는 다음과 같이 3 가지 점에서 문제가 된다[5].

1) 시청자 버퍼의 기아로부터의 방지
2) 버퍼 공간의 최소화
3) 개시(start) 지연의 최소화

본 고에서는 VOD 시스템에서의 버퍼링 기법들에 대하여 기술하였으며, 향후 연구를 위한 기초적인 자료로 사용될 수 있을 것이다.

감사의 글

본 연구는 HAN/B-ISDN 출연 초고속 정보통신 서비스 기반 기술 개발 과제로 수행되었습니다.

참고문헌

[1] P. Venkat Rangan, Harrick M. Vin, and Srinivas Ramanathan, "Designing an On-Demand Multimedia Service", IEEE Comm. Magazine, July 1992, pp. 56-64.

[2] Huanxu Pan, Lek Heng Ngoh and Aurel A. Lazer, "A Time-Scale Dependent Disk Scheduling Scheme for Multimedia-on-Demand Servers", Proc. MULTIMEDIA'96, Sep. 1996, pp. 572-579.

[3] Ernst Biersack, Frederic Thiesse, Christoph Bernhardt, "Constant Data Length Retrieval for Video Servers with Variable Bit Rate Streams", Proc. MULTIMEDIA'96, Sep. 1996, pp. 151-155.

[4] Antoine N. Mourad, "Issues in the design of a storage server for video-on-demand", Multimedia Systems (1996) 4: 70-86.

[5] D. James Gemmell, Harrick M. Vin, Dilip D. Kandlur, P. Venkat Rangan, Lawrence A. Rowe, "Multimedia Storage Servers: A Tutorial", IEEE Computer, May 1995, pp. 40-49.

전 용 희

- 1978년 2월 : 고려대학교 전기공학과 졸업
- 1985년 9월~87년 8월 : 미국 플로리다공대 대학원 수학
- 1989년 8월 : 미국 노스캐롤라이나주립대 Elec. and Comp. Eng. 석사(MS)
- 1992년12월 : 미국 노스캐롤라이나주립대 Elec. and Comp. Eng. 박사(Ph.D)
- 1978년 1월~78년11월 : 삼성중공업(주) 근무
- 1978년11월~85년 7월 : 한국전력기술(주) 근무
- 1989년 1월~92년 9월 : 미국 노스캐롤라이나주립대 부설 CCSP(Center For Comm. & Signal Processing) 연구원
- 1992년10월~94년 2월 : 한국전자통신연구소 선임연구원
- 1994년 3월~현재 : 대구효성가톨릭대학교 공과대학 전자정보공학부 조교수, 학부장

박 정 숙

- 1994년 2월 : 효성여자대학교 전자계산학과 학사
- 1996년 8월 : 대구효성가톨릭대학교 전산통계학과 석사
- 1996년 9월~현재 : 대구효성가톨릭대학교 전산통계학과 박사과정 재학중

박 영 덕

- 1984년 : 성균관대학교 전자공학과 학사
- 1987년 : 성균관대학교 전자공학과 석사
- 1990년 : 성균관대학교 전자공학과 박사
- 1983년~85년 : 삼성전자 연구원
- 1994년~95년 : 일본 ATR 연구소 초빙연구원
- 1990년~현재 : 한국전자통신연구원 책임연구원 멀티미디어통신연구실장