접근 요청 빈도에 기반한 멀티미디어 뉴스 데이터의 선별적 버퍼 캐쉬 관리 전략

박 용 운† · 서 원 일†† · 정 기 동†††

요 약

대규모의 실시간 주문형 뉴스 제공 시스템(Real Time News On Demand)에서는 다수의 사용자들이 디스크에 저장된 뉴스 데이터를 실시간으로 동시에 접근하며 최대로 수용할 수 있는 사용자 수는 총 디스크 대역폭의 제한을 받는다. 본 연구에서는 이러한 디스크 대역폭의 한계를 극복하기 위하여 디스크 비용의 일부로 버퍼 캐쉬를 구성하여 실시간 뉴스 데이터에 적합하도록 버퍼를 블록 단위가 아닌 오브젝트 단위로 할당하는 버퍼 캐쉬 정책을 사용하고, 캐싱 대상 뉴스 데이터를 현재의 디스크 대역폭의 사용 정도와 해당 뉴스 데이터의 평균 요청 간격을 고려하여 선별함으로써 재 접근 가능성이 낮은 데이터의 경우 캐싱 대상에서 제외시켜 불 필요한 버퍼의 재 할당에 의한 메모리 오버헤드를 방지하는 실시간 뉴스 데이터에 적합한 캐싱 방법을 제안한다. 이렇게 함으로써 접근 빈도수가 높은 데이터의 경우 디스크의 접근 없이도 데이터의 획득이 가능하게 되어 디스크만으로 저장 시스템을 구성할 때와 비교하여 저 비용으로 저장 시스템을 구성할 수 있다. 본 논문에서 제안한 알고리즘의 성능을 시뮬레이션을 통하여 평가한 결과, 본 논문에서 제안한 캐싱 방법으로 뉴스 데이터에 대한 사용자의 요청을 처리했을 경우, 디스크만으로 저장 서버를 구성하였을 경우보다 30% 이상의 사용자를 지원할 수 있었다.

Access Frequency Based Selective Buffer Cache Management Strategy For Multimedia News Data

Yong-Woon Park† · Won-Il Seo†† · Ki-Dong Chung†††

ABSTRACT

In this paper, we present a new buffer pool management scheme designed for video type news objects to build a cost-effective News On Demand storage server for serving users requests beyond the limitation of disk bandwidth. In a News On Demand Server where many of users requests for video type news objects have to be serviced keeping their playback deadline, the maximum numbers of concurrent users are limited by the maximum disk bandwidth the server provides. With our proposed buffer cache management scheme, a requested data is checked to see whether or not it is worthy of caching by checking its average arrival interval and current disk traffic density. Subsequently, only granted news objects are permitted to get into the buffer pool, where buffer allocation is made not on the block basis but on the object basis. We evaluated the performance of our proposed caching algorithm through simulation. As a result of the simulation, we show that by using this caching scheme to support users requests for real time news data, compared with serving those requests only by disks, 30 % of extra requests are served without additional cost increase.

† 준 회 원 : 부산대학교 대학원 전자계산학과
†† 정 회 원 : 지산대학 전산정보처리과 교수
††† 종신회원 : 부산대학교 전자계산학과 교수
논문접수 : 1999년 2월 26일, 심사완료 : 1999년 8월 23일

1. 서 론

통신 기술과 컴퓨팅 기술의 발전은 연속형 미디어 데이터를 이용한 멀티미디어 응용들을 가능하게 하였으며 이러한 멀티미디어 응용 가운데에서 NOD(News On Demand)는 가장 대중적이며 광범위한 사용자 층을 대상으로 한 응용의 하나이다. 그러나 이러한 멀티미디어 응용에 대한 연구는 대부분 VOD(Video On Demand)에 초점을 맞추어 이루어지고 있다. 그러나 NOD 데이터의 경우 VOD 데이터와 상이한 점이 존재하며 그 중 본 연구와 관련된 차이점을 들면 다음과 같다. 첫째, 데이터의 길이이다. 통상적인 비디오 프로그램의 재생시간은 80분 전후인 반면 뉴스 데이터는 3분을 초과하는 경우는 드물다. 만약 MPEG II로 압축된 데이터가 정상적인 재생을 위해서는 8 Mbps 정도의 전송 대역폭이 필요하다고 하면 80분의 비디오 프로그램이 약 5 기가 바이트 정도의 저장 공간이 필요한 반면 뉴스 데이터의 경우 180 메가 바이트 정도의 저장 공간만이 필요하다[1]. 둘째, 시간에 따른 사용자 수의 변화가 비디오 데이터의 경우보다는 뉴스 데이터가 심하다. 뉴스 데이터의 경우, 전체 참조 횟수 중의 많은 부분이 하루 24시간 중에 5시간 이내의 피크타임 내에서 이루어진다[2]. 이러한 뉴스 데이터 크기와 요청의 시간적 지역성을 이용하여 접근 확률이 높은 뉴스 데이터를 버퍼 캐쉬를 통하여 지원한다면 대역폭 확보를 위하여 필요한 디스크의 수를 상당 부분 줄일 수 있다. 그러나 NOD 데이터의 경우 재래식 데이터나 VOD 데이터와는 특성이 상이하므로 재래식 데이터나 VOD 데이터의 캐쉬 기법을 그대로 적용하기 곤란하다. 그러므로 NOD 데이터의 적절한 캐쉬을 위해서는 NOD 데이터의 접근 유형을 분석하여 캐쉬에 적용하여야만 효과적으로 NOD 저장 서버를 운영할 수 있다. 본 연구에서는 이러한 뉴스 데이터의 특성 및 접근 패턴의 특성을 고려하여 캐쉬될 데이터를 전체 디스크 대역폭 사용 정도와 요청된 데이터들의 접근 패턴을 이용하여 선별하는 실시간 타입 뉴스 비디오 데이터의 캐쉬 기법을 제안한다.

본 논문의 구성은 다음과 같다. 제2장에서는 멀티미디어 데이터의 입출력 향상을 위한 연구들과 멀티미디어 데이터의 캐쉬과 관련된 연구들을 소개한다. 제3장에서는 본 논문에서 제안하고 있는 NOD 데이터 캐쉬 정책 및 알고리즘을 소개한다. 제4장에서는 이 논문에서 제안하는 캐쉬 정책을 시뮬레이션을 통하여 분석하고 마지막 장인 5장에서는 결론을 내린다.

2. 관련 연구 및 뉴스 데이터의 접근 유형

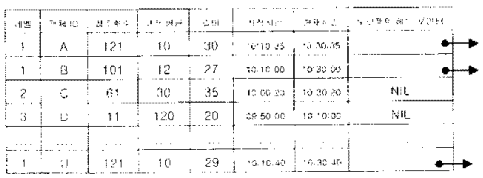
멀티미디어 데이터의 버퍼링에 관한 연구의 대부분은 데이터의 공유를 통한 재사용 측면보다는 효과적인 디스크 스케줄링을 지원하기 위한 측면으로 초점이 맞추어져 있다[3, 4, 5, 6, 7, 8, 9, 10]. 본 연구에서 제안하고자 하는 데이터의 공유를 통한 데이터 재 사용도의 향상에 관한 연구는 그다지 많지 않으며 대표적인 연구로는 다음과 같은 사례가 있다. [11, 12]에서는 동일 비디오 데이터의 요청에 대한 선 후행 스트림간의 구간을 캐쉬함으로써 후행 스트림은 디스크 읽기를 하지 않고 선행 스트림이 버퍼에 읽어 들인 데이터를 서비스를 받음으로써 디스크 읽기를 피할 수 있는 구간 캐쉬(interval)을 제안하고 있다. 이 방법은 요청되는 비디오 프로그램이 많고 특정 프로그램을 요청하는 사용자 수가 많아질 때에는 버퍼 풀의 전체가 하나의 비디오 오브젝트에 의해 점유될 수 있는 위험이 있으며 캐쉬되는 구간을 구간 길이의 함수만으로 선택함으로써 짧은 시간 동안의 특정 비디오에 대한 변화만이 적용되나 시간에 따른 비디오의 점진적인 변화를 반영하지 못한다. 이상의 연구에서 보듯이 기존의 캐쉬 연구 중 뉴스 비디오 오브젝트 즉, 데이터의 길이가 비디오 데이터에 비해 짧으며 짧은 시간 동안에 특정 데이터에 대한 중복된 요청이 다수 발생하는 특징을 가진 실시간 뉴스 데이터의 특성을 고려한 캐쉬에 관한 연구는 없다. 따라서 실시간 뉴스 데이터의 캐쉬 전략은 뉴스 데이터의 성격을 고려하여 그러한 성격에 맞게 재정립되어야 한다. [2]에 따르면 뉴스 데이터에 대한 사용자의 요청 회수는 시간 대별로 차이가 많으며 상대적으로 짧은 기간 동안에 사용자의 요청이 집중적으로 몰린다. 또한 전체 기사 중 10%의 기사에 전체 빈도수의 80% 이상의 편중되어 있었다. 이는 모든 뉴스 데이터를 캐쉬 대상으로 한다면 기사 요청의 편중되는 형태와 데이터의 대용량성, 그리고 순차적 접근성을 고려할 때, 불필요한 버퍼 재할당으로 인한 시스템 부하가 증가할 가능성이 높다는 것을 의미한다. 그러므로 이러한 데이터의 경우 재 참조의 가능성이 낮은 데이터에 대해서는 캐쉬 대상에서 제외시키는 정책을 사용하는 것이 유리하다.

3. 뉴스 데이터를 위한 선별적 캐슁 메커니즘

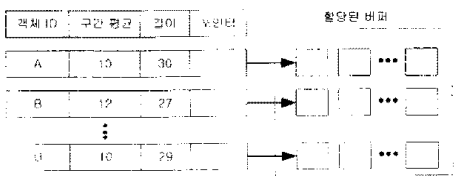
3.1 버퍼 할당을 위한 메타데이터 구조

본 논문에서 제안하는 캐슁은 버퍼 할당의 단위가 블록이 아닌 오브젝트 단위이다. 재래식 캐슁의 경우에는 대부분의 트랜잭션의 처리 단위가 블록인 반면 뉴스 데이터의 경우 처리 단위가 블록이 아닌 오브젝트이다. 따라서 블록 단위로 캐쉬를 할당받게 되면 오브젝트 간의 캐쉬 확보를 위한 경쟁으로 인한 오버헤드가 발생한다. 또한 멀티미디어 데이터의 특성 상 캐쉬를 블록 단위로 할당 받을 경우 캐쉬를 마감시간(deadline)내에 할당받지 못하였을 때 전송 지연이 발생할 수도 있다. 그러므로 실시간 멀티미디어 데이터의 경우에는 버퍼 할당을 오브젝트 단위로 하여야 한다. (그림 1)에 오브젝트 단위로 버퍼를 할당하기 위하여 필요한 자료구조인 오브젝트 리스트(a)와 오브젝트 헤드(b)가 나타나있다. 오브젝트 라스트는 디스크에 존재하는 뉴스 데이터에 대한 메타 데이터를 저장하고 있으며 해당 뉴스 데이터에 대한 사용자의 요청이 있을 때 마다 데이터가 갱신된다. 객체 ID는 뉴스 데이터의 ID를, 레벨은 해당 데이터의 등급을 의미하며 캐슁의 선별 기준이 된다. 참조 횟수는 해당 오브젝트에 대한 누적 참조 횟수이며, 시작시간은 해당 오브젝트에 대한 최초의 요청시간을, 현재 시간은 해당 오브젝트에 대한 가장 마지막 요청 시간을 나타낸다. 구간 평균은 현재 시간을 기준으로 해당 오브젝트에 대한 요청들의 평균 참조 간격으로써 재배치될 오브젝트를 선정하는 기준이 된다. 길이는 해당 오브젝트의 길이를 나타낸다. 오브젝트 헤드 포인터는 오브젝트 헤드를 가리키는 포인터이다. 오브젝트 헤드는 현재 버퍼 풀에 캐슁되어 있는 오브젝트들만의 정보를 가지고 있으며 오브젝트 리스트에 존재하는 일부 메타 정보와 해당 오브젝트가 캐슁된 버퍼 풀의 시작 포인터를 가지고 있다.

(a) 오브젝트 리스트



(b) 오브젝트 헤드



(그림 1) 캐슁을 위한 데이터 구조

3.2 캐슁 알고리즘

본 논문에서 제안하는 캐슁 전략은 모든 데이터에 대해서 캐슁을 적용하지 않고 주어진 4개의 한계 값과 현재의 디스크 대역폭의 활용도와 요청된 데이터의 평균 요청 구간의 함수에 의해 그 결과 값이 캐슁을 만족하는 데이터에 대해서만 캐슁을 허락한다. 이러한 4개의 한계 값들은 각각 일, 이차 요청 한계 구간 (first and second request distance threshold) 및 일, 이차 디스크 대역폭 한계(first and second disk bandwidth threshold)로 부른다. 일,이차 데이터 한계 구간은 각각의 데이터에 대한 요청간의 간격을 말하며 각각, dt_α, dt_β로 나타내고 다음과 같다.

$$0 \le dt_\alpha,\ dt_\beta,\ dt_\alpha \le dt_\beta \tag{1}$$

일,이차 디스크 대역폭 한계치는 각각, bd_α, bd_β로 나타내고 임의로 주어지며 다음과 같다

$$0 \le bd_\alpha,\ bd_\beta \le 1,\ bd_\alpha \le bd_\beta \tag{2}$$

뉴스 데이터의 경우 특정 시간대에 집중적으로 접근되는 경향이 있으므로 데이터 요청의 빈도에 관계없이 모든 데이터를 캐슁 대상으로 하면 요청이 집중할 때에는 버퍼 재 배치가 빈번하게 발생하여 버퍼 캐쉬가 효율적으로 운영되지 못할 가능성이 높다. 따라서 캐슁될 오브젝트를 선별적으로 둠으로써 버퍼 재 할당에 따른 오버 헤더를 줄인다. 이를 위하여 먼저 뉴스 데이터 i의 요청의 평균 도착률 $AVG(dt)_i$를 다음과 같이 정한다.

$$AVG(dt)_i = 1/\left(\sum_{j=2}^{n} (T_{\langle i,j\rangle} - T_{\langle i,j-1\rangle}) / n-1 \right) \tag{3}$$

n : 뉴스 데이터의 i의 총 요청 수

$T_{\langle i,j\rangle}$: 뉴스 데이터 i의 j번째 요청시간

오브젝트 i의 데이터 레벨 $L(O)_i$ 는 다음과 같이 주어진다.

$$L(O)_i = \begin{cases} 1, & AVG(dt)_i > dt_\alpha \\ 2, & dt_\alpha \le AVG(dt)_i < dt_\beta \\ 3, & AVG(dt)_i < dt_\alpha \end{cases} \quad (4)$$

즉, 뉴스 오브젝트 i의 $AVG(dt)_i$ 가 dt_α보다 작을 때에는 3(third level), dt_α와 dt_β 사이일 경우에는 2(second level), dt_β 이상의 경우에는 1(first level)이 된다.

그 다음에는 현재 모든 사용자에 의해 사용중인 총 디스크 대역폭에 따라 일 이차 디스크 대역폭 한계치 bd_α, bd_β를 기준으로 현재 디스크 대역폭의 활용도를 3등급(level)즉, IDLE, MODERATE, INTENSIVE등으로 나누어 단계에 따라 캐싱할 데이터를 선별하여 정한다. C를 현재 동시에 지원 중인 총 디스크 스트림 수라고 하면 현재의 디스크 입출력 BD_{ratio}은 사용중인 디스크 대역폭/디스크 총 디스크 대역폭의 식으로 나타낼 수 있으며 다음과 같다.

$$BD_{ratio} = \frac{\sum_{i=1}^{C} PL_i}{TR_{disk} * TOT_{disk}}, \quad 0 \le BD_{ratio} \le 1 \quad (5)$$

TR_{disk} : 디스크의 평균 전송률/초
PL_i : 디스크 스트림 i의 재생률/초
TOT_{disk} : 총 디스크 수

현재 서버의 디스크 대역폭 사용도 T_{mode} 는 다음과 같이 주어진다.

$$T_{mode} = \begin{cases} INTENSIVE, & BD_{ratio} \ge bd_\alpha \\ MODERATE, & bd_\alpha \le BD_{ratio} < bd_\beta \\ IDLE, & BD_{ratio} < bd_\beta \end{cases} \quad (6)$$

즉, 현재의 총 디스크 스트림에 의해 점유된 총 디스크 대역폭과 전체 디스크 대역폭과의 비율(BD_{ratio})를 구하여 bd_α보다 작으면 T_{mode} 가 IDLE, bd_α보다 크고 bd_β보다 작으면 MODERATE, bd_β보다 크면 INTENSIVE가 된다. 이를 기준으로 캐싱될 데이터는 현재 서버의 디스크 대역폭 사용도 T_{mode} 와 요청된 오브젝트 i의 데이터 레벨 $L(O)_i$ 에 따라 차등적으로 적용된다. T_{mode} 가 IDLE일 때는 모든 뉴스 데이터가 캐싱 대상으로 간주되며 MODERATE 때는 데이터 레벨 $L(O)$ 가 3인 데이터를 제외한 1,2 인 데이터에 대해서만 캐싱을 적용한다. 마찬가지로 T_{mode}가 INTENSIVE일 때는 $L(O)$가 1인 뉴스 데이터에 대해서만 캐싱을 허용한다. 개략적인 알고리즘은 그림 2에 나타나있다.

4. 시뮬레이션

4.1 한계치 설정

일반적으로 데이터를 저장할 디스크의 수를 정하는 방법은 다음과 같은 두 가지 방법 중 하나에 의해 결정된다[11]. 첫째는 암 바운드 방식으로써 단위 시간당 읽혀지는 데이터를 실시간으로 지원할 수 있는 대역폭에 의해 필요한 디스크의 수를 결정하는 방법이고 두 번째는 공간 바운드로써 데이터의 저장에 필요한 디스크의 공간에 의해 필요한 디스크의 수를 결정하는 방법이다. NOD 서버에 M개의 뉴스 데이터가 저장되어 있다고 하고 동시 예상 최대 사용자의 수가 U라고 하자. U명의 동시 사용자를 지원하기 위하여 암바운드 방식으로 구한 디스크의 수를 $DISK^{U}_{arm_bound}$로 표시하고 공간 바운드 방식으로 구한 디스크의 수를 $DISK^{M}_{space_bound}$라고 하자. 시스템 구성에 필요한 디스크의 수는 두 방식 중에 큰 것을 선택하여야 한다.

```
Process Object Handler(user_request)

BEGIN {
   Receive a users request
   Increase the reference count of the news data in the
Object list the use want to access and Set its data level
(among 1,2,3);
   Update the average request interval of the news data
   Check out which traffic mode (among IDLE, MODERATE,
INTENSIVE) the current I/O status is;

   If IDLE mode then go to Buffer_pool for data access;
   Else if (MODERATE mode && the datas level >= 2)
            then go to Buffer_pool  for data  access;
   ELSE if (INTENSIVE && data level = 3)
            then go to Buffer_pool for data access;
   ELSE disk read;

}END

Function Buffer_pool(news_ID) {
   Do while until the object with news_ID found {
     If the news object found in the buffer pool {
        Then update its meta data in the Object Header and
        read buffered news data until end of object
                                        and then Return; }
   } End of do_while

   If (available free buffer in the Buffer pool) then
     Create new Object Head and load data from disk(s)
                                        until end of object;
```

```
    else {
       Sort the Existing Object Header(s) in the Buffer Pool
                    by average interval in descending order;
       Do while until buffer space to load
            the requested news_IDs data from disk(s) is reserved
       {
            If  ((news_IDs average interval <= existing Objects
average interval) &&  (The reserved space is large enough
to hold news_IDs data)) then
              Load and read the requested news_IDs data
                                  from disk(s) and Return;
       } End of do_while
    } End_if

    Go to disk_read and return;

} End Function Buffer_pool
```

(그림 2) 캐싱 알고리즘

$$DISK_{total} = MAX(DISK^{U}_{arm\_bound}, DISK^{M}_{space\_bound}) \quad (7)$$

서버 구성에 필요한 디스크의 수 $DISK_{total}$는 대규모 실시간 뉴스 시스템의 경우 통상적으로 $DISK^{U}_{arm_bound} \gg DISK^{M}_{space_bound}$가 되며 $DISK^{U}_{arm_bound}$는 원본 데이터 저장에 필요한 디스크 $DISK^{M}_{space_bound}$와 $DISK^{M}_{space_bound}$가 제공하는 디스크 대역폭 만으로는 지원되지 못하는 사용자들의 요청을 지원하기 위해 필요한 디스크 대역폭 제공을 위한 추가적인 디스크의 수인 XR_{disk} 의 합으로 표현되며 이때 XR_{disk} 은 $DISK^{M}_{space\_bound}$ 의 복제(replication)들이 저장된다.

$$DISK_{total} = DISK^{U}_{arm\_bound} = DISK^{M}_{space\_bound} + XR_{disk} \quad (8)$$

식 (1)의 일,이차 데이터 한계 구간, dt_α, dt_β는 다음과 같이 정하였다.

$$dt_\alpha = 1/\left(\sum_{i=1}^{M} Leng(O)_i/M\right),\ dt_\beta = 1/(C_{stream}/C_{buffer}) \quad (9)$$

$Leng(O)_i$: 뉴스 오브젝트 i의 크기(단위 MB)

M : 총 뉴스 데이터 수

C_{stream} : 디스크 스트림당 비용

C_{buffer} : MB당 메모리 가격

예를 들면 디스크 대당 가격이 30만원이고 평균 재생률을 1MB/초라 하고 모든 디스크 스트림들의 평균 재생률을 8 mbps라고 하자. 이때의 동시 최대 지원 스트림의 수는 10이 되며 스트림 당 비용은 3만원이 된다. 버퍼의 가격(C_{buffer})이 MB 당 2천원으로 두면 하나의 디스크 스트림 비용에 해당하는 버퍼 메모리는 15MB가 되며 dt_β는 1/15 가 된다. 즉, 뉴스 오브젝트 i의 $AVG(dt)_i$가 dt_β보다 크다면 스트림 당 비용이 디스크를 통하여 서비스 받을 때 보다 버퍼 캐쉬를 통하는 것이 비용이 적게 든다. dt_α는 1/뉴스 오브젝트의 평균 길이로 하였다. 일,이차 디스크 대역폭 한계치는 bd_α, bd_β는 각각 다음과 같이 구하였다.

$$bd_\alpha = \frac{DISK_{space\_bound}}{DISK_{total}},\ bd_\beta = \frac{1}{DISK_{space\_bound}} \quad (10)$$

bd_α는 사용자의 데이터 입출력 요청을 원래의 데이터를 저장하는데 사용된 디스크의 대역폭만으로 지원 가능한 시점으로 두었다. 특정 데이터 요청에 대한 입출력이 승인되지 못하는 시점은 해당 데이터가 저장 되어있는 디스크 모두의 대역폭이 포화 상태에 이르렀을 때를 의미한다. 총 뉴스 데이터는 $DISK^{M}_{space_bound}$의 디스크에 저장되며 이때 특정 데이터는 $DISK^{M}_{space\_bound}$ 중 임의의 한 디스크에 존재하게 되어 식 10의 bd_β과 같이 표현된다. 이 값은 임의의 데이터에 대한 요청이 지원되지 못할 경우가 발생하였을 때의 전체 누적 디스크 대역폭 중 현재 사용중인 디스크 대역폭의 비율의 최저값이며 사용자의 증가에 따라 디스크가 증가하더라도 동시에 선형적으로 증가하므로 고정적이다.

4.2 시뮬레이션 파라메터

본 연구에서 제안하는 알고리즘의 평가를 위하여 유닉스 시스템 환경에서 시뮬레이션을 하였다. 멀티미디어 데이터의 접근 패턴은 *Zipf* 분포를 이용하는 것이 일반적이나 본 연구에서는 실제 사용자들의 접근 패턴을 반영하기 위해 [2]에서 조사한 데이터를 사용하였으며 각 사용자가 요구한 데이터의 전송율은 8 Mbps로 가정하였다. 시뮬레이션에 사용된 파라메터들이 <표 1>에 나타나있다.

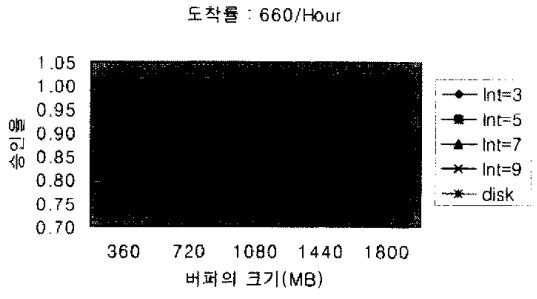
〈표 1〉 시뮬레이션 파라메터

파 라 메 터	값
디 스 크 저 장 공 간	3.2 GB
디 스 크 평 균 대 역 폭	10 MBytes/sec
뉴 스 데 이 터 길 이	30-180 Mbytes
뉴 스 데 이 터 수	60개
도 착 률 / 시 간 당	660~3300

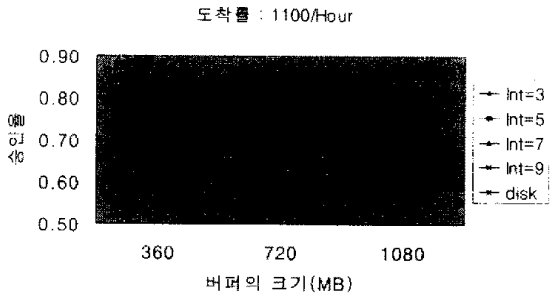
[2]의 조사에 의하면 하루에 생성되는 데이터의 개수는 총 600 개 정도이고 하루 평균 10,000 건 정도 접근되며 이 중에서도 사용자가 10번 이상 언급한 경우는 60개 정도였다. 따라서 이를 근거로 총 데이터 수를 60개로 정하였다. 데이터의 길이는 본 실험을 위하여 국내외의 뉴스 길이를 측정한 결과 대부분의 경우 2분 내외의 길이라는 사실에 근거하여 60개의 데이터는 각각 30 MB의 크기가 5개, 60 MB의 크기가 10개 , 90 MB의 크기가 20개 , 120 MB의 크기가 15개, 150 MB의 크기가 10개로 정하였다. 이를 근거로 계산한 dt_α, dt_β의 값은 각각 0.01과 0.05였다. bd_α는 모집단의 크기에 따라 유동적이며 bd_β는 본 논문에서 디스크 사양과 데이터의 수와 크기를 기준으로 계산한 결과 0.5가 되었다. 본 실험에서는 식 8로 구한 디스크로 지원할 수 있는 스트림의 수보다 도착률을 10% 증가 시켜서 각각 시간당 도착률을 660, 1100, 1700, 3300으로 두고 시뮬레이션 하였다.

4.3 버퍼 캐쉬 크기에 따른 사용자의 승인율 변화

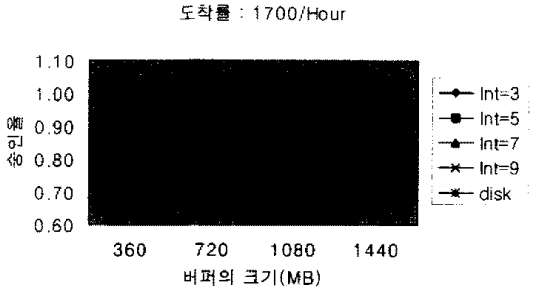
첫 실험은 시간당 사용자의 도착률을 기준으로 버퍼 캐쉬의 크기의 변화가 사용자의 뉴스 데이터에 대한 요청 건수의 승인율에 어떠한 변화를 주는가를 측정하였다. 여기서의 승인율이란 데이터를 읽기 위하여 디스크 접근을 한 전체 요청 건수 중 해당 디스크의 대역폭이 존재하지 않아서 요청이 기각되는 경우를 제외한 승인된 비율을 말한다. (그림 3)부터 (그림 6)에 도착률과 dt_β의 크기에 따른 승인율을 그래프로 나타내었다. dt_β는 그림에서 int로 표시되어있으며 식 (10)에 의해 구해진 값 0.05(int = 5)를 기준으로 각각 0.03(int = 3), 0.07(int = 7), 0.09(int = 9)로 변화시키면서 측정하였다. *disk*는 버퍼를 통한 데이터의 공유를 하지 않고 디스크만을 통하여 데이터를 접근하였을 경우를 나타낸다. 대부분의 경우 버퍼 캐쉬를 사용하여 사용자들의 요청을 처리한 경우에 디스크만으로 지원한 경우보다 성능이 우수하게 나왔다. 단, 모집단이 작은 경우 즉, 660의 경우에는 버퍼 캐쉬의 크기가 1080MB의 경우 디스크 비용의 일부를 버퍼 풀로 대치한다 하더라도 별 효과가 없음을 알 수 있다. 이는 모집단이 작음으로 인해서 필요한 디스크의 수가 상대적으로 적은 반면 디스크 비용의 일부로 버퍼캐쉬를 구성하였으나 사용자의 뉴스 오브젝트에 대한 요청 빈도수가 적음으로 인하여 데이터가 재 사용될 확률이 떨어지기 때문이다. 시간 당 도착률이 1100명에 이르면서 버퍼 풀을 통한 데이터의 캐싱이 효과를 발휘함을 알 수 있으며 도착률이 3300에 이르면서 버퍼 캐쉬의 크기가 720MB 이상일 때 대부분의 경우 승인률이 100%에 이르고 있



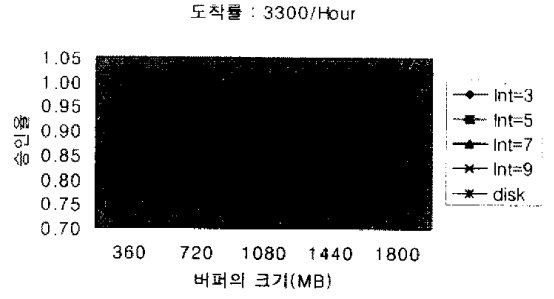
(그림 3) 평균 도착률에 따른 승인율의 변화(1)



(그림 4) 평균 도착률에 따른 승인율의 변화(2)



(그림 5) 평균 도착률에따른 승인율의 변화(3)

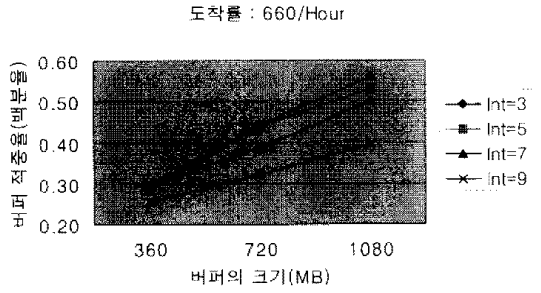


(그림 6) 평균 도착률에 따른 승인률의 변화(4)

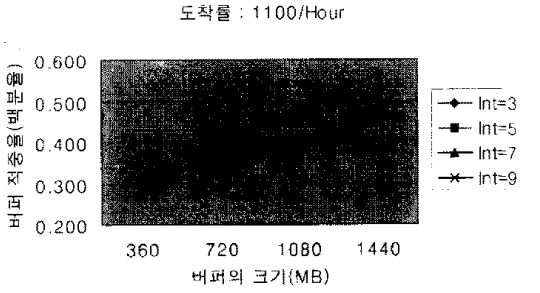
어 디스크를 사용하여 지원하는 경우 보다 20%이상의 요청을 지원할 수 있음을 알 수 있다. 또한 그림에는 나타나있지 않지만 도착률이 3300인 경우 20%이상의 사용자를 지원하고도 10%정도의 추가적인 사용자를 지원할 수 있는 디스크의 대역폭이 존재하였다. 이는 만약 추가적인 요청이 있더라도 승인될 수 있음을 의미하며 실제 30% 이상의 성능 향상 효과가 있음을 의미한다. 도착률이 1100명을 초과하는 경우에 버퍼 풀의 크기에 따라 승인율의 차이가 나타나며 전체적으로 볼 때 모집단이 660인 경우를 제외하고는 버퍼 풀의 크기가 720MB 1080MB 사이에서 최적화된 결과를 얻을 수 있다는 것을 알 수 있다. 이는 본 논문에서 제안하는 알고리즘이 단순히 버퍼 풀만을 고려하여 버퍼 풀의 성능을 측정하는 것이 아니라 디스크에 사용된 비용의 일부를 버퍼 풀의 비용으로 대치하여 버퍼 풀을 운영하기 때문이다. 따라서 버퍼 풀이 커진다는 것은 상대적으로 디스크의 수가 감소된다는 의미이므로 단순히 버퍼 풀을 크게한다고 해서 승인율이 증가하지는 않는 것이다. 승인율에 dt_β의 크기가 미치는 영향은 식 (10)에서 계산된 int = 5보다 int = 3의 경우가 약간 좋은 것으로 나타났으나 5보다 큰 경우에는 떨어지는 것으로 나타났다. dt_β가 커진다는 의미는 버퍼 대상으로 선택되는 데이터의 평균 요청 간격이 짧아지게 된다는 의미가 되어 캐싱 대상이 되는 뉴스 오브젝트가 dt_β가 작을 때보다 줄어들게 되어 버퍼 캐쉬의 사용률을 떨어뜨리는 반면 디스크의 사용률을 증가시키는 결과가 초래되어 승인률이 떨어지는 것이다.

4.4 버퍼 캐쉬의 크기에 따른 버퍼 적중률의 변화

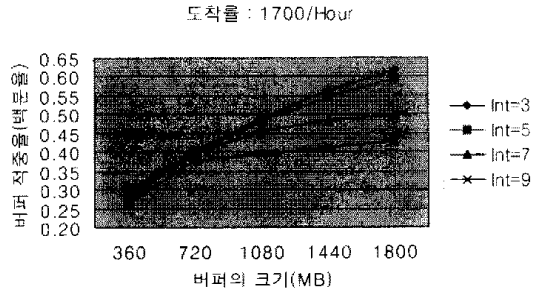
(그림 7)부터 (10)까지는 시간 당 사용자의 도착률을 기준으로 전체 요청 중 버퍼 캐쉬에 존재하는 데이터로 서비스한 요청의 비율을 버퍼 적중률로 표시하고 dt_β의 변화에 따른 버퍼 적중률의 변화를 측정하였다. 이 횟수는 재래식 버퍼 운영 정책에서의 버퍼의 히트율(hit ratio)에 해당되며 비율이 낮을 경우에는 버퍼 캐쉬의 효율이 떨어짐을 의미한다. 그림에서 보듯이 버퍼 적중률은 dt_β의 값이 커질 수록 감소하는 경향이 있다. 이는 본 논문에서 제안하는 버퍼 캐쉬 알고리즘이 오브젝트에 대한 요청간의 구간(interval)에 기반하고 있기 때문에 dt_β가 작을수록 버퍼 캐쉬를 이용할 수 있는 오브젝트의 수가 증가하여 사용자가 요구하는 데이터가 버퍼 캐쉬에 존재할 확률이 높아진다. 따라서 버퍼 적중률은 모집단의 크기에 관계없이 버퍼 캐쉬의 크기가 클수록 그리고 dt_β가 작을수록 증가한다. 본 논문의 경우를 예를 든다면 dt_β=0.05일 때(int = 5)



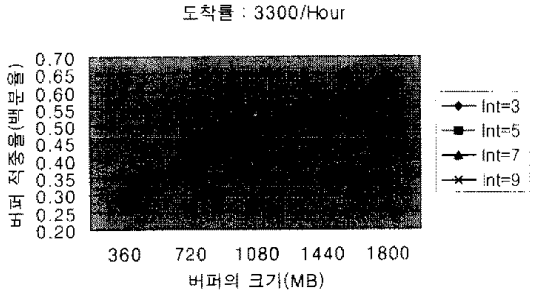
(그림 7) 평균 도착률에 따른 버퍼율 변화(1)



(그림 8) 평균 도착률에 따른 버퍼율 변화(2)



(그림 9) 평균 도착률에 따른 버퍼율 변화(3)



(그림 10) 평균 도착률에 따른 버퍼율 변화(4)

와 0.07(line 7)일 때를 비교해 보면 5일 경우가 버퍼 높다. 이는 dt_{β}가 커질수록 보다 빈번하게 요청된 소수의 데이터 만이 버퍼 캐쉬에 존재하게 되어 버퍼 캐쉬로 지원 가능한 뉴스 데이터의 수가 줄어들게 되는 반면 dt_{β}가 작은 경우에는 버퍼 캐쉬의 공간이 보다 많은 데이터 들에 의해 공유되기 때문에 보다 많은 오브젝트들이 버퍼캐쉬에 존재하게 되어 버퍼 캐쉬를 통한 데이터의 지원이 증가한다.

5. 결 론

실시간 뉴스 데이터의 경우 지원할 수 있는 사용자의 수가 디스크의 총 대역 폭의 영향을 받게 되며 사용자의 수가 증가하면 할수록 증가하는 사용자를 지원하기 위한 디스크의 수 또한 선형적으로 증가하게 된다. 그러나 뉴스 데이터의 경우 짧은 시간 동안 집중적으로 사용자의 요구가 발생하므로 적절한 캐싱 알고리즘이 적용된다면 버퍼 풀을 이용하여 재 사용도가 높은 데이터를 버퍼 풀에 수용함으로써 디스크만으로 사용자를 지원할 때보다 경제적이고 효율적인 저장 장치를 구성할 수 있다. 이를 위하여 본 논문에서는 뉴스 데이터를 접근 빈도에 따라서 여러 등급으로 분리한 다음 전체 디스크 교통량에 따라 버퍼 풀을 탄력적으로 운영하여 서버의 디스크 대역폭 사용 정도와 데이터의 요청 정도에 따라 버퍼 풀을 접근할 수 있는 오브젝트를 제한하는 오브젝트 단위의 선별적 다단계 캐싱 알고리즘을 제안하였다. 이 캐싱 방법을 사용함으로써 불 필요한 버퍼 재할당을 방지하고 데이터의 재 사용도를 높여 디스크 만을 사용하여 사용자의 요청을 지원할 때 보다 최고 30%이상의 추가적인 요청을 지원할 수 있었다.

참 고 문 헌

[1] 이주경, 박용운, 김영주, 정기동, "NOD 데이터를 위한 데이터 배치 방법", 한국 정보 과학회 추계 학술 발표회, 제24권 2호, pp.55-58, 1997.

[2] 백건효, 박용운, 서원일, 정기동, "뉴스 비디오 데이터의 효율적인 운영 정책 수립을 위한 사용자 접근 패턴 분석", 한국 정보 과학회 춘계학술 발표회, 제25권 제1호, pp.485-487, 1998.

[3] A. L. Narashima Reddy, Jim Wylle, "Disk Scheduling in a multimedia I/O system," ACM multimedia 93 proceedings, pp.225-233, 1993.

[4] Mon-Song Chen, Dillip D. Kaundlur, Philp S Yu, "Optimization of the Grouped Sweeping Scheduling(GSS) with Heterogeneous Multimedia Streams," ACM multimedia 93 proceedings, pp.235-242, 1993.

[5] Raymond T. Ng, Jinhai Yang, "An analysis of Buffer Sharing and Prefetching Techniques for Multimedia Systems," Tech. Report, Dept. of CS, Univ. of British Columbia, Ca, 1995.

[6] Mohan Kamath, K. Ramamrithan, D. Towsley, "Continuous Media Sharing in Multimedia Database Systems," Tech. Rept of CS, U. of Massachusetts

[7] Asit Dan, Dinka Sitaram, "Multimedia Caching Strategies for Heterogeneous Application and Server Environment," IBM Tech. Rept, 1996

[8] Steven W. Carter, Darrell D. E. Long, "Stream Tapping : a System for Improving Efficiency on aVideo-On-Demand Server," Tech-rept UCSC-CRL-97-11, U of California, Santa Cruz

[9] Edward Chang, H. G. Molina, "Cost-Based Media Server Design," Proceedings of the 8th Research Issues in Data Engineering, Feb., 1998.

[10] Kun Lung Wu, Philip S Yu, "Consumption based Buffer management for Maximizing System Throughputs of News On Demand Multimedia system," IBM Technical Report, 1995.

[11] Renu Tewari, Asit Dan, etal "Buffering and Caching in Large-Scale Video Servers," Proceedings of COMPCON 1995, 1995

[12] Weifeng Shi etal, "Trading memory for disk bandwidth in Video on Demand," Proceedings of 13th ACM Symposium on Applied Computing, Feb., 1998.

박 용 운

e-mail : ywpark@melon.cs.pusan.ac.kr

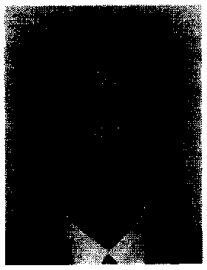
1988년 부산대학교 계산통계학과 졸업(학사)

1988~1995 ㈜LG-EDS 근무

1997년 부산대학교 전자계산학과 졸업(석사)

1997년~현재 부산대학교 전자계산학과 박사과정

관심분야 : 병렬 파일 시스템, 멀티미디어, 캐싱



서 원 일

e-mail : wiseo@jisan.ac.kr

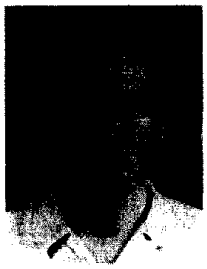
1988년 서강대학교 전자계산학과 졸업(학사)

1993년 서강대학교 전자계산학과 졸업(석사)

1995년~1997년 부산대학교 전자계산학과 박사 수료

1993년~현재 지산 대학 전산 정보처리과 조교수

관심분야 : 멀티미디어, 병렬파일시스템, 캐싱 등



정 기 동

e-mail : kdchung@ hyowon.cc.pusan.ac.kr

1973년 서울대학교 졸업(학사)

1975년 서울대학교 대학원 졸업(석사)

1986년 서울대학교 대학원 계산통계 학과 졸업(박사)

1990년~1991년 MIT, South Carolina 대학 교환교수

1978년~현재 부산대학교 전자계산학과 교수

1993년~현재 부산대학교 부설 컴퓨터 및 정보통신 연구소 운영위원

관심분야 : 병렬처리, 멀티미디어