VOD 시스템에서 효율적인 영화할당을 위한 버퍼관리 알고리즘의 연구

유근식° , 최수영, 윤남균, 구용완
수원대학교 전자계산학과

Issue of Buffer Management Algorithm for efficient Movie Allocation on Video-on-Demand System

Kun-Sik You° , Su-Young Choi, Nam-Kyun Yun, Young-Wan Koo
Dept. of Computer Science, University of Suwon

요 약

주문형 비디오 시스템의 비용과 성능에 실제 영향을 미칠 수 있는 주요 설계 문제로서 실시간 디스크 스케쥴링, 버퍼 관리, 프리페칭, 영화 할당, 그리고 장치 대역폭 감소등이 있다. 여기서 영화 할당은 영화 사본의 개수를 결정하고, 그리고 서버들에게 영화 사본들이 배치될 위치를 결정한다. 본 논문에서는, 인기도를 고려해서 영화 사본들의 개수를 결정하는 알고리즘과 영화 사본들이 디스크에 배치될 때 topic별로 분리해서 각각의 topic들을 서로 다른 서버에 배치시키는 알고리즘을 제안한다. 그리고 본 논문에서는 각 서버에 한 개의 영화를 저장할 수 있는 버퍼를 두어 가장 인기있는 비디오만 버퍼를 사용하는 버퍼관리 알고리즘을 제안한다.

1. 서 론

정보 기술, 저장 매체 등의 발달과 함께 멀티미디어 시스템의 보급이 확산되면서 주문형 비디오(video-on-demand;VoD), 화상 회의, 전자게임, 전자서적 등의 응용 분야가 빠르게 현실화 되어가고 있다. 이중에서도 가장 활발히 연구되는 분야가 주문형 비디오 시스템(VoD)이다.

주문형 비디오 서비스는 ATM과 같은 고속의 통신망을 통한 비디오 대여 서비스라 할 수 있으며, 사용자는 원하는 시간에, 원하는 형태로 자유로운 조작을 통해 서비스를 제공받을 수 있는 대화형 서비스이다[1].

주문형 비디오(VoD) 서버는 지원해야할 스트림의 크기가 매우 크고, 온라인으로 사용자의 실시간 서비스를 보장해야 하는 특징을 가지고 있다. 최근 데트워크의 발전과 저장 장치의 대용량화, 고성능화에 의해서 주문형 비디오 서버의 가능성이 매우 높아지고 있다. 이러한 VoD에서 취급하는 비디오 데이터와 같은 연속 매체는 시간 종속적인 매체로 실시간 요구 조건을 충족시키면서 연속적으로 제공될 때에만 의미 전달이 가능하다[4]. 또한 연속 매체 데이터는 기존 데이터에 비해 크기가 커서 많은 입출력이 요구되는 반면에 입출력 요구마다 디스크에 접근하면 시스템 성능이 크게 저하될 수 있다. 따라서 본 논문에서는 영화를 할당할 때 일반 비디오 숍과 같이 topic별로 분리하고, 각 topic별 인기도 순으로 영화를 할당하는 알고리즘과 영화를 관리할 때 topic별로 저장한 각각의 서버 내에 가장 인기 있는 영화를 버퍼(buffer)에 유지하고 이 영화를 요청하는 클라이언트의 요청을 버퍼 캐쉬에서 제공함으로써 디스크 사용을 감소시키며 빠른 응답을 제공하는 버퍼 관리 알고리즘을 제안한다. 그리고 영화들의 인기도에 따라 사본의 개수를 결정하고, 그 사본들을 디스크에 저장하는 알고리즘으로 주사를 이용한 순차정책을 사용한다.

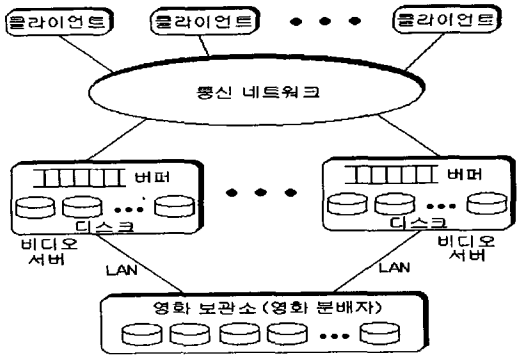
본 논문의 구성은 다음과 같다. 2장에서는 주문형 비디오 시스템을 살펴보고, 영화 사본 생성 알고리즘과 영화를 디스크에 할당하는 알고리즘을 설계한다. 3장에서는 버퍼관리 기법을 살펴보고, 4장에서 본 논문에서 제안한 알고리즘에 대한 성능의 분석과 결론 및 향후 연구 방향을 제시하였다.

2. 영화 사본 생성 및 할당 알고리즘 설계

본 논문에서는 [그림 2-1]과 같은 영화 저장 서버 시스템 모델을 가정한다. 각각의 클라이언트는 ATM과 같은 고속 통신망을 통해 비디오 서버와 연결되어 있다. 비디오 서버들은 영화의 topic별로 분리되어 있으며, 비디오들은 같은 영화에 대해서 다중 액세스를 지원하기 위해 비디오 서버내의 모든 디스크들에 걸쳐 스트라이프 되어진다. 비디오 스트라이핑이 갖는 장점은 시스템내의 모든 디스크들의 총 누적 대역폭이 가장 인기 있는 영화를 관람하는데 유용하다는 것이다. 영화 보관소에서는 모든 가용 영화들을 포함하고 있으며, 또한 주기적으로 각 서버에 있는 영화들의 참조 회수를 모니터링하여 인기도를 계산하고, 그 인기도에 따라 필요한 만큼의 영화 사본을 생성하여 오프-피크(off-peak)시간에 비디오 서버들에게 할당한다.

2.1 사본 생성 알고리즘

영화의 사본은 인기도를 고려하여 사본을 생성한다. 인기도를 구함에 있어 기존의 Zipf의 법칙을 사용하지 않고 새로운 방법을 제안



[그림 2-1] 영화 저장 서버 시스템 모델

한다. Zipf의 법칙에서는 사용자수에 의해서 인기도가 달라지고 사용자의 수에 따라 디스크의 개수를 결정하지만 현재의 웹 상에서 사용자의 수를 결정하기란 어려운 일이다. 따라서 본 논문에서 제안하는 방법은 현재 비디오 숍에서 인기도를 구하는 방법과 유사하며, 한 주간의 인기도가 그 다음주의 영화 사본 생성의 자료가 된다. 여기서 가장 중요한 요소는 영화의 참조 회수이다. 참조를 많이 한 영화가 인기도가 높다. 따라서 인기도를 구하는 방법은 i번째 영화의 참조 회수를 영화의 개수로 나눈 것이 각각의 인기도가 되고, 그 다음 전체 인기도의 합으로 영화 각각의 인기도를 나눈다. 그리고 나서 그 값과 각 서버에 저장할 수 있는 영화의 총수를 곱하면 각각의 영화에 대한 사본의 개수를 구할 수 있다. 그리고 각각의 영화에 대한 사본의 하한값은 하나이며 상한값은 서버에 저장할 수 있는 디스크의 수만큼 한다. 그리고 전체 사본 개수의 총합은 하나의 서버에 저장할 수 있는 영화의 총수이다. 본 논문의 사본 생성 알고리즘은 [그림 2-2]과 같다.

```
// 인기도를 구하고 그 합을 구한다.
for( i = 1; i <= M;  i++)
{ Pi[i] = float(Msui[i]) / M;
  SumPi = SumPi + Pi[i]; }
// 인기도 순으로 정렬한다.
for( i = 1; i < M; i++)
   for( j = i+1; j <= m; j++)
         if( Pi[i] < Pi[j]) swap(Pi[i] , Pi[j]);
Buffer = Pi[1];
// 사본을 생성한다.
for(i=2; i<=M; i++)
{ SabonX[i] = int(Pi[i] / SumPi * SAMsu);
    if(SabonX[i] < SabonMin) SabonX[i] = SabonMin;
    else if(SabonX[i] > SabonMax) SabonX[i] = SabonMax;
// 사본의 총 합이 가용 영화 저장 수 보다 적을 경우
if(sum < SAMsu)
   i = 2;
         while(sum < SAMsu)
         {    if(SabonX[i] < 6)
                  { SabonX[i] = SabonX[i] + 1;
                    sum++;  }
              i++;
         }
```

[그림 2-2] 사본 생성 알고리즘

2.2 할당 알고리즘

멀티미디어 시스템에서 영화의 크기는 매우 크므로 영화를 이주시키거나, 인기 없는 영화를 인기 있는 영화로 교체하는데 아주 많은 비용이 든다. 따라서 영화의 할당은 매우 중요하다. 이러한 영화 할당 알고리즘은 크게 정적과 동적 영화 할당 알고리즘으로 분류할 수 있다. 정적 영화 할당 알고리즘은 영화 분배자가 주기적으로 영화들의 인기도를 계산하고 그것에 따라 영화들의 사본의 개수와 비디오 서버들로의 배치를 결정한다. 그리고 동적 영화 할당 알고리즘은 시스템의 현재 또는 예측 부하에 따라 적응적으로 영화 사본들을 제거하거나 또는 생성한다.

Chen[2]은 복사 한계치가 주어지면 디스크 배열에 의해 지원되는 비디오 스트림의 최대 개수, 관련된 액세스 빈도를 갖는 영화의 개수, 그리고 비디오 서버를 위해 요구되는 디스크 배열의 개수를 먼저 결정한 후 영화 배치를 결정한다. 여기서는 영화를 디스크에 배치하는 방법으로는 순차, 교대, 랜덤의 세 가지 방법을 사용했다. 순차방법은 영화의 인기도 순으로 인기 있는 영화부터 사본들을 모두 할당하고, 그 다음 인기 있는 영화의 사본을 할당한다. 그리고 교대방법은 처음에는 인기 있는 영화의 사본을 할당하고, 다음 번에는 인기 없는 영화의 사본을 할당, 다시 인기 있는 영화를 할당하는 방식으로 영화를 할당한다. 마지막으로 랜덤방식은 무작위로 영화를 할당한다. 여기서는 영화들을 topic별로 분류하지 않고 모든 서버들에 배치하였다. 이렇게 하면 새로운 영화사본을 배치할 경우 모든 서버들의 디스크에 저장되어 있는 영화들을 다시 배치해야만 하는 문제점이 발생한다. 따라서, 본 논문에서 제안하는 영화 할당 알고리즘은 각각의 서버들에게 topic별로 분리되어 있는 영화들의 인기도를 고려하여 할당하는 알고리즘을 사용한다. 그리고 영화를 배치하는 알고리즘으로 Chen[2]의 순차배치를 변형한 주사를 이용한 순차배치를 사용하며, 각 topic별 서버에 한 개의 영화를 저장할 수 있는 버퍼를 두고 가장 인기 있는 영화는 디스크에 사본을 할당하지 않고 버퍼에만 할당한다. 따라서 가장 인기 있는 영화는 디스크에서 제공하는 것이 아니라 버퍼에서 바로 서비스를 한다. 그리고 영화를 할당하는 순서는, 먼저 가장 인기 있는 영화의 사본은 버퍼에 할당하고, 두 번째로 인기 있는 영화의 사본들 중 첫 번째 사본 → 첫 번째 디스크, 두 번째 사본 → 두 번째 디스크에 할당, 세 번째 사본 → 세 번째 디스크 순으로 할당되며, 그 영화의 사본이 모두 디스크에 할당되면 그 다음 인기 있는 영화의 사본들이 디스크에 순서대로 할당된다. 즉, 각각의 서버들은 topic별로 분리되어 있는 영화들만 있으며, topic별 영화들은 영화의 인기도 순으로 디스크에 할당된다. 예를 들어 하나의 비디오 서버 내에 다섯 개의 디스크가 있고 하나의 디스크에 다섯 편의 비디오를 저장할 수 있는 서버를 가정하고, 첫 번째 서버에 "topic=액션"이라는 비디오가 저장된다면, 액션 비디오들은 먼저 인기도를 구하고 인기도가 가장 높은 영화를 제외한 영화들을 인기도 순으로 사본의 개수를 결정한다. 그리고 가장 인기 있는 영화는 버퍼에 할당하고 그 다음 인기 있는 영화의 사본들 중 첫 번째 사본은 해당 서버의 첫 번째 디스크에 할당되고, 두 번째 사본은 그 다음 디스크에 할당된다. 이렇게 사본이 모두 할당된 다음 그 다음으로 인기 있는 영화의 사본들이 디스크에 할당된다. 즉, $V_iD_1 \leftarrow M_{1,1}$, $V_iD_2 \leftarrow M_{1,2}$, $V_iD_3 \leftarrow M_{1,3}$ ⋯, $V_iD_j \leftarrow M_{n,k}$ (M_{nk}는 n 번째 영화의 k 번째 사본, V_iD_j는 i 번째 서

버의 j 번째 디스크) 과 같이 순차적으로 영화들이 할당된다.

영화를 topic별로 각 서버의 디스크에 할당하는 알고리즘은 [그림 2-3]과 같다.

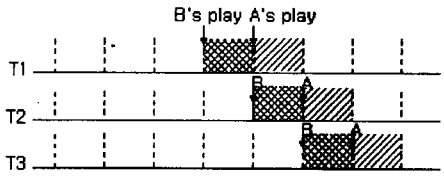
```
// 생성된 사본을 디스크에 순차적으로 할당한다.
i=2, j=1, k=1, p=1;
while(i <= M)
{  for(w=1; w<=SabonX[i]; w++)
    { SD[p][k] = SabonX[i];
      k++;
      if(k==6) {  k = 1;
                  p++;  }
    }
   if(p>Dsu) break;
}
```

[그림 2-3] 영화 할당 알고리즘

2.3 버퍼관리 기법

기존의 영화 배치 기법은 서버에 버퍼를 두지만 이 버퍼는 현재 읽혀지고 있는 영화를 위해 사용되어진다. 그러나 본 논문에서는 이러한 버퍼 이외에 각 topic별 서버에 하나의 영화를 저장할 수 있는 버퍼를 두고 각 topic 별 영화 중에 인기도가 가장 높은 영화를 버퍼에 두는 기법을 제안한다. 가장 인기 있는 영화를 요청하는 클라이언트는 대기 큐에 저장되며, 순차적으로 영화를 서비스 받게 된다. 버퍼내에 있는 영화는 1초 분량의 프레임으로 되어있으며, 요청한 클라이언트들에게 순차적으로 할당한다. 이렇게 함으로써 가장 인기 있는 즉, 가장 참조율이 높은 영화를 버퍼에 둠으로써 디스크 부하를 줄일 수 있다. 그리고 버퍼를 관리함에 있어서는 주기적으로 인기도를 계산하여 인기도가 가장 높은 영화로 교체한다.

가장 인기 있는 영화를 버퍼에 둠으로써 디스크로부터 영화를 읽어오지 않고 클라이언트의 요청을 버퍼에서 제공하므로 디스크 부하가 감소하여 연속 매체 시스템이 지원할 수 있는 최대 클라이언트 수가 증가할 수 있다. 예를 들어, 클라이언트 A, B가 가장 인기 있는 영화를 요청했을 때 단위 시간별 버퍼 참고 형태를 보면 [그림 2-4]와 같다.



[그림 2 - 4] 단위 시간별 CM 파일의 접근 형태

T1 시점에서 클라이언트 A가 참조한 데이터는 T2 시점에서 클라이언트 B에 의해 재사용 되어지고, T2 시점에서 A가 참조한 데이터는 T3 시점에서 클라이언트 B에 의해 재 참조되어진다. 동일한 연속 매체 파일을 다수의 클라이언트들이 참조할 때 버퍼로부터 읽어온 연속 매체 데이터는 시간적으로 인접한 후속 클라이언트에 의해 재 참조될 수 있으므로, 디스크의 부하를 그 만큼 줄일 수 있다.

4. 결론 및 향후연구

본 논문에서는 주문형 비디오 시스템에서 영화 할당 알고리즘과 버퍼 관리에 대해 연구하였다. 본 논문에서 제안한 영화 관리 알고리즘은 우선 영화를 topic별로 분류하여 서버에 저장하고, 각 topic별 서버 내에서 영화에 대한 참조회수를 기반으로 인기도를 구하고, 그 인기도의 값에 따라 영화 사본의 개수를 결정하는 알고리즘을 제안하였다. 그리고 영화 사본을 디스크에 할당하는 알고리즘으로 주사를 이용한 순차정책을 사용하였으며, 각 서버에 영화 사본 한 개를 저장할 수 있는 버퍼를 따로 두어 각 서버에서 가장 인기 있는 영화는 사본을 생성하지 않고 버퍼에 저장하여 그 영화를 요청하는 클라이언트에게 직접 서비스하는 알고리즘을 제안하였다. 이러한 알고리즘을 사용하였을 경우의 장점으로는 영화 보관소에서 영화를 관리할 때 각 topic별로 관리하기 때문에 관리가 쉽고 비용도 감소할 수 있다. 그리고 가장 인기 있는 영화는 버퍼에서 제공하므로 디스크의 부하를 줄일수 있을뿐더러 클라이언트들의 영화 요청에 대한 평균 응답시간을 줄일 수 있다. 그리고 본 논문에서 제안한 인기도를 구하는 방법은 기존의 Zipf의 법칙을 사용하지 않고 한 주간의 영화 참조 회수를 가지고 인기도를 산정 하였다. 향후 연구과제로는 네트워크 대역폭을 고려한 영화 할당과 예측 부하를 고려한 적응적 동적 영화 할당 등이 있다.

[1] D.C Thomas, et al., "Prospects for Interactive Video-on-Demand", IEEE Multimedia, Fall 1944, PP. 14-24

[2] M-S Chen and H-I Hsiao, C-S Li, and P. S. YU, " Using Rotational Mirrored Declustering for Replica Placement in a Disk-Array-Based Video Server", ACM Multimedia '95, PP. 121~130, Nov. 1995.

[3] E. J. O'Neil and G. W. Weikum, "The LRU-K Page Replacement Algorithm For Database Disk Buffering," Proc of the ACM SIGMOD, pp. 297~306, 1993.

[4] H. M. Vin, P. Goyal "A Statistical Admission Control Algorithm for Multimedia Server" proc. of the ACM Multimedia 94, PP. 33~40, Oct 1994.

[5] C. S Freedman and D. J. Dewitt, "The SPIFFI Scalable Video-on-Demand System", SIGMOD Record, Vol. 24, No. 2, pp. 352~363, June 1995.

[6] J. L. Wolf, P. S. Yu and H. Shachnai, "DASD Dancing: A Disk Load Balancing Optimization Scheme for Video-on-Demand Computer Systems", ACM Sigmetrics, Vol. 23, No. 1, pp. 157~166, May 1995.