동적 버퍼 분할을 사용한 클러스터 VOD 서버 부하 분산 기법

권춘자°, 최황규
강원대학교 전기전자정보통신공학부
e-mail:kwoncj@mail.kangwon.ac.kr, hkchoi@kangwon.ac.kr

A Load Balancing Technique Based on the Dynamic Buffer Partitioning in a Clustered VOD Server

Chun-Ja Kwon, Hwang-Kyu Choi
Dept. of Electrical and Computer Engineering, Kangwon University

요 약

본 논문은 클러스터 기반의 VOD 서버에서 동적 버퍼 분할을 이용한 새로운 부하 분산 기법을 제안한다. 제안된 기법은 사용자 요청을 처리하는 서비스 노드간의 버퍼 성능과 디스크 접근 빈도를 고려하여 전체 부하를 고르게 분산하도록 한다. 또한 동적 버퍼 분할 기법은 동일한 연속매체에 접근하려는 여러 사용자에게 평균 대기시간을 감소시킬 수 있도록 버퍼를 동적으로 분할한다. 시뮬레이션을 통한 성능분석 결과에서 제안된 기법은 기존의 기법보다 부하량을 적절히 조절하면서 평균 대기시간을 감소시키고 각 노드의 처리량과 병행 사용자 수를 증가시킴을 보인다.

1. 서 론

최근 들어 컴퓨팅 기술, 압축 기술, 저장 장치, 그리고 네트웍 장치 등의 기술 진보는 인터넷상에서 실시간 멀티미디어 서비스의 증대에 큰 영향을 미쳤으며, 이러한 환경의 변화로 멀티미디어 정보에 대한 요청도 크게 증가하고 있다[1]. 멀티미디어 연속 매체는 기존의 텍스트 데이터뿐만 아니라 비디오, 오디오, 정지 화상 등의 다양한 데이터로 이루어져 있다. 또 그 특성상 그 용량이 매우 크고 실시간 검색이 가능해야 하며, 서로 다른 스트림들 사이의 동기화를 필요로 하는 특성을 갖는다[7].

일반적으로 멀티미디어 서비스를 위한 시스템은 VOD 서버, 통신망, 단말기 등의 구성요소로 이루어진다. 이중 VOD 서버는 대용량 저장장치와 연속매체 서비스의 병행처리가 가능한 실시간 구조를 필요로 한다. 따라서 고가용성 및 확장성을 지원하기 위한 클러스터 VOD 서버 구조를 활용하는 많은 연구가 진행되고 있다. 비교적 저가의 고성능 PC 또는 워크스테이션들을 고속의 네트워크로 연결한 클러스터 VOD 서버는 확장성과 신뢰성이 높고 가격대 성능 면에서 우수하므로 이에대한 많은 연구가 이루어지고 있다[2].

본 논문은 클러스터 기반의 VOD 서버 환경에서 효율적인 버퍼관리 기법과 이를 활용한 부하 분산 기법을 제안한다. 제안된 기법은 연속 매체의 특성을 고려하여 각 노드의 버퍼를 동적으로 분할하고, 클러스터 서버 노드간 버퍼 성능과 디스크 접근 빈도를 고려하여 부하를 고르게 분산하는 기법이다. 제안된 기법을 이용함으로써 버퍼의 활용도를 높일 수 있고, 병행 사용자들의 평균 대기 시간을 줄여 VOD 서버의 성능을 향상시킬 수 있다.

본 논문의 구성은 먼저 2장에서 관련연구에 대하여 기술하고, 3장에서 제안된 부하 분산 기법과 동적 버퍼 분할 기법에 대해 설명한다. 4장에서 제안된 알고리즘의 성능분석을 수행하며, 마지막으로 5장에서 결론 및 향후 연구방향에 대해 기술한다.

2. 관련연구

클러스터 기반의 VOD 서버에 대한 연구는 크게 서버 구성, 데이터 배치, 디스크 스케줄링, 버퍼 관리 분야 등 4가지 분야로 구분할 수 있으며, 여기에서 가장 중요한 문제의 하나는 각 서버 노드간의 부하를 균등화하는 것이다. 각 노드의 부하를 균등화하기 위해서 일반적으로 사용되는 방법은 각 노드의 접속 수를 기준으로 하는 방법이다. 그러나 VOD 서버의 경우 접속 수가 노드의 부하를 의미하지 않는다. 각 노드의 중요한 부하는 디스크 접근 빈도수이며, 각 노드의 버퍼 관리 기법에 따라 디스크 접근 수

를 줄일 수 있으므로 디스크 접근 빈도 수와 사용자 접속 수가 비례하지 않는다. 따라서 기존의 부하 분산 방식을 클러스터 VOD 서버에 적용할 경우 부하의 불균등을 초래할 수 있다.

멀티미디어 객체는 연속 접근 특성을 갖는다. 따라서 멀티미디어 서버를 위한 많은 버퍼 관리 기법들이 제안되었다. 가장 대표적인 버퍼 관리 기법으로 버퍼 선인출 기법과 버퍼 교체 기법이 있다[3]. 여기서 버퍼 선인출 기법은 일정량의 데이터 블럭을 버퍼에 미리 저장하여 서비스를 수행한다. 기존의 버퍼 선 인출 기법으로는 버퍼 비분할 기법, 정적 버퍼 분할 기법, 적응 버퍼 분할 기법 등이 있다[4]. 그러나 전통적인 버퍼 선인출 기법을 사용할 경우 버퍼 낭비 등 효율적인 버퍼 관리가 불가능하다.

버퍼 교체 기법은 버퍼가 필요할 때마다 각 기법의 알고리즘에 따라 선택된 버퍼를 교체하는 방법이다. 여기에는 FFU , LRU, MRU 등의 교체 알고리즘이 있다[3]. 그러나, 최적 버퍼를 교체하는 것이 불가능하다는 단점이 있다. 이외에 연속 매체를 위한 대표적인 버퍼 관리 알고리즘으로 BASIC/DISTANCE, Generalized Interval Caching 방법이 있다[5].

3. 동적 버퍼 분할을 이용한 부하 분산 기법

3.1 클러스터 서버 구조

본 논문에서 사용하는 클러스터 서버의 구조는 기존의 two-tier 방식을 변형한 것이다. 일반적인 two-tier 방식에 부하 분산기를 추가한 구조로 리눅스 가상 서버 구조와 유사하며, 부하 분산기를 통한 효율적인 부하 분산이 가능하다는 장점을 갖는다. 본 논문에서 가정하는 서버구조는 다음 그림 1과 같다.

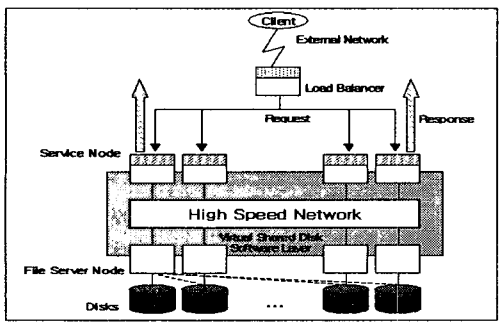

그림 1. 클러스터 서버 구조

3.2 부하 분산기(Load Balancer)

부하 분산기의 목적은 클러스터 내의 서버간 부하를 분산시키는 것이다. 부하가 집중된 서버를 자동 감지해 다른 서버로 트래픽을 분산시킴으로 전체 서버의 성능을 균등화하고 빠른 응답속도를 유지하게 한다.

본 논문의 부하 분산기는 각 서비스 노드의 부하 정보와 열려진 스트림 블록에 대한 정보, 동적 버퍼 정보를 이용해 서비스 노드간 부하를 적절히 유지하며 서버 노드의 성능을 최대로 높일 수 있도록 요청을 분배한다. 이 기법의 처리절차는 표 1의 알고리즘과 같으며, 부하 분산 기법은 크게 2 단계로 나누어 처리된다.

표 1. 동적 버퍼 분할을 이용한 부하 분산 알고리즘

```
새로운 request 도착;
Enrollment Window Table를 검색;

If(request 처리 노드 존재 == TRUE)
    해당 노드로 요구 분배;
else
{
    노드 관리 테이블 검색;
    if(가용 버퍼 존재 == TRUE )
      전체 그룹 수와 현재 그룹 수의 차이가 가장 큰 곳으로
      요구 분배;
    else if(최대 인터벌 > 0)
          해당 노드로 요구 분배;
        else
          대기큐에서 대기;
}
```

- 단계 1 : 사용자의 요청 파일을 서비스하고 있는 노드가 있는지 검색하여 해당 노드의 버퍼를 재 참조할 수 있다면 노드의 부하량에 상관없이 요청을 분배하는 단계이다.
- 단계 2 : 요청을 재 참조할 수 있는 버퍼가 존재하지 않을 경우 각 서비스 노드의 부하량을 측정하여 부하가 가장 적은 노드로 요청을 분배하는 단계이다.

제안된 부하 분산 기법을 통한 요구 분배 과정은 그림 2와 같다. 부하 분산기는 각 서비스 노드의 버퍼 정보와 부하량 정보를 유지해야 하며 각 노드 정보는 표 2에 나타나 있다.

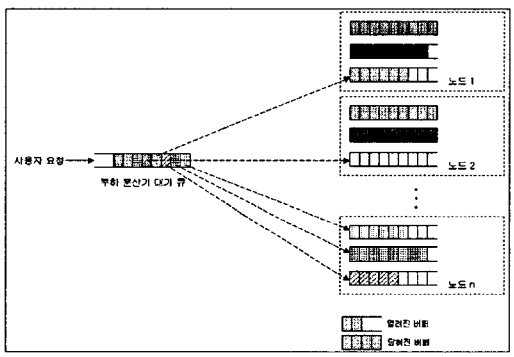

그림 2. 부하 분산기의 요구 분배

3.3 동적 버퍼 분할 기법

동적 버퍼 분할 기법은 연속 매체 스트림을 요청하는 다수의 사용자들에게 최소의 평균 대기 시간을 제공할 수 있는 버퍼 선인출 기법이다. 여기서, 부하 분산기를 통해 분배된 요구는 각 서비스 노드에 의해 처리되며, 각 서비스 노드는 동적 버퍼 분할 기법을 통해 사용자의 요구를 처리한다.

표 2. 부하 분산기에서 유지하는 각 노드의 정보

유지 정보	설명
Node ID	각 노드를 식별할 수 있는 식별자
각 노드의 그룹 수	각 서비스 노드에서 실질적으로 사용하고 있는 그룹 수
최대 그룹 수	각 노드에서 제공될 수 있는 최대 그룹 수
가용 버퍼량	각 노드에 남아있는 버퍼의 양
최대 버퍼 인터벌	각 노드의 서비스 그룹들 내에서 사용되는 버퍼의 최대 참조 간격
열려진 Enrollment Window	서비스 노드의 버퍼를 재 참조 할 수 있을 경우 재 참조가 가능한 영화에 대한 이름과 상태 정보

동적 버퍼 분할 기법은 처음에는 전체 버퍼를 하나의 버퍼로 관리하고, 필요할 때마다 버퍼를 그룹으로 분리하여 관리한다. 이 기법의 알고리즘과 처리 절차는 표 3과 그림 3에 나타낸다.

표 3. 동적 버퍼 분할 알고리즘

```
if (새로운 request를 위한 버퍼 == free)
{
  새로운  request를 위한 버퍼할당;
  버퍼의 세크먼트들을 참조하여 진행한 마지막 사용자까지
     하나의 그룹이 됨;
  if(request == 각 그룹의 처음 request)
    저장 장치로부터 새로운 블럭을 읽으면서 진행;
  else
    첫 request를 따라 버퍼를 재 참조하며 진행;
}
else
{
  최대 인터벌을 갖는 두 사용자를 선택;
  요구가 속한 그룹을 두개의 그룹으로 분리;
  두 요구 사이의 버퍼를 회수, 새로운 요구에 할당;
}
```

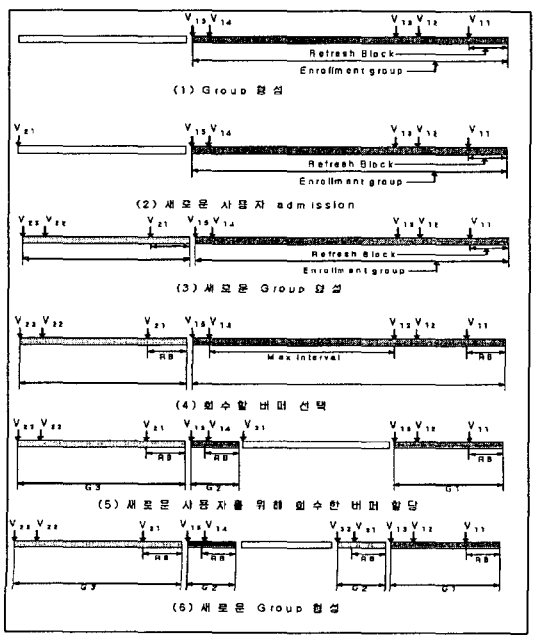

그림 3. 동적 버퍼 분할 기법

4. 성능평가

본 논문에서 제안한 부하 분산 기법과 동적 버퍼 분할 기법의 성능 분석을 위해 시뮬레이션 하였다. 시뮬레이션에 적용된 파라미터는 표 4에서 정의된다. 각 그래프에서 DBP-LB는 동적 버퍼 분할 기법에 부하 분산 기법을 적용했음을 나타내고, DBP-RR은 동적 버퍼 분할 기법에 라운드 로빈 요구 분배 방식을, GIC-RR은 Generalized Interval Caching 기법에 라운드 로빈 요구 분배 방식을 적용했음을 나타낸다. 또한, GIC-LB는 Generalized Interval Caching 기법에 부하 분산 기법을 적용한 결과이다.

그림 4는 서비스 시간 경과에 따라 각 노드의 부하량(디스크 접근 빈도)이 어떻게 변하는지를 나타낸다. 그래프에서 보면 서비스 시간이 증가해도 그룹의 수가 일정하게 유지되는 것을 볼 수 있다.

표 4. 시뮬레이션 파라미터

파라미터	값	단위
전체버퍼 크기	500(100..1000)	Mbytes
압축된 블록의 크기	0.06	Mbytes
재생률	4	Bblocs/sec
디스크속도	20 (20,30,...100)	Mbytes/sec
영화 재생 시간	6000	sec
평균서비스요청간격	10(5,10,...30)	sec
Refresh Slack	0.012	block

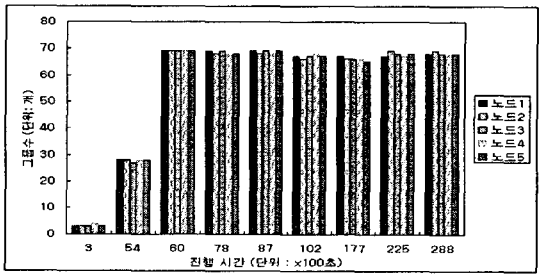

그림 4. 서비스 진행 시간에 따른 각 노드의 부하량

그림 5는 노드 수를 5개로 유지하고 각 서비스 노드의 버퍼량을 증가시켰을 때 평균 대기 시간을 측정한 그래프이다. 제안된 DBP-LB 기법이 DBP-RR 기법보다 성능이 우수함을 볼 수 있다. 부하분산 기법을 적용한 DBP-LB는 같은 영화에 대해 이전 요구와의 관계를 고려하여 버퍼 재참조가 가능한 쪽으로 요구 분배가 이루어지므로 버퍼량이 커질수록 평균 대기시간이 줄어든다.

그림 6은 30초당 들어오는 요청의 평균 도착 수를 정량화해서 구한 처리량(throughput)이다. 30초부터 1초까지 평균 도착 시간이 변화할 때의 처리량을 구했다. 제안된 기법의 처리량은 DBP-RR, GIC-RR보다 약 4.2배 뛰어남을 보였으며, GIC-LB보다 1.6배 우수함을 보였다.

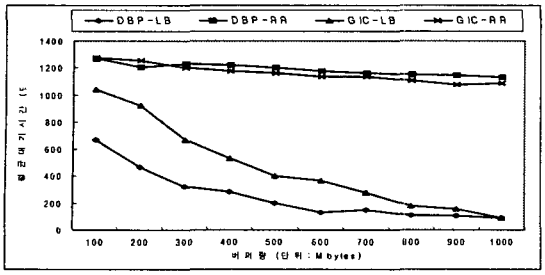

그림 5. 버퍼량 변화에 따른 평균 대기 시간 측정

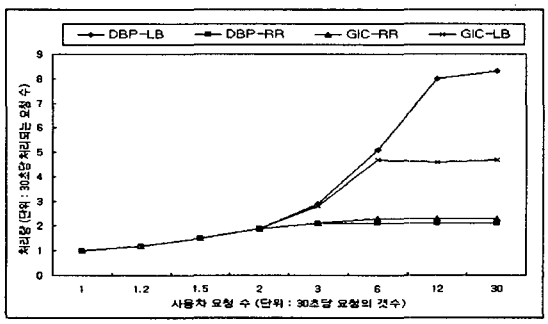

그림 6. 30초당 요청 빈도 수에 따른 처리량

마지막으로 그림 7은 노드 수가 증가함에 따른 병행 사용자 수를 보여준다. 노드 수가 작을 때 DBP-LB와 GIC-LB의 경우 부하 분산기 대기 큐에서 서비스 노드로 분배 시 참조 간격이 짧아져 더 많은 병행 사용자를 지원하는 특성이 나타났다.

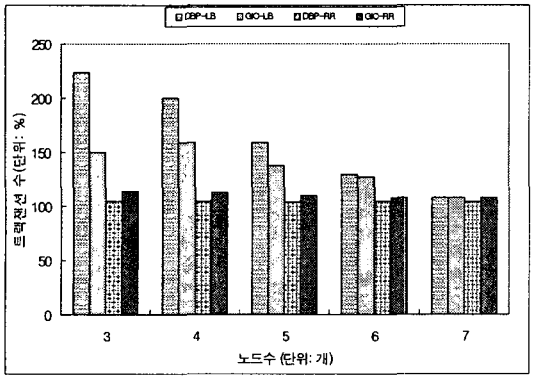

그림 7. 노드 수 변화에 따른 병행 사용자 수

5. 결 론

본 논문은 최근 관심의 대상이 되고 있는 클러스터 기반의 VOD 서버 환경에서 효율적인 버퍼관리 기법과 이를 활용한 부하 분산 기법을 제안한다. 제안된 기법은 연속 매체의 특성을 고려하여 각 노드의 버퍼를 동적으로 분할하고, 클러스터 서버 노드간 버퍼 성능과 디스크 접근 빈도를 고려하여 부하를 고르게 분산하는 기법이다. 제안된 기법을 이용함으로써 버퍼의 활용도를 높일 수 있고, 병행 사용자들의 평균 대기 시간을 줄여 VOD 서버의 성능을 향상시킬 수 있다. 시뮬레이션 성능 분석을 통하여 클러스터 VOD 서버 환경에서 제안된 기법은 기존의 다른 버퍼 관리 기법과 부하분산 기법을 적용했을 경우에 비하여 노드 부하 균등화, 사용자 평균 대기시간 감소, 병행 사용자 수 증가 등의 성능 향상을 얻었다. 특히, 클러스터 서버가 과부하 상태일 때 제안된 기법은 기존의 GIC-LB보다 약 1.5~2배의 우수한 성능을 나타내었으며, 시뮬레이션을 통하여 제안된 기법의 우수함을 보였다.

참고 문헌

[1] Wu D.P. Hou Y.W.T. and Zhu W.W., "Streaming video over the Internet: Approaches and directions", IEEE Transactions on Circuits & Systems for Video Technology, Vol. 11 No. 3, 2001.

[2] 최재영, 최종명, "고가용성 리눅스," 한국정보처리학회, Vol. 6, 1999.

[3] R. Tewari, R. Mukherjee, "Design and Performance Tradeoffs in Clustered Video Servers," Proceedings of the International Conference on Multimedia Computing and Systems, Los Alamitos, CA, 1996.

[4] D. Rotem and J. L. Zhao. "Buffer Management in Multimedia Database Systems," In Proc. of IEEE International Conference on Multimedia Computing and Systems, Hiroshima, Japan, June 1996.

[5] A. Dan and D. Sitaram, "A generalized interval caching policy for mixed interactive and long video environments," In IS&T SPIE Multimedia Computing and Networking Conference, San Jose, CA, January 1996.

[6] A. Garica-Martinez, J. Fernandez-Conde, "Efficient memory management in video on demand servers," Computer Communications 23, 2000.

[7] D. Jadav and A. Houdhary, "Design Issues in High Performance Media-on-Demand Servers," IEEE Parallel and Distributed Technology Systems and Applications, Summer 1995.