韓國컴퓨터情報學會 論文誌
第14卷 第5號, 2009. 5.

2009-14-5-1-5

대규모 멀티미디어 서버에서 효율적인 데이터 이동/중복 기법

김 은 삼*

An Efficient Data Migration/Replication Scheme in a Large Scale Multimedia Server

Eunsam Kim*

요 약

최근 멀티미디어 데이터의 품질이 높아짐에 따라 서버는 더 큰 용량의 저장 공간과 더 높은 I/O 대역폭을 요구하고 있다. 이러한 대규모 멀티미디어 서버에서는 각 객체의 인기도에 따른 접근 빈도의 차이로 인한 디스크 간의 부하 균형 문제가 성능에 중요한 영향을 미친다. 따라서 이 문제를 해결하기 위해 여러 데이터 중복 기법들이 제안되었다. 본 논문에서는 멀티미디어 서버에서 사용되는 대표적인 데이터 중복 기법인 동적 중복 기법보다 저장 효율이 뛰어나고 성능이 우수한 이동/중복 기법을 제안한다. 이 기법은 기존 중복 기법의 단점인 각 객체의 복사본 수를 줄임으로써 중복으로 인해 추가로 요구되는 저장 공간을 크게 감소시킨다. 또한 각 객체에 대한 요청들 간의 간격을 줄임으로써 캐싱 효과를 높여서 기존 중복 기법에 비해 동시에 지원할 수 있는 최대 사용자 수를 증가시킨다.

Abstract

Recently, as the quality of multimedia data gets higher, multimedia servers require larger storage capacity and higher I/O bandwidth. In these large scale multimedia servers, the load-unbalance problem among disks due to the difference in access frequencies to multimedia objects according to their popularities significantly affects the system performance. To address this problem, many data replication schemes have been proposed. In this paper, we propose a novel data migration/replication scheme to provide better storage efficiency and performance than the dynamic data replication scheme which is typical data replication scheme employed in multimedia servers. This scheme can reduce the additional storage space required for replication, which is a major defect of replication schemes, by decreasing the

• 제1저자 : 김은삼
• 투고일 : 2008. 08. 14, 심사일 : 2008. 10. 08, 게재확정일 : 2009. 05. 10.
* 홍익대학교 컴퓨터공학과 조교수
※ 이 논문은 2007학년도 홍익대학교 학술연구진흥비 및 2008년도 정부(교육과학기술부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2008-313-D00812)

number of copies per object. The scheme can also increase the number of concurrent users by increasing the caching effect due to the reduced lengths of the intervals among requests for each object.

▸ Keyword : 멀티미디어 서버(Multimedia Server), VOD, 데이터 이동/중복(Data Migration/Replication), 구간 캐싱(Interval Caching)

Ⅰ. 서 론

저장 장치와 네트워크 기술의 발전으로 VOD나 IPTV와 같은 주문형 멀티미디어 서비스가 일반화되고 있다. 이러한 멀티미디어 서비스를 지원하기 위해서는 대용량의 저장 공간과 높은 I/O 대역폭이 요구된다. 따라서 멀티미디어 서버에서는 다수의 디스크를 병렬로 구성하여 이러한 요구 사항을 만족시킨다[1]. 더구나 최근 HD와 같은 고품질의 멀티미디어 데이터가 등장함에 따라 멀티미디어 서버는 더 큰 저장 용량과 I/O 대역폭을 요구하므로 더 많은 수의 디스크를 필요로 하게 되었다. 이를 위해 SAN(Storage Area Network)이나 NAS(Networked Stroage System) 기반의 멀티미디어 서버가 일반화되고 있다[2][3].

멀티미디어 객체들은 인기도에 따라 사용자가 접근하는 빈도에 상당한 차이가 발생한다. 즉, 인기도가 높은 소수의 객체에 요청이 집중되기 쉽다. 따라서 각 멀티미디어 객체를 분산된 디스크들에 효율적으로 배치하지 않으면 인기도가 높은 객체를 저장하고 있는 특정 디스크에 부하가 집중되는 문제가 발생하게 된다. 이를 해결하기 위해 스트라이핑(striping) 기법이 연구되어 왔다[4][5]. 스트라이핑 기법은 각 멀티미디어 객체를 모든 디스크에 분산해서 저장함으로써 특정 디스크에 부하가 집중되는 것을 방지한다. 하지만 이 기법은 서버를 확장할 때 문제를 발생시킨다. 즉, 디스크를 추가하는 경우에는 모든 멀티미디어 객체를 다시 스트라이핑해야 한다. 또한 디스크 수가 증가함에 따라 재생 요청 후 서비스를 시작할 때까지의 대기 시간이 길어진다. 이것은 스트라이핑 기법이 각 요청에 대한 시작 시간을 지연함으로써 디스크 간의 부하 균형을 맞추므로 디스크 수에 비례해서 평균 대기 시간이 길어지기 때문이다.

이러한 스트라이핑 기법의 문제점를 해결하기 위해서는 각 객체가 스트라이핑되는 디스크의 수 (즉, 스트라이핑 그룹의 크기)를 제한하는 것이 필요하다. 즉, 디스크들을 여러 그룹으로 나누고 각 객체가 속한 그룹 내에서만 스트라이핑을 하는 것이다. 이 경우에는 인기도가 높은 객체에 대해 집중되는 요청들을 최대한 수용하기 위해 그 객체들을 여러 그룹에 중복 저장할 필요가 있다. 이와 같이 전체 디스크를 여러 그룹으로 나누거나 물리적으로 분산된 노드를 단위로 인기도가 높은 객체를 여러 노드에 중복 저장하는 기법을 데이터 중복 기법이라 한다[6][7][8][9][10].

본 논문에서는 각 객체의 인기도의 변화에 효율적으로 동작하는 대표적인 멀티미디어 데이터 중복 기법인 동적 중복 기법[9]보다 저장 효율과 성능이 우수한 이동/중복 기법을 제안한다. 이 기법은 기존 중복 기법의 단점인 각 객체의 복사본 수를 줄임으로써 중복으로 인해 추가로 요구되는 저장 공간을 크게 감소시킨다. 기존 중복 기법에서는 특정 객체에 대해 중복을 결정할 때 그 객체의 복사본을 저장하고 있는 노드들이 사용할 수 있는 디스크 대역폭의 크기만을 고려하였다. 하지만 제안하는 이동/중복 기법은 기존 기법과 달리 중복을 수행하기 전에 각 객체들이 저장되어 있는 위치의 연관성을 고려하여 이동 연산을 먼저 수행한다. 따라서 중복 연산 수를 줄임으로써 디스크 저장 공간 및 대역폭을 줄일 수 있다. 또한 이 기법은 기존 중복 기법에 비해 동시에 지원하는 사용자 수를 증가시킨다. 이것은 각 객체에 대한 복사본의 수가 감소하기 때문에 각 객체에 대한 요청들이 소수의 디스크에 집중되는 효과가 있기 때문이다. 다시 말해서 각 객체에 대한 요청 사이의 간격이 줄어들기 때문에 캐싱 효과로 인해 성능을 향상시킬 수 있다. 시뮬레이션 실험을 통해서 본 논문에서 제안하는 이동/중복 기법과 기존 중복 기법에 대해 각 객체에 대한 평균 복사본 수와 동시 지원하는 최대 사용자 수를 비교한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 멀티미디어 서버의 성능 향상에 관련된 연구를 살펴본다. Ⅲ장에서는 시스템 모델과 본 논문에서 제안하는 이동/중복 기법을 설명한다. Ⅳ장에서는 시뮬레이션을 통한 성능평가 결과를 기술한다. 마지막으로, Ⅴ장에서는 본 논문의 결론을 기술한다.

Ⅱ. 관련 연구

다수의 디스크들을 병렬로 연결해서 대규모의 저장 공간과 대역폭을 제공하는 멀티미디어 서버에 대한 연구에서는 디스크 간의 부하 균형 문제가 성능을 결정하는 중요한 문제로 인식되어 왔다.

위에서 언급한 것처럼 이 문제를 해결하기 위해 두 가지 접급 방

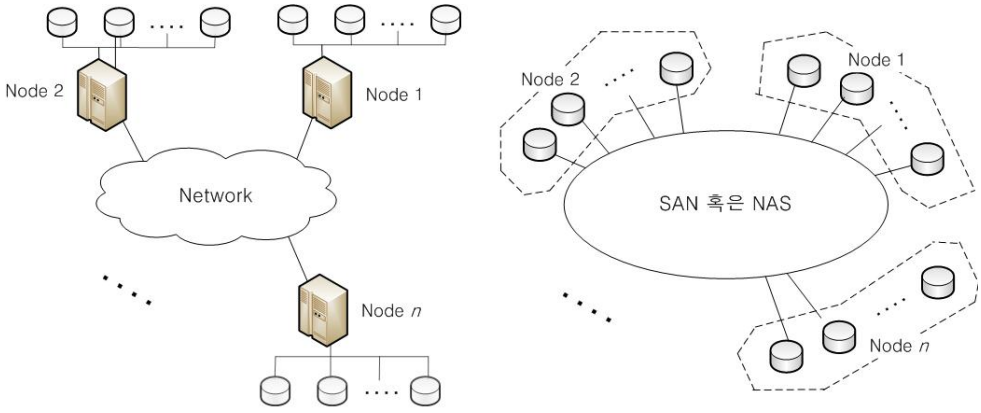

그림 1. 대규모 멀티미디어 서버 구조: (a) 분산 시스템 기반 구조, (b) SAN 혹은 NAS 기반 구조

Fig 1. The structure of large scale multimedia servers: (a) Structure based on distributed systems, (b) Structure based on SAN or NAS

법이 제안되었다. 먼저 각 객체의 세그먼트를 모든 디스크들에 분산해서 저장하는 스트라이핑 기법이다[4][5]. 스트라이핑 기법의 문제점인 낮은 확장성과 긴 초기 대기 시간을 해결하기 위해 데이터 중복 기법들이 제안되었다[6][7][8][9] [10]. 정적으로 중복 연산을 수행하는 여러 중복 기법들이 제안되었지만[6][7] 시간에 따라 각 객체의 접근 빈도의 변화에 대응하지 못하는 단점이 있었다. 이를 해결하기 위해 주기적으로 중복 연산을 수행하여 복사본의 수를 조절하거나[8] 임계값을 이용하여 동적으로 중복 연산을 결정하는 기법들이 제안되었다 [9][10]. 하지만 중복 연산을 결정하는 기준으로 각 객체를 저장하고 있는 노드들의 사용 가능한 서비스 용량만을 고려했기 때문에 복사본의 수가 크게 증가하는 문제가 발생하였다.

또한, 멀티미디어 서버에서는 멀티미디어 데이터의 순차적인 접근 패턴과 긴 재생 시간을 가지는 특성을 이용한 구간 캐싱 기법을 통해 성능 향상과 초기 대기 시간을 줄였다[11][12]. 구간 캐싱 기법은 동일한 객체에 대한 연속적인 요청들 사이의 간격들을 캐싱함으로써 디스크 대역폭을 사용하지 않고 추가적인 요청을 수용할 수 있다. 본 논문에서는 멀티미디어 서버에서의 캐싱 기법으로 구간 캐싱을 사용한다고 가정한다.

일반 웹 데이터에 대한 이동 및 중복 기법에 대한 많은 연구가 진행되어 왔다[13][14]. 이 기법들은 웹 데이터의 중복을 통해 서버들 간에 부하가 균형을 이루도록 하는 점은 멀티미디어 동적 중복 기법과 유사하다. 하지만 멀티미디어 데이터는 갱신이 거의 발생하지 않는 반면 이 기법들은 갱신이 빈번하게 발생하는 웹 환경에서 적용되기 때문에 데이터의 일관성 문제를 고려해야 한다. 또한 일반 웹 데이터에 비해 멀티미디어 객체는 훨씬 길기 때문에 이동/재생의 수를 제한하는 것이 더욱 중요하다.

표 1 기호 정의
Table 1 Definition of Symbols

기호	의미
S_i	멀티미디어 객체 i
N_x	노드 x
$L_x(t)$	t 시각에서 N_x 의 부하 (스트림 수)
$R_i(t)$	S_i를 저장하고 있는 노드의 집합
$A_i(t)$	t 시각에서 $R_i(t)$에 속한 노드들이 사용 가능한 총 서비스 용량 (스트림 수)
m_i	S_i의 재생 길이 (초)
λ	재생 요청의 평균 도착율 (요청수/초)
$p_i(t)$	t 시각에서 사용자가 S_i를 요청할 확률 (인기도)
$T_i(t)$	S_i에 대한 중복 연산을 시작하기 위한 임계값 (스트림 수)
$C_i(t)$	t 시각에서 S_i를 제외하고 $R_i(t)$에서 현재 서비스 중인 객체의 집합 (노드와 객체의 쌍으로 구성)
$K_i(t)$	t 시각에서 $R_i(t)$를 제외하고 $C_i(t)$를 저장하고 있는 노드의 집합 (객체와 노드의 쌍으로 구성)
n_s	이동 혹은 중복 연산에서 송신 노드
n_t	이동 혹은 중복 연산에서 수신 노드

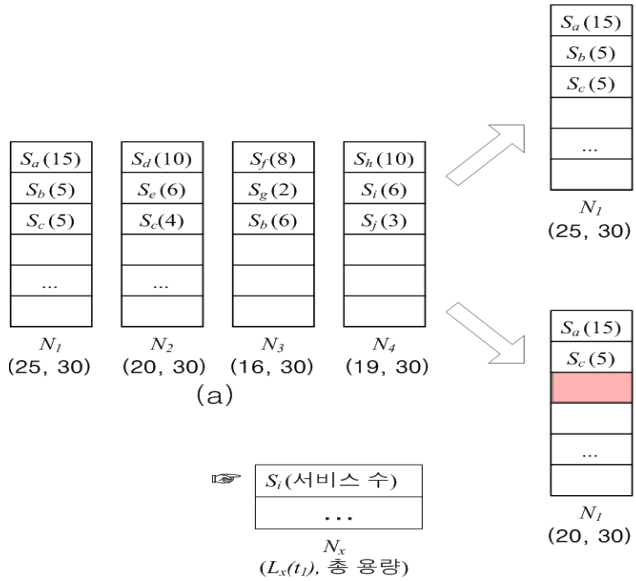

그림 2. 기존 동적 중복 기법과 이동/중복 기법의 동작 예: (a) 요청 처리 이전 저
결과, (c) 이동/중복 기법 사용 후 결과
Fig 2. Operation examples of the conventional dynamic replication a
storage state before processing the request, (b) result after app
replication scheme, (c) result after applying the migrat

III. 동적 이동/중복 기법

본 장에서는 멀티미디어 서버에서 저장 효율과 성능을 향상시킨 데이터 이동/중복 기법을 제안한다. 3.1절에서는 본 논문에서의 멀티미디어 서버 시스템 모델을 기술하고 3.2절에서는 기존의 동적 중복 기법의 문제점을 설명하고 이를 해결하는 이동/중복 기법을 제안한다.

.

3.1 시스템 모델

본 논문에서는 대규모 멀티미디어 저장 시스템을 구성하는 전체 디스크를 여러 노드로 분할한다. 각 노드는 스트라이핑을 독립적으로 수행하는 기본 단위로서 논리적인 디스크 그룹으로 정의한다. 예를 들어, 하나의 노드는 〈그림 1(a)〉와 같이 분산 시스템 구조에서 디스크 배열을 가진 하나의 서버나 〈그림 1(b)〉와 같이 SAN과 NAS와 같은 네트워크 저장 시스템에서의 분할된 디스크 그룹이 될 수 있다. 모든 노드들은 네트워크를 통해 연결되어 있다고 가정한다. 〈표 1〉은 본 논문에서 사용되는 주요 기호들이다.

멀티미디어 서버에서 동적 중복 기법은 다음과 같이 동작한다[9]. 각 노드는 한정된 저장 용량과 디스크 대역폭을 제공한다. $L_x(t)$는 노드 x(즉, N_x)의 현재 부하를 서비스 중인 스트림 수로 나타낸다. 각 객체는 여러 노드에 중복 저장될 수 있으며 객체 i(즉, S_i)를 저장하고 있는 노드의 집합은 $R_i(t)$로 나타낸다. $R_i(t)$에 속하는 노드 수(즉, S_i의 복사본 수)는 S_i의 인기도에 의해 결정되므로 시간에 따라 변할 수 있다. $A_i(t)$는 $R_i(t)$에 속한 모든 노드가 현재 사용 가능한 총 서비스 용량을 스트림 수로 나타낸 것이다. 사용자의 재생 요청은 λ 비율로 도착하며 S_i를 요청할 확률은 $p_i(t)$로 나타낸다. 각 객체의 재생 길이는 m_i라고 가정한다. 중복 연산을 위한 임계값인 $T_i(t)$는 $\lambda^* m_i^* p_i(t)$로 계산한다.

$A_i(t)$값이 $T_i(t)$값보다 작거나 같으면 중복 연산을

시작해야 한다. 즉, S_i의 인기도를 고려할 때 추가적인 서비스 용량을 확보하기 위해서는 t 시각에서 m_i 시간이 소요되는 중복 연산을 시작해야 한다는 의미이다. 그렇지 않으면 다른 노드에 S_i의 복사가 완료되기 전에 도착하는 요청을 모두 수용할 수 없을 가능성이 높다는 것이다. 중복 연산에서 송신 노드인 n_s는 $R_i(t)$에 속하는 노드들 중에서 부하가 가장 작은 노드를 선택하며 수신 노드인 n_t는 $R_i(t)$에 속하지 않는 노드들 중에서 부하가 가장 작은 노드를 선택한다. 시간이 지남에 따라 특정 객체의 인기도가 낮아지면 저장 공간을 확보하기 위해 복사본의 수를 줄일 필요가 있다. 복사본을 삭제하는 시점을 결정하는 기준은 중복 연산을 결정하는 기준과 반대로 $A_i(t)$가 $T_i(t)$보다 커지는 시점이다.

3.2 이동/중복 기법

멀티미디어 서버에서는 일반적으로 인기도에 따라 각 객체에 대한 요청 빈도의 차이가 크기 때문에 인기도가 높은 객체를 저장하는 노드에 부하가 집중될 수 있다. 이로 인해 다른 노드들은 충분한 서비스 용량이 남아 있음에도 불구하고 부하가 집중된 노드로 인해 시스템의 성능이 저하될 수 있다. 위에서 언급했듯이 이를 해결하기 위한 하나의 접근 방법으로 데이터를 동적으로 중복하는 기법이 사용되어 왔다. 〈그림 2〉는 기존의 동적 중복 기법과 제안되는 이동/중복 기법을 비교하여 설명하기 위한 예를 나타낸다. 이 예에서는 현재 시각은 t_1, 각 노드가 동시에 서비스할 수 있는 최대 스트림 수는 30, 그리고 $T_a(t_1)$ = 5, $T_b(t_1)$ = 3, $T_c(t_1)$ = 2 라고 가정한다.

〈그림 2(b)〉는 기존의 동적 중복 기법을 사용한 결과를 보여준다. 〈그림 2(a)〉에서는 N_1만이 S_a를 저장하고 있으므로 $A_a(t_1)$값은 5이다. $T_a(t_1)$값도 5이므로 현재 시점에서 중복 연산을 시작해야 한다. 기존 기법에서는 중복 연산을 수행할 때 n_s와 n_t를 결정하는 기준으로 각 노드의 현재 부하(즉, $L_x(t_1)$)만을 고려했기 때문에 N_3을 n_t로 선택하게 된다. N_1이 S_a를 저장하고 있는 유일한 노드이므로 n_s는 N_1이 된다. 따라서 〈그림 2(b)〉와 같이 N_1에 저장되어 있는 S_a를 N_3으로 복사하게 된다. 하지만 N_3은 복사본을 위한 추가적인 저장 공간을 사용해야 하는 단점이 있다. 또한 N_1에서 N_3로 S_a를 복사하기 위해 N_1과 N_3이 추가적인 디스크 대역폭도 필요하다.

이동/중복 알고리즘

(input : i /*객체 ID*/, t /*현재 시각*/)

```
if (A_i(t) > 0)
  가용 디스크 대역폭이 가장 큰 노드에 할당;
else
  요청 거절;
if (A_i(t) ≤ T_i(t)) { /* 이동 수행 */
  C_i(t) 구함;
  K_i(t) 구함;
  이동할 객체와 n_s 및 n_t는 C_i(t)와 K_i(t)의
    가능한 모든 조합을 비교하여 이동으로 인해
    n_t의 L_x(t)값 증가가 가장 작은 경우 선택;
  이동 연산 수행;
  n_s에서 이동된 객체 삭제;
}
if (A_i(t) ≤ T_i(t)) { /* 중복 수행 */
  R_i(t)에 속하는 노드 중 L_x(t)가 가장 작은
    노드를 n_s로 선택;
  R_i(t)에 속하지 않는 노드 중 L_x(t)가 가장
    작은 노드를 n_t로 선택;
  중복 연산 수행;
}
```

이러한 중복 연산을 위해 필요한 추가적인 자원 사용을 줄이면서 성능을 향상시키는 이동/중복 알고리즘은 다음과 같이 동작한다.

먼저 S_i에 대한 새로운 요청이 도착하면 $A_i(t)$가 소진되지 않았다면 요청을 수락한다. 그리고 기존 중복 연산을 수행해야 하는 조건인 $A_i(t)$가 중복 임계값인 $T_i(t)$보다 작거나 같은지를 검사한다. 이 조건을 만족한다는 것은 $R_i(t)$에 속한 노드들이 향후 예상되는 S_i에 대한 요청 빈

도를 고려하면 서비스 용량이 부족하기 때문에 S_i를 위한 추가적인 서비스 용량을 확보할 필요가 있다는 의미이다. 제안하는 기법에서는 이 조건이 만족될 때 중복 연산을 수행하기 전에 이동 연산을 먼저 수행한다.

이를 위해 $R_i(t)$에 속한 노드들이 현재 서비스하고 있는 객체의 집합인 $C_i(t)$를 구한다. 이때 새로운 요청의 대상인 S_i는 제외한다. 또한 $C_i(t)$를 저장하고 있는 노드의 집합인 $K_i(t)$를 구한다. 이때 $R_i(t)$에 속하는 노드들은 제외한다. 즉, $R_i(t)$에서 향후 S_i에 대한 요청을 수용하기 위한 추가적인 디스크 대역폭을 확보하기 위해 $R_i(t)$에서 서비스하고 있는 객체들 중 하나를 $R_i(t)$에 속하지 않는 노드로 이동시키는 것이다. 이때 이동할 객체와 n_s 및 n_t는 $C_i(t)$와 $K_i(t)$의 가능한 모든 조합을 비교하여 이동으로 인해 n_t의 $L_x(t)$ 값의 증가가 가장 작은 경우를 선택한다. 그런 다음 선택된 객체를 n_s에서 n_t로 복사한 후 n_s에서 그 객체를 삭제한다.

이동 연산을 수행한 후에도 $A_i(t)$가 $T_i(t)$보다 작거나 같으면 중복 연산을 수행한다. 하지만 이동 연산을 수행함으로써 노드간의 부하를 분산시켰기 때문에 중복 연산을 수행해야 할 경우를 줄일 수 있다.

〈그림 2(a)〉에서 제안된 이동/중복 기법을 적용하는 경우를 살펴보자. 현재 $A_a(t_1)$와 $T_a(t_1)$ 값이 모두 5이므로 중복 연산을 수행하기 전에 먼저 이동 연산을 수행한다. 이를 위해 $C_a(t_1)$를 구한다. 현재 $R_a(t_1)$에 속한 유일한 노드인 n_1이 서비스하고 있는 객체들은 S_a를 제외하면 S_b와 S_c이므로 $C_a(t_1)$ = {(N_1, S_b), (N_1, S_c)}이다. 또한, N_1을 제외하고 $R_a(t_1)$에 속한 S_b와 S_c를 저장하고 있는 노드는 N_2와 N_3이므로 $K_a(t_1)$ = {(S_b, N_3), (S_c, N_2)}이다. 이때 $C_a(t_1)$와 $K_a(t_1)$의 모든 가능한 (n_s, 이동할 객체, n_t) 조합은 (N_1, S_b, N_3)와 (N_1, S_c, N_2)이다. 이 둘 중에서 이동 후 n_t의 $L_x(t_1)$ 값 증가가 더 작은 경우는 (N_1, S_b, N_3)이므로 〈그림 2(c)〉에서와 같이 S_b를 N_1에서 N_3으로 이동함으로써 $R_a(t_1)$에 속한 노드들의 디스크 대역폭을 확보하였다. 이동 후에는 $A_a(t_1)$ 값이 $T_a(t_1)$보다 작기 때문에 더 이상 중복 연산을 수행할 필요가 없다.

〈그림 2〉의 예에서 보듯이 제안된 이동/중복 기법은 크게 두 가지 장점을 가진다. 먼저 중복 연산수를 줄임으로써 복사본의 수가 증가함으로써 추가적으로 요구되는 저장 공간의 사용을 줄일 수 있다. 또한 특정 객체를 복사하기 위해 필요한 두 노드의 디스크 대역폭도 줄임으로써 서비스할 수 있는 요청 수를 그 만큼 증가시킬 수 있다. 또한 이동 연산 후에는 특정 객체를 저장하는 노드의 수가 줄어들기 때문에 그 객체에 대한 요청은 소수의 노드에 집중되게 된다. 이로 인해 각 객체에 대한 요청들 사이의 간격을 줄일 수 있다. 이것은 멀티미디어 서버에서 사용하는 대표적인 캐싱 기법인 구간 캐싱의 성능을 향상시킨다. 구간 캐싱에서는 구간 길이가 줄이들면 주어진 캐쉬 크기로 캐싱할 수 있는 구간의 수가 증가하기 때문에 디스크 대역폭을 사용하지 않고 서비스할 수 있는 요청 수가 증가하게 된다.

Ⅳ. 성능 평가

이 장에서는 기존의 동적 중복 기법과 본 논문에서 제안한 이동/중복 기법의 성능을 비교하는 시뮬레이션 실험 결과를 보여준다. 〈표 2〉는 시뮬레이션에 사용된 디스크와 멀티미디어 객체에 대한 실험 변수들이다.

표 2 시뮬레이션 변수
Table 2 Simulation Parameters

변수	값
디스크 캐쉬 크기	256 MB
디스크 블록 크기	1024 KB
유효 디스크 대역폭	37.5 MBytes/s
객체 재생율	19.4 Mbits/s (HD)

시스템은 각 5개의 디스크를 가지는 100개의 노드로 구성된다. 각 노드는 75개의 요청을 동시에 서비스할 수 있다. 또한 400개의 서로 다른 객체가 존재하고 각 객체는 90분의 재생 시간을 가진다. 시스템 동작 초기에는 각 객체 당 하나의 복사본을 노드들에 라운드 로빈 방식으로 배치하였다. 객체들의 인기도를 반영하기 위해 VOD에서 일반적으로 사용하는 0.271의 변수를 가지는 Zipf 확률 분포를 사용하였다. 재생 요청의 빈도에 따른 성능을 비교하기 위해 평균 요청 도착 간

격 (즉, $1/\lambda$) 값은 0.6에서 1.1까지 변화시켰다. 또한, 구간 길이의 차이에 따른 성능의 변화를 보이기 위해 멀티미디어 서버에서의 대표적인 캐슁 기법인 구간 캐슁 기법을 사용하였다. 캐슁은 각 노드에서 분산되어 있는 디스크의 캐쉬 메모리에서 수행된다. 본 시뮬레이션은 C++ 언어를 사용하여 이벤트 구동 방식으로 구현했다.

4.1 객체의 평균 복사본 수

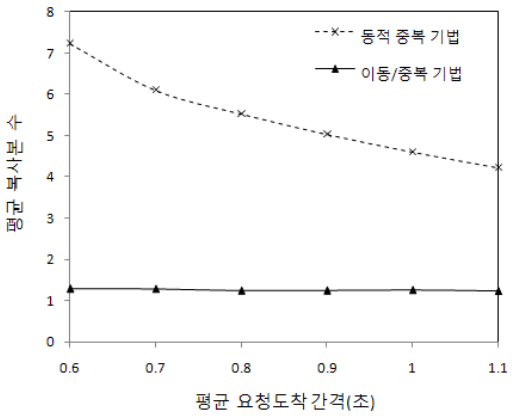

그림 3. 평균 복사본 수 비교
Fig 3. Comparison of the average number of copies per object

〈그림 3〉은 기존 동적 중복 기법과 이동/중복 기법에서 요청들의 평균 도착 간격에 따른 각 객체의 복사본 수를 보여준다. 실험 결과를 통해서 이동/중복 기법이 모든 도착 간격 값에서 훨씬 적은 수의 복사본을 유지한다는 것을 알 수 있다. 즉, 평균적으로 약 3.3배 적은 복사본 수를 유지한다. 특히 도착 간격이 작을수록 (즉, 요청 빈도가 높을수록) 그 차이가 크다는 것을 알 수 있다. 평균 도착 간격이 초당 0.6인 경우에는 평균 복사본의 수가 약 4.5배 차이가 난다. 이것은 이동/중복 기법이 이동 연산을 통해 최대한 중복 연산을 피하면서 부하 균형을 맞추기 때문이다. 부하가 어느 정도 이상이 되면 중복 연산이 수행되지만 본 실험 결과에서 보듯이 복사본의 수는 크게 증가하지 않는다. 다시 말해서 이동 연산은 부하를 전체 노드에 공평하게 분배하는 역할을 하면서도 추가적인 많은 저장 공간을 요구하지 않다는 것이다. 이것은 중복 기법의 가장 큰 단점인 복사본을 위한 추가적인 저장 공간을 사용해야 하는 문제를 해결할 수 있다는 점에서 의미가 있다.

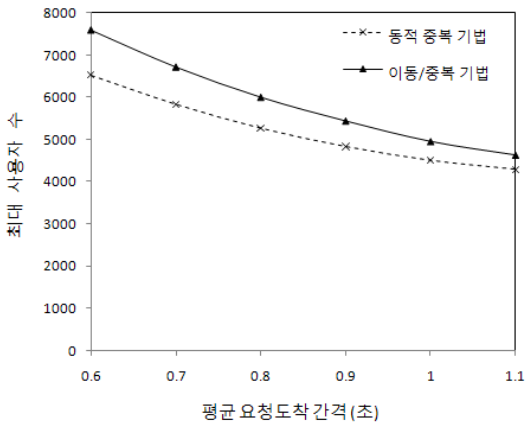

그림 4. 최대 동시 사용자 수 비교
Fig 4. Comparison of the total number of concurrent users

4.2 최대 사용자 수

〈그림 4〉는 평균 도착 간격에 따라 두 기법이 지원하는 최대 동시 사용자 수를 비교하고 있다. 이동/중복 기법이 모든 도착 간격에서 더 좋은 성능을 보이며 평균적으로 12.8% 더 많은 사용자를 지원했다. 특히, 평균 도착 간격이 초당 0.6인 경우에는 16.5% 성능 향상을 보였다. 이러한 성능 향상은 크게 두 가지 요인 때문이다. 첫째, 각 객체에 대한 복사본 수를 줄임으로써 각 객체가 소수의 노드에 저장되게 된다. 이로 인해 각 객체에 대한 요청들 또한 소수의 노드에 집중되므로 요청들 사이의 구간 길이를 줄어들게 하는 효과를 나타낸다. 실험은 통해서 이동/중복 기법의 구간 길이가 기존 동적 중복 기법보다 약 47% 정도 짧다는 것을 알 수 있었다. 따라서 주어진 캐쉬 메모리로 더 많은 수의 요청을 디스크 대역폭을 사용하지 않고 캐슁으로 지원할 수 있기 때문에 동시 지원하는 사용자 수를 증가시킨다. 둘째, 이동/중복 기법에서는 중복 연산을 수행하기 이전에 이동 연산을 수행함으로써 중복 연산의 수를 줄인다. 따라서 중복 연산에 사용해야 할 디스크 대역폭으로 추가적인 요청을 지원함으로써 성능 향상에 기여한다.

V. 결론

본 논문에서는 기존의 멀티미디어 데이터를 위한 중복 기

법의 문제점인 복사본의 수를 줄이고 동시 지원 가능한 사용자 수를 증가시키는 이동/중복 기법을 제안하였다. 이 기법은 중복을 결정하기 전에 각 노드가 저장하고 있는 객체에 대한 정보를 이용하여 현재 서비스되고 있는 요청을 이동함으로써 중복 연산을 줄이고 각 객체에 대한 요청 사이의 간격을 줄였다. 또한 실험을 통해 이동/중복 기법이 기존의 동적 중복 기법에 비해 각 객체에 대한 복사본의 수가 최대 4.5배 줄었고 성능도 최대 16.5%까지 향상된다는 것을 보였다.

멀티미디어 데이터의 품질이 지속적으로 높아감에 따라 더 높은 디스크 대역폭을 요구할 것으로 예상된다. 따라서 본 논문에서 제안하는 이동/중복 기법은 향후 더 많은 수의 디스크를 필요로 하는 대규모 멀티미디어 서버에 더 효과적으로 적용될 것으로 기대된다. 본 논문의 향후 연구과제는 이동/중복 기법이 최적으로 동작할 수 있도록 스트라이핑 그룹의 크기를 결정하는 것이다.

참고문헌

〔1〕 정현식, "VOD 시스템에서의 실시간 응용 프로그램을 위한 디스크 배열 구조의 성능 분석,"한국정보컴퓨터학회논문지, vol. 5, no. 4, pp. 106-111, 2000년 4월.

〔2〕 G. Gibson and R. Meter, "Network attached storage architecture", Communications of the ACM, vol. 43, no. 11, pp. 37-45, 2000.

〔3〕 G. Ma and A. Khaleel and N. Reddy, "Performance evaluation of storage systems based on network-attached disks," IEEE Transactions on Parallel and Distributed Systems, vol. 11, no. 9, pp. 956-968, 2000.

〔4〕 S. Berson and S. Ghandeharizadeh and R.R. Muntz and X. Ju, "Staggered Striping in Multimedia Information Systems", Proc. ACM SIGMOD Conf., pp. 79-90, 1994.

〔5〕 J. Ding and Y. Huang, "Resource-Based Striping: An Efficient Striping Strategy for Video Servers Using Heterogeneous Disk-Subsystems", Multimedia Tools and Applications, vol. 19, no. 1, pp. 29-51, 2003.

〔6〕 D. Serpanos, L. Georgiadis, and T. Bouloutas, "MMPacking: A Load and Storage Balancing Algorithm for Distributed Multimedia Servers", IEEE Transactions on Circuits and Systems for Video Technology, vol. 8, no. 1, pp. 13-17, 1998.

〔7〕 N.Venkatasubramanian and S.Ramanathan, "Load Management in Distributed Video Servers", Proc. Conf. on Distributed Computing Systems, pp. 31-39, 1997.

〔8〕 J. Wolf, P. Yu, and H. Shachnai, "DASD Dancing: A Disk Load Balancing Optimization Scheme for Video-on-Demand Computer", Proc. ACM SIGMETRICS Conf., pp. 157-166, 1995.

〔9〕 P. Lie and J.. Lui, "Threshold-Based Dynamic Replication in Large-Scale Video-on-Demand Systems", Multimedia Tools and Application Journal, vol. 11, no. 1, pp. 35-62, 2000.

〔10〕 E. Kim and J. Liu, "Design of HD-quality Streaming Networks for Real-time Content Distribution," IEEE Transactions on Consumer Electronics, Vol. 52, No. 2, pp. 392-401, 2006.

〔11〕 A. Dan and D. Sitaram, "A Generalized Interval Caching Policy for Mixed Interactive and Long Video Workloads", Proc. Multimedia Computing and Networking Conf., pp. 344-351, 1996.

〔12〕 N. Sarhan and C. Das, "Caching and Scheduling in NAD-based Multimedia Servers," IEEE Transactions on Parallel and Distributed Systems, vol. 15, no. 10, pp. 921-933, 2004.

〔13〕 N. Fujita, Y. Ishikawa, A. Iwata, and R. Izmailov, "Coarse-grain replica management strategies for dynamic replication of Web contents," Computer Networks, vol. 45, no. 1, pp.19-34, 2004.

〔14〕 M. Bhide, P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham, and P. Shenoy, "Adaptive push - pull desseminating dynamic web data," IEEE Transactions on Computers, vol. 51, no. 6, pp. 652 - 668, 2002.

저 자 소 개

김 은 삼(Eunsam Kim)

1994: 서울대 컴퓨터공학과 학사

1996: 서울대 컴퓨터공학과 석사

1996-2002: LG전자 선임연구원

2006: University of Florida 컴퓨터공학과 박사

현재: 홍익대학교 컴퓨터공학과 조교수

관심분야 : 분산 멀티미디어 시스템, 컴퓨터 저장시스템, IPTV 시스템