클러스터 비디오 서버에서 에너지 감소를 위한 캐싱 기법

이 범 선[1], 송 민 석[2]
[1,2]인하대학교 컴퓨터정보공학과
[1]mrbumm@gmail.com, [2]mssong@inha.ac.kr

An Energy-Aware Caching Scheme for Clustered Video Servers

Bumsun Lee[1], Minseok Song[2]
[1,2]School of Computer Science and Information Engineering, Inha University, Korea

요 약

최근 인터넷의 발달과 더불어, 멀티미디어 네트워크 서비스가 크게 활성화됨에 따라서 해당 정보를 저장하는 저장 장치의 크기가 기하급수적으로 늘고 있으며, 서버에서의 전력 소모 문제가 큰 이슈로 대두되었다. 서버 구성 요소 중에 디스크와 같은 저장장치가 전력 소모에 큰 부분을 차지하고 있으며, 이를 감소시키기 위해 디스크는 여러 모드를 지원하며, 그 중 저전력 모드에서 소비되는 전력이 다른 모드에 비해서 훨씬 적다. 본 논문에서는 클러스터 비디오 서버에서 최대한 많은 디스크를 저전력 모드로 동작하게 하는 캐싱(caching) 기법을 제안한다. 제안하는 기법은 클러스터 별로 캐쉬를 할당하며, 할당된 캐쉬 크기에 따라서 각 클러스터에서 소모되는 디스크 이용률과 전력을 분석한다. 이에 기반하여, 전체 클러스터에서 소모되는 전력을 최소화하는 새로운 캐싱 알고리즘을 제안하며, 시뮬레이션을 통해 해당 기법의 효용성을 분석한다.

1. 서 론

최근 멀티미디어 및 네트워크 기술의 발달로 인해, 디지털 도서관, 주문형 교육, 원격 교육 및 주문형 비디오와 같은 다양한 응용에서 비디오 서비스를 제공하는 것이 중요하게 되었다. 비디오 데이터는 높은 대역폭 및 큰 저장 공간을 요구하기 때문에, 비디오 서버는 일반적으로 수백 개의 디스크 배열 (disk array) 로 구성된 디스크 클러스터링 기법을 사용한다.

최근 서버에서의 전력 소모 감소에 관한 이슈가 큰 관심을 끌기 시작하였다. 최근 Energy User News [8]에서, 일반적인 서비스 공급자들은 현재 150-200 W/ft^{2}의 전력을 필요로 하고 있으며, 가까운 미래에는 200-300 W/ft^{2}까지 필요로 하게 될 것이라고 보고하고 있다. 이와 같이 증가하고 있는 전력 수요는, 서비스 공급자들에게 있어서 심각한 경제적 문제를 초래한다. 예를 들어, 중간 크기인 30,000 ft^2 데이터 센터는 15MW를 필요로 하는데, 이는 연간 $13,000,000의 비용에 해당한다[2,3]. 소비전력 증가와 관련된 또 다른 문제는 열(heat)의 발생이다 [1,7]. 예를 들면, 주위 온도가 15℃ 높게 동작시킬 경우, 디스크 드라이브 고장률(failure rate)이 2배로 높아질 수 있는 것으로 알려져 있다 [3]. 디스크의 열 문제를 해결하기 위해 필요한 냉각 시스템은 엄청나게 고가이며, 냉각 시스템 사용으로 인한 소비전력 증가는 불가피하다. 특히 최근 인터넷에서 비디오 트래픽의 양이 기하급수적으로 증가하고 있으므로, 비디오 서버를 위한 전력 감소가 중요하게 대두되었다.

서버 구성 요소 중에 디스크와 같은 저장장치가 전력 소모에 큰 부분을 차지한다. 최근 분석에 의하면, 디스크가 전체 전기 비용의 27%를 차지한다고 조사되었다.[3,9,8,10]. 특히 최근에는 점차 많은 데이터를 저장하기 위해서 고성능 디스크를 사용하므로, 저장 장치에서 전력 소모를 줄이는 것이 매우 중요하다. 디스크에서 소모되는 전력을 줄이기 위해서 디스크는 최대 속도로 돌면서 데이터를 읽고, 쓰고, 탐색(seek) 하는 활동(active) 모드, 최대 속도로 돌지만, 실질적인 디스크 읽기, 쓰기, 탐색 동작이 없는 유휴 (idle) 모드, 회전 동작이 완전히 멈추어서 전력이 다른 두 가지 모드보다 훨씬 적게 소모되는 저전력(low-power) 모드의 세 가지 모드[1,2,3]를 지원한다.

본 논문에서는 클러스터 비디오 서버에서 최대한 많은 디스크를 저전력 모드로 동작하게 하는 캐싱(caching) 기법을 제안한다. 클러스터 별로 메모리를 따로 할당하며, 할당된 메모리 크기에 따라서 각 클러스터에서 소모되는 디스크 이용률과 전력의 상관관계를 분석한다. 이를 바탕으로, 전체 클러스터에서 소모되는 전력을 최소화하는 모델을 제시하고 이를 위한 캐싱 알고리즘을 제안한다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 본 연구의 배경 지식을 설명한다. 3장에서 새로운 캐싱 기법을 제안하고, 4장에서 시뮬레이션을 통해서 기존 연구와 제안한 기법을 비교한다. 마지막으로 5장에서 결론을 맺는다.

2. 배 경

2.1 시스템 모델

멀티미디어 서버는 일반적으로 버퍼 관리자(Buffer manager), 입/출력 관리자(I/O manager), 네트워크 관리자(network manager)의 세 가지 기능적 요소로 구성되어 있다. 버퍼 관리자는 메모리 버퍼를 선반입 버퍼(read-ahead buffer)와 캐쉬로 나눈다. 선반입 버퍼는 재사용할 가능성이 높은 오브젝트를 캐쉬에

	cluster 1				cluster 2			
	D_1^1	D_1^2	D_1^3	D_1^4	D_2^1	D_2^2	D_2^3	D_2^4
원본 (primary copy)	$S_{i,1}$	$S_{i,2}$	$S_{i,3}$	$S_{i,4}$	$S_{m,1}$	$S_{m,2}$	$S_{m,3}$	$S_{m,4}$
	$S_{i,5}$	$S_{i,6}$	$S_{i,7}$	$S_{i,8}$	$S_{m,5}$	$S_{m,6}$	$S_{m,7}$	$S_{m,8}$
	...	...	...	...	...	...	...	...
백업본 (backup copy)	$S_{i,4}$	$S_{i,1}$	$S_{i,2}$	$S_{i,3}$	$S_{m,4}$	$S_{m,1}$	$S_{m,2}$	$S_{m,3}$
	$S_{i,8}$	$S_{i,5}$	$S_{i,6}$	$S_{i,7}$	$S_{m,8}$	$S_{m,5}$	$S_{m,6}$	$S_{m,7}$
	...	...	...	...	...	...	...	...

<그림 1> chained declustering을 이용한 Q = 4인 클러스터 비디오 서버에서의 데이터 배치의 예

저장하는 동안에 디스크로부터 읽혀진 데이터와 클라이언트에게 즉시 전송될 데이터를 저장한다. 클라이언트가 비디오를 요청하면, 서버는 일단 캐쉬를 체크한다. 만약 요청된 데이터가 캐쉬에 있다면, 데이터는 네트워크 관리자에 의해서 클라이언트에게 직접 전송된다. 그렇지 않으면, 입/출력 관리자는 디스크로부터 데이터를 읽어서 데이터가 네크워크로 전송될 때까지 버퍼에 데이터를 임시 저장한다.

비디오 서버에서는 비디오 데이터의 주기적 성질을 이용한 라운드 기반 스케줄링(round-based scheduling)을 사용한다[4,5]. 전체 시간을 고정된 크기의 라운드 주기로 분할하며, 연결된 각 클라이언트는 각 라운드에 한 번씩 실시간 재생에 필요한 데이터 크기만큼 데이터를 받아서 재생한다. 예를 들면, 1.5Mb/sec의 재생률을 갖는 비디오 데이터의 경우, 라운드 길이가 1초라면, 매 초마다 1.5Mb를 지속적으로 디스크에서 읽어서 서비스한다. 한 개의 디스크의 전송률은 한 개의 비디오 스트림의 재생률보다 훨씬 높기 때문에, 비디오 서버의 각 디스크는 동시에 여러 스트림을 서비스하는 것이 가능하다.

디스크의 대역폭을 효율적으로 이용하기 위해, 한 개의 비디오 개체는 세그먼트들로 나누어지고, 여러 디스크에 스트라이핑(striping)되어 저장된다. 세그먼트는 한 개의 디스크에 저장되어진 데이터의 최대 크기를 나타낸다. 시스템의 용이한 확장을 위해서 디스크 배열은 여러 개의 클러스터에 나누어져 있고, 하나의 비디오는 하나의 클러스터 내에서 스트라이핑 된다[7]. 본 논문에서 세그먼트의 크기는 한 라운드에서 읽는 데이터의 크기로 정의한다. 한 개의 세그먼트는 디스크에 연속적으로 저장되었으므로, 한 라운드에 한 개의 클라이언트를 위해서 한 번의 탐색(seek)이 요구된다.

디스크의 고장에 강인하기 위해서, 본 연구에서는 복제 기법을 사용한다. 복제 기법에서 오리지널 데이터는 같은 클러스터의 다른 디스크에 복사되어 있고, 오리지널 데이터를 원본(primary copy), 복사된 데이터를 백업본(backup copy)라 한다. 복제 기법에서 chained declustering(CD) 기법이 주로 사용된다[6]. C개의 클러스터로 이루어진 시스템에서 각 클러스터는 동종의 디스크 Q개로 구성되어 있다고 가정하자. CD 기법에서는 클러스터 k의 j 번째 디스크 D_k^j의 원본은 $D_k^{(j+1) \bmod Q}$에 백업본을 가지고 있을 것이다. 본 논문의 CD 기법에서는 백업본을 한 디스크에 위치시킨다. N개의 비디오가 저장되어 있고, 각 비디오 V_i는 제한된 개수의 세그먼트($S_{i,1}$, $S_{i,2}$, ...)로 나누어져 있다고 가정하면, CD 기법을 이용한 데이터의 배치는 <그림 1> 과 같이 표현된다.

디스크는 저전력 모드에 있을 경우 전력을 적게 소모하므로, 디스크가 저전력 모드에 있는 시간을 가능한 최대화하는 것이 중요하다. 그러나, 디스크가 대기 모드에서 활동 모드로 전환하는데 디스크를 다시 스핀업(spinup) 해야 하며 이는 오랜 시간과 에너지를 필요하기 때문에, 오히려 전력 소모를 증가시킬 수 있다. 대기 모드로의 전환이 전력 소모를 줄일 수 있는 최소의 유휴(idle) 시간을 break-even time이라고 하며, 디스크를 접근하지 않는 유휴시간은 break-even time 보다 커야 한다. 그러나 비디오 데이터는 실시간성을 요구하기 때문에 지속적으로 데이터를 읽어야 하므로, 디스크는 저전력 모드에 들어갈 기회가 없게 된다.

비디오서버에서 디스크의 저전력 모드 진입을 허용하기 위해서, 복제 데이터를 이용한 기법이 연구되었다[11]. 디스크 D_k^j의 데이터가 백업카피 $D_k^{(j+1) \bmod Q}$에 복제되어 있기 때문에, D_k^{2j}에서 데이터를 읽는 대신, $D_k^{(2j+1) \bmod Q}$ (j = 1, ..., $\left\lfloor \frac{1}{2}Q \right\rfloor$)의 백업본에서 데이터를 대신 읽을 수 있다. 이런 방법을 사용할 경우, D_k^{2j}는 전혀 접근되지 않을 수 있기 때문에, D_k^{2j}는 저전력 모드로 진입할 수 있다. 이를 이용하여, 본 연구에서는 두 가지 상태를 정의한다. 원본 (k = 1, ..., C and j =

1, …, Q)로 부터 읽는 상황에서, 클러스터 k의 j번째 디스크 D_k^j의 디스크 대역폭 이용도를 DU_k^j라 하자. 만약 $\forall j$, $DU_k^j \leq 0.5$ 이면, 클러스터 k는 에너지 감소(energy reduction(ER)) 상태에 있고, 그렇지 않으면 클러스터 k는 정상(normal) 상태에 있다고 정의한다. 정상 상태에서는 디스크 이용률이 0.5보다 크므로, 홀수 번째 디스크의 부하가 짝수 번째 디스크로 이동될 경우, 짝수 번째 디스크의 이용률이 1을 넘을 수 있다. 따라서 정상 상태에서는 부하 이동이 불가하다. 그러나 ER 상태에서 부하가 이동된다 하더라도, 디스크 이용률이 1을 넘지 않으므로, 짝수 번째 디스크는 저전력 모드로의 진입이 가능하다. 디스크의 대역폭 이용도를 0.5이하로 줄이는 것은 ER 상태로의 진입을 의미하는 것이기 때문에 에너지를 줄이는데 매우 중요하다. 본 연구에서는 이와 같은 방법을 이용하여 디스크가 저전력 모드로 진입하도록 허용한다.

2.2 인터벌 캐싱 (Interval Caching)

멀티미디어 스트리밍 서버에서의 캐싱 기법은 서버 시스템의 성능 향상과 사용자의 서비스 대기 시간을 줄이는 효과적인 방법이다. 멀티미디어 객체는 크기가 크고 순차적으로 참조되므로 LRU(Least Recently Used) 알고리즘과 같은 전통적인 버퍼 캐쉬 관리 기법을 멀티미디어 서버에 그대로 적용하는 것은 효율적이지 못하다. 이러한 문제를 해결하기 위해 기존에 많은 연구들이 수행되었고, 그 대표적인 버퍼 관리 기법이 스트리밍 서비스의 요청 간격에 기반 한 인터벌 캐싱(IC) 기법이다[12,13]. 인터벌 캐싱 기법은 같은 비디오 개체에 대한 두 개의 연속적인 요청이 있을 때, 이들 요청 사이의 간격을 인터벌(interval)이라 정의하고, 만약 먼저 도착한 요청이 디스크로부터 읽은 데이터를 캐쉬에 저장해 두었다면, 나중에 도착한 요청은 디스크 I/O 없이 캐쉬로부터 데이터를 직접 서비스해 주는 방법이다. 인터벌 캐싱 기법은 캐쉬로부터 데이터를 얻는 요청의 개수를 최대화 시킨다. 이를 위해 인터벌의 크기로 요청들을 정렬하고, 캐쉬 공간이 존재하는 한 가장 작은 인터벌을 가진 요청부터 캐쉬에 저장한다.

인터벌 캐싱 기법은 짧은 기간 내의 두 연속적인 요청에 대한 시간 지역성(temporal locality)만을 고려하였다. 하지만, 멀티미디어 스트리밍 서비스가 서버환경에서 동작하기 때문에, 서버에서의 중요한 문제인 전력 소비문제를 고려하지 않았다는 것은 큰 약점이 될 것이다. 이러한 문제를 해결하기 위해 본 연구에서는 서버 환경에서 클러스터에서 소모되는 전력을 최소화하는 모델을 제시하고 이를 위한 캐싱 알고리즘을 제안한다.

3. 에너지 인지 인터벌 캐싱 기법

3.1 모델링

본 논문의 주목적은 인터벌을 적절히 캐싱시켜 전력 소비를 최소화하는데 있다. 즉, 각 클러스터마다 인터벌 큐를 관리하고, 클러스터의 디스크 대역폭 이용도를 조절하여 최대한 많은 클러스터를 ER 상태로 만드는 것이다. 그러나 이 방법은 더 큰 인터벌을 캐싱하여, 디스크 입/출력의 개수를 늘어나게 할 수 있으므로, 현명한 캐싱 기법이 필요하다.

클라이언트가 비디오 스트림을 요청하거나, 서비스 중지를 했을 때, 서버는 그 인터벌을 캐싱할 것인지 아닌지를 결정해야 한다. 각 비디오 개체에 대한 요청이 도착했을 때, 이전 비디오 스트림은 캐쉬에 데이터 저장을 시작하기 때문에 요청이 도착하자마자 캐쉬로부터 데이터를 얻어올 수 없고, 일단 디스크로부터 데이터를 얻어야 한다. 따라서 각 비디오 스트림은 다음의 3가지 상태에 있을 수 있다.

- ICM : 인터벌이 캐싱되어 있고, 데이터가 캐쉬로부터 서비스된다. 이 상태는 디스크 입/출력이 없다.
- ICD : 인터벌이 캐싱되어져 있지만, 아직 디스크로부터 데이터를 읽는다. 새로운 인터벌이 도착했을 때 스트림은 캐싱되기 시작하고, 그 스트림은 ICD 상태가 된다. 스트림이 충분히 캐싱되면(인터벌 만큼의 시간이 지나면) 그 스트림은 ICM 상태가 된다.
- INS : 인터벌이 캐싱되어 있지 않고, 디스크로부터 데이터를 읽기 위해 디스크 I/O가 필요하다.

스트림은 위와 같은 세 가지 상태로 상황에 따라 변할 수 있다. 예를 들어, ICM 상태에 있는 스트림은 이전 스트림이 캐쉬로부터 제거되어졌을 때, INS 상태로 바뀔 수 있다. 또한 한 클라이언트가 종료되어 캐쉬 공간이 확보되었을 때, INS 상태의 스트림은 ICD 상태가 될 수 있다.

현재 V_i에 대한 요청의 개수를 NA_i, 현재 생성된 비디오 i에 대한 j번째 인터벌의 크기를 IS_i^j라 하면(j = 1, …, NA_i-1), 클러스터 k의 인터벌들의 집합은 SI_k, SI_k = {0, 클러스터 k의 저장된 모든 비디오 i에 대한 IS_i^j, (j = 1, …, NA_i-1)} (k = 1, …, Q) 로 나타낼 수 있다. 여기에 어떤 인터벌도 캐싱되지 않은 상태 0을 추가한다. 한 클라이언트가 요청 또는 종료되었을 때, 서버는 SI_k의 원소를 인터벌 크기에 대한 오름차순으로 정렬한다. SI_k의 m번째 원소를 IL_k라 하자. SI_k의 첫 번째 원소는 0이고, m번째 원소는 m-1번째로 인터벌이 작다(m > 2). SI_k의 원소의 개수를 NE_k라 하자.

서버는 SI_k의 첫 번째 원소부터 m번째 원소까지 캐싱했을 때(m = 1, …, NE_k-1)[1], 클러스터 k(k = 1, …, Q), $CU_k(m)$ 에 대한 디스크 대역폭 이용도의 집합을 유지한다. 클러스터 k에서 SI_k의 첫 번째 원소부터

[1] SI_k에서 첫 번째 원소가 캐쉬되어졌을 때, 그 상태는 ICD가 되므로 실제 캐쉬된 것은 아무것도 없음.

m번째 원소까지의 인터벌이 캐슁되었을 때의 ICM, ICD, INS 각각의 개수를 $NM_k(m)$, $ND_k(m)$, $NS_k(m)$라 하자. 서버는 매 라운드마다 각 스트림의 상태를 체크하여 이 값을 갱신하는데, 이 라운드를 R이라 한다. 모든 스트림의 지속적인 재생을 보장하기 위해 스트림 정보를 얻어오는데 걸리는 전체 시간은 그 라운드만큼의 시간을 넘을 수 없다. 디스크 대역폭 이용도는 보통 라운드 길이[4]당 전체 서비스 시간의 비율로서 정의된다. 편의를 위해 본 논문에서는 모든 비디오 스트림들은 같은 재생률(display rate)인 dr bits/sec를 갖는다고 가정한다. 하지만 다른 재생률을 갖는 시스템에도 우리의 연구는 쉽게 적용 가능하다 [5]. 디스크의 일반적인 탐색 시간을 T_s, 연속적인 데이터를 한번 읽는데 요구되는 회전 지연 시간을 T_d라 하자 [4]. 디스크에서 tr이 디스크의 데이터 전송률일 때, 하나의 비디오 스트림을 얻는데 걸리는 오버헤드는 T_s+T_d 와 읽는 시간 $R\times\frac{dr}{tr}$이다. 따라서 비디오 스트림을 서비스하는 시간은 $T_s + T_d + R\times\frac{dr}{tr}$ 만큼 증가한다. 각 클러스터는 Q개의 디스크를 가지고 있기 때문에, $ND_k(m)$ + $NS_k(m)$의 디스크 I/O가 필요하며, $CU_k(m)$는 다음의 식으로 정의 될 수 있다[2].

$$CU_k(m) = (ND_k(m)+NS_k(m)) \times \frac{T_s + T_d + R\times\frac{dr}{tr}}{R\times Q} \quad (1)$$

탐색하는데 요구되는 전력을 P_s, 데이터를 읽고 쓰는데 요구되는 전력을 P_a, 아이들 모드에서 소비되는 전력을 P_i, 저전력 모드에서 소비되는 전력을 P_l이라 하자. 이제 클러스터 k에서 SI_k의 첫 번째부터 m번째 원소까지 캐슁 되어졌을 때의 에너지 사용량을 구하는 공식을 얻을 수 있다.

1) R동안의 전체 탐색 시간은 $(ND_k(m)+NS_k(m)) \times T_s$ 이다. 따라서 R 동안 탐색하는데 소비되는 전력 $E_k^s(m)$은 $(ND_k(m)+NS_k(m)) \times T_s \times R$ 이다.

2) R동안 읽거나 쓰는데 소비되는 전력 $E_k^a(m)$은 $(ND_k(m)+NS_k(m)) \times (R\times\frac{dr}{tr}) \times P_a$ 이다.

3) 만약 디스크 동작이 전혀 없거나, 디스크의 헤드가 섹터에 도착하기를 기다린다면 소비되는 전력은 서버가 디스크의 상태가 정상인지 ER 상태인지에 따라 다른 방법으로 계산해야한다.

- 정상 상태에서는 저전력 모드로 진입할 수 있는 디스크가 없기 때문에, R 동안의 소비전력 $E_k^n(m)$은 다음과 같이 표현된다.

$$E_k^n(m) = P_i \times (Q\times R - ND_k(m) + NS_k(m)) \times (T_s + R\times\frac{dr}{tr}))$$

- ER 상태에서는 $\left\lfloor \frac{1}{2}Q \right\rfloor$ 의 디스크는 저전력 모드로 진입할 수 있기 때문에, R 동안의 소비전력 $E_k^e(m)$은 다음과 같이 표현된다.

$$E_k^n(m) = P_i \times (\left\lceil \frac{1}{2}Q \right\rceil \times R - (ND_k(m) + NS_k(m)) \times (T_s + R\times\frac{dr}{tr})) + P_i \times \left\lfloor \frac{1}{2}Q \right\rfloor$$

만약, $CU_k(m) > 0.5$이면 클러스터 k는 정상 상태이고, 아니라면 ER 상태다. 위에 표현된 공식으로 SI_k의 첫 번째부터 m번째 원소의 인터벌을 캐쉬에 저장했을 때의 $E_k(m)$은 다음과 같이 나타낼 수 있다.

$$E_k(m)\begin{cases} E_k^s(m)+E_k^a(m)+E_k^n(m) & \text{if } CU_k(m) > 0.5 \\ E_k^s(m)+E_k^a(m)+E_k^e(m) & \text{if } CU_k(m) \le 0.5 \end{cases}$$

3.2 에너지 인지 캐슁 알고리즘

본 연구는 에너지 소비를 최소화시키기 위해 각 클러스터의 캐슁된 인터벌의 개수를 동적으로 결정한다. 하지만 디스크 대역폭 이용도의 제약인 $CU_k(m) \le 1$을 만족해야 한다. 한 클라이언트가 비디오 스트림을 요청했을 때, 서버는 이 상태를 체크한다. $CU_k(m) \le 1$, $(m = 1, \cdots, NE_k)$을 만족하는 m의 가장 작은 값을 SV_k라 하자. 만약 $CU_k(NE_k) > 1$ 이면, 이후의 클러스터 k에 대한 요청은 거부될 것이다. 전체 캐쉬의 크기를 B라 하자. 만약 $\sum_{k=1}^{C}\sum_{m=1}^{SV_k} IL_k^m \le 1$ 라면, 캐쉬 공간이 부족하기 때문에 새로운 클라이언트 역시 거부될 것이다.

SP_k가 선택된 변수라면, 클러스터 k에서 SI_k의 첫 번째부터 SP_k번째까지의 인터벌은 캐슁된다. $ES_k(m)$은 $SP_k = m$일 때, SP_k로서 SV_k가 선택된 경우에 에너지 소비의 감소를 나타내며, $ES_k(m) = E_k(m) - E_k(SV_k)$ 이다. 따라서 $E_k(SV_k) = 0$가 된다. SP_k의 값이 증가하면, 디스크 대역폭 이용도는 감소하고, 만약 $m > n$ 이면 $ES_k(m) > ES_k(n)$ 다.

본 연구의 목적은 캐쉬 크기 제한 $\sum_{k=1}^{C}\sum_{m=1}^{SP_k} IL_k^m \le 1$를 초과하지 않는 범위 안에서 라운드 길이 R동안의 에너지 소비 감소량을 최대화하는 것이다. 아래의 정의 1은 앞에서 언급한 내용을 구체화시켜 인터벌 길이 결정 문제로 정의한 것이다.

[2] 편의를 위해 부하는 디스크 전체에 고르게 분포되어 있다고 가정했다.

알고리즘 1 인터벌 길이 결정 알고리즘 (Interval Length Determination Algorithm)

1: Set of $IF_k(n)$'s : SIF;
2: Temporary variable for storage usage : TS;
3: $TS \leftarrow \sum_{i=1}^{C} \sum_{m=1}^{SV_k} IL_k^m$;
4: $SP_k \leftarrow SV_k$;
5: **while** $TS \leq B$ and $SD \neq \Phi$ **do**
6: Find the highest value, $IF_k(h) \in SIF$;
7: $SIF \leftarrow SIF - IF_k(h)$;
8: **if** $h > SP_k$ **then**
9: $TS \leftarrow TS - \sum_{m=1}^{SP_k} IL_k^m + \sum_{m=1}^{h} IL_k^m$;
10: $SP_k \leftarrow h$;
11: **end if**
12: **end while**

정의 1 : 인터벌 길이 결정 문제 (Interval Length Determination Problem (ILDP))
각 라운드에서 모든 클러스터 k에 대해 $\sum_{k=1}^{C} \sum_{m=1}^{SP_k} IL_k^m \leq B$ 이고, $\sum_{k=1}^{C} ES_k(SP_k)$가 최대인 SP_k를 찾는다.

ILDP에서 각 개체는 제한된 아이템들의 집합을 갖고 각 개체에서 하나의 아이템을 선택하여 전체 이익을 최대화하는 multiple-choice knapsack problem의 변형이다. 특히, 각 클라이언트의 상태가 ICM, ICD, INS로 동적으로 변할 수 있기 때문에, 최적의 해결책을 찾는 것은 불가능하므로, 휴리스틱 알고리즘을 제안한다.

클러스터 k에 대하여 $IF_k(n)$, $(n = SV_k+1, \cdots, NE_k)$은 변수의 집합으로 정의하며, 다음과 같이 계산된다.

$$IF_k(n) = \frac{ES_k(n) - ES_k(SV_k)}{\sum_{m=1}^{n} IL_k^m - \sum_{m=1}^{SV_k} IL_k^m}$$

위 공식은 클러스터 k에 대한 선택된 변수 SP_k가 SV_k부터 n까지 증가할 때까지의, $IF_k(SP_k) \sim IF_k(n)$ 값을 구하며, 분모는 필요한 버퍼의 크기를, 분자는 감소되는 전력 소비량을 나타낸다. 이에 기반 하여, 알고리즘 1과 같이 인터벌 길이 결정 알고리즘 (Interval Length Determination Algorithm (ILDA)) 로 기술된 휴리스틱 알고리즘을 제안한다 .

4. 실험 결과

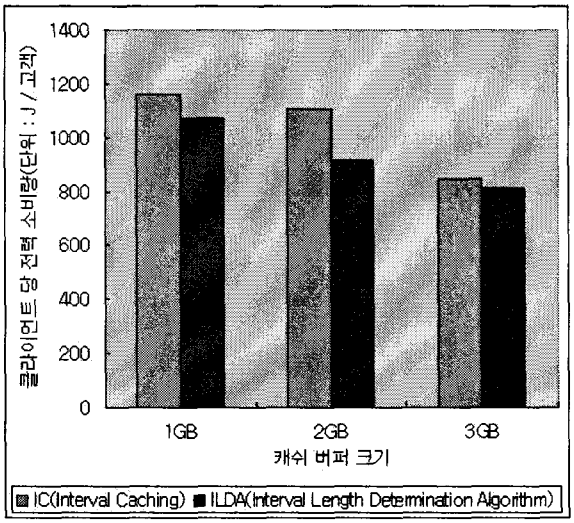
IC과 ILDA의 소비 전력을 비교하기 위해 tr(transfer rate) : 50MB/s, T_s(seek time) : 10ms, T_d (rotational delay) : 4ms 의 성능을 갖는 가상의 디스크를 가정하였다. 비디오 서버는 8개의 클러스터를 보유하고 있으며, 각 클러스터는 4개의 디스크로 구성되어 있다. 사용자 요청의 도착 시간은 Poisson 분포를 사용하였으며, 각 비디오에 대한 선호도는 α = 0.271을 갖는 Zipf 분포를 따른다. 모든 비디오는 90분 길이이며, 50개의 영화가 각각의 클러스터에 분산되어 있으며, R은 1초로 가정하였다. 각 모드에서 소비되는 디스크의 전력은 활성 모드에서 13.5W, 아이들 모드에서 10.2W, 저전력 모드에서 2.5 W이고, 측정 시간은 24시간이다.

<그림 2>는 사용자 요청의 평균 도착 시간은 3초이며, 비트 전송률은 800Kb/sec인 상황에서 캐쉬 버퍼 크기의 변화에 따른 IC와 ILDA의 전체 전력 사용량을 나타내고 있다. 캐쉬 버퍼가 2G인 경우의 전력 감소량이 약 17.6%로 가장 높게 나타났으며, 이때 IC의 각 클러스터의 대역폭 이용도의 분포는 50%~60%로 모든 클러스터가 활성모드에서 동작하였다. 반면 ILDA 각 클러스터의 캐쉬 버퍼를 조절하여 최대한 많은 클러스터를 저전력 모드로 진입시켜 소비전력을 최소화 시켰다.

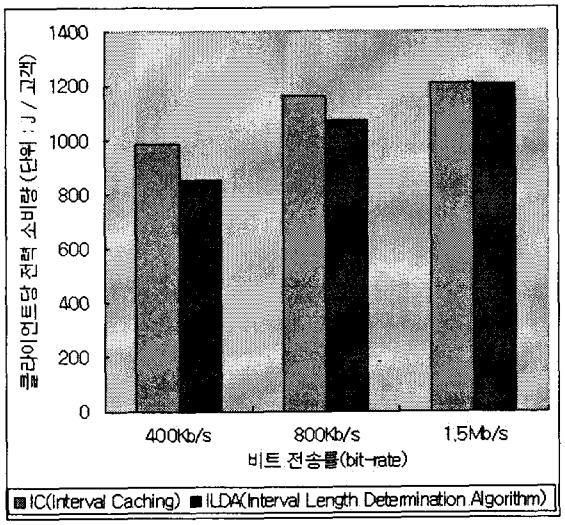
<그림 3>은 앞의 실험과 동일한 조건에서 캐쉬 버퍼 크기를 1G로 고정하고, 비트 전송률(bit-rate)을 바꾸어서 측정한 실험 결과다. 비트 전송률이 800kb일 때의 전력 감소율이 약 13.7%로 가장 크게 나타났으며, 이것은 앞의 실험과 같은 이유로 나타난 결과이다.

실험 결과 전체 클러스터의 대역폭 이용도의 분포가 50~80%에서 약 1%~18%의 성능 향상을 보였으며, 이것은 본 연구에서 제안한 ILDA 기법이 멀티미디어 서버 환경에서 전력 소비량을 줄일 수 있는 효율적인 방법임을 나타내고 있다.

5. 결 론



<그림 2> 캐쉬 버퍼 크기에 따른 IC와 ILDA의 사용 전력 (24 시간)



<그림 2> 비트 전송률(bit-rate)에 따른 IC와 ILDA의 사용 전력 (24 시간)

본 논문에서는 비디오 서버에서, 최대한 많은 디스크를 저전력 모드로 동작하게 하는 동적 캐쉬 방법을 제안하였다. 클러스터 별로 메모리를 따로 할당하며, 할당된 메모리 크기에 따라서 각 클러스터에서 소모되는 디스크 이용률과 전력의 상관관계를 모델링하였으며, 이를 바탕으로, 전체 클러스터에서 소모되는 전력을 최소화 하는 캐쉬 알고리즘을 제안했다. 시뮬레이션을 통한 실험 결과는 제안한 기법이 디스크에서 소모되는 전력 소비량을 감소시킬 수 있음을 보여준다.

6. 참 고 문 헌

[1] S. Gurumurthi. "Power management of enterprise storage systems," Ph.D. dissertation, Pennsylvania State University, 2005.

[2] Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes. "Hibernator: helping disk arrays sleep through the winter," *ACM Operating Systems Review*, vol. 39, no. 5, pp. 177--190, 2005.

[3] Q. Zhu and Y. Zhou. "Power aware storage cache management," *IEEE Transactions on Computers*, vol. 54, no. 5, pp. 587--602, 2005.

[4] E. Chang., "Storage and retrieval of compressed video," Ph.D. dissertation, University of California at Berkeley, 1996.

[5] E. Chang and H. Garcia-Molina, "Effective memory use in a medis server." in *Proceedings of VLDB Conference*, Aug. 1997, pp. 496--505.

[6] M. Song and H. Shin. "Replication and retrieval strategies for resource-effective admission control in multi-resolution video servers." *Multimedia Tools and Applications Journal*, vol. 28, no. 3, pp. 89--114, Mar. 2006.

[7] L. Golubchik, J. Lui, and M. Papadopouli. "A survey of approaches to fault tolerant vod storage servers: Techniques, analysis, and comparison.", *Parallel Computing*, 24(1):123-155. january 1998.

[8] B. Moore. "Taking the data center power and cooling challenge." *Energy User News*, 27, August 2002.

[9] A. Dan, D. Sitaram, and P. Shahabuddin. "Dynamic batching policies for an on-demand video sever." *ACM/Springer multimedia Systems Journal*, 4(3):112-121, 1996.

[10] E. Pinheiro and R. Bianchini. "Energy conservation techniques for disk-array-based servers." In *Proceedings of the ACM/IEEE Conference on Supercomputing*, pages 88-95, june 2004.

[11] M. Song, "Dynamic Buffer Allocation for Conserving Disk Energy in Clustered Video Servers Which Use Replication." *Embedded and Ubiquitous Computing*, LNCS 4096, pp 193-203, August. 2006.

[12] A. Dan and D. Sitaram, "Buffer Management Policy for an On-Demand Video Server." *IBM Research Report RC19347*, T.J. Watson Research Center, Yorktown Heights, NY.

[13] A. Dan and D. Sitaram, "A Generalized Interval Caching Policy for Mixed Interactive and Long Video Environments." *Proceedings of SPIE Multimedia Computing and Networking Conference*, San Jose, CA, 1996.