패칭 기법을 이용한 프락시 관리 정책에 기반한 연속형 미디어 스트림 서비스

백 건 효[†] · 박 용 운[††] · 정 기 동[†††]

요 약

연속형 미디어는 대용량이고 실시간으로 전송되어야 하므로 전송에 요구되는 네트워크 자원에 부하를 준다. 일반적으로 프락시는 자주 참조되는 객체를 캐싱하여 서버로의 네트워크 전송 대역폭을 줄이기 위한 기법으로 사용되고 있으나 기존의 연구들이 이미지나 텍스트 등의 비 실시간성 객체 지향적으로 설계되어 실시간성 객체의 서비스에는 적합하지 않다. 그러므로 본 논문에서는 패칭 기법[10]을 사용한 프락시 관리 기법을 사용하여 연속형 스트림 서비스를 하는 스트림 서비스 기법을 제안한다. 제안한 기법에서는 프락시에 캐싱된 데이터의 양에 따라 스트림 전송 방식을 달리한다. 첫째, 요청된 객체 전체가 캐싱되어 있을 경우 프락시 만으로 서비스 한다. 둘째, 요청된 객체가 전혀 캐싱되어있지 않을 경우 후행 스트림들이 서버로부터 객체를 전송할 때 발생하는 초기 지연을 상쇄할 만큼의 데이터를 선반입한다. 셋째, 일부분 만이 캐싱된 경우에는 해당 객체를 요청하는 스트림 사이에 존재하는 양만큼의 데이터를 프락시에 패칭하며 이 경우에는 사용자 노드는 두 개의 채널을 열어 하나는 프락시에 패칭된 데이터를 읽는데 사용하며 또 하나의 채널로는 서버로 부터 나머지 부분을 읽어오는데 사용한다.

Continuous Media Stream Service Using Proxy Caching Based on Patching Scheme

Keon-Hyo Baek[†] · Yong-Woon Park[††] · Ki-Dong Chung[†††]

ABSTRACT

The continuous media objects, due to large volume and real-time constraints in their delivery, are likely to consume much network bandwidth. Generally, proxy servers are used to hold the frequently requested objects so as to reduce network traffic for the central server but most of them are designed for text and image data ; they do not go in harmony with continuous media objects.

Therefore, in this paper, to resolve this problem, we propose a stream service mechanism using patch-based proxy caching policy. With the proposed stream service mechanism, the service mode of a client depends on the amount of cached portion of the requested object. First, a client is serviced fully from the proxy server if the requested object is cached fully in it. Second, only the initial portion of the requested object is cached to hide the initial latency for succeeding streams accessing the same object if it is not cached at all. Third, if the requested object is cached partially, the un-cached part of the requested object is cached step by step as a function of time discrepancy among streams if more than two concurrent streams exist. In this case, the client is assigned two channels ; the one is to read the cached data from the proxy server and the other is to read and then load the remaining part, i.e., the un-cached portion of the requested object from the central server.

키워드 : 프락시 캐싱, 연속형 미디어 스트림, 패칭

1. 서 론

프락시 또는 네트워크 캐쉬는 사용자와 서버의 중간 지점에 캐쉬를 설치함으로써 중앙 서버로의 네트워크 교통량을 줄이기 위한 목적으로 소개되었다[1, 8, 12]. 그러나 기존의 프락시는 주로 텍스트, 이미지 등의 기존의 데이터의 캐싱에 적합하도록 설계되었으며 캐싱 정책도 파일 서버의 캐싱 정책을 그대로 적용하였으므로 대용량의 저장 공간을 차지하고 실시간으로 전송되어야 하는 연속형 미디어 오브젝트의 캐싱에는 적합하지 않다[13, 14, 16, 17]. 또한 [7]에 따르면 2005년 경에는 인터넷 상에서 접근되는 데이터의 50%가 오디오 또는 비디오 등의 연속형 미디어 오브젝트가 될 것으로 추정되므로 네트워크 자원의 효과적인 관리를 위해서는 연속형 미디어 오브젝트의 네트워크 캐싱에 관한 연구가 보다 활발히 진행되어야 한다.

그러므로 본 논문에서는 패칭 기법[10]을 사용한 프락시 관리 기법을 사용하여 연속형 스트림 서비스를 하는 스트

† 정 회 원 : 동의공업대학 사무자동화과 교수
†† 준 회 원 : 동의공업대학 전자계산과 교수
††† 종신회원 : 부산대학교 전자계산학과 교수
논문접수 : 2001년 3월 15일, 심사완료 : 2001년 5월 16일

림 서비스 기법을 제안한다. 제안한 기법에서는 프락시에 캐싱된 데이터의 양에 따라 스트림 전송 방식을 달리한다. 첫째, 요청된 객체 전체가 캐싱되어 있을 경우 프락시 만으로 서비스한다. 둘째, 요청된 객체가 전혀 캐싱되어있지 않을 경우 후행 스트림들이 서버로부터 객체를 전송할 때 발생하는 초기 지연을 상쇄할 만큼의 데이터를 선반입한다. 셋째, 일부분 만이 캐싱된 경우에는 해당 객체를 요청하는 스트림 사이에 존재하는 양만큼의 데이터를 프락시에 패칭하며 이 경우에는 사용자 노드는 두 채널을 열어 하나는 프락시에 패칭된 데이터를 읽는데 사용하며 또 하나의 채널로는 서버로 부터 나머지 부분을 읽어오는데 사용한다. 또한 공간 재 할당 정책으로는 기존의 캐쉬 공간을 새로이 요청된 오브젝트의 캐싱을 위해 무조건 할당하지 않고 조건을 만족하는 경우에만 공간 재할당을 하는 선별적인 공간 재할당 방법을 적용하여 불필요한 공간 재할당을 방지함으로써 캐쉬의 히트율을 높이고 할당 오버헤드를 줄일 수 있다.

2. 관련 연구

일괄 처리[2, 4]는 동일한 오브젝트를 요구하는 사용자들을 그룹으로 묶어서 하나의 스트림으로 그룹에 속한 사용자 모두를 서비스하는 채널 할당 방식이다. 그러나 일괄 처리의 문제는 초기 지연 시간이 길어진다는 것이며 일괄 처리를 위한 주기의 선정은 사용자의 인내 정도와 깊은 관계가 있다. 통상 일괄 처리의 주기를 늘리게 되면 전송 채널을 효율적으로 사용할 수 있는 반면 사용자들의 대기 시간은 그만큼 길어진다. 패칭(patching)[6, 10]은 멀티 캐스팅과 유사하나 기존의 멀티캐스팅 기법과는 달리 멀티캐스팅과 유니 캐스팅(unicasting)을 적절히 혼합하여 스트림을 지원하는 기법이다. 그러나 이 방법은 네트워크 자원의 사용에는 효율적이지만 사용자 노드에 패칭에 사용될 대역폭과 캐싱 공간이 충분히 있어야 하며 또한 멀티캐스팅이 지원되지 않는 환경에서는 적합하지 않다. 또한 패칭이 사용자 노드에서 이루어지므로 패칭된 부분의 재사용이 불가능하며 요청된 오브젝트의 선두 부분은 항상 서버로부터 읽어와야 하므로 초기 지연의 향상에는 도움을 주지 못한다.

네트워크의 병목 현상 해소를 통하여 웹 교통량을 줄이고 서버의 확장성을 향상시키는 또 다른 방안으로써 프락시 캐싱 기법[1, 3, 5, 8, 9]을 들 수 있다. 그러나 지금까지의 프락시 캐싱에 관한 많은 연구가 진행되었음에도 불구하고 대부분의 연구는 주로 이미지나 텍스트 형태의 재래식 파일에 관한 것이 주류를 이루며 연속형 미디어의 캐싱에 관한 연구는 최근에 들어서야 진행되고 있으나 주로 캐싱 정책 전반에 관한 내용보다는 초기 지연을 줄이기 위한 선반입 기준[14, 18], VBR 스트림의 불규칙성(burstiness)을 평활화(bandwidth smoothing)하기 위한 연구[13, 18] 등의 국부적인 주제에 맞추어서 진행되어 왔다.

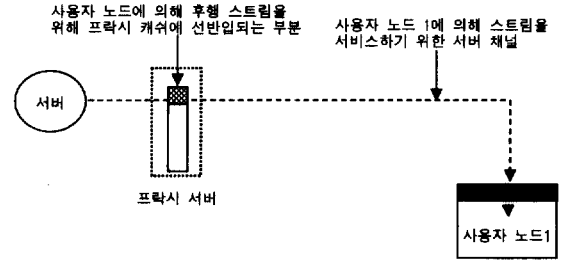
3. 패칭에 기반한 프락시 서버 운영 기법

3.1 프락시 서버를 이용한 패칭의 목적

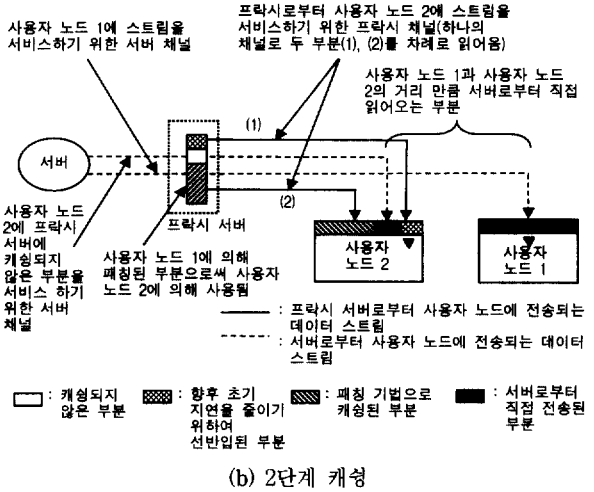
패칭[10]은 데이터 요청이 발생하였을 때 서버 측으로부터 스트림 서비스를 받는 동시에 동일한 데이터를 서비스 받고 있는 스트림의 전송 채널을 공유한 다음 사용자 측 노드의 메모리 또는 디스크 등의 자원을 사용하여 데이터를 캐싱하여 서비스 함으로써 캐싱한 부분 만큼의 서버의 로드를 줄일 수 있다. 그러므로 서버의 채널 사용을 최소화하기에는 적합한 방법이지만 사용자 노드의 자원을 이용하여 데이터를 캐싱함으로써 캐싱된 데이터를 재 사용할 수 없다는 문제점이 있다. 따라서 패칭이 프락시 캐쉬 상에 이루어져야 오브젝트 전송에 필요한 자원 절약 효과를 배가시킬 수 있다.

3.2 프락시를 이용한 패칭의 개념

기존의 네트워크 캐싱에서는 캐싱의 대상이 주로 텍스트나 이미지 등 비교적 크기가 크지 않기 때문에 캐싱 단위가 주로 오브젝트가 된다[21]. 그러나 이미지나 텍스트와는 달리 오디오나 비디오 등의 연속형 미디어 오브젝트의 경



(a) 1단계 캐싱



(b) 2단계 캐싱

(그림 1) 프락시 서버를 이용한 패칭 및 선반입

우 압축된 형태라고 하더라도 대용량의 저장 공간이 필요하다. 따라서 캐싱되는 단위가 오브젝트라면 캐싱할 수 있는 오브젝트의 수가 제한되므로 이로 인한 공간 재할당 오버헤드가 커진다. 이러한 이유로 인하여 멀티미디어 오브젝트의 캐싱에 관한 기존의 연구[14, 18, 19]에서도 캐싱 단위를 오브젝트가 아닌 오브젝트의 일부분 만을 캐싱하는 부분 캐싱 기법을 사용하고 있다. 본 논문에서는 (그림 1)에서 처럼 프락시에 캐싱되는 부분을 두 스트림 사이에 존재하는 분량 만큼으로 두는 패칭 기반의 공간 할당 기법을 사용하여 패칭된 부분을 후행 스트림들이 공유할 수 있도록 한다. 그러나 패칭의 경우 요청된 오브젝트의 선두 부분은 항상 서버로부터 읽어와야 하므로 이러한 문제를 해결하기 위하여 요청된 오브젝트가 전혀 캐싱되어 있지 않은 경우 서버로부터 오브젝트를 전송할 때 발생하는 전송 지연을 상쇄할 만큼의 데이터 분량을 선반입하고 이를 본 논문에서는 전방 분할로 칭한다.

정의 1

모든 오브젝트 O_i는 2개의 논리적 분할 O_i^P로 구성되며 이를 각각 전방 분할(front-end partition) 인 O_i^F와 후방 분할(rear-end partition)인 O_i^R라고 하며 다음과 같이 정의한다.

$$\forall o_i (1 \le i \le n),\ \exists o_i^P (P = F \text{ or } R) \text{ such that}$$

$$1)\ S(o_i^F) = \sum_{j=0}^{m} b_j(o_i),\ S(o_i^R) = \sum_{j=m+1}^{r} b_j(o_i)$$

$$2)\ S(o_i) = S(o_i^F) + S(o_i^R)$$

단, $S(o_i), S(o_i^F), S(o_i^R)$은 각각 o_i, o_i^F, o_i^R의 크기
$b_i(o_i)$는 o_i를 구성하는 디스크 블록 번호 j(전체 디스크 블록의 수는 r)

3.3 프락시를 이용한 패칭 알고리즘

패칭을 이용한 프락시 서버의 운영시 오브젝트의 캐싱 상태는 두 단계(two phase)로 구성된다.

1) 1단계(first phase) : 요청된 오브젝트가 전혀 캐싱되어 있지 않을 때
동일 오브젝트에 대한 후행 스트림이 서버를 접근할 때 발생되는 초기 지연을 줄이기 위하여 오브젝트의 초기 일부분 또는 전방 분할을 캐싱 단위로 하여 선반입(prefetching)한다. 즉, 임의의 오브젝트 o_i에 대하여 오직 하나의 스트림이 존재할 때의 캐싱 단위 $S_c^s(o_i)$는 다음과 같다.

$$S_c^s(o_i) = \begin{cases} 0, & \textit{if already cached} \\ S(o_i^F) \end{cases} \quad (1)$$

2) 2단계(second phase) : 요청된 오브젝트의 캐싱 상태가 1단계를 거친 상태일 때
요청된 오브젝트에 이미 선행 스트림이 존재한다고 하고 계속해서 동일한 오브젝트를 요청하는 새로운 스트림이 발생한다고 하면 선행 스트림과 후행 스트림과의 간격을 측정한 다음 해당 간격에 해당하는 분량의 데이터를 캐싱하며 캐싱의 주체는 선행 스트림이 된다. 이 때 두 스트림이 형성한 간격에 의해 정해진 캐싱되는 분량(선행 스트림은 p이고 후행 스트림은 f)을 $S_c^{M(p,f)}(o_i)$로 표시하고 다음과 같이 결정한다.

$$S_c^{M(p,f)}(o_i) = \begin{cases} S(o_i^R) - S_C^A(o_i)_{L_p(o_i)},\ S(o_i^F) \ge S(o_i^{p-f}) \\ S(o_i) - S(o_i^{p-f}) - S_C^A(o_i)_{L_p(o_i)},\ S(o_i^F) < S(o_i^{p-f}) \end{cases} \quad (2)$$

where $L_p(o_i)$: 선행 스트림의 O_i상의 전송 위치
$S_c^{M(p,f)}(o_i)$: O_i의 캐싱된 부분에서 $L_p(o_i)$의 위치만큼을 뺀 부분
$S(o_i^{p-f})$: O_i의 두 스트림(선행 p, 후행 f) 사이에 존재하는 데이터 량

위와 같이 캐싱이 결정되면 사용자는 주 채널을 사용하여 프락시 서버를 탐색하여 캐싱된 부분을 서비스받는 동시에 프락시 서버에서 발견되지 않는 부분에 대해서는 부 채널로 동시에 서버로부터 직접 다운로드 받아서 주 채널로 서비스가 되지 않는 부분에 대한 서비스를 한다.

3.4 캐쉬 공간의 재 할당

인터넷 상에서의 연속형 미디어 오브젝트 요청의 인기도는 *Zipf* 분포와 유사한 형태를 띄며 오브젝트의 접근 빈도수는 오브젝트의 순위에 비례하며 오브젝트 접근의 시간적 국부성도 관찰되었다[11, 12, 15]. 또한 네트워크 캐싱에서는 공간 할당 단위가 디스크 블록인 기존의 파일 캐싱 정책과는 달리 블록이 아닌 오브젝트 전체가 캐싱의 단위가 되며 이 때 오브젝트의 크기는 서로 다르다. 그러므로 동일한 참조 수의 오브젝트의 경우 크기가 작은 오브젝트를 캐싱하는 것이 캐싱 공간 측면에서 유리하다[20]. 따라서 본 논문에서는 오브젝트 분할을 공간 할당 단위로 하여 접근 빈도수 및 참조의 시간적 지역성 및 오브젝트의 크기를 동시에 고려한 캐쉬 공간 할당 정책을 사용한다. 제안된 공간 재할당 알고리즘을 개략적으로 설명하면 다음과 같다. 먼저 제안한 알고리즘은 빈도수에 오브젝트의 접근 패턴의 변화를 반영하기 위하여 요청 간의 평균 간격의 변화를 접근 빈도수에 적용한 가중 접근 빈도수(weighted access frequency)를 구한다. 이렇게 구한 가중 접근 빈도 수에 오브젝트의 크기를 반영하여 단위 공간별 가중 접근 빈도수를 함수를 사용하여 구한 다음 공간 재 할당에 이용한다.

정의 2

캐쉬 v에서 임의의 오브젝트 i에 대한 k번 째 참조 시의 가중 접근 빈도수를 $W_v^{freq}(o_i)^k$로 표시하고 절대 접근 빈도수 k와 k번째 요청까지의 평균 구간 차분(average interval difference)의 누적합으로 정의한다.

$$W_v^{freq}(o_i)^k = k + \sum_{j=1}^{k}\left(\frac{AVG_v(I(s_i)_1^j)}{AVG_v(I(s_i)_1^{j-1})}\right)$$

단, $O=\{o_1, \cdots, o_n\}$, $AVG_v(I(s_i)_1^k)$: 캐쉬 v에서의 오브젝트 i에 대한 k번째 요청이 발생하였을 때의 첫번째 요청부터 k번째 요청까지의 평균 요청 간격

그러나 요청되는 오브젝트는 서로 크기가 상이하므로 동일한 가중 접근 빈도수를 가진다고 하더라도 오브젝트의 크기가 각각 다를 때 가중 접근 빈도수 만으로 공간 재할당 후보를 결정할 경우 오브젝트의 크기에 관계 없이 동일한 가중 접근 빈도수를 가진 오브젝트는 동일하게 취급된다. 따라서 오브젝트의 크기가 아닌 고정된 단위 공간 당 가중 접근 빈도수를 사용하여야만 적절한 재 할당 정책이 가능하다. 그러므로 이러한 고정된 단위 공간 당 가중 접근 빈도수는 앞에서 구한 가중 접근 빈도수를 오브젝트의 크기로 나눔으로써 구하며, 이 값을 구하는 함수를 캐쉬 v에서 임의의 오브젝트 i의 k번째 참조 시의 캐쉬 공간 할당에 사용되는 할당 함수(*replacement function*) $f_v^R(o_i)^k$라고 부르고 공간 할당에 사용하며 다음과 같이 정의한다.

정의 3

캐쉬 v에 존재하는 오브젝트 i의 k번째 참조 시의 가중 접근 빈도수를 오브젝트의 크기로 나눔으로써 단위 공간 당 가중 접근 빈도수를 구하는 함수를 할당 함수 $f_v^R(o_i)^k$로 정의하며 이를 오브젝트 i의 가중 접근 빈도 수와 크기를 고려하여 다음과 같이 구할 수 있다.

$$f_v^R(o_i)^k = \frac{W_v^{freq}(o_i)^k}{S_C^A(o_i)}$$

단, $S_C^A(o_i)$는 오브젝트 i의 캐쉥 된 부분의 크기

4. 실험 결과

본 장에서는 앞 장에서 제안한 네트워크 캐쉥 정책의 성능을 기존의 알고리즘과의 비교를 통해서 비교 분석한다. 실험 결과의 분석에 앞서 먼저 실험 환경과 실험에 사용된 파라메터들에 대하여 설명을 한다.

4.1 캐쉬 설정과 사용자 도착 유형

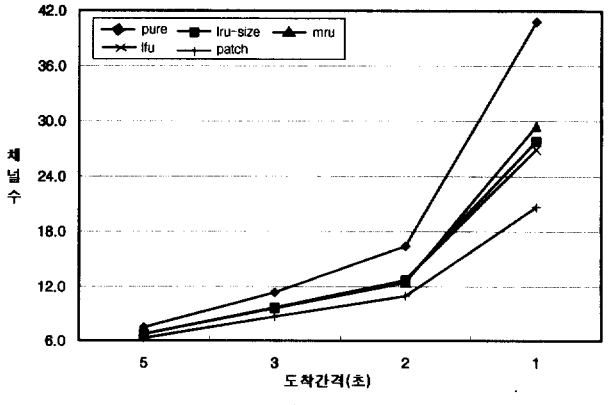
실험은 썬 마이크로 시스템즈 사의 ES3000(OS 버전 솔라리스 2.6)상에서 C로 쓰레드 프로그램을 작성하여 시뮬레이션을 실행하였다. 스트림의 요청 형태는 일반적으로 널리 알려진 Zipf 분포(Θ = 0.271)를, 도착 간격은 포아송 분포를 따르는 것으로 가정하였다. 또한 통상 CPU나 메모리 버스 등의 속도가 전체 시스템 성능에 영향을 미치지만 본 실험의 목적은 연속형 미디어 전송에 필요한 서버와 네트워크 채널의 경제적인 운영에 있으므로 저장 서버의 CPU나 메모리의 전송 속도는 충분하다고 가정하였다. 이 밖의 본 실험에 사용된 파라메터가 <표 1>에 나타나있다. [14]에 따르면 라운드 트립 지연(round trip delay)은 네트워크의 교통량에 따라 변화가 심하며 때에 따라서는 수 초에 이를 때도 흔히 있다고 한다. 그러므로 수 초(예를 들면 5초 정도의 분량)에 해당하는 오브젝트의 초기 부분을 캐쉬에 선반입하므로써 중앙에 존재하는 서버로부터 데이터를 전송할 때 발생하는 초기 지연을 줄일 수 있다.

〈표 1〉 네트워크 캐쉥에 사용된 파라메터

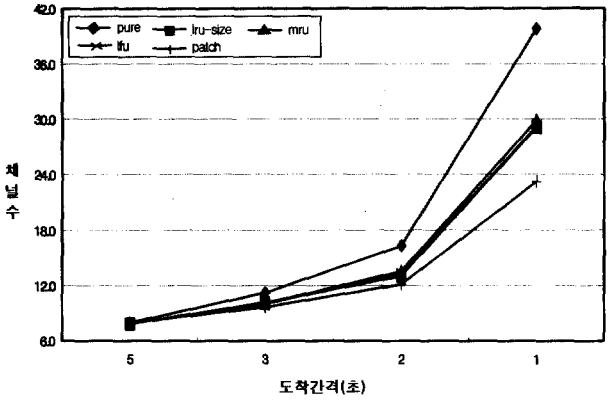
파라메터	값
오브젝트의 수	300, 600,1200
오브젝트의 길이(단위 초)	30 180(sec)
스트림 도착 간격(초)	1, 2, 3, 5
초당 스트림 재생률/초	4 Mbps
디스크(캐쉬) 전송율/초	8 Mbytes/sec
디스크 용량(바이트)	3 GB
디스크 블록 크기(바이트)	512
디스크 회전율(RPM)	7,200
최대 탐색 지연 (ms)	8

4.2 네트워크 채널의 사용도

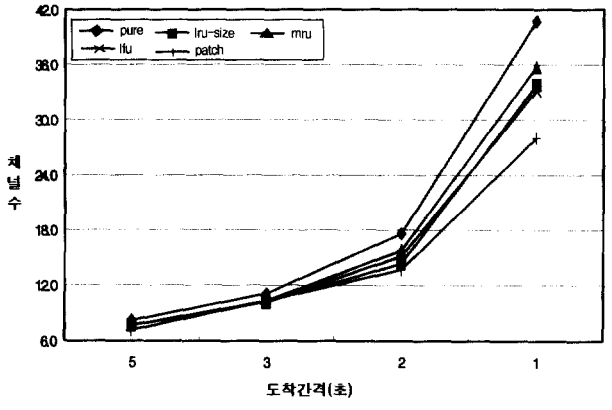
첫 번째 실험은 스트림의 서비스에 필요한 네트워크 채널의 사용도를 스트림의 도착 간격과 오브젝트의 수를 변화시켜가면서 행하였다. 채널의 사용도는 요청된 오브젝트를 네트워크를 통하여 전송할 때 필요한 네트워크 자원을 버퍼란 연속형 미디어 오브젝트처럼 데이터 전송에 등시성을 가진 응용의 경우 등시성을 유지하기 위하여 디스크와



(a) 오브젝트 수 : 300



(b) 오브젝트 수 : 600



(c) 오브젝트 수 : 1200

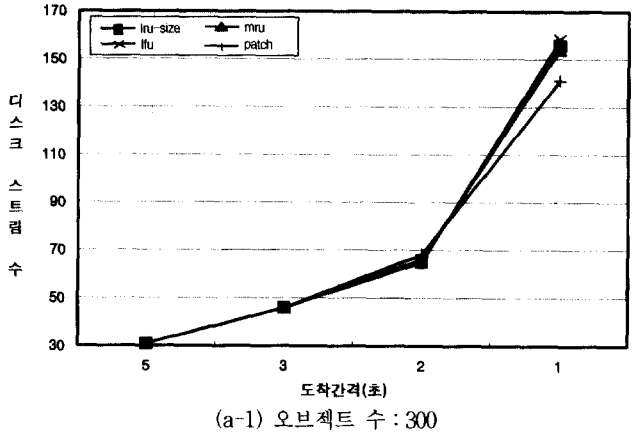
(그림 2) 채널 사용도의 변화

네트워크의 데이터 전송 속도차를 완충하기 위한 버퍼를 말하는 것으로 실시간 전송을 위해서는 스트림이 종료될 때까지 확보되어야 한다. 본 실험에서는 이를 위하여 피크 타임 때의 각각의 네트워크 경로의 사용도를 측정한 다음 해당 값들의 평균을 구하여 (그림 2)에 나타내었다. 그림에서 *pure*로 나타난 부분은 네트워크 캐싱의 적용없이 중앙 서버에서 바로 오브젝트를 전송하였을 경우를 나타내며, lru-size는 LRU-SIZE 알고리즘을, lfu는 LFU 알고리즘을, mru는 MRU 알고리즘을 적용한 결과를 나타낸다. 본 논문에서 제안한 알고리즘은 *patch*로 표현하였다.

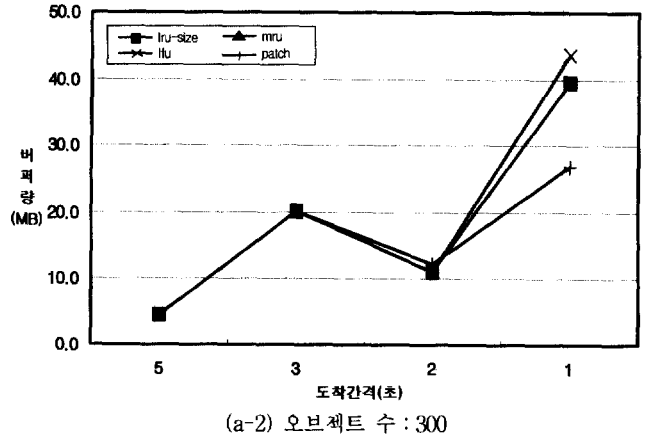
실험 결과를 보면 사용자의 도착 간격이 5초인 경우를 제외하고는 캐싱 정책을 사용하는 것이 캐싱 정책을 사용하지 않는 *pure*의 경우보다 적은 네트워크 채널이 필요하였다. 캐싱 정책을 사용하지 않는 경우와 캐싱 정책을 사용하는 경우의 성능 차이는 사용자의 도착율이 높을수록 많이 났다. 특히 본 논문에서 제안하는 방식(*patch*)과 캐싱을 적용하지 않은 *pure* 방식의 차이는 거의 두 배에 이르며 캐싱 정책을 사용하는 경우 사용자의 도착률이 높아질수록(도착 간격이 1 또는 3초) 오브젝트의 수의 변화가 캐싱 정책에 영향을 미치고 있었다. 이는 오브젝트의 수가 증가하는 반면 네트워크 캐쉬의 크기는 고정되어 있으므로 재 할당이 빈번하게 일어나기 때문이다. 따라서 오브젝트의 수가 증가 할수록 캐싱되는 오브젝트를 더욱 선별적으로 두어야 할 것으로 판단된다. 그러나 캐싱을 적용하지 않은 *pure*의 경우 요구되는 네트워크 채널의 수가 오브젝트의 수의 변화에 영향을 받지 않는다. 이는 캐싱을 적용하지 않는 경우 요청되는 오브젝트에 상관 없이 사용자와 서버간의 네트워크 채널이 독립적으로 형성되기 때문이다. 본 논문에서 제안한 알고리즘은 사용자의 도착률이 낮은 경우에는 성능 향상을 보이지 않았지만 사용자의 도착률이 높아질수록 다른 캐싱 정책에 비해 성능이 증가하였다. 특히 도착 간격이 1 초인 경우는 다른 알고리즘 대비 최고의 성능을 보였다.

4.3 서버의 선반입 버퍼량의 변화

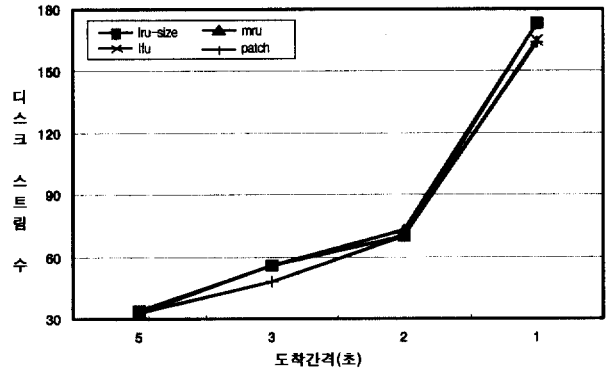
본 절에서는 두 번째 성능 평가의 척도로써 네트워크 캐싱의 적용으로 인한 서버의 자원 변화 정도를 서버에서 피크 타임때의 디스크 스트림의 수와 디스크 스트림을 서비스하기 위한 선반입 버퍼의 양으로써 평가하였다. 선반입 의미하며 선반입 버퍼의 양은 디스크 스트림의 수가 증가함에 따라 동시에 증가한다. (그림 3)에 요청된 오브젝트의 수와 사용자 도착률의 변화에 따른 디스크 스트림의 수와 그에 따른 선반입 버퍼량의 변화를 나타내었다((그림 3)의 위쪽이 디스크 스트림의 수를, 아래쪽이 해당 디스크 스트



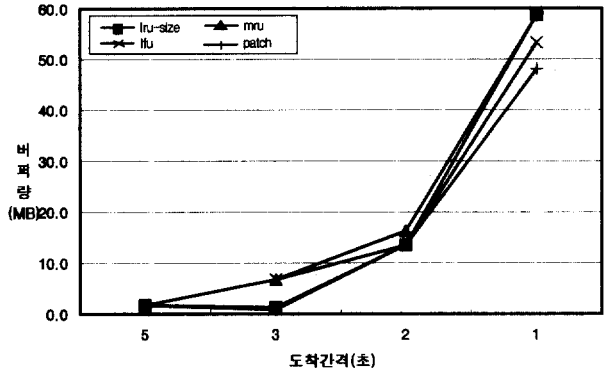
(a-1) 오브젝트 수 : 300



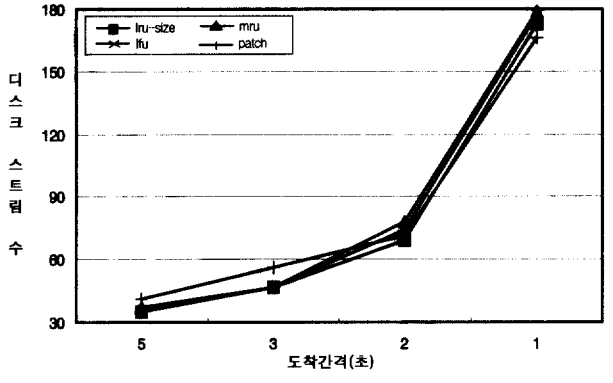
(a-2) 오브젝트 수 : 300



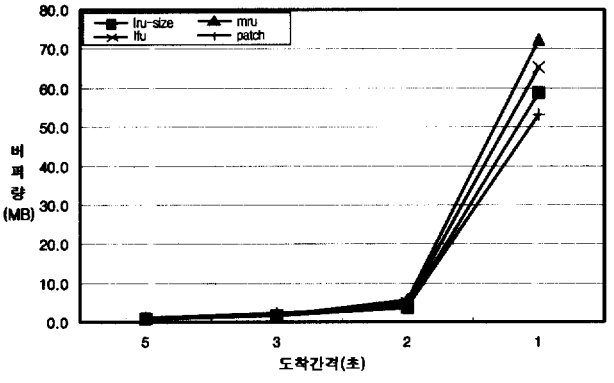
(b-1) 오브젝트 수 : 600



(b-2) 오브젝트 수 : 600



(c-1) 오브젝트 수 : 1200



(c-2) 오브젝트 수 : 1200

(그림 3) 서버의 디스크 스트림과 선반입 버퍼량 변화

림들을 서비스하기 위한 선반입 버퍼의 양이다). 그림에서 보는 것처럼 전체적으로 피크 타임 때의 디스크 스트림의 수의 경우 사용자의 도착률과 오브젝트 수에 관계없이 제안한 알고리즘을 적용한 경우가 가장 좋은 결과를 나타내었다. 그러나 도착률이 낮거나 보통의 경우(도착 간격이 2, 3, 5초)에는 알고리즘 별 디스크 스트림의 차이와 이에 따른 선반입 버퍼량의 차이는 그다지 크지않다. 반면 사용자 도착 간격이 1초인 경우를 보면 제안한 알고리즘과 다른 알고리즘과의 차이가 분명하게 나타난다. 이는 앞에서 언급한 히트율의 결과와도 관계가 있으며 사용자 도착 간격이 클수록 캐싱된 오브젝트의 재 사용율이 떨어진다. 따라서 어떠한 캐싱 알고리즘을 사용하더라도 그 차이는 크지않다는 의미가 된다. 그러나 도착 간격이 좁아질수록 캐싱 정책의 영향을 받으며, 저장 서버의 부하 정도도 알고리즘의 영향을 많이 받는다는 것을 알 수 있다. 또한 디스크 스트림이 점유한 디스크 대역폭이 디스크가 제공하는 최대 대역폭에 가까워 질수록 필요한 선반입 버퍼의 양의 급격하게 증가하는 선반입 버퍼량의 증가 패턴을 고려한다면 사용자 도착 간격이 좁아질수록 성능의 차이는 더욱 더 커질것으로 판단된다.

5. 결론 및 향후 연구

본 논문에서는 인터넷 상에서 연속형 미디어 오브젝트를 서비스할 때의 네트워크 자원을 효율적으로 관리하기 위하여 선반입과 패칭을 동시에 적용하여 스트림을 서비스하는 정책을 제안하였다. 제안한 알고리즘에서는 오브젝트를 논리적으로 전방 분할과 후방 분할로 나눈 다음 오브젝트 전체를 캐싱 단위로 하는 것이 아니라 서비스의 요청 빈도에 따라 단계적으로 오브젝트가 통합 캐싱되는 정책을 적용하였다. 또한 공간 할당 알고리즘으로는 요청된 오브젝트의 빈도수를 기본으로 하되 요청의 최근성을 반영하기 위하여 요청 도착 간격의 변화량을 측정하여 이를 빈도수에 적용하여 가중 접근 빈도수를 구하고 이렇게 구해진 값을 오브젝트의 크기로 나누어서 구한 값을 공간 재할당 정책에 사용하였다.

제안한 알고리즘은 실험을 통하여 평가하였으며 성능의 비교 평가를 위해서 기존의 캐싱 알고리즘의 성능을 본 논문에서 제안한 알고리즘과 동일한 환경에서 측정하였다. 실험 결과 일부 경우를 제외하고는 제안한 캐싱 정책의 성능이 위에서 언급한 성능 평가 척도의 모든 면에서 다른 알고리즘의 결과보다 우수한 것으로 나타났으며 네트워크 채널의 경우 다른 캐싱 알고리즘을 적용하였을 때에 비하여 30%, 스트림 서비스를 위한 선반입 버퍼의 경우 10%의 개선 효과가 있었다.

참 고 문 헌

[1] M. Abrams, C. Standridge, G. Abdulla, S. Williams and E. Fox, "Caching Proxies : Limitations and Potentials," Proc. Fourth International World Wide Web Conference, Boston, 1995.

[2] C.C Aggarwal and J. L. Wolf and P. S. Yu, "On Optimal Batching Policies for Video-On-Demand Storage Server," Proc. of the IEEE Int'l Conf. On Multimedia Systems. June 1996.

[3] Aggarwal, C., Wolf, J. L., Yu, P.S., "Caching on the World Wide Web," Knowledge and Data Engineering, IEEE Transactions on Volume : 111, Jan.-Feb. 1999, Page(s) : 941.

[4] Asit Dan, D. Sitaram and P. Shahabuddin. "Scheduling Policies for an On-Demand Video Server with Batching," In Proceedings of ACM Multimedia, pp.15-23, San Fransisco, California, Oct. 1994.

[5] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrel, "A Hierarchical Internet Object Cache," In Proceedings of the 1996 USENIX Technical Conference, San Diego, CA, January 1996.

[6] L. Gao, D. Towsley, "Supplying Instantaneous Video-on Demand Services Using Controlled Multicast," Proc. IEEE Multimedia Computing Systems'99, (June 1999).

[7] Garth A. Gibson, Jeffrey Scott Vitter, John Wilkes, "Strategic directions in storage I/O issues in large-scale computing," ACM Computing Surveys Vol.28, No.4 (Dec. 1996), Pages 779-793.

[8] S. Glassman, "A Caching Relay for the World Wide Web," Proc. of First International World Wide Web Conference, Geneva, 1994.

[9] Greg Barish and Katia Obraczka , "World Wide Web Caching : Trends and Techniques," Univ. of Southern California USC tech reports, available at ftp : //ftp.usc.edu/pub/csinfo/ tech-reports/papers/99-713.ps.Z.

[10] Kien A. Hua, Ying Cai, Simon Sheu, "Patching : A Multicast Technique for True Video-on-Demand," ACM Multimedia Bristol, 1997, UK 191-200.

[11] C. Maltzahn, K. Richardson, and D. Grunwald, "Performance Issues of Enterprise Level Web Proxies," In Proceedings of the SIGMETRICS Conference on Measurement an Modeling of Computer Systems, June 1997.

[12] Pei Cao and Sandi Irani, "Cost-Aware WWW Proxy Caching Algorithms," In Proceedings of the 1997 USENIX Symposium on Internet Technology and Systems, Dec. 1997.

[13] Reza Rejaie, Mark Handley, Haobo Yu, Deborah Estrin, "Proxy Caching Mechanism for Multimedia Playback Streams in the Internet," 1999, USC tech reports, available at ftp : //ftp.usc.edu/pub/csinfo/tech-reports/papers/99-693.ps.gz.

[14] Rexford S., J. Towsley, "Proxy Prefix Caching for Multimedia Streams," INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE Volume : 3, 1999, Page(s) : 1310-1319 Vol.3.

[15] Rizzo, L., Vicisano, L., "Replacement Policies for A Proxy Cache," Networking, IEEE/ACM Transactions on Volum e : 82, April 2000, Page(s) : 158-170.

[16] R. Tewari, H. M. Vin, A. Dan, and D. Sitaram, "Resource-based Caching for Web Servers," In Proceedings of ACM/SPIE Multimedia Computing and Networking 1998 (MMCN'98), San Jose, pp.191-204, January 1998.

[17] Wessels, D., Claffy, K "ICP and the Squid web cache," Selected Areas in Communications, IEEE Journal on Volume : 163, April 1998, Page(s) : 345-357.

[18] Zhi-Li Zhang, Du, D.H.C., Dongli S, Yuewei Wang, "A network-conscious approach to end-to-end video delivery over wide area networks using proxy servers," INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE Volume : 2, 1998, Page(s) : 660-667 Vol.2.

[19] Haobo Yu, Deborah Estrin, Ramesh Govindan, "A Hierarchical Architecture for Internet-scale Event Services," Techincal Report of Dept. of CS, Univ. of Southern California, 1999.

[20] Charu C. Aggarwal, Joel L. Wolf and Philip S. Yu, "A Permutation-Based Broadcasting Scheme for Video-On-Demand Systems," Proc. of the IEEE International Conference on Multimedia Systems. June 1996.

[21] Jia Wang, "A Survey of Web Caching Schemes for the Internet," Cornell Network Research Group (C/NRG) Department of Computer Science, TR99-1747, May 12, 1999. Cornell University Ithaca, NY 14853-7501, available at http : //cs-tr.cs.cornell.edu : 80/Dienst/UI/2.0/Describe/ncstrl.cornell/TR99-1747.

백 건 효

e-mail : ghbaek@dit.ac.kr

1988년 부산대학교 계산통계학과 졸업 (학사)

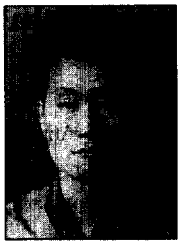
1989년~1992년 공군 전산실 근무

1994년 부산대학교 전자계산학과 졸업 (석사)

1996년~1998년 부산대학교 전자계산학과 박사 수료

1999년~현재 동의 공업대학 사무자동화과 조교수

관심분야 : 멀티미디어, 병렬파일시스템, 캐싱 등



박 용 운

e-mail : ywpark@dit.ac.kr

1988년 부산대학교 계산통계학과 졸업 (학사)

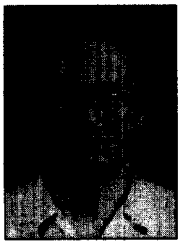
1988년~1995년 ㈜LG-EDS 근무

1997년 부산대학교 전자계산학과 졸업 (이학 석사)

2000년 부산대학교 전자계산학과 졸업(이학 박사)

2000년~현재 동의 공업대학 전자계산과 전임강사

관심분야 : 저장 서버, 캐싱, 전자 상거래 등



정 기 동

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

1973년 서울대학교 졸업(학사)

1975년 서울대학교 대학원 졸업(석사)

1986년 서울대학교 대학원 계산통계학과 졸업(박사)

1990년~1991년 MIT, South Carolina 대학 교환교수

1978년~현재 부산대학교 전자계산학과 교수

관심분야 : 병렬처리, 멀티미디어