인터넷 캐싱에서의 대체 알고리듬의 설계 및 성능평가

문진용[U] 구용완
수원대학교 전자계산학과
igis152@yahoo.com

Design and Performance Evaluation of Replacement Algorithm in the Internet Caching

Jin-Yong Moon[U] Yong-Wan Koo
Dept. of Computer Science, The University of Suwon

요 약

1990년대 초반에 등장한 인터넷을 기반으로 하는 서비스 중에 하나인 웹은 대중적인 인기를 확보하여 사용량이 기하급수적으로 증가하고 있다. 인터넷 사용자와 서비스 제공자의 증가에 따라 같은 객체에 대한 중복요청이 네트워크 대역폭의 상당 부분을 차지하여 불필요하게 낭비되며 일부 인기있는 서버로 부하가 집중되어 응답시간이 길어지게 된다. 이와 같은 문제를 해결하기 위해 인터넷 캐싱기법이 도입되어 연구되어 왔다.

사용자의 지연시간 및 네트워크 전송용량의 사용효율은 어떻게 캐싱을 설계하고 운영하는가에 따라 많은 영향을 받게 된다. 본 논문에서는 인터넷 캐싱을 위해 설계된 기존의 전략들을 살펴보고 우리의 새로운 알고리즘을 제안한다. 그리고, 각 알고리즘의 성능을 trace-driven 모의실험을 통해 검증한다.

1. 서론

최근의 인터넷의 인기는 네트워크의 트래픽을 폭발적으로 증가시켰다. 그 결과로 인터넷은 네트워크 병목 현상의 주된 원인이 되었다. 서버에 접속한 사용자는 느린 네트워크로 인하여 클라이언트 쪽의 지연을 경험하게 된다. 더욱이, 네트워크를 통한 크기가 큰 인터넷 객체의 반복적인 재전송은 네트워크의 트래픽을 더욱 증가시킨다. 따라서 네트워크의 사용 가능한 대역폭을 감소시켜 다른 사용자의 네트워크 지연을 증가시킨다. 따라서 접속 시간을 줄이기 위해 인기 있는 객체를 복제하여 사용자에게 좀 더 가까이 저장하는 것이 바람직하다[ABC96, ASA95, 문00].

그 결과로 현재 인터넷 캐싱에 대한 연구가 활발히 진행되고 있다[ASA95, BCF98, PH97, WAS96]. 캐싱은 자주 요청되는 인터넷 객체를 사용자에게 보다 가까이 저장함으로써 인터넷 성능을 전반적으로 향상시킨다. 캐싱은 인기 있는 서버의 부하와 전체 네트워크의 트래픽을 감소시키며 인터넷 사용자에게는 캐싱된 문서로 빠른 응답시간을 제공한다.

캐싱은 네트워크의 여러 지점에서 구현할 수 있다. 본 논문에서는 인터넷 캐쉬에서 적용될 수 있는 일반적인 주기억장치 캐쉬 대체 전략에 관한 것이다. 따라서, 이 결과를 인터넷 서버, 대리 서버(Proxy server), 그리고 인터넷 클라이언트에 적용할 수 있다.

캐싱되는 위치는 다르지만 어떻게 한정된 인터넷 캐쉬 공간을 효율적으로 사용할 것인가라는 공통된 문제를 가지고 있다. 따라서 캐싱 전략의 문제는 한정된 저장공간을 효율적으로 관리하여 적중률(Hit ratio)을 높이는 것이다. 그러므로, 대체 전략의 성능은 캐싱 전략의 효율성에 중요한 요소가 된다. 이를 위해, FIFO, LRU, LFU, 그리고 SIZE(Largest file removed first)와 같은 여러 대체 알고리듬들이 연구되어 왔다[ASA95, WAS96].

인터넷 캐싱은 파일 시스템, 가상 메모리 시스템과 같은 전통적인 대체 전략과는 다르다. 인터넷 캐싱에서는 가변크기의 인터넷 컨텐츠를 지원해야 한다. 예를 들어, 인터넷 컨텐츠의 크기는 수B에서 수십MB까지 매우 다양하며 각 객체의 인기도 또한 매우 가변적이다[ABC96, YM98]. 그 결과로 인기가 없는 매우 큰 객체의 삽입으로 인하여 크기나 작으나 매우 인기 있는 수많은 객체들을 제거되어야 하는 상황이 발생할 수 있다. 이 경우 인터넷 캐싱의 효율은 매우 떨어지게 된다.

본 논문에서는 인터넷 캐쉬를 관리하는 새로운 전략을 제시한다. 모든 객체들을 하나의 캐쉬에 관리하는 것이 아니라 객체들의 크기에 기반하여 저장하는 부분을 논리적으로 달리하는 새로운 캐싱 전략을 제안한다. 캐쉬는 3부분으로 나누어 관리하여 객체의 크기를 기초하여 저장·관리한다. 따라서, 각 부분에서 관리하는 객체들의 크기는 다소 동일해져서 전통적인 대체 알고리듬을 적용하는데 큰 기여를 하게 된다.

본 논문의 구성은 다음과 같다. 2장에서 기존의 인터넷 캐싱 전략들에 대해 소개하고, 3장에서는 제안한 알고리듬을 상세하고, 4장에서는 소개된 알고리듬들의 성능을 평가하기 위해 trace-driven 성능평가 및 그 결과에 대해 논한다. 마지막으로 5장에서는 현재까지 수행되어온 본 논문의 성과에 대해 정리하고 앞으로의 개선방향에 대해 기술한다.

2. 관련연구

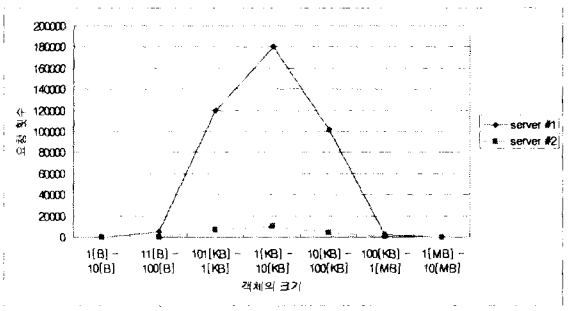
[ASA95, WAS96]에서 평가된 다음과 같은 캐쉬 대체 전략들이 있다. 본 논문에서는 제안된 전략과의 성능평가를 하여 본다.

LRU 확장 전략은 크기가 서로 다른 객체를 처리하기 위해 기존의 LRU 전략의 확장이라 할 수 있다. 이 전략은 새롭게 들어오는 인터넷 객체의 여유공간을 마련하기 위하여 가장 최근에 사용되지 않은 객체부터 삭제하는 방식이다. 이렇게 하면, 삭제되는 객체는 0개에서부터 수많은 객체들이 캐쉬에서 삭제될 수 있다. 이 방식은 실제적으로 매우 낮은 성능을 보인다는 것을 4장에서 보인다.

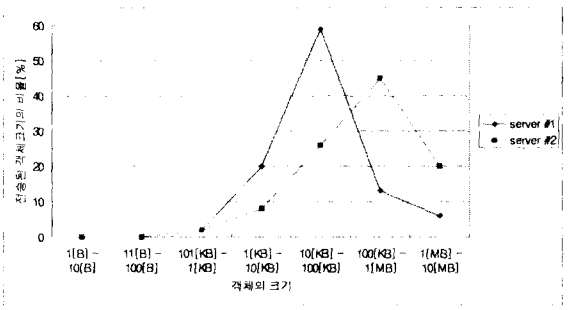
LRUMIN 전략은 삭제되는 객체의 수를 최소화하기 위하여 크기가 작은 객체에 비중을 두는 전략이다. 예를 들면, 캐쉬에 새롭게 들어오는 객체의 크기를 S로 하고 S보다 큰 여유공간이 없다고 하면 다음과 같은 순서로 처리한다. S/2보다 큰 객체들 중에서 LRU 전략으로 제거하고, S/4보다 큰 객체 중 LRU 방식으로 여유공간이 생길 때까지 이와 같은 전략으로 계속해서 제거한다.

SIZE 전략은 인기가 있는 작은 크기의 객체가 큰 객체의 삽입으로 제거되는 상황을 막기 위해 삽입되는 객체의 여유공간이 생길 때까지 크기가 가장 큰 객체부터 차례로 제거하는 전략이다.

본 논문에서는 제안한 인터넷 캐쉬 대체 알고리듬이 위의 기존 알고리듬에 비해 보다 좋은 성능을 보인다는 것을 4장에서 보인다.



<그림 1> 객체 크기별 요청 횟수



<그림 2> 객체 크기별 총전송량 비율

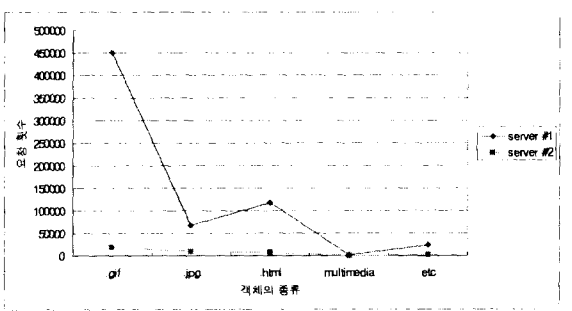
3. 인터넷을 위한 캐쉬 전략

본 연구의 접근 방법은 인터넷 객체의 크기에 기반한 분할된 인터넷 캐쉬 전략이다. 분리된 캐쉬 전략의 원리는 참조의 지역성을 유지하며 각 분할된 부분 캐쉬의 적중률을 높이는 방법이다. 각 부분에 저장된 인터넷 객체는 같은 등급의 객체들간에만 대처된다. 작은 크기의 인터넷 객체의 참조 지역성이 캐쉬에 없는 등급이 다른 크기의 큰 객체의 도착으로 혼란 받는 일이 없어진다. 이 전략을 이용하면 각 분할된 부분의 캐쉬는 시간적 지역성이 개선된다.

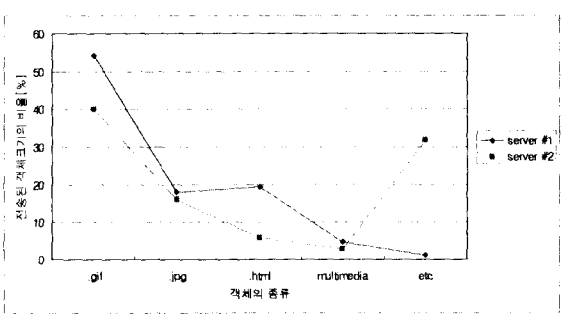
캐쉬에서의 분할된 부분의 수, 각 부분의 크기, 각 부분에서 객체를 지원하는 크기의 상한과 하한 값의 설정, 그리고 각 부분에서 사용되는 대체 알고리듬이 설계에 관련된 파라메타이다. 본 논문에서는 이를 위해 실제 사용되고 있는 인터넷 서버의 로그를 분석하여 다음과 같이 설정하였다.

캐쉬의 구조는 하나의 캐쉬를 논리적으로 3부분으로 나누고 각각을 MIN, NOR, MAX로 한다. 각 부분의 크기비율은 MIN을 1/10, NOR을 2/10, 그리고 MAX를 7/10로 나누었다. 각 부분이 지원하는 인터넷 객체의 상한과 하한의 크기 범위는 4.2절에 나타나 있는 인터넷 부하의 특성에 기반한다. 이를 통해, 인터넷 객체 중에서 가장 많은 요청이 있는 객체의 평균 크기는 1-10KB인 것을 알 수 있다. 우리는 NOR은 1-10KB의 인터넷 객체를 지원하며, MIN은 1KB보다 작은 객체들을 관리하며, MAX는 10KB보다 큰 객체를 지원한다고 정하였다. 각 부분은 LRU 대체 알고리듬으로 작동한다.

캐쉬의 동작구조는 다음과 같다. 캐쉬 관리기는 캐쉬에 저장된 인터넷 객체의 리스트를 관리한다. 요구가 들어오면, 캐쉬 관리기는 요구된 객체를 리스트에서 검색한다. 만일 존재하지 않으며, 관리기는 원래의 인터넷 서버에 요청한다. 전송받은 파일은 등급에 맞는 캐쉬 부분에 저장되게 된다. 만일 전송받은 인터넷 객체를 위한 여유공간이 없다면 바로 그 등급의 캐쉬 부분에서 여유공간이 생길 때까지 LRU 알고리듬으로 삭제할 객체를 선택하게 된다.



<그림 3> 객체 포맷별 요청횟수



<그림 4> 객체 포맷별 총전송량 비율

4. 성능평가

이 장에서는 관련연구에서 살펴본 기존의 LRU 확장, LRUMIN, SIZE 전략들을 본 논문에서 제안한 알고리듬과 실제 운영중인 인

터넷 서버에서 수집된 요청 기록에 대해 적용한 시뮬레이션 방법과 그 결과에 대해 기술한다.

4.1 실험 환경 및 가정

이 실험은 다음과 같은 환경 및 가정에서 수행되었다. 우선, 각 객체를 저장 공간에 저장하는 방법에 따라 발생할 수 있는 단편화 현상은 전혀 없는 것으로 한다. 그리고, 각 인터넷 객체의 변경은 전혀 없는 것으로 가정하였다. 이는 일치성 유지 전략이 대체 알고리듬의 결과에 영향을 주는 것을 방지하기 위함이다.

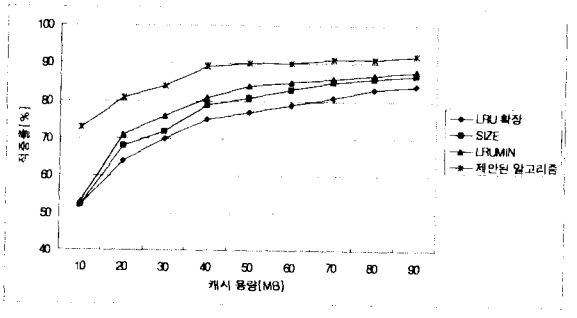
본 논문에서 사용된 trace는 본 대학에서 공개적으로 운영하고 있는 메인 인터넷 서버(이하 server #1)와 학과 홈페이지 서버(이하 server #2)의 로그를 이용하였다. 이 trace는 1999년 11월 한달 간의 요청 기록이며 인터넷 사용자의 요청 횟수는 각각 약 70만 회와 4만 5천 회이며, 총전송된 인터넷 컨텐츠의 양은 각각 약 3GB와 550MB의 요청이 포함되어 있다.

4.2 성능평가 결과

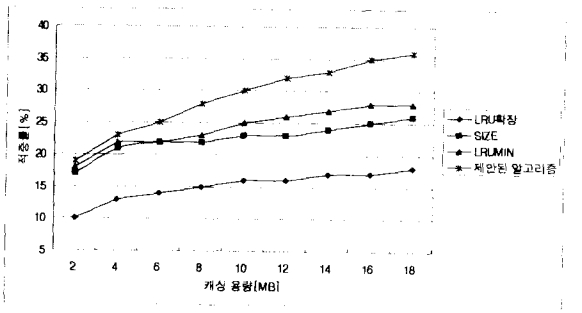
<그림 2>는 객체 크기별로 총전송된 전송량의 비율이다. <그림 1>과 비교하면 상대적으로 인기가 없는 큰 객체가 네트워크 트래픽의 상당부분을 차지함을 알 수 있다. 예를 들어, 1MB ~ 10MB 범위의 인터넷 객체들의 요청 횟수는 아주 미약하나 총전송된 크기에서 차지하는 비율은 각각 6%와 20%로 아주 높음을 알 수 있다.

객체 크기의 분포를 이해하기 위해 컨텐츠를 구성하는 객체 포맷의 종류를 조사하였다. <그림 4>는 객체 종류별로 총전송된 크기 비율로 mp3, mid, swf와 같은 멀티미디어 포맷 데이터의 경우 요청횟수는 <그림 3>과 같이 아주 미약했으나 전송 비율에서 차지하는 비중은 아주 높음을 알 수 있다. 특히, server #2의 경우 zip, hwp, exe와 같은 기타 포맷들은 요청 횟수는 1088회로 아주 작았으나 총전송량에서 차지하는 비율은 31%로 아주 높았다.

본 논문에서는 캐쉬의 크기를 달리하여 캐쉬의 적중률을 비교하였다. <그림 5>과 <그림 6>은 성능평가의 결과를 나타낸 것이다. 우리가 제안한 전략은 기존의 전략들에 비해 높은 적중률을 갖는 것을 알 수 있다. 객체의 크기에 비중을 둔 LRUMIN과 SIZE 전략은 비슷한 성능을 보였다. 이 그래프에서 LRU 확장 전략이 가장 낮은 성능을 보임을 알 수 있다. 이는 기존의 다른 파일 시스템과 같은 대체 전략들에서는 캐싱되는 객체들은 페이지라는 고정된 크기를 가지고 있으나, 인터넷 캐싱에서는 인터넷 객체를 분할하여 기존의 고정크기의 대체 전략을 사용하면 캐싱을 하는 아무런 유용성을 얻을 수 없기 때문에 가변적인 크기의 대체 전략을 사용해야 하므로 단지 앞으로의 사용자의 성향을 최적화한 LRU 확장 전략을 인터넷에서 그대로 사용하기에는 적합하지 않음을 알 수 있다.



<그림 5> 전략별 적중률 비교(server #1)



<그림 6> 전략별 적중률 비교(server #2)

5. 결론

본 논문에서는 인터넷 캐싱의 여러 가지 대체전략들에 대해 다루었다. 우리는 캐쉬를 3부분으로 나누어 처리하는 새로운 전략을 제안하였다. 또한, trace-driven 성능평가로 본 논문에서 제안한 인터넷 캐싱 대체 알고리듬이 다른 기존의 캐싱들에 비해 좋은 성능을 보인다는 것을 보였다. 인터넷 객체의 가변적인 크기 때문에, LRU 확장 전략은 다른 전략에 비하여 전반적으로 성능이 떨어졌다. 제안된 알고리듬은 실제 사용중인 인터넷 서버를 기반으로 시뮬레이션한 결과 다른 전략에 비해 높은 적중률을 보임을 알 수 있었다.

참고 문헌

[ABC96] V. Almeida, A. Bestavros, M. Crovella, and A. Oliveira, "Characterizing reference locality in the WWW," In Proc. of the 4th Int'l Conf. on Parallel and Distributed Information Systems, 1996.

[ASA95] M. Abrams, C. Standridge, G. Abdulla, S. Williams, and E. Fox, "Caching Proxies: Limitations and Potentials," Proc. 4th Int'l World Wide Web Conf., 1995.

[BCF98] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "On the Implications of Zipf's Law for Web Caching," In 3rd Int'l WWW Caching Workshop, 1998.

[PH97] D. Povey and J. Harrison, "A Distributed Internet Cache," In Proc. of the 20th Australian Computer Science Conf., 1997.

[WAS96] S. Williams, M. Abrams, C. R. Standridge, G. Abdulla, and E. A. Fox, "Removal Policies in Network Caches for World Wide Web Documents," Proc. ACM SIGCOMM, pp. 293-304, 1996.

[YM98] P. S. Yu and A. MacNair, "Performance Study of a Collaborative Method for Hierarchical Caching in Proxy Servers," In Proc. of the 7th WWW Conf., 1998.

[문00] 문진용, 구용완, "인터넷 지리 정보 시스템을 위한 HVF의 개발," 한국정보처리학회 논문지, Vol. 7, No. 2, 2000.