웹과 멀티미디어 요청이 혼재한 환경에서 네트워크 전송 비용을 고려한 프락시 캐시 교체 정책

서진모[0] 강지숙 남동훈 박승규
아주대학교 정보통신전문대학원
(seo007[0], orion, ndh, sparky)@madang.ajou.ac.kr

Proxy Cache Replacement Policy reflecting Network Transmission Costs in Web and Multimedia Environments

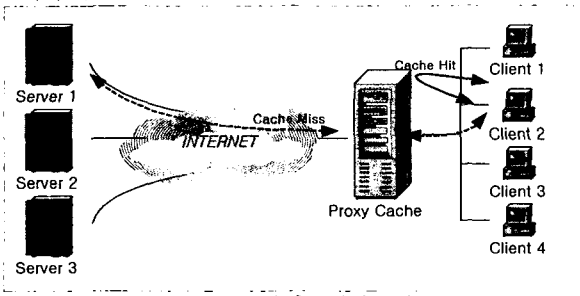
Jin-Mo Seo[0] Ji-Sook Gang Dong-Hoon Nam Seung-Kyu Park
Graduate School of Information and Communication, Ajou University

요 약

사용자의 요구와 인터넷 어플리케이션의 발달로 규모가 큰 미디어 오브젝트의 수가 급증하고 있다. 따라서 네트워크 전송비용은 반드시 고려해야 하는 중요한 요소이다. 본 논문에서는 기존 프락시 캐시 교체 정책들을 분석하고, 이를 개선한 G-N 및 L-N 정책을 제안한다. 이것은 프락시 캐시 소프트웨어인 ' Squid' 에서 채택하고있는 GDSF와 LFU-DA 정책에 네트워크 전송 비용을 추가하여 확장한 알고리즘이다. 시뮬레이션을 통하여 기존의 알고리즘과 비교해 본 결과, 평균 응답 시간을 10%이상 감소시킬 수 있었으며, 추가로 드는 비용(Processing Overhead)은 크게 증가하지 아니 하였음을 확인하였다.

1. 서 론

인터넷은 해마다 폭발적인 성장을 거듭하고 있다. 또한, 최근 전자 상거래를 포함한 다양한 인터넷 어플리케이션의 발달로, 동영상과 같은 연속 미디어 형태의 규모가 큰 오브젝트가 급증하고 있다. 이에 따라 네트워크의 부하는 급격히 증가하고 있다. 인터넷 네트워크 트래픽의 부하를 줄이고, 사용자에게 신속한 응답 속도와 일정한 응답 시간을 제공하기 위해 도입된 것이 프락시 캐시이다.



[그림 1] 프락시 캐시의 동작

[그림 1]과 같이 프락시 캐시는 클라이언트와 서버 사이에 위치해 대리 서버의 역할을 한다. 클라이언트로부터 웹 페이지에 대한 요청을 받아, 자신의 저장장치(메모리, 혹은 디스크)에 있는지를 확인해서 있으면 자신이 가지고 있는 웹 페이지를, 없으면 웹 서버에 접속하여 페이지를 받아와 이를 클라이언트에 전송한다. 따라서 프락시 캐시의 성능은 캐시 교체 정책에 크게 의존한다고 할 수 있다.

캐시 교체 정책에 대한 많은 연구들이 진행되어 왔으며, 대부분 Hit Ratio나 Byte Hit Ratio를 높이는데 그 목적을 두고있다. Hit Ratio는 오브젝트 요청에 대한 적중률을 의미하는 반면, Byte Hit Ratio는 오브젝트의 크기에 따른 네트워크 전송 비용을 함께 고려하기 위한 측정 값이라 할 수 있다

정적 파일(Text, image)위주에서 오디오나 비디오 등의 규모가 크고 가변적인 동적파일위주의 오브젝트가 증가하는 경향으로 볼 때, 오브젝트의 크기를 함께 고려하는 Byte Hit Ratio를 높이는 것에 초점을 두는 것이 더 효과적이라 할 수 있다.

본 논문에서는 Byte Hit Ratio 향상에 목표를 둔 기존 알고리즘(GDSF, LFU-DA)을 기반으로 오브젝트의 크기와 네트워크 전송 비용을 충분히 고려하는 G-N과 L-N 정책을 제안하고, 시뮬레이션을 통하여 성능을 비교 분석한다.

2. 관련 연구

프락시 캐시의 성능향상을 위해 연구된 기존의 알고리즘들은 접근 시간을 고려한 LRU, 접근 빈도를 고려한 LFU, 오브젝트의 크기를 고려한 SIZE, LRU에 크기한도를 고려한 LRU-Threshold, 평균 latency를 최소화 하기 위한 Lowest-Latency-First등 여러 가지가 있다. 논문[1]을 통하여 각 알고리즘들에 대한 성능 분석 결과를 보면 Greedy-Dual-Size알고리즘이 여러 가지 측면에서 좋은 성능을 나타냄을 알 수 있다.

이에 Hewlett Packard(HP) 연구소에서는 위의 알고리즘에 접근 빈도를 고려한 GDSF(Greedy Dual-Size with Frequency) 정책과 LFU에 Aging을 고려한 LFU-DA(Least Frequently Used with Dynamic Aging) 정책을 제안하여 실험한 결과 Byte Hit Ratio와 Hit Ratio 측면에서 보다 향상된 결과를 도출하였다.[4][5] 이 두 알고리즘은 캐시 오브젝트에 우선 순위 값(Ki)을 부여하여 교체 상황이 발생시 가장 낮은 Ki값을 갖는 오브젝트를 캐시에서 삭제하는 방식이다.

HP 연구소에서 제안한 교체 정책은 다음과 같다.

➢ GDSF

$$K_i = F_i \times \frac{C_i}{S_i} + L$$

➢ LFU-DA

$$K_i = C_i \times F_i + L$$

F_i : 오브젝트 i의 접근 빈도

C_i : 오브젝트 i를 캐시로 가져오는 비용

S_i : 오브젝트 i의 크기

L_i : 쫓겨나는 오브젝트의 K_i 값

오브젝트 i를 캐시로 가져오는 비용 값 Ci값은 Hit Ratio를 목적으로 할 경우와 Byte Hit Ratio를 목적으로 할 경우에 따라 다르게 계산이 되는데, Hit Ratio를 높이고자 할 경우는 Ci값을 1로 동일하게 두고, Byte Hit Ratio를 높이고자 할 경우는 Ci값을 2+size/536로 적용한다. Byte Hit Ratio를 위한 비용 함수는 캐시 미스 처리를 위해 보내고 받는 네트워크 패킷의 수를 예측하는 식으로 만들어졌다.[5]

3. 기존 알고리즘의 취약성

인터넷 네트워크의 상태는 일정할 수 없다. 시간과 공간에 따라 사용자의 요청률이 다르고, 응답 시간도 상당히 가변적이다. 최근 국내 한 단체에서 5개 분야 상위 10개 기업을 대상으로 웹 사이트 성능평가를 하였다. 그 결과 각 분야별 접속응답시간은 최소 2초대에서 최대 93초대로 고르게 나타났으며 4초 미만의 응답시간을 보인 웹 사이트는 7개에 불과했다.

이처럼 네트워크의 대역폭은 가변적이다. 그러므로, 모든 웹 서버의 대역을 동일시 처리하는 기존의 방식은 실제 상황에 적합치 못하다. 또한 임의의 오브젝트에 정해진 우선 순위 값(Ki)도 (삭제되기 전까지) 그대로 유지됨으로 인해 현재의 정확한 네트워크 상태를 반영한 값이라 볼 수 없다.

이에 G-N과 L-N 정책을 통하여 이 문제점들을 해결하고자 한다.

4. 제안하는 알고리즘(G-N & L-N)

서버와 프락시 캐시 사이의 네트워크 상태를 최대한 반영하기 위한 G-N(GDSF reflecting Network) 정책과 L-N (LFU-DA reflecting Network) 정책을 제안한다. 본 알고리즘은 기존의 알고리즘과 2가지 측면에서 차이를 갖는다.

첫째, 오브젝트를 서버로부터 프락시 캐시로 가져오는 비용을 실제에 맞도록 하기위해 Ki 함수를 수정하였다.

◆ G-N 정책

$$K_i = F_i \times \frac{C_i}{S_i} \times (1 + CS_i) + L \quad \text{--------①}$$

◆ L-N 정책

$$K_i = C_i \times F_i \times (1 + CS_i) + L \quad \text{---------②}$$

여기에서 CSt 값은 서버와 연결 시간이다.

둘째, 현재의 네트워크 상황을 최대한 고려하여 우선 순위 값을 결정하도록 한다. 이것을 해결하기 위해서는 캐시 되어있는 오브젝트의 해당 서버의 네트워크 상태를 수시로 모니터링하는 것이 가장 정확하겠지만, 네트워크에 커다란 부하를 주게됨으로 적절하지 못하다. 본 논문에서는 유효성 검사 메커니즘이 동작 시, IMS(If-Modified-Since)를 서버측에 요청 할 때 서버로의 연결 시간(CSt)을 뽑아내어 적용시킴으로써 보다 최근의 네트워크 상태를 반영시키는 방식을 사용하였다

제안하는 캐시 교체 정책을 정리하면 다음과 같다.

1. 캐시 Hit 일 경우
   Used Cache Size는 변경 없음.
   참조 회수 F_i 는 1증가
   해당 웹 서버와의 CS_i 를 다시 측정하여 저장
   ① 또는 ②식에 의해 K_i 값을 재 계산하여 저장
2. 캐시 Miss 일 경우
   (경우 A) Total Cache - Used Cache >= S_i
   해당 오브젝트를 캐시 함.
   ① 또는 ②식에 의해 K_i 값을 계산하여 저장
   (경우 B) Total Cache - Used Cache < S_i
   while ((Total Cache - Used Cache) < S_i)
   {
   가장 작은 K_i 를 갖는 오브젝트(EvictObj)를 찾음.
   Used Cache = Used Cache + EvictObj Size
   $L = L + L_{EvictObj}$
   }
   ① 또는 ②식에 의해 K_i 값을 계산하여 저장
   Used Cache = Used Cache + S_i

5. 성능 측정 요소

제안하는 알고리즘을 적용하면, Original 웹 서버와 프락시 캐시 사이의 네트워크 상태가 열악한 연결일수록 캐시될 확률이 더 높아지게 된다. 그러므로 사용자의 평균 응답 시간은 줄어들 것이다. 그러나, 추가되는 작업으로 인한 CPU 처리 비용은 증가하게 될 것이다. 본 논문에서는 실험을 통하여 다음의 2가지 값을 측정하고자 한다.

평균 응답 시간

$$AVE = (HR \times H) + (MR \times (100 - H))$$

HR_i : Hit Response Time

H_i : Hit Ratio

: Miss Response Time

Total Processing Time

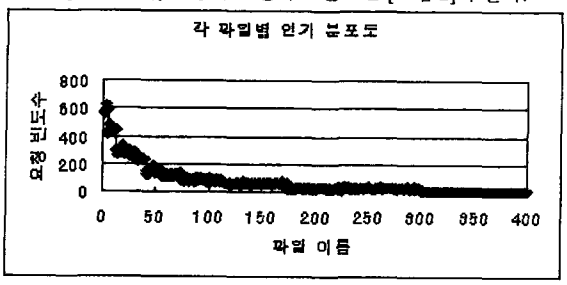
기존의 알고리즘과 비교하여 추가로 드는 처리 비용(Overhead)을 알아 보기 위한 값이다. 본 논문에서는 동일한 요청에 대해 각 정책별 총 처리 시간을 계산하여 비교하기로 한다

6. 시뮬레이션

6.1 시뮬레이션 환경

순수 웹 오브젝트만을 놓고 볼 때, 사용자가 요청하는 오브젝트의 74%가 8K Byte 이하이다.[6] 본 논문에서는 순수 웹 오브젝트의 분포 중에서 전체 요청의 30%가 미디어 스트림이라 가정하였다.

또한 미디어 스트림의 캐시 방법으로 Prefix Caching 방식을 적용하여,전체 스트림의 크기에 관계없이 10Mbyte 만 캐시 한다고 가정한다. 그리고 각 오브젝트의 인기도는 Zipf-like 분포를 따른다고 가정하였으며, 요청되는 인기도 분포는 [그림 2]과 같다.



[그림 2] 각 파일별 인기 분포도

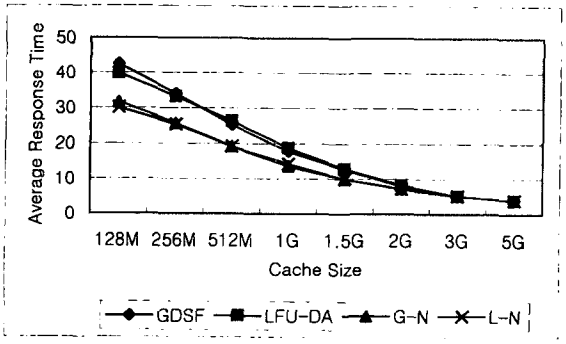
또한 웹 서버의 수는 10개로 한정하였으며, 각 서버의 연결 시간은 [표 1]과 같이 0.1초에서 30초 사이에서 변하는 값을 갖게 하였다.

서버	연결시간(초)	서버	연결시간(초)
A	0.1 ~ 0.5	F	2 ~ 4
B	0.3 ~ 0.7	G	4 ~ 7
C	0.5 ~ 1	H	5 ~ 10
D	0.7 ~ 2	I	10 ~ 20
E	1 ~ 3	J	20 ~ 30

[표 1] 각 웹 서버별 연결 시간

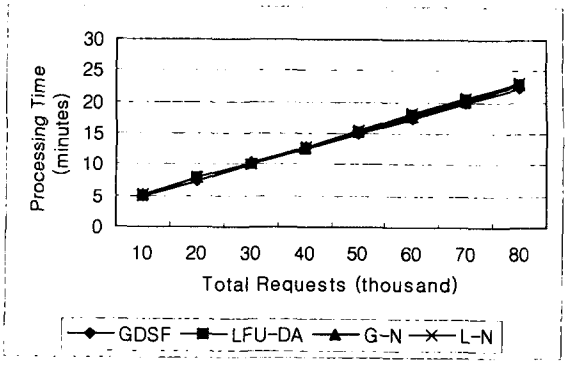
6.2 결과 분석

위와 같은 환경에서 총 30000 회의 요청에 수행하였을 때,대해 각 알고리즘의 평균 응답 시간은 그림[3]과 같다.



[그림 3] 각 정책별 평균 응답 시간

그림에서 보면, G-N과 L-N 정책이 기존의 정책에 비해 평균 응답 시간이 평균 10%이상 감소하였음을 볼 수 있다. 또한 캐시의 크기가 2.5G이상인 경우 프락시 캐시에서 대부분의 요청이 Hit되기 때문에 평균 응답시간은 일정 값으로 수렴됨을 확인할 수 있다. [그림 4]는 캐시의 크기를 1GByte로 고정하였을 때, 각 정책별 사용자 요청 회수에 따른 처리 시간에 대한 결과이다.



[그림 4] 사용자 요청 회수에 따른 처리 시간

그림에서 보듯이 G-N과 L-N이 다른 정책에 비해 추가로 드는 처리 비용이 크게 들지않음을 알 수 있다.

7. 결론 및 향후 연구

인터넷 어플리케이션의 발달로 규모가 큰 여러 미디어 오브젝트의 수가 급증하는 시점에서, 네트워크 전송비용은 반드시 고려해야 하는 중요한 요소이다.

본 논문에서는 기존의 웹 오브젝트 기반의 프락시 캐시 교체 알고리즘인 GDSF와 LFU-DA에 네트워크의 상태와 전송 비용을 최대한 고려하는 G-N정책과 L-N정책을 제안하여 실험하였다. 그 결과 제안한 정책을 통하여 평균 응답시간을 10%이상 감소시켰으며, 추가로 드는 오버헤드가 크지 않음을 확인하였다.

GDSF와 LFU-DA는 Squid 프락시 서버의 구현되어 상용화되고 있는 알고리즘이다. Squid는 GNU license를 갖는 리눅스용 소프트웨어로서, C언어로 구현되어있다. 차후 G-N과 L-N정책을 Squid에 적용하여 실질적인 성능 평가를 할 계획이다.

또한 멀티미디어 스트림의 캐시 정책을 보다 깊게 연구하여 순수 웹 오브젝트만을 처리하는 Squid에 미디어 스트림의 캐시 기능을 추가하고자 한다.

참고 문헌

[1]P.Cao and S.Irani, "Cost Aware WWW Proxy Caching Algorithms", USITS, pp.193-206, December 1997.

[2]M. Arlitt, L. Cherkasova, J. dilley, R. Friedrich and T. Jin, "Evaluating Content Management Techniques for Web Proxy Caches", Technical Report HPL-98-173, HP Laboratories, April, 1999.

[3]J. Dilley, M. Arlitt and S. Perret, "Enhancement and Validation of Squid's Cache Replacement Policy", Technical Report HPL-1999-69, HP Laboratories, May, 1999.

[4]L. Cherkasova, "Improving WWW Proxies Performance with Greedy-Dual-Size-Frequency Caching Policy", Technical Report HPL-1998-69R1, HP Laboratories, November, 1998.

[5]C. Maltzahn and K. J. Richardson, "Reducing the Disk I/O of Web Proxy Server Caches", Proceedings of the USENIX Annual Technical Conf., pp.6-11, June, 1999.