캐시 서버를 위한 웹 트래픽 데이터 분석

정슬기*, 이일병*
*연세대학교 컴퓨터과학과
e-mail:lovehard@casi.yonsei.ac.kr

Web Traffic Data Analyze for Cache Server

Seulki Jung*, Yillbyung Lee*
*Dept of Computer Science, Yonsei University

요 약

전체 웹 트래픽 요소 중 가장 큰 비중을 차지하는 HTTP 트래픽을 대상으로 하여 과거의 데이터와 비교 분석해 보았다. 현재의 웹 페이지의 경우 최소 10개~ 20개 이상의 또 다른 객체를 요청 하게 되고 있음을 발견했다. 이는 텍스트가 주를 이루었던 과거의 객체들과 매우 다른 양상을 보인다. 최근의 웹 트레이스 로그를 분석하여 기존 알고리즘들의 문제점을 발견하여 지적 하며 새로운 캐싱 알고리즘의 개념을 제안한다.

1. 서론

웹 트래픽은 전체 네트워크의 트래픽에서 많은 부분을 차지하고 있다. 따라서 네트워크 관리의 효율을 평가하는데 있어서 HTTP 요구의 효과적인 처리는 매우 중요한 요소이다. 웹 캐시는 웹에서의 HTTP요구와 성능 향상을 효율적으로 처리할 수 있다[2,7].

위의 처리를 위해 CDN(Content Delivery Network)을 사용 하고 있다. CDN이란 각 ISP에 전략적으로 배치된 캐시 서버를 통하여 컨텐츠를 캐싱한 후 클라이언트에게 제공함으로써 인터넷 이용자에게 효율적으로 컨텐츠를 전송하고 CP(Contents Provider)등의 인터넷 사업자에게는 분산 처리를 통하여 이용자를 수용할 수 있도록 네트워크 인프라를 제공하는 서비스이다.

CDN의 핵심 기술로는 Content Distribution Management, Content Router, Content Edge Delivery, Content Switch로 구성되어 있다[3]. CDN은 병목 현상과 데이터 손실이 발생하는 인터넷 교환 및 네트워크 연결 구간을 우회하여 인터넷 이용자로부터 가장 가까운 서버에서 컨텐츠를 전송도록 해 빠른 속도를 보장한다.

또한 주요 ISP에 분산된 캐시 서버를 통해 컨텐츠를 전송하는 분산 처리 시스템임으로 동시 접속자가 증가 하여도 동일한 속도와 품질을 보장한다. 그리고 트래픽 폭주로 인한 서버의 성능 저하를 방지하므로 클라이언트에게 안정적인 서비스가 가능하다. 더불어 빠른 속도와 안정성을 통해 향상된 서비스를 제공하며 전체적인 COST를 낮추는 효과를 가질 수 있다. 마지막으로 여러 대의 서버를 클러스터 방식으로 연결하여 확장할 수 있어 확장성이 뛰어나며 각 서버가 ISP망에 분산되어 있어서 서버의 부하를 낮춘다.

CDN구성 요소 중 프록시 캐시 서버가 실제 캐시 역할을 담당하는 핵심 요소라고 볼 수 있다. 프록시 캐시간 로드 밸런싱도 중요하지만 프록시 캐시 서버의 알고리즘 효율이 나쁘다면 로드 밸런서에 의해 개선되는 시스템 전체 효율이 좋아질 수 없기 때문이다. 물론 현재 상용 프록시 서버에서 사용 중인 캐싱 알고리즘이나 잘 알려진 프록시 캐싱 알고리즘의 효율은 이미 검증 되었다. 하지만 현재 웹 트래픽의 환경을 생각할 때 과거와 다른 많은 변화가 있으며 현존하는 알고리즘의 문제점을 찾아 이를 보완 개선 하고자 한다.

전체 웹 트래픽 데이터에는 TCP, UDP, ARP등의 데이터가 있다. 이중 제일 많은 사용량을 차지하는 TCP는 HTTP데이터가 주를 이루고 있다. UDP는 스트리밍 데이터의 경우 사용되고 있으며 ARP는 캐시 될 필요가 없는 데이터이다.

이에 따라 본 논문에서 주된 관심으로 다룰 데이터는 실제 웹 트래픽 데이터 중 HTTP를 위한 데이터들을 대상으로 하여 분석 하고자 한다. 현제 웹 트래픽 데이터의 패턴을 분석한다면 과거의 캐싱 알고리즘들에 비해 좋은 효율의 캐싱 알고리즘을 찾을 수 있을 것이다.

2. 관련 연구

웹 캐싱은 여러 가지 타입들로 분류되어 진다. 클라이언트 캐싱, 서버 캐싱, 프록시 캐싱, 계층적 캐싱과 협력적 서버 캐싱 등이 있다. 이 웹 캐싱 메소드들의 공통적인 목표는 제한된 저장 공간의 한계 속에서 효율적인 처리를 하는 것이다[1,3,5].

캐시의 저장 공간 안에 빈번하게 사용되는 객체들을 저장함으로서 성능을 향상 시킬 수 있다. 따라서 효과적인 객체 교체 알고리즘은 캐싱의 성능을 향상시키기 위한 중

요 척도가 된다[1,6].

잘 알려진 교체 알고리즘들 중엔 LRU, LFU, Size, LRU-min, Hybrid, LNC-R-W3, LRV, GD-size, sw-LFU, Mix, SLRU, LUV등이 있다.

3. 트래픽 데이터 특성

과거와 비교할 트래픽 데이터를 얻기 위해 동국 대학교의 프록시 서버의 로그를 추출하였다. 프록시 서버를 거친 대부분의 데이터는 동국 대학교의 웹 서버를 대상으로 하고 있었다. <표 1>은 동국과 과거의 공개 프록시 서버 두 곳인 Digital Equipment Corp. (DEC; later acquired by Compaq)와 National Laboratory for Applied Network Research (NLANR)에서 얻은 데이터를 보여주고 있다. 이 두 곳의 데이터는 반효경의 논문에서 사용된 데이터이다[4].

동국의 경우 방학이 시작해서 사용자 수가 줄었음에도 불구하고 과거 10년 전 트래픽 자료들과 비교하여 상당한 차이를 보여주고 있다. 테이블 정보를 살펴보면 동국의 경우 짧은 기간에 불구하고 Total requests와 Total bytes가 가장 큼을 보이고 있다. 그러나 반대로 Total unique request와 Total unique bytes는 가장 작다. Total unique request를 Total unique bytes로 나눈 결과 값인 객체당 평균 사이즈가 동국은 65956, DEC는 12476, NLANR은 11620으로 동국의 경우 5배 이상 크며 데이터 취득 기간까지 고려해 본다면 20~25배의 트래픽이 증가했음을 볼 수 있다.

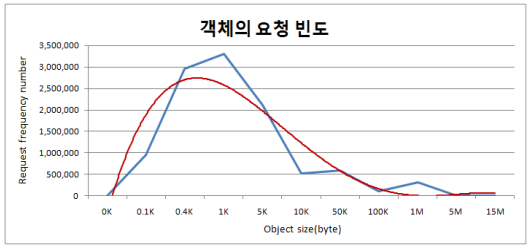
종합적으로 볼 때 동국의 경우 객체의 접근이 매우 빈번하며 객체 사이즈도 크다는 것을 알 수 있다. 공개 프록시 서버 두 곳과 대학 내의 사설 프록시 서버와의 비교임을 고려할 때 현재의 사설 프록시 서버의 경우 이보다 더 큰 차이를 보일것이라고 예상된다. 이는 과거에 비해 요청되는 객체 사이즈가 커졌으며 접근 빈도도 높아 졌다고 생각해 볼 수 있다.

Trace	Duration	Total requests	Total unique requests	Total bytes	Total unique bytes
동국	2007.7.2 ~ 7.8	10, 857,914	9,916	100,098, 348,086	604, 445,099
DEC	1996.9.1 ~ 9.22	609,951	306,685	6,415, 640,000	3,826, 340,000
NLANR	1999.7.18 ~ 8.28	4, 175,435	1, 852,920	39,111, 400,000	21,531, 100,000

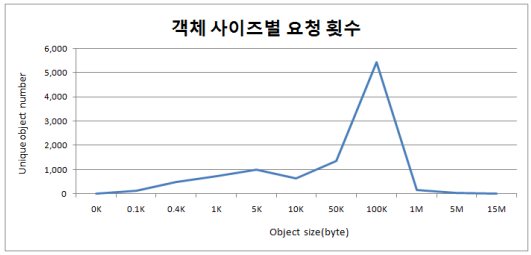
<표 1> 웹 프록시 트레이스의 특성(동국, DEC, NLANR)

4. 트래픽 데이터 분석

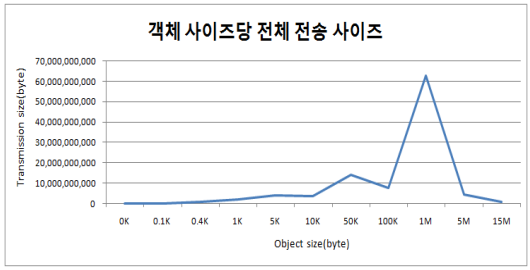
아래의 그래프와 표는 동국의 트레이스를 분석한 결과이다. (그림 1)은 요청되었던 전체 객체들을 일정 범위로 그룹화한 후 해당 범위의 객체들이 얼마만큼 요청 되었는지를 나타낸 것이다. 파란색은 실제 데이터이며 빨간색 선은 추세 선이다. (그림 2)는 각 객체 별로 일정 범위로 그룹화한 후 해당 범위의 객체들이 얼마만큼 요청 되었는지를 보인다. (그림 3)은 객체 사이즈를 기준으로 일정범위로 그룹화 한 후 해당 그룹의 전체 전송한 사이즈를 보인다.



(그림 1) 객체의 요청 빈도



(그림 2) 객체 사이즈별 요청 횟수

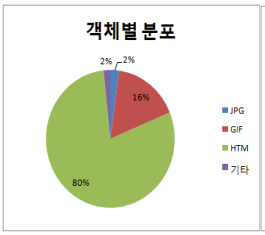


(그림 3) 객체 사이즈당 전체 정송 사이즈

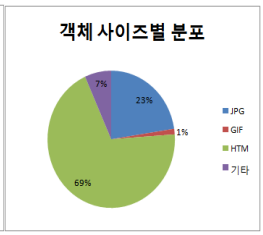
type	unique object	object size	frequency
jar	1	353	1
png	2	762	34
php	2	1,335	4
txt	6	7,284	227
xls	31	2,116,096	832
swf	31	2,495,360	187,805
js	36	256,280	466,158
pdf	59	35,786,295	1,039
jpg	222	136,255,866	2,109,309
gif	1,637	7,915,672	7,532,923
htm	7,889	419,609,796	559,582
Total	9,916	604,445,099	10,857,914

<표 2> 객체별 특성

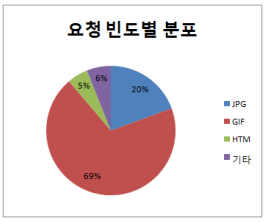
<표 1>의 자료를 사용하여 (그림 4)에서 (그림 6)의 그래프를 작성 하였다. 객체 수, 사이즈, 접근 수가 큰 HTM, GIF, JPG 파일과 그 외의 타입들의 데이터를 묶어서 분류한 후 표현했다. 기타 타입들은 대부분이 HTM파일에서 부수적으로 요청되어 같은 웹 브라우저에 보이는 타입들이 해당되며 JPG와 GIF는 객체의 요청 빈도가 높기에 따로 작성 되었다. 전체 데이터들을 추적해본 결과 98%이상이 HTM 페이지를 요청함에 따라서 부수적인 요청으로 내포된 객체들이 다시 한 번 요청되었음을 알 수 있었다.



(그림 4) 객체별 분포



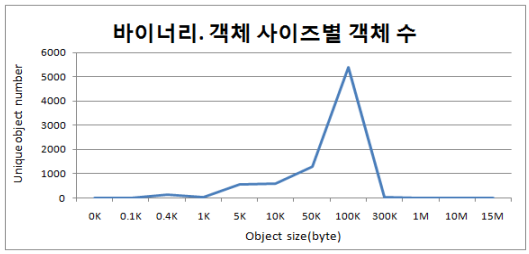
(그림 5) 객체 사이즈별 분포



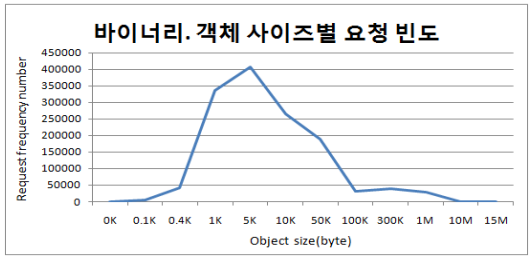
(그림 6) 요청 빈도별 분포

(그림 4)에서 (그림 6)을 살펴보면 HTM이 객체의 수가 가장 많으며 그다음이 GIF, JPG순이다. 객체 사이즈에서도 HTM이 가장 크며 JPG가 GIF에 비해 사이즈는 좀 더 큰 모습을 보여 주고 있다. 이는 JPG가 GIF보다 해상도가 큰 이미지 파일이기 때문에 용량이 큰 것을 보여주고 있다. 빈도수에서는 매우 다른 모습을 보여 주고 있는데 HTM의 요청 빈도가 가장 낮으며 이미지 파일들이 79%, 다른 타입의 파일들이 6%의 빈도를 보였다. 이는 하나의 HTM파일이 평균적으로 16개의 이미지 파일을 링크 하고 있으며, 다른 타입들의 파일을 1개 링크 하고 있음을 보인다. 또한 서로 다른 HTM 파일들에서 같은 객체들이 빈번히 링크 되고 있음을 보여준다.

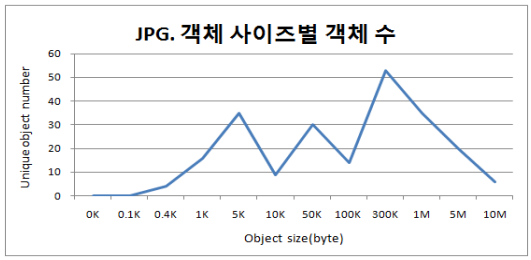
바이너리 파일의 대부분은 HTM 파일이다. (그림 7)에서 100KB 부근의 파일이 많이 존재하는 것을 볼 수 있다. 그러나 (그림 8)을 보면 0.4KB에서 10KB인 사이즈의 파일들이 가장 빈번히 요청되고 있음을 알 수 있다. 실제 로그 파일을 본 결과 이는 100KB부근의 경우 커뮤니티 보드를 요청하는 페이지였다. 동국 웹 서버의 가장 많은 용량을 차지하는 페이지는 커뮤니티 보드였지만 가장 많이 접근하는 페이지들은 인덱스페이지와 학교 정보가 있는 페이지였다. (그림 8)의 그래프를 보면 100KB 영역 이후는 빈도가 낮은 Long-Tail 현상을 보이고 있다.



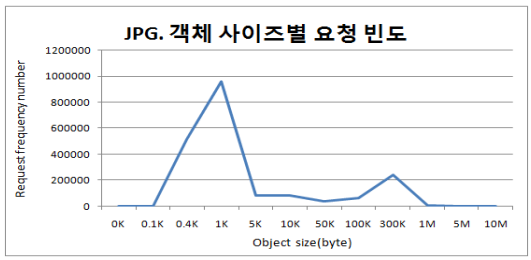
(그림 7) 바이너리 타입. 객체 사이즈별 객체 개수



(그림 8) 바이너리 타입. 객체 사이즈별 요청 빈도



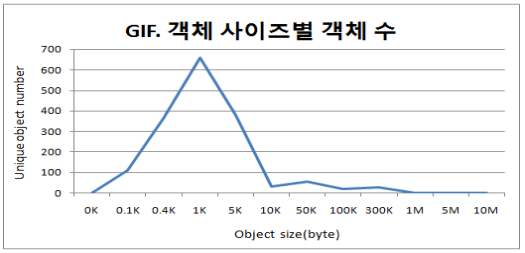
(그림 9) JPG 타입. 객체 사이즈별 객체 수



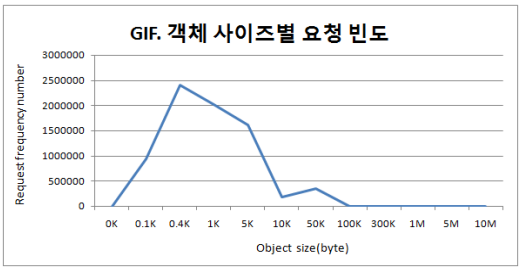
(그림 10) JPG 타입. 객체 사이즈별 요청 빈도

(그림 9)를 보면 JPG파일의 특성상 5KB에서 1MB에 집중된 분포를 보이고 있다. (그림 10)을 보면 요청 빈도가 가장 높은 것은 0.4KB에서 5KB 영역임을 알 수 있고

Long-Tail 현상이 발생하다가 100KB에서 1MB영역에서 다시 한 번 요청이 빈번해 졌다. 가장 빈도가 높은 영역과는 차이가 있지만 파일 포맷의 특성상 일부 파일들은 사이즈가 큼에도 접근이 이루어지고 있다. 이는 배너나 작은 이미지 파일들이 HTM파일 안에 많이 포함 되어있음을 나타낸다.



(그림 11) GIF 타입. 객체 사이즈별 객체 수



(그림 12) GIF 타입. 객체 사이즈별 요청 빈도

(그림 11)을 보면 GIF의 파일 특징대로 0.1KB에서 10KB사이에 가장 많이 차지하고 있다. (그림 12)에서의 요청 빈도도 역시 마찬가지 이다. JPG파일 보다 작은 사이즈의 이미지들이 있기에 1KB이전 사이즈의 파일들이 많이 존재하고 있다.

5. 결론 및 향후 연구

과거의 프록시 트레이스 데이터들은 현재와 마찬가지로 대부분이 웹 트래픽 데이터들이였다. 당시 웹 페이지를 이루는 것은 다수의 바이너리 파일과 몇 몇의 이미지 파일들 이였다.

과거와 비교해서 현재 트래픽 데이터를 분석한 결과 종합적으로 볼 때 하나의 HTM파일의 요청이 이루어 질 경우 실제 HTM파일의 사이즈는 작지만 이에 포함되어 부수적으로 요청되는 약 19개의 다른 포맷의 객체들이 존재했다. 빈도수를 고려해 볼 때, 예를 들어 5KB의 HTM 파일을 웹 브라우저가 요청한다면 1KB 이상의 파일이 19개 더 요청되게 된다. 이는 첫 번째 요청된 객체 사이즈에 비해 최소 4배 이상의 트래픽 발생이 요구 되는 결과이다.

앞서 말했듯이 동국의 프록시 서버에서 가장 많이 요청 했던 사이트는 2006년도에 새롭게 디자인된 동국대학교의 웹사이트이다. 현재의 기업이나 학교들의 웹 사이트는 대부분 이와 비슷한 형식으로 이루어져 있다. 이에 따라 현재 공개 프록시 트레이스 데이터 역시 위의 분석결과와 유사할 것이라고 예측 가능하다.

현재의 특성을 볼 때 과거의 단일 객체들만을 대상으로 하는 교체 알고리즘은 교체 효율을 나쁘게 할 수 있다. 웹 트래픽 데이터의 대부분을 차지하는 웹 사이트 페이지를 고려할 때, 과거와 달리 하나의 요청에 다수의 객체를 제공해야 한다. 이에 따라 한 그룹의 객체들이 평가를 받을 때 같은 평가를 받아야 한다.

다시 말하면 과거의 교체 알고리즘의 개념은 보존하되 평가의 대상을 하나의 객체로 하는 것이 아니라 연관된 다수의 객체들을 그룹으로 묶어 하나의 객체로 보고 이를 하나로서 평가해야 한다는 것이다. 이렇게 됨으로서 각 객체의 평가를 위한 시간을 줄일 수 있고, 교체 알고리즘의 문제가 되는 교체지연시간의 낭비를 줄일 수 있다고 본다.

현재와 미래에서 고려해야할 동영상과 같은 멀티미디어 객체들을 위한 교체 알고리즘 역시 그룹화 교체 알고리즘을 사용할 때 지금보다 좋은 캐싱 효율을 보일것이라고 본다.

향후 시뮬레이션 실험을 통해 그룹정책을 사용한 교체 알고리즘의 효율을 증명하고자 한다.

참고문헌

[1] C. Aggarwal, J. Wolf and P. Yu, "Caching on the World Wide Web," IEEE Trans. Knowledge and Data Engineering, vol 11, no.1, pp.94-107, 1999.

[2] CDN Conference, NetTrend 2001, 2001.

[3] G. Barish, K. Obraczka, World Wide Web Caching: Trends and Algorithms. IEEE Communications, Internet Technology Series, May 2000.

[4] H. Bahn, S. Noh, S. L. Min, and K. Koh, "Efficient Replacement of Nonuniform Objects in Web Caches," IEEE Computer, Vol.35, No.6, pp.65-73, June 2002.

[15] J. Wang, "A Survey of Web Caching Schemes for the Internet," ACM Computer Communication Review, 29, pp.36-46, October, 1999.

[6] N. Niclausse, Z. Liu, P. Nain, "A New and Efficient Caching Policy for the World Wide Web," Proc. Workshop on Internet Server Performance(WISP 98), pp.94-107, 1998.

[7] RFC 3568 Barbir, A., Cain, B., Nair, R., Spatscheck, O.: "Known Content Network (CN) Request-Routing Mechanisms," July 2003

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