파일 타입에 의한 웹 캐쉬 분할기법

오윤주[O*], 한지영*, 이은화**, 윤성대***
부경대학교 전산교육학과[O*]
부경대학교 전산정보학과**
부경대학교 전자계산학과***
{ohbear[O*], hanzero*, eunhwa**}@dol.pknu.ac.kr
sdyoun@pknu.ac.kr***

Web Cache Partition Strategy based on File Type

Yunju Oh[O*], Jiyoung Han*, Eunhwa Lee**, Sungdae Youn***
Dept. of Computer Science Education, Pukyong National University[O*]
Dept. of Computer and Information, Pukyong National University**
Dept. of Computer Science, Pukyong National University***

요 약

최근 웹 사용자의 수가 급격히 증가하고 있다. 이로 인해 인터넷 객체의 캐슁 기법이 매우 중요한 문제가 되었다. 웹 캐쉬는 전통적인 캐쉬와는 달리 다양한 종류와 크기를 가진 개체를 다루어야 하므로 웹 캐쉬에서 보다 나은 적중률을 얻기 위해 여러 접근 방법들이 연구되고 있다. 그에 대한 방법으로 동적인 웹 환경을 고려하면서 캐쉬 공간 관리의 효율성을 높이기 위한 분할캐쉬 접근 방법, 또한 웹 캐쉬의 성능향상에 중요한 역할을 하는 교체 알고리즘에 대한 연구도 계속 진행되어 왔다. 본 논문에서는 캐쉬를 파일 타입에 기반을 둔 클래스별로 분할하고, 각각의 클래스에 효율적인 교체 알고리즘을 사용한다. 그리고 사용자의 객체 요구 변화에 대응하기 위해 일정 간격으로 클래스별 분할비를 갱신한다.

1. 서 론

최근 웹 사용자의 수가 급격히 증가함에 따라 웹 캐슁 기능은 그 중요성이 날로 증가하고 있다. 캐슁은 서버의 부하를 줄이고, 웹 문서의 요청들로 인해 발생하는 네트워크 트래픽을 줄이는데 도움이 된다. 그리고 캐슁된 문서는 웹 사용자들에게 빠른 응답시간을 제공한다. 인터넷에는 다양한 파일 타입이 존재하며 파일 타입마다 평균 크기, 접근 횟수 등이 다르다. 웹 캐쉬는 전통적인 캐쉬와는 달리 다양한 종류와 크기를 가진 개체를 다루고 있다. 웹 캐쉬는 클라이언트 캐쉬, 서버 캐쉬, 프락시 캐쉬, 계층적 캐쉬, 협력적 캐쉬[5] 등 여러 접근 방법이 있다. 이런 모든 접근방법들은 보다 나은 적중률을 얻기 위해 웹 캐쉬의 한정된 저장 공간을 어떻게 효율적으로 관리할 것인지가 공통의 문제이다. 그에 대한 방법으로 동적인 웹 환경을 고려하면서 캐쉬 공간 관리의 효율성을 높이기 위한 분할 캐쉬 접근 방법[1,4]이 있다. 또한 웹 캐쉬의 성능 향상에 중요한 역할을 하는 것이 교체 알고리즘이다. 이를 위해 FIFO(First In First Out), LRU(Least Recently Used), LFU(Least Frequently Used) 그리고 SIZE 등의 많은 교체 알고리즘의 연구가 진행되어 왔다[1,6]. 기존의 캐쉬 공간을 분할하지 않거나, 캐쉬공간을 특정 비율로 고정시켜 분할하는 방법은 분할된 각 영역에 같은 교체 알고리즘을 적용함으로써 분할된 캐쉬의 장점을 다 활용하지 못한다는 단점이 있다.

본 논문에서는 캐쉬를 파일 타입별 클래스로 분할하고, 각 클래스마다 다른 교체 알고리즘을 사용한다. 그리고 주기적으로 캐쉬 클래스의 분할비를 갱신하여 효율적인 캐슁 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 캐슁 기법에 대해 살펴보고, 3장에서는 기존 알고리즘의 단점을 해결하기 위한 분할된 캐슁 기법을 제안한다. 4장에서는 실험 결과 및 성능평가를 보이고, 마지막으로 5장에서는 결론 및 향후연구 과제에 대해 기술한다.

2. 관련연구

2.1 Web Cache 교체정책

캐슁 기법의 성능은 한정된 캐쉬 공간하에서 캐쉬에 보관하고 있을 객체와 삭제할 객체를 동적으로 결정하는 온라인 기법인 캐쉬 교체기법에 의해 크게 좌우된다. 캐쉬 교체 기법의 효율성은 궁극적으로 미래의 참조 가능성을 얼마나 잘 예측하는가에 좌우된다. 캐쉬 교체정책에는 다음과 같은 것들이 있다.

2.1.1 LRU(Least Recently Used)

캐쉬내의 객체들 중 최근 참조된 시각이 가장 오래된 객체를 교체하는 정책이다. 이 정책은 최근에 사용된 객체가 가까운 시간 내에 다시 사용될 가능성이 높다는 점을 이용한다. LRU의 주요 약점중의 하나는 오직 한번 참조된 객체에 의해서 재사용 될 가능성이 높은 객체가 교체될 수도 있다는 점이다. 일반적으로 인터넷 캐쉬에서는 다른 교체정책에 비해 크게 우수한 성능을 갖지 못한다[1,2].

2.1.2 LFU(Least Frequently Used)

이 정책은 참조 횟수가 가장 작은 객체를 삭제하는 교

체 정책으로서, 많이 참조된 객체는 계속해서 참조 될 가능성이 높다는 점에 기반한다. LFU를 사용하면 참조 횟수는 많지만 더 이상 사용되지 않는 객체가 캐쉬에서 제거되지 않는 경우가 발생한다. 이는 이른바 "캐쉬 오염(cache pollution)" 현상의 원인이 되고, 유효 캐쉬 크기의 감소를 일으킨다[2].

2.1.3 SIZE

사이즈가 가장 큰 객체를 삭제하는 교체정책이다. 인터넷 캐쉬는 하드웨어 캐쉬나 파일 시스템 캐쉬와는 달리 캐쉬의 단위가 파일 단위이기 때문에 캐쉬 할 파일의 크기가 각기 다르다. 따라서 크기가 큰 하나의 파일이 크기가 작은 여러 개의 파일을 교체하는 경우가 종종 발생한다. 이 정책은 캐쉬 된 파일 중에서 크기가 가장 큰 파일을 먼저 교체함으로써 이러한 문제점을 개선한다. SIZE 교체 정책은 인터넷 캐쉬에서 우수한 성능을 갖은 것으로 평가하고 있다[1].

2.1.4 LRFU(Least Recently/Frequently Used)

각각의 참조 시점을 그 최근성에 근거하여 고려하는 방법으로, 과거 시점에서의 각각의 참조가 현재 객체의 가치를 높이는 데에 기여하는 바를 독립적으로 계산하여 더하는데 최근의 참조일수록 가치 기여도를 더 높게 계산한다[7].

2.2 분할 캐쉬

웹에서 인식되는 객체 크기의 다양성을 고려하기 위해 캐쉬 저장 공간을 관리하기 위한 방법인 분할 캐쉬가 제안되었다[1]. 분할 캐쉬란 모든 문서를 저장하는 단일 캐쉬를 가지는 것 대신, 캐쉬 저장 공간을 여러 개의 분할 영역으로 나누어서 문서들의 크기나 타입에 따라 각각의 문서들을 저장하는 방법이다. 캐쉬를 분할하는 근본적인 이유는 각각의 분할 영역에서 참조 지역성을 보존함으로써 전체 캐쉬 공간의 적중률을 높이고자 하는데 있다. 분할기법에 관한 기존연구로 파일 타입기반 캐슁 기법[8]과 크기에 따른 캐쉬 분할기법[9]이 있다. 파일 타입을 기반으로 캐슁한 연구의 경우 3가지 파일 타입으로 캐쉬를 분할하고, 각 분할캐쉬에 동일한 교체정책을 사용하여 통합된 캐쉬보다 더 좋은 성능을 보여주고 있다[8]. 하지만 이들 연구는 웹 캐쉬를 분할했지만 각 영역에 동일한 교체정책을 사용하여 분할캐쉬의 최적의 성능을 보여주지는 못하였다.

3. 분할캐쉬 기법의 제안

본 논문에서는 파일 타입별 클래스로 분할된 캐쉬구조를 제안한다. 이는 웹에서 요청되는 객체가 같은 파일 타입이면 크기 또한 비교적 비슷한 분포를 갖는다는 Martin F.Arlitt의 연구[3]를 바탕으로 한다. 즉 비교적 비슷한 크기를 갖는 파일타입별 클래스로 구분함으로써 각 클래스가 최적의 상태를 갖도록 한다. 캐쉬는 Graphic, Text, Etc 3개의 클래스로 분할하는데, Graphic 클래스에는 gif, jpg, jpeg 등의 확장자를 가진 파일이 포함되고, Text 클래스의 경우는 txt, html, htm와 같은 확장자를 가진 파일이 포함된다. 그리고 Etc 클래스는 그 외의 video, audio, 그리고 알려지지 않은 확장자가 포함된다. 이때 각 클래스별 분할비(%)는 '클래스별 전송량/전체 전송량'으로 하고, 각 클래스 분할비는 주기적으로 변경된다. 웹 사용자는 매번 정해진 비율로 객체를 요청하는 것이 아니고, 요청하는 객체 타입들의 비율이 계속 바뀌기 때문에 주기는 일반적인 활동 주기인 1주일로 정한다.

클래스별로 분할된 캐쉬는 각 분할영역에 맞는 교체 알고리즘을 적용한다. 즉, LRU, LFU, SIZE, LRFU를 각각의 분할캐쉬에 한번씩 적용시켜 좀더 나은 성능을 보이는 교체 알고리즘을 선택한다. 이때 교체 알고리즘의 성능을 나타내는 적중률(hit rate)과 바이트 적중률(byte hit rate) 중 어느 쪽을 기준으로 하느냐에 따라 결과가 달라질 수 있는데, 두 가지를 모두 적절히 고려하여 교체 알고리즘을 선택하기로 한다.

4. 분석 및 성능 평가

이 장에서는 로그파일의 분석결과로 정해진 교체 알고리즘을 캐쉬의 파일타입별 클래스에 각각 적용시켜 1주일마다 클래스 분할비를 갱신시키는 방법을 사용한다.

4.1 실험 환경

시뮬레이션을 위한 시스템으로 개인용 펜티엄급 컴퓨터를 사용하였으며 운영체제는 리눅스, 시뮬레이터로는 ns-2(Network Simulator-2)를 사용하였다.

시뮬레이션은 신뢰성을 높이기 위해 인터넷상에 공개된 trace[1]를 사용하여 제시된 trace에 대해 hit rate와 byte hit rate에서 우수함이 입증된 알고리즘과 본 논문에서 제시한 알고리즘을 비교평가 하였다. 이때 웹 캐쉬의 크기는 trace 총량의 10%(새로운 개체에 의해 캐쉬 내의 어떤 개체도 제거되지 않을 최소한의 크기)로 설정했다.

4.2 trace 특징

시뮬레이션의 결과는 입력으로 사용된 트레이스에 의존한다. 본 시뮬레이션에 사용된 trace에 대한 통계를 표 1에 나타냈다. 이를 보면 웹 trace의 높은 다양성과 일반적으로 크기가 작은 Graphic이나 Text/html 타입의 파일에 요청이 많아서 전체 전송량의 대부분을 차지한다는 것을 알 수 있다. 그리고 Audio, Video 타입은 요청은 미비하지만 크기가 크기 때문에 전체 전송량의 20% 정도를 차지함을 알 수 있다.

표 1 trace 통계

File Type	%Refs	%Bytes
Graphic	51.13	46.26
Text/html	43.38	29.30
Audio	0.25	17.91
Video	0.04	3.58
CGI	0.95	0.05
Unknown	4.25	2.89

4.3 클래스별 교체 알고리즘

우선 공개 trace를 이용하여 분할된 각 클래스의 교체 알고리즘을 찾기 위해 모든 클래스에 LRU, LFU, SIZE, LRFU 교체정책을 한번씩 적용한 후 그 결과를 관찰하였다. 즉 매번 동일한 trace를 사용하고 매번 다른 교체 알고리즘을 모든 클래스에 적용하여, 특정 클래스에서 좋은 성능을 보이는 교체 알고리즘들을 찾아내고자 했다.

표 2는 3개의 클래스에 각각의 교체 알고리즘을 적용하였을 때, 캐쉬의 성능 척도인 hit rate와 byte hit rate를 고려하여 각 클래스에 최적인 교체 알고리즘을 보여준다.

표 2 클래스별 교체 알고리즘

Graphic class	LRFU
Text class	LRFU
Etc class	SIZE

4.3 성능 평가 결과

일반적으로 웹 캐쉬의 성능평가 척도로는 hit rate, byte hit rate, 사용자 응답시간, 그리고 전체 비용 등이 사용되고 있다. 인터넷 객체의 크기가 가변적인 성격을 가지기 때문에 본 논문에서는 성능평가 척도를 hit rate와 byte hit rate로 정한다.

그림 1과 그림 2는 시뮬레이션 결과를 보여준다. 그림 1은 날짜 흐름에 따른 hit rate를 나타내고, 그림 2는 날짜 흐름에 따른 byte hit rate를 보여준다. 그림에 의하면 hit rate는 1~6%의 성능향상이 있었고, byte hit rate는 2~9%의 성능향상이 있음을 알 수 있다.

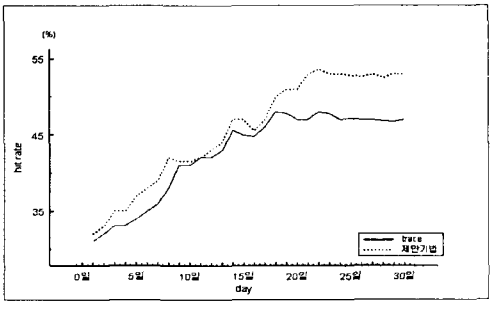

그림 1 날짜 흐름에 따른 hit rate

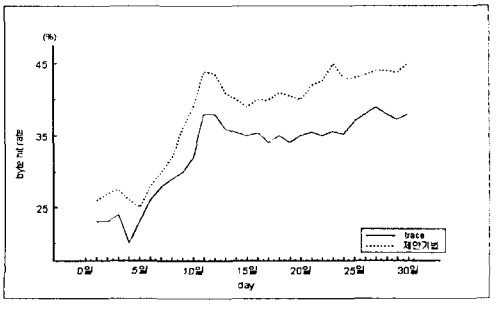

그림 2 날짜 흐름에 따른 byte hit rate

5. 결 론

웹에는 다양한 크기의 파일 타입이 존재하는데, 기존의 웹 캐쉬에서는 요청되는 객체의 가변적인 크기를 고려하지 않아 캐쉬의 hit rate와 byte hit rate를 감소시키는 문제점이 있다. 이를 해결하기 위해 캐쉬공간을 특정 비율로 고정시켜 분할하는 방법은 분할된 각 영역에 같은 교체 알고리즘을 적용함으로써 분할된 캐쉬의 장점을 다 활용하지 못한다는 단점이 있다.

본 논문에서는 웹 캐쉬를 요청되는 객체의 파일 타입에 따라 Graphic, Text, Etc 3부분의 클래스로 분할하고 각각의 클래스에 LRFU, LRFU, SIZE 교체 알고리즘을 적용시켜, 1주일 단위로 분할비를 갱신시키는 알고리즘을 제시하여 성능이 향상됨을 알 수 있었다.

향후 연구로는 보다 효율적인 파일 타입별 클래스 분류와 캐쉬 분할비 갱신을 위한 최적의 주기를 도출하는 것이다.

참고 문헌

[1] S. Williams, M. Abrams, C. R. Standrige, G. Abdulla, and E. A. Fox, "Removal policies in network caches for world-wide web document", In Processings of ACM SIGCOMM '96, Aug. 1996.

[2] S. Williams, M. Abrams, C. R. Standrige, G. Abdulla, and E. A. Fox, "Caching proxies : *Limitations and potentials*", *Proc. of 4th International* WWW conference, Dec. 1995

[3] M. F. Arlitt and C. L. Williamson "Web server workload characterization: The search for invariants", In Proc. SIGMETRICS, Philadelphia, PA, Apr. 1996. ACM

[4] Cristina Duarte Murta, Virgilio Almeida, and Wagner Meira, Jr. "Analyzing performance of partitioned caches for the WWW". In Proceeding of the 3rd International WWW Caching Workshop, June, 1998.

[5] Jia Wang. "A survey of web caching schemes for the Internet". AGM Computer Communication Review, October, 1999

[6] Pei Cao and Sandy Irani. "Cost-aware WWW proxy caching algorithms". In Processings of USENIX Symposium on Internet Technology and Systems, December, 1997.

[7] D. Lee, J. Choi, J. Kim, S. Noh S. Min Y. Cho, and C. Kim, "LRFU replacement policy: a spectrum of block replacement policies," In IEEE Transactions on Computers, 1996.

[8] 임재현, "프락시 서버 성능 향상을 위한 파일 타입 기반 캐슁 기법", 정보과학회 논문지(A), 제25권, 제8호 1998

[9] 최수영, 문진용, "인터넷 서버 부하 경감을 위한 효율적인 캐쉬 알고리즘", 한국정보처리학회 논문지, 제 8-C권, 제 2호, 2001