동적 공간분배에 의한 적응형 웹캐시 대체정책

이수행[0] 정진하 최상방
인하대학교 정보통신공학과
leesean@shinbiro.com, g2001131@inhavision.inha.ac.kr, sangbang@inha.ac.kr

Adaptive Web Cache Replacement Policy using Dynamic Distribution of Partitions in Proxy Server

Soo-Haeng Lee[0] Jin-Ha Chung Sang-Bang Choi
Dept. of Information and Communications Engineering, Inha University

요 약

인터넷이라는 환경이 우리의 생활의 일부로 자리잡으면서, 급속히 늘어난 사용자들의 요구는 그 만큼 네트워크의 현저한 부하증가를 일으키고 네트워크의 성능저하를 유발하며 속도 면에서도 빠른 응답을 원하는 사용자들의 욕구를 충족시키지 못하게 된다. 이에 대한 하나의 대응책이, 프락시 서버를 사용함으로써 네트워크 대역폭을 효과적으로 절약하고 서버 측의 부하를 감소시키며 사용자의 요청에 대한 빠른 응답이 가능하게 하는 것이다. 그러나 프락시 서버는제한된 캐시용량 때문에 새 개체를 위한 공간확보를 위해 기존 개체를 제거해야 하는데, 캐시의 성능을 최대화하도록 하는 효율적인 캐시대체정책이 필요하다. 기존의 대체정책이 캐시성능판단의 두 기준인 히트율(Hit Rate)과 바이트히트율(Byte Hit Rate)을 만족시키지 못하던지 혹은 불필요한 개체에 공간을 낭비하는 등 최대한의 공간활용을 못하는 단점을 가지고 있다. 본 논문에서는 캐시를 상위층과 하위층의 2단계로 나누어 운용하면서, 상위층은 분할된 여러개의 파티션으로 관리하여 히트율과 바이트율을 높게 유지하고 하위층은 상위의 각각의 파티션들에 추가적으로 필요한 캐시공간을 제공함으로서 동적인 파티션공간분할 효과를 제공하는 프락시 서버의 캐시구조와 캐시대체정책을 제안한다.

1. 서론

인터넷의 사용자가 급격히 늘어가면서 기하급수적으로 증가하는 네트워크부하를 감당해내기 위한 많은 대안들 중의 하나가, 인터넷의 다양한 사용분야 중 큰 비중을 차지하는 World Wide Web(WWW)에서의 네트워크 성능 향상의 방안으로서 프락시 서버를 사용하는 것이다.

프락시 서버란 서버와 클라이언트간에 위치하여 보안과 캐시를 목적을 하는 서버를 일컫는다. 전자의 경우는 방화벽으로서의 역할을 의미하며, 후자의 경우는 임의의 클라이언트에 의해 접근된 객체를 저장하였다가 다음 번에 다시 그 객체가 요구될 때 빠르게 처리할 수 있도록 미리 저장해 두는 것을 의미한다. 결과적으로 네트워크 트래픽이 줄고, 클라이언트는 좀 더 빠르게 그 객체에 접근할 수 있게 되는 것이다. 그런데, 캐시공간은 한정되어 있으므로 계속적인 클라이언트의 수많은 접근들을 모두 수용하는 것은 불가능하다. 결국, 기존에 캐시에 저장된 객체들을 제거하여 새로운 객체를 저장하기 위한 공간을 마련해야한다. 이를 위해 필요한 것이 캐시성능을 최선으로 유지할 수 있도록, 캐시 내에 존재하는 기존 객체들 중에서 제거되기 위한 최적의 객체를 선정하는 일련의 과정인 캐시대체정책이다. 이에 대한 많은 연구들이 있었고 여전히 계속되고 있다.

2. 웹캐시의 성능판단 기준

웹캐시는 전통적 파일캐시나 가상메모리 시스템의 캐시를 위한 성능판단 기준으로 평가하기에 무리가 있다. 왜냐하면 웹캐시 상에 저장되는 객체의 크기는 수백 바이트에서 수십 메가바이트까지 다양하기 때문에 일정크기의 객체만을 고려한 단순한 히트율(Hit Rate)만으로는 성능판단을 할 수 없는 것이다. 그러므로 객체의 크기를 고려한 바이트 히트율(Byte Hit Rate)이라는 또 하나의 성능판단 기준이 요구된다. 히트율은 클라이언트들이 요청한 개체 수에 대한 캐시에 의해 응답되는 개체 수의 비율을 뜻하며, 바이트히트율은 클라이언트들이 요청한 개체들의 전체 바이트수에 대한 캐시에 의해 응답되는 개체들의 바이트수의 비율을 의미한다.

3. 기존의 대표적 캐시대체 알고리즘

대표적인 것들로 FIFO, SIZE, LFU, LRU 등이 있으며 최근 논문들에서도 나름대로의 최선의 알고리즘들(GDSF, LFU-DA, TYPE 등)을 제시하고 있다[1,2].

3.1 기존의 대표적 알고리즘과 최근의 연구들

*SIZE:*추가적 공간확보가 필요할 시에 크기가 큰 개체를 우선적으로 대체시킨다.

*LFU(Least Frequently Used):*캐시에 저장된 후로부터 참조된 횟수가 제일 적은 개체를 우선적으로 대체시킨다.

*LRU(Least Recently Used):*참조된 후 가장 오래 경과

된 개체를 우선적으로 대체시킨다.
*GDSF, LFU-DA:*개체의 크기, 참조회수 및 에이징 요소를 모두 고려하여 계산한 키값을 기준으로 대체시킬 개체를 결정한다. 전자의 경우, Size-based Algorithm인 GDS에 Frequency를 추가하여 개체의 요구빈도를 수용하였으며, 후자는 Frequency-based Algorithm인 LFU에 에이징요소를 적용함으로써 다른 알고리즘들에서 볼 수 있는, 사용도가 없는 불필요한 개체로 인해 유발되는 캐시오염문제를 개선한다[1].

3.2 기존 대체 알고리즘의 문제점 및 개선방안

최근의 연구들을 포함한 지금까지의 알고리즘들은 두 성능판단 조건 중 하나만을 만족하거나 아니면 어느 조건도 만족하지 못한다. 또한 필자가 참여한 선행된 연구에서 제안된 TYPE 알고리즘[2]에서는 두 성능판단 조건에서 어느 정도 만족할 만한 성능을 보였음에도 불구하고, 불필요한 개체에 공간을 낭비하는 등 최대한의 공간 활용을 못하는 단점을 가지고 있다.
그러므로 앞에서 언급한, 에이징요소를 포함한 알고리즘과 이전 연구의 TYPE 알고리즘을 조합하여 히트율과 바이트히트율의 개선 및 캐시오염문제 해결의 두 성과를 얻고자 한다. 아울러 동적인 캐시공간 분배를 제공함으로써, 변화되는 클라이언트의 관심도 또한 수용하고자 한다.

4. 제안된 캐시대체 정책

본 논문에서는 캐시를 상위층과 하위층의 2단계로 나누어 운용하면서, 상위층은 분할된 여러개의 파티션으로 관리하여 히트율과 바이트율을 높게 유지하고, 하위층은 상위의 각각의 파티션들에 추가적으로 필요한 캐시공간을 제공함으로서 동적인 파티션공간분할 효과를 제공하는 알고리즘을 선보인다.

.

4.1 캐시구조 및 대체 알고리즘의 특징

이중 캐시구조:캐시는 상위층과 하위층을 갖춘 이중 캐시구조를 가진다. 상위층은 파일타입에 기초한 파티션들로 구성되고 하위층은 모든 파일타입을 수용할 수 있는 하나의 파티션으로 구성된다.
파일타입에 기초한 상위층 캐시의 분할(TYPE):상위층 캐시는 파일타입에 의해 graphic, text/html, audio, video, CGI, formatted, unknown 의 7개 파티션으로 분할된다. 이는 Martin F. Arlitt의 연구[3]에서 알려진 대로, 같은 타입의 파일들은 비교적 비슷한 크기의 파일로 구성되기 때문에 타입별로 일정량의 캐시공간을 보장하면 히트율면에서 우수한 SIZE알고리즘에 준하는 히트율을 얻을 수 있음에 기인한다.
상위층 각 파티션 및 하위층에 LFU-DA적용:분할된 상위층캐시의 파티션 내부와 하위층캐시에서는 Martin F. Arlitt의 최근 연구에서, 최적의 바이트히트율을 보이는 것으로 알려진 LFU-DA알고리즘을 적용한다. LFU-DA는 높은 바이트히트율을 제공하는 동시에 캐시오염문제를 해결한다.

4.2 캐시의 운용

그림 1은 다음의 내용을 도식화한 것이다.

Step1. 새로이 들어온 개체는 상위층 캐시에 타입별 해당 파티션에 저장된다.
Step2. 상위층 해당파티션이 가득찬 경우

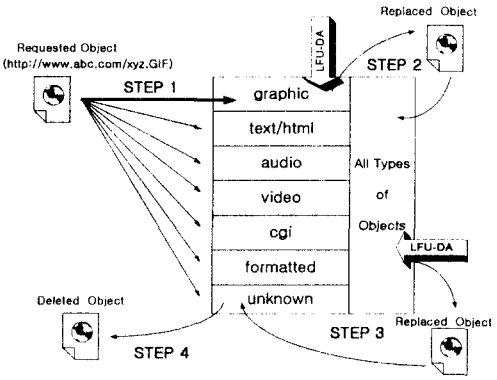

그림 1. 이중 캐시구조 분할 캐시 알고리즘

- 새로운 개체를 위한 공간 마련을 위해 상위층에서 대체 알고리즘(LFU-DA)이 적용된다.
- 밀려난 개체는 하위층으로 이동하여 저장된다.

Step3. 하위층 공간이 가득찬 경우
- 상위층에서 밀려나 하위층으로 이동되는 개체를 위한 공간확보를 위해 대체 알고리즘(LFU-DA)이 적용된다.
- 밀려난 개체는 상위층의 비사용 공간에 저장된다.

Step4. 상위층 비사용 공간에 임시저장된 개체는 새로운 개체를 위한 공간이 필요시 Step2에 우선해서 제거된다.
- 여기에서 밀려난 개체는 캐시로부터 완전히 제거된다.

4.3 캐시의 동적 공간 재분배효과

하위층 캐시공간은 상위층 캐시에서 밀려난 모든 타입의 파일을 모두 수용하며, 이 곳에 저장된 파일들이 히트하게 되면 히트된 파일은 상위층으로 옮겨 저장된다. 하위층 캐시는 인기도에 의한 임의의 타입에 편중된 사용자요청의 충격에 대해 추가적 공간을 제공함으로써 완충작용을 하며 이는 단기간 적으로 클라이언트의 요구가 급격히 늘어나는 개체를 위해 상대적으로 많은 추가공간을 제공하게 되어 각 파티션간에 동적인 공간재분배 효과를 가져다 주게 된다. 그리고 상위층 캐시에 주기적으로 하위층공간 사용 비율을 참조하여 상위 파티션들의 공간을 재분배하여 하위층 사용도가 높은 타입의 파티션에 더 많은 추가공간을 배정을 하게 된다. 상위층에 임시 저장된 개체 역시 히트되면 상위층 해당 파티션에 저장되게 된다. 결국 캐시 내의 모든 공간의 사용도를 극대화 시키게 되는 것이다.

5. 트레이스-드리븐 시뮬레이션

본 논문의 시뮬레이션에서 사용하게 될 트레이스 데이터는 Virginia Tech에서 공개한 웹 트레이스 데이터 중 workload U를 사용하였는데 이는 이전의 연구[2]에서도 사용된 바 있다.

Undergrad(U): 1995년 4월부터 10월까지 190일간 2.19GB의 정적인 웹 페이지를 요청한 173,384개의 유효한 접근을 포함한다.

시뮬레이션은 위에 제시된 웹 트레이스 데이터에 대한 히트율과 바이트히트율에 대해서 기존의 알고리즘들과 본 논문에서 제안한 알고리즘의 성능을 비교하는 방식을 채택할 것이다. 캐시를 완전히 비운 상태에서 웹 트레이스에 기초해 제공되는 데이터를 사용자 요청으로서 받아들여 히트율과 바이트히트율을 측정하게 된다. 결과는 사용자요청의 증가에 따른 히트율 및 바이트히트율의 변화를 그래프로 나타낸다.

6. 시뮬레이션 결과

시뮬레이션 결과는 다음의 그래프에서 보여지듯이 히트율과 바이트히트율의 그래프에서의 성능 비교로 이루어진다.

그림2는 트레이스 데이터의 연속요청이 증가할때 각 알고리즘이 보이는 히트율의 변화를 나타낸 것이다. 그림에 따르면 SIZE알고리즘이 히트율면에서 최상의 결과를 보인반면 LRU알고리즘은 가장 떨어지는 히트율을 보였다. 그 이유는 SIZE알고리즘이 타 알고리즘에 비해 동일공간내에 크기가 작은 많은 수의 파일을 수용하게 되고, 대부분의 파일 요청이 작은 크기의 개체에 집중하기 때문이다. 본 논문에서 제안된 이중구조의 분할캐시알고리즘은 TYPE알고리즘에서의 성능을 갖게되므로 SIZE알고리즘에 준한 높은 히트율을 보였다. 이는 분할된 파티션에 존재하는 파일들은 비슷한 크기를 갖게 되어 일정수준의 히트율을 보장하기 때문이다. 그뿐만 아니라, 사용자요청이 한 타입에 급격히 몰릴 때에는 그 타입의 요청수에 비례하는 하위의 추가공간을 배정받게 되어 SIZE알고리즘보다 오히려 나은 성능을 나타냈다

그림3은 데이터 연속요청에 대한 바이트 히트율을 보여준다. 결과는 예상대로 LRU가 높은 바이트히트율을 나타냈고, SIZE알고리즘이 가장 좋지않은 바이트히트율을 나타냈다. 바이트히트율면에서도 제안된 이중구조의 분할캐시알고리즘은 LRU 만큼이나 좋은 성능을 보여주었고 특히나 평균크기가 큰 타입에 요청이 몰릴 때에는 두드러진 성능을 보였고 평균크기가 작은 타입에 요청이 몰릴 때에도 미미하지만 성능향상을 볼 수 있었다. 그 이유는 분할된 파티션내에 바이트히트율이 우수한 LFU-DA알고리즘을 사용하여 캐시오염문제를 해결함으로써 공간활용의 효율도를 높이고 큰 사이즈의 개체들이 그만큼 많이 히트되게 하여 총 히트된 바이트의 양이 늘어났기 때문이다..

7. 결과분석

이전에 제안된 TYPE알고리즘과 비교할 때 본 논문에서 제안한 2중구조 캐시대체정책이 전반적으로 어느정도 향상된 성능을 보였음에도 불구하고 그 차이는 그리 현저하지 않았다. 그러나 우리가 주목할 점은 다음에 있다. 전체적인 성능향상 외에도 제안된 알고리즘은 클라이언트의 관심이 반영된 단기간동안의 임의의 파티션에 편중되는 요청을 효과적으로 수용하여 어느 정도의 급격한 히트율 하락을 방지한다는 점이다. 이는 인기도에 민감한 Frequency-Based알고리즘인 LFU-DA의 특징이기도 하며 공동사용공간인 하위층 캐시공간을 많이 배정함으로써 얻게 되는 동적공간재분배 효과때문이기도 하다.

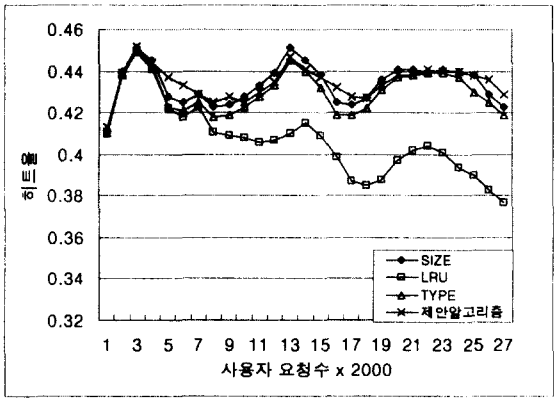

그림 2. 사용자의 요청 증가에 따른 히트율의 변화

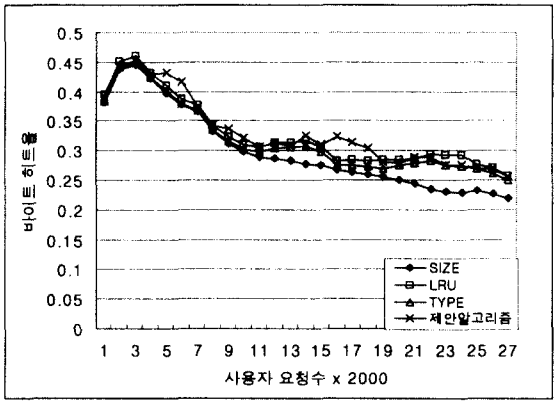

그림 3. 사용자의 요청 증가에 따른 바이트히트율의 변화

8. 참고 문헌

[1] M.Arlitt, L.Cherkasova, J.Dilley, R.Friedrich and T. Jin, "Evaluating Content Management Techniques for Web Proxy Caches", ACM SIGMETRICS Performance Evaluation Review, Vol. 27, No. 4, pp. 3-11, March 2000.

[2] H.J.Doo, S.H.Lee, J.H.Chung and S.B.Choi, "Cache Replacement Policy for Proxy Server using Type Based Partitioning," In Proceedings of IC2001, Las Vegas, USA, June 2001.

[3] M.Arlitt and C.Williamson,"Web Server Workload Characterization: The Search for Invariants", In Proceedings of the 1996 ACM SIGMETRICS Conference on the Measurement and Modeling of Computer Systems, Philadelphia, PA, pp. 126-137, May 23-26, 1996