웹 캐싱에서 갱신 위험도 기반 TTL 추정 방식*

이정준† °, 황규영†, 이병석‡
†한국과학기술원 전산학과, 첨단정보기술 연구센터
‡버먼트대학교 전산학과

Update-Risk based TTL Estimaiton in Web Caching

Jeong-Joon Lee†, Kyu-Young Whang†, Byung-Suk Lee‡
†Department of Computer Science and
Advanced Information Technology Research Center(AITrc)
Korea Advanced Institute of Science and Technology(KAIST)
‡Department of Computer Science
University of Vermont

요 약

웹 캐싱은 캐쉬 액세스를 통해 웹 서버와 네트워크의 부하를 감소시켜 웹 응용을 가속화하는 중요한 기술이다. 전통적인 데이타 캐싱과 마찬가지로, 웹 캐싱은 캐쉬 일관성 유지라는 문제를 안고 있다. 그러나, 기존의 캐싱과는 달리 웹 캐싱에서는 웹 서버의 데이타 갱신을 지연하여 반영하는 약 일관성이 허용된다. 이러한 조건은 TTL(time-to-live, 캐쉬 서버가 캐쉬된 데이타 아이템이 유효하다고 기대하는 시간)이 일관성 유지를 위해 사용되는 것을 허용한다. 이것은 효과적인 TTL 추정방법의 개발이 필요하도록 하였다. 그러나, 현재까지 소개된 두 가지 추정방법(고정 TTL방법과 휴리스틱 방법)은 직관적 해석이 어렵고, 이론적인 추정근거가 빈약하다. 본 논문에서는 이러한 단점을 보완하기 위하여 확률적 분석에 기반하여 정형적이고, 직관적인 의미를 갖는 위험도 기반 TTL 설정 방법을 제안한다. 이 방법에서는 위험도를 TTL 이내에 원본 데이타가 갱신될 확률로 정의하고, 갱신분포를 포아송 과정으로 가정한 후, 주어진 위험도에 기반한 TTL 식을 유도한다. 위험도 기반 TTL 설정 방법은 기존방법과 비교하여 위험도란 개념을 통하여 보다 직관적이고, 확률적 유도를 통하여 TTL 설정에 대한 이론적인 근거를 제공한다.

1. 서론

웹이 주요한 정보전달(information dissemination)을 위한 기반으로 성장함에 따라, 웹 상에서 교환되는 데이타의 양도 증가하고 있다. 이러한 웹 데이타의 폭발적인 증가는 웹 서버와 네트워크에 과부하를 일으켜 성능저하를 유발한다. 이 문제를 해결하기 위해서, 웹 캐싱은 자주 사용되는 웹 페이지를 클라이언트 쪽에 보관한다 [1, 2]. 최근들어, 이 분야에 대한 활발한 연구와 상용화가 활발히 이루어져 Inktomi, Imimic, CacheFlow, InfoLibria과 같은 상용제품들이 출시되었다 [2].

다른 종류의 캐쉬된 데이타와 마찬가지로 웹 캐쉬에 캐쉬된 웹 페이지는 웹 서버에 있는 원본 데이타의 복사본이다. 따라서, 캐쉬된 웹 페이지는 웹 서버의 원본 데이타와 동기화되어야 한다 [8]. 이와 같이, 원본 데이타와 캐쉬된 데이타를 동일하게 유지하는 것을 일관성 유지라 한다 [5, 8]. 그러나, 다른 종류의 캐쉬된 데이타와는 달리 웹 페이지에 적용되는 일관성 요구는 비교적 약하다. 즉, 동기화가 지연되어 사용자가 갱신전 데이타를 보는 것을 인정한다 [5, 6].

웹 캐쉬에서는 일관성을 유지하기 위하여 TTL 방법을 많이 사용한다 [6]. TTL 방법에서는 캐쉬된 시점[a]부터 데이타의 유효기간을 의미하는 TTL 동안 데이타의 유효성을 인정한다 [6]. TTL은 웹 서버로부터 데이타와 함께 받는 *Expire* 필드나 *max-age* 필드 값을 이용하여 결정된다. 그러나, 현실적으로 이 값을 갖지 않는 데이타의 비율이 크므로 [1], 다른 값을 이용하여 TTL을 추정하는 여러 방법들이 제안되었다 [3, 4, 6]. 그 예로는 고정시간 방법 [7]과 휴리스틱 방법 [3, 6]이 있다. 전자는 모든 데이타 아이템에 동일한 TTL값을 설정하는 방법인 반면, 후자는 원본 데이타의 마지막 갱신시점 (*Last-Modified* 필드에 저장됨)부터 캐싱 시점까지 구간의 일정비율을 할당하는 것이다.

기존의 두 가지 방법은 그 근거가 약하고, 직관적으로 이해하기 어렵다. 첫째, 두 가지 방법은 정형적인 이론에 근거하지 않는다. 둘째, 추정한 TTL에 직관적인 의미를 부여하기 어렵다. 실례로, 고정시간 방법은 각 웹 페이지의 갱신빈도를 무시하므로, 웹 서버의 갱신 경향을 반영하지 못한다. 그리고, 휴리스틱 방법은 현재부터 마지막 참조 시간까지의 시간적 거리의 비율을 임의로 설정하고 있으므로, 사용자에게는 특별한 의미를 주지 못한다. 그 해결책으로서, 본 논문에서는 타당한 근거와 직관적인 의미를 갖는 위험도기반 TTL 추정방법을 제안한다.

위험도기반 TTL 추정방법에서의 위험도란 설정된 TTL이 만료되

* 본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원을 받았음.

[a] 본 논문에서는, 시간의 구간과 구별하여 한 순간을 의미하는 용어로 "시점"이란 용어를 사용한다.

기 전에 원본 데이타가 갱신되어 캐쉬된 데이타와 원본 데이타와 달라질 확률로 정의한다. 즉, TTL 동안에는 캐쉬된 데이타가 유효하다는 가정이 틀릴 확률이다. 반면, 위험도의 1 보수(1's complement)는 TTL이 만료할 때까지 캐쉬된 데이타가 최신 데이타로 남아 있을 확률이므로 신뢰도를 의미한다. 이와 같은 위험도(혹은 신뢰도) 개념은 사용자에게 직관적이다. 본 논문에서는 갱신 회수를 포아송 과정으로 모델하고, 주어진 위험도를 갖는 TTL을 결정하는 확률적 방법을 제안한다. 이 방법에서는, 동일한 신뢰도를 유지하기 위해서, 모든 데이타 아이템이 동일한 위험도를 갖도록 각 데이타 아이템의 TTL 값이 결정된다.

본 논문의 구성은 다음과 같다. 제 2 절 관련 연구에서는 기존의 TTL 설정 방법을 설명한다. 그리고, 제 3 절에서는 위험도를 제안하고, 제 4 절에서 이를 기반으로 TTL 결정식을 유도한다. 그리고, 제 5 절에서 결론을 맺는다.

2. 기존의 TTL 추정방법

본 절에서는 기존의 TTL 추정방법에 대해 설명한다. TTL을 설정하는 기존의 방법들인 고정시간 방법(fixed time method) [3]과, 휴리스틱 방법(heuristic method) [6]은 순수한 시간기준 방법이다. 즉, 이 방법들은 TTL을 설정하는 기준으로 시간만을 고려하였다. 본 절에서는 이 방법들에 대해 간단히 설명하고, 그 문제점들을 지적한다.

고정시간 방법은 마이크로소프트의 IIS [7]에 사용되는 방법이다. 이 방법은 모든 데이타에 대하여 동일한 TTL을 사용하는 방법으로 구현이 단순하고 용이하다. 그런데, 실제로는 데이타 아이템들의 갱신빈도가 각기 다르고, 동일한 데이타 아이템도 갱신 시간에 따라 다른 갱신빈도를 가질 수 있다. 따라서, 이 방법은 데이타 아이템의 종류나 갱신 시간에 따라 다른 갱신 경향을 전혀 고려하지 않는다. 결국, 이 방법은 적정한 TTL 설정에 있어서 신뢰할 수 없다는 단점이 있다. 만일 TTL이 너무 크게 설정되면, 갱신전 데이타를 필요 이상으로 오래 사용하게 된다. 그리고, TTL이 너무 작게 설정되면, 웹 서버에 불필요한 재유효화를 너무 자주 수행하여 시스템과 네트워크 자원을 낭비하게 된다.

HTTP/1.1 표준 [6]과 참고문헌 [3]에 제안된 휴리스틱 방법은 캐쉬 진입시점인 현재시점에서 마지막 갱신시점까지 거리의 일정비율을 TTL로 사용한다. 즉, 그림 1에서와 같이 $(now - Last\text{-}Modified) \times M$값을 TTL로 설정한다. 이 때, M값은 휴리스틱하게 결정되는 임의의 비례상수로서, HTTP/1.1에서는 0.1이하의 임의의 값을 권유하고, 참고문헌 [3]에서는 0.5를 사용한다. 그런데, 이 방법들은 추측과 경험으로 M값의 상한을 선택했을 뿐, 이 값의 설정에 대한 아무런 논리적인 근거가 없다. 이 방법은 갱신경향 반영에는 고정시간 방법보다 우수하지만, 갱신이 규칙적으로 일어나지 않으면, 고정시간 방법과 마찬가지로 유연하지 못하다. 즉, 마지막 참조시간만을 이용하여 갱신경향을 결정한다. 또한, 사용자에게 M값의 의미가 직관적이지 못하다.

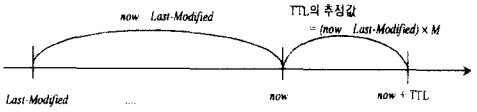

그림 1. 휴리스틱 방법의 TTL추정.

이러한 문제점들로 인해, 실제 갱신 발생에 보다 잘 적응할 수 있는 새로운 TTL 추정방법이 요구된다. 그리고, 추정한 TTL에 대하여 타당한 의미해석이 가능하도록 이론적인 배경이 있는 정형적인 방법이 필요하다.

3. 위험도

캐쉬된 데이타 아이템에 대한 TTL 설정은 그 데이타 아이템이 TTL동안에 유효할 것이라는 가정에 기반한다. 그러나, 이 가정은 틀릴 수 있으므로, 그 기간에도 원본 데이타는 갱신될 수 있다. 이와 같이 TTL 동안에 원본 데이타가 갱신될 확률을 위험도라고 하며, 다음과 같이 정의한다.

정의 1: 캐쉬된 데이타 아이템 x의 **위험도** ρ_x란 설정된 TTL l_x동안 x의 원본 데이타가 갱신되어 캐쉬된 데이타와 달라질 확률이다. □

그림 2는 데이타가 갱신될 확률밀도분포로부터 얻을 수 있는 위험도를 그래프 상에서 나타낸 것이다. 정의 1에 따르면, 위험도(ρ_x)는 캐쉬된 현재시점 now에서 미래시점 $(now + l_x)$ 사이에 갱신이 일어날 확률이다. 이 확률 ρ_x는 그림에서와 같이 now에서 $(now + l_x)$까지의 구간에 대하여 확률밀도분포를 적분하여 구할 수 있다.

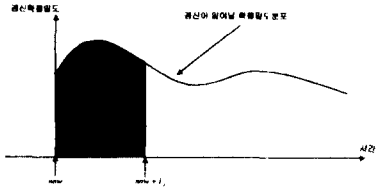

그림 2. 데이타의 갱신확률밀도분포와 위험도와의 관계.

정의 1에서는 위험도를 웹 서버 관점에서 설명한 것이다. 반면에, 정의 2에서는 클라이언트의 관점에서 위험도를 다르게 정의할 수 있다.

정의 2: 캐쉬된 데이타 아이템 x의 **위험도** ρ_x란 설정된 TTL l_x동안 x에 대한 임의의 접근이 무효화된 데이타 아이템을 참조할 최대 확률이다(즉, 클라이언트가 무효(invalid) 데이타 아이템을 참조할 확률은 위험도를 넘지 않는다.). □

그림 3에서 t_{ref}는 클라이언트가 x를 참조하는 시점이다. 그러면, t_{ref}에서 무효화된 x를 참조할 확률은 now부터 t_{ref}까지의 구간에서 x가 갱신될 확률이다. 따라서, 이 확률은 now부터 t_{ref}에서 갱신확률밀도분포를 적분한 값이며, 그림에서 회색 부분에 해당한다. 그리고, now에서 캐쉬된 데이타 아이템 x는 $(now + l_x)$ 이후에는 유효하지 않은 것으로 간주하므로 이후의 요청에서는 원본 데이타를 다시 참조하여 캐쉬한다. 따라서, 위험도는 참조시점 t_{ref}가 $(now + l_x)$일 때, 최대가 되며 그 값은 갱신확률밀도를 $(now \sim now + l_x)$동안 적분한 값과 동일한 값이다. 따라서, 클라이언트가 무효 데이타를 참조할 확률은 위험도 이하이다.

4. 위험도 기반 TTL 추정

본 절에서는 확률적 분석 방법을 통하여 TTL 추정식을 유도한다. 본 논문에서는 갱신사건이 독립이라는 가정하에 갱신사건의 수를 포아송 과정으로 모델한다. 전통적인 포아송 과정과 같이 $N(T)$는 시간 T 동안 발생한 사건의 수를 의미하고, $P[C]$는 조건 C가 참일 확률을 의미한다. 그리고, 포아송 과정의 발생율인 μ_x는 단위시간당 발

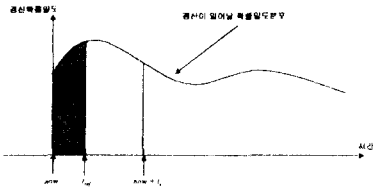

그림 3. 캐쉬 참조시점 t_{ref}에서 무효 데이타를 참조할 확률.

생한 사건의 평균회수(즉, 평균 갱신빈도)를 의미한다. 이 때, 각 데이타 아이템의 위험도를 정리 1로 구할 수 있다.

정리 1: 데이타 아이템 x의 단위시간당 평균 갱신회수가 μ_x이고, TTL이 l_x일 때, x의 위험도 ρ_x는 다음과 같다.

$$\rho_x = 1 - e^{-\mu_x l_x} \quad (1)$$

증명: 위험도 ρ_x는 l_x 동안 갱신이 한번 이상 발생할 확률이므로, $\rho_x = P[N(l_x) > 0] = 1 - P[N(l_x) = 0]$이 성립한다. 그리고, 포아송 과정의 정의에 따라, $P[N(l_x) = 0] = e^{-\mu_x l_x}$이다. 따라서, ρ_x는 $1 - P[N(l_x) = 0] = 1 - e^{-\mu_x l_x}$이다. □

정리 1에 따라 캐쉬된 데이타 아이템 x의 위험도 ρ_x를 TTL l_x의 함수로 나타내면, 그림 4과 같다. 그림을 보면, l_x가 작은 구간에서는 l_x의 증가에 따라 위험도가 급격히 증가하고, l_x가 큰 구간에서는 위험도가 완만하게 증가하며 1로 수렴함을 알 수 있다.

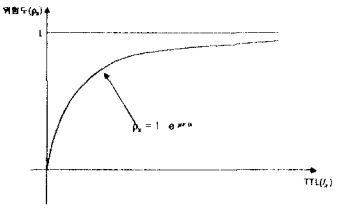

그림 4. 데이타 아이템 x에 대하여 TTL의 함수로 표현된 위험도.

식 (1)로부터 TTL l_x를 위험도 ρ_x의 함수로 다음과 같이 나타낼 수 있다.

$$l_x = -\frac{1}{\mu_x}\log(1-\rho_x) \quad (2)$$

식 (2)을 그래프로 나타내면, 그림 5과 같다. 그림을 보면, 위험도 ρ_x가 커질수록 x의 TTL 값인 l_x가 커짐을 알 수 있다. 이는 위험도가 커질수록 큰 TTL값을 부여해야 함을 의미한다.

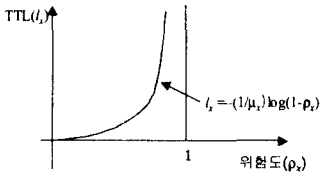

그림 5. 데이타 아이템 x에 대하여 위험도의 함수로 표현된 TTL.

본 논문에서는 데이타 아이템 x의 평균 갱신빈도를 μ_x로 표기한다. 그리고, 실제 μ_x는 LRU-K와 같이 캐쉬로그 화일에 기록된 과거의 K번의 갱신사건 동안에 포함된 연속된 갱신사건의 평균거리의 역수로 추정할 수있다.

5. 결론

본 논문에서는 클라이언트쪽의 캐쉬 서버에서 요구되는 약 일관성 환경에 적합한, 새로운 TTL 추정방법을 제안하였다. 제안한 방법은 TTL 동안 원본 데이타가 갱신될 확률로 정의되는 위험도에 기반하고 있다. 주어진 최대 허용 위험도에 기반하여 캐쉬된 데이타 아이템의 TTL이 추정된다. 이에 따라, 갱신사건의 수가 포아송 과정을 따른다는 가정하에 주어진 위험도에서 데이타 아이템의 TTL 값을 결정하는 식을 유도하였다. 평균 갱신빈도로 정의되는 포아송 상수는 이전에 발생한 K개의 연속된 갱신사건의 평균거리를 이용하여 추정할 수있다.

위험도 기반 TTL 설정방법은 TTL을 설정하는 새로운 방법으로서, 화일 형태로 존재하는 정적인 데이타 뿐만 아니라, 요청시점에 생성되는 동적 데이타에도 유용하게 사용될 것으로 기대된다. 왜냐하면, 웹 서버는 동적인 데이타에 대해서 TTL을 부여하지 않으므로, 캐쉬 서버가 추정해야하기 때문이다.

제안한 방법은 사용자가 직관적으로 이해할 수 있는 정형적인 TTL 설정 방법으로서, TTL기반 일관성 유지에 대한 명확한 이해와 정형적인 기반을 제공한 것으로 사료된다.

Reference

[1] Arlitt, M. F. and Willaiamson, C. L. "Web Server Workload Characterization: The Search for Invariants," In *Proc. Int'l Conf. on Measurement and Modeling of Computer Systems*, ACM SIGMETRICS, Philadelphia, pp. 126-136, May 1996.

[2] Barish, G. and Obraczka, K. "World Wide Web Caching: Trends and Techniques," *IEEE Communications*, Vol. 38, No. 5, pp. 178 184, May 2000.

[3] Chankhunthod, A., Danzig, P. B., and Neerdaels, C., "A Hierarchical Internet Object Cache," In *Proc. USENIX Technical Conf.*, USENIX Association, San Diego, Calif., pp. 153 164, Jan. 1996.

[4] Colajanni, M. and Philip, S. Y. "Adaptive TTL Schemes for Load Balancing of Distributed Web Servers," In *Performance Evaluation Review*, ACM SIGMETRICS, Vol. 25, No. 2, pp. 36 42, Sept. 1997.

[5] Gwertzman, J. and Seltzer, M., "World-Wide Web Cache Consistency," In *Proc. USENIX Technical Conf.*, USENIX Association, San Diego, Calif., pp. 141-152, Jan. 1996.

[6] Fielding, R., Gettys, J., Mogul, J. C., Frystyk, H., and Berners-Lee, T., Hypertext Transfer Protocol - HTTP/1.1. RFC 2616, http://nic.ddn.mil/ftp/rfc/rfc2616.txt, June 1999.

[7] Internet Information Services 5.0 Technical Overview, http://www.microsoft.com/windows2000/docs/, 2001.

[8] Liu, C., and Cao, P., "Maintaining Strong Cache Consistency in the World-Wide-Web," *IEEE Trans. on Computers*, Vol. 47, No. 4, pp. 445 457, Apr. 1998.