웹 캐쉬에서 만기시간의 영향을 고려한 유효참조확률

(Effective Reference Probability Incorporating the Effect of Expiration Time in Web Cache)

이 정 준 † 문 양 세 † 황 규 영 †† 홍 의 경 †††

(Jeong-Joon Lee) (Yang-Se Moon) (Kyu-Young Whang) (Eui-Kyung Hong)

요 약 웹 캐쉬는 웹 응용의 성능향상을 위한 중요한 문제가 되었다. 본 논문에서는 웹 데이타에 부여되는 만기시간(expiration time)을 활용하여 웹 캐쉬의 성능을 향상시키는 방법을 제안한다. 먼저, 기존 캐쉬 교체 알고리즘에서 사용된 참조확률에 만기시간의 영향을 반영한 **유효참조확률(effective reference probability)**의 개념을 제시한다. 그리고, 유효참조확률을 정형적으로 정의한 후, 확률적 모델 하에서 이론적으로 유도한다. 기존의 교체 알고리즘에서 참조확률을 유효참조확률로 대체하면 기존 교체 알고리즘에 만기시간의 영향을 반영할 수 있다. 성능평가 결과, 유효참조확률을 사용한 알고리즘이 그렇지 않은 알고리즘보다 항상 우수한 결과를 보였다. 이것은 제안한 방법이 만기시간을 고려하여 캐쉬효과를 얻을 이론적인 확률을 정확히 반영한 방법으로서, 만기시간의 영향을 보다 효과적으로 반영하기 때문이다. 특히, 유효참조확률은 캐쉬비율이 0.05이고 갱신이 비교적 자주 일어날 경우(갱신횟수가 참조횟수의 1/10 이상인 경우)에 LRU-2의 경우 30% 이상, Aggarwal의 방법(refresh overhead factor가 통합된 PSS)의 경우 13% 이상 성능을 개선하였다. 이 같은 결과는 유효참조확률이 만기시간이 주어지는 웹 캐쉬에서 크게 성능을 개선할 수 있음을 보여주고 있다.

Abstract Web caching has become an important problem addressing the performance issues in web applications. In this paper, we propose a method that enhances the performance of web caching by incorporating the expiration time of web data. We introduce the notion of the ***effective reference probability*** that incorporates the effect of expiration time into the reference probability used in the existing cache replacement algorithms. We formally define the effective reference probability and derive it theoretically using a probabilistic model. By simply replacing the reference probabilities with the effective reference probability in the existing cache replacement algorithms, we can take the effect of expiration time into account. The results of performance evaluation through experiments show that the replacement algorithms using the effective reference probability always outperform the existing ones. The reason is that the proposed method precisely reflects the theoretical probability of getting the cache effect, and thus, incorporates the influence of the expiration time more effectively. In particular, when the cache fraction is 0.05 and data update is comparatively frequent (i.e., the update frequency is more than 1/10 of the reference frequency), the performance enhancement is more than 30% in LRU-2 and 13% in Aggarwal's method (PSS integrating a refresh overhead factor). The results show that effective reference probability contributes significantly to the performance enhancement of the web cache in the presence of expiration time.

· 본 연구는 첨단정보기술연구센터를 통하여 한국과학재단의 지원을 받았음.

† 학생회원 : 한국과학기술원 전산학과
jjlee@mozart.kaist.ac.kr
ysmoon@mozart.kaist.ac.kr
†† 종신회원 : 한국과학기술원 전산학과 교수
kywhang@mozart.kaist.ac.kr
††† 종신회원 : 서울시립대학교 전산통계학과 교수
ekhong@venus.uos.ac.kr

논문접수 : 2000년 6월 23일
심사완료 : 2001년 7월 26일

1. 서 론

최근 들어, 웹은 주요한 정보전달 시스템(information dissemination system)으로 빠르게 성장하고 있다. 이러한 빠른 성장으로 인한 웹 데이타의 폭발적인 증가는 네트워크 상에서 병목현상(bottleneck)을 일으키는 주요한 원인이 되고 있다. 그리고, 병목현상을 해소하기 위해서, 자주 사용되는 데이타 아이템[1]을 클라이언트 가까이에 두는 캐쉬 기술에 관한 연구가 활발히 이루어지고 있다[1, 2, 3, 4, 5, 6, 7, 8].

웹 캐쉬는 여러 가지 면에서 전통적인 캐쉬와 다르다. 전통적인 페이지 기반 캐쉬는 크기, 비용, 타입에서 동일한 페이지나 블록을 하나의 캐쉬 단위로 다루므로 데이타 아이템의 특성이 다른 경우를 고려하지 않고 참조확률만을 사용하여 교체대상을 선택한다[9, 10]. 반면에 웹 캐쉬는 웹 문서를 하나의 단위로 캐쉬하며, 웹 문서는 기존의 캐쉬 단위와는 달리 다양한 크기, 참조 비용 및 만기시간(expiration time)과 같은 특성을 갖고 있다[2, 7, 8]. 이 중에서도 특히 만기시간이 캐쉬에 미치는 영향에 대한 이론적인 연구가 충분히 이루어지지 않았다.

만기시간이란 서버의 원본 데이타가 갱신되어 캐쉬된 데이타와 원본 데이타가 서로 달라져 캐쉬된 데이타가 유효하지 않게 되는 실제시점[2]으로 HTTP(HyperText Transfer Protocol)의 TTL(Time To Live)필드에 부여되어 웹 문서와 함께 전송된다[5, 10, 11]. 특히, 신문 데이타, 일기예보 데이타, 주식 데이타, 스포츠 데이타 등과 같이 주기적으로 갱신되는 데이타에 많이 사용된다. 또한, 주기적으로 갱신되지 않는 데이타라 할지라도 기존의 갱신기록을 이용하여 만기시간을 추정하는 방법들이 사용되고 있다[8, 12]. 만기시간이 경과한 데이타의 경우 캐쉬되어 있다 하더라도 다시 원본 데이타를 참조해야 하므로 참조비용[3]의 감소와 같은 캐쉬를 사용한 효과를 얻지 못한다. 따라서, 동일한 조건이라면 만기시간까지 남은 시간적 거리가 작은 데이타 아이템을 교체대상으로 선택하는 것이 캐쉬를 사용한 효과를 얻을 확률이 높아진다.

최근 들어 웹 환경이 보편화되고 많은 데이타가 만기시간을 가짐에 따라, 기존의 교체 알고리즘에 만기시간까지 남아 있는 시간적 거리를 반영하는 방법이 요구되고 있다. LRU-K[13], LFU[9]와 같은 전통적인 교체 알고리즘들에서는 각 데이타 아이템이 다음 번에 참조될 확률을 비교하여 교체 대상을 선택한다. 본 논문에서는 이 확률을 **참조확률**이라고 부른다. 만기시간이 없다면 참조확률은 캐쉬효과[4]를 얻을 확률과 동일하다. 그러나, 만기시간이 있다면 만기된 데이타 아이템은 서버로부터 원본 데이타를 새롭게 참조해야 하므로 캐쉬효과를 얻지 못한다. 따라서, 이 경우 만기시간 이전에 참조되어 캐쉬효과를 얻을 새로운 확률이 필요하다. 본 논문에서는 이와 같이 데이타 아이템이 만기시간 이전에 발생한, 다음 참조시점에서 참조될 확률을 **유효참조확률**이라고 정의한다.

본 논문에서는 유효참조확률을 정형적으로 정의하고 확률적 모델을 바탕으로 유도한다. 즉, 교체시점부터 다음 참조시점까지 시간거리에 대한 확률분포를 유도한다. 그리고, 이를 이용하여 다음 참조에서 특정 데이타 아이템이 참조되고, 참조된 실제시점이 참조된 데이타 아이템의 만기시간 이전일 확률을 유도한다. 이와 같이 유도된 유효참조확률은 참조확률과 만기시간까지 남은 시간거리에 의해 결정되는 값이다. 왜냐하면, 참조확률이 클수록 다음 참조에서 참조될 확률이 크고, 만기시간까지 남은 거리가 클수록 참조된 데이타 아이템이 유효할 확률이 크기 때문이다. 유효참조확률은 참조확률을 포괄한다. 즉, 만기시간이 없다면(무한대가 된다면), 유효참조확률은 참조확률과 동일하다. 또한, 기존의 교체 알고리즘은 참조확률 대신 유효참조확률을 사용하여 만기시간의 영향을 반영한 교체 알고리즘으로 확장될 수 있다.

본 논문의 구성은 다음과 같다. 제 2 절에서는 전통적인 교체 알고리즘의 기본 개념과 기존의 교체 알고리즘들을 설명한다. 제 3 절에서는 만기시간에 의해 교체 대상이 바뀌는 예를 통해 연구 동기를 설명한다. 제 4 절에서는 유효참조확률의 개념과 정의를 설명하고, 이론적으로 유도한다. 제 5 절에서는 기존 교체 알고리즘에 유효참조확률을 적용하는 방법을 설명한다. 제 6 절에서는 성능평가를 통하여 유효참조확률을 사용한 알고리즘이 기존 알고리즘보다 우수함을 보이고, 제 7 장에서 결론을 맺는다.

2. 관련 연구

본 절에서는 필요한 용어를 소개하고, 전통적인 페이지 기반 교체 알고리즘 연구에서부터 사용되던 기본 개

1) 크기와 참조비용이 가변적일 수 있다는 것을 나타내기 위해 페이지가 아닌 데이타 아이템이란 용어를 사용함.

2) 참조순번에 의한 이산시간이 아닌 실제시간 상의 시점

3) 본 논문에서 참조비용은 주어진다고 가정한다. 참조비용은 참조확률과는 독립적으로 결정되는 요소로서, 이전 참조시점에 이미 결정되기 때문이다.

4) 캐쉬로 인해 데이타 아이템의 참조비용이 발생하지 않는 것을 말함.

념과 기존의 교체 알고리즘들을 설명한다.

2.1 교체 알고리즘의 기본 개념

먼저, 교체 알고리즘 설명에 필요한 용어와 표기법을 소개한다. 페이지들의 집합은 $N=\{1,...,n\}$으로 표시하고, 페이지들의 읽기 요청들의 순열(sequence)을 **참조 스트링**(*reference string*)이라 하며, $\omega=r_1r_2...r_i...(r_i\in N)$과 같이 표현한다[9]. 여기에서 $r_i=x$는 i번째 참조에서 페이지 x가 참조되었음을 의미한다. 또한, i를 참조순번을 이용한 이산시간(discrete time)에서의 시점으로 해석하기도 한다[9]. 즉, i는 순열상의 위치로 해석할 수도 있고, 또한 이산시간 축상에서의 하나의 시점으로 해석할 수도 있다. 편의상 본 절에서 사용하는 '시점'은 이산시간에서의 시점을 의미한다.

교체 알고리즘은 캐쉬 미스(혹은, 버퍼 폴트) 횟수를 최소화하기 위해서 앞으로 어떤 페이지에 대해서 참조가 일어날 것인지를 예측하여 이를 교체 전략에 사용한다. 페이지 x가 시점 i에서 시작하여 처음 참조될 때까지의 시간 차이(또는 거리)를 **전방거리**(*forward distance*)라 하고 $d_i(x)$로 표시한다[14]. 만약 시점 i에서 시작하여 $l(l\geq 0)$ 번째 참조에서 x가 처음 참조되었다면(즉, $r_{i+l}=x$이면) $d_i(x)=l$ 이고, x가 한번도 참조되지 않았으면 $d_i(x)=\infty$라 정의한다. 예를 들어, 참조 스트링이 $\omega=1\ 2\ 1\ 4\ 3\ 2$라 할 때, 시점 $3(=i)$에서 페이지 $2(=x)$에 대한 전방거리 $d_3(2)$는, $r_6=2$이므로 $3(=6-3)$이 된다.

일반적으로, 교체 알고리즘이 사용하는 최적의 교체 전략은 전방거리의 기대값이 가장 큰 페이지를 교체하는 것이다[9, 13, 14]. 이와 같은 교체 전략을 사용하는 이유는, 전방거리의 기대값이 최대인 페이지를 교체하면 버퍼 폴트간의 거리를 최대화하여 폴트율의 기대값을 최소화할 수 있기 때문이다. 이러한 전략을 사용하는 최적의 알고리즘에는 페이지들에 대한 전방거리를 미리 알 수 있는지 여부에 따라서 Belady의 B_0 알고리즘[15]과 Denning 등의 A_0 알고리즘[16]이 있다. B_0 알고리즘은 참조 스트링이 미리 알려져 있다고 가정하고 현재 버퍼에 있는 페이지 중에서 전방거리가 가장 큰 것을 교체한다. 이 알고리즘은 최적의 알고리즘으로서 어떠한 참조 스트링을 입력으로 받아도 항상 최소의 비용을 발생시킨다. 이 알고리즘은 참조 스트링을 미리 알 수 있는 경우에 한하여 사용될 수 있으며, 참조 스트링을 미리 알 수 없는 경우에는 최소의 버퍼 폴트 횟수를 계산하는 기준으로만 활용되고 있다. A_0 알고리즘은 간단한 확률적 모델에 기반한 알고리즘으로서, 앞으로 발생할 참조 스트링을 모르는 상태에서 전방거리의 기대값(expected value)이 가장 큰 페이지를 교체한다. 모든 x에 대해서 $P[r_{i+l}=x]$[5] 값이 주어진다는 확률적 모델 하에, 전방거리의 기대값 $E[d_i(x)]$는 x에 대한 다음 첫 참조까지 시간적 거리의 기대값이므로, 그 값은 x가 참조될 확률의 역수 $1/P[r_{i+l}=x]$이다[9, 13]. 그래서, 이 알고리즘은 어떤 시점 i에서 $P[r_{i+l}=x]$가 가장 작은 페이지 x를 교체한다. 참조 스트링의 형태가 다음에 설명하는 독립참조모델(independent reference model)이란 확률적 모델을 따른다는 가정하에서 이 알고리즘에 의해 발생하는 비용의 기대값은 최소이다[9].

독립참조모델이란 참조 스트링 $r_1r_2...r_i...$가 독립적인 확률변수(random variable)들의 순열이고, 모든 시점 i (≥ 1)에서 이들은 모두 동일한 안정적(stationary) 확률분포(probability distribution) $\{\beta_1,...,\beta_n\}$, $\beta_x=P[r_i=x]$, $x\in N$을 갖는다는 모델이다. 다시 말해서, 각각의 참조들은 서로 독립적이고 어떤 페이지 x가 참조될 확률은 시점과 무관하게 일정하다는 것이다. 그러면, $d_i(x)$는 시점 i에서 페이지 x에 대한 전방거리를 나타내는 확률변수로서 다음과 같은 안정 상태의 기하분포(geometric distribution)를 갖는다[14].

$$P[d_i(x)=k]=\beta_x(1-\beta_x)^{k-1},\ k=1,\ 2,...$$

따라서, 시점 i에 상관없이 페이지 x에 대한 전방거리의 기대값 $E[d_i(x)]$은 항상 $1/\beta_x$이 된다[17].

그러나, 실제 상황에서는 페이지의 참조확률 $P[r_{i+l}=x](=\beta_x)$가 주어지지 않는다. 따라서, 그 값을 추정하여 교체대상을 선정하는 많은 알고리즘들이 제안되었고, 이들이 추정한 $P[r_{i+l}=x]$값에 따라 교체대상이 달라지고, 동일한 참조 스트링을 처리하는 비용도 서로 달라진다. 다음 절 2.2 절에서는 기존의 교체 알고리즘들을 $P[r_{i+l}=x]$ 값을 추정하는 관점에서 살펴보도록 한다.

2.2 기존의 교체 알고리즘

본 절에서는 기존의 교체 알고리즘들을 고정 크기의 페이지를 기반으로 한 전통적인 알고리즘과 크기와 참조비용이 가변인 데이타 아이템을 기반으로 한 알고리즘으로 분류하여 설명한다. 참조확률은 전통적인 교체 알고리즘과 최근에 제안된 다양한 교체 알고리즘에서 모두 중요한 역할을 하고 있다. 그러나, 만기시간이 주어지면 캐쉬효과를 얻을 확률이 참조확률과 달라지므로 그 영향을 반영하고자 하는 방법들이 연구되고 있으나 이론적으로 접근한 연구 결과는 많지 않은 상태이다.

전통적인 페이지 기반 교체 알고리즘은 이전의 참조 기록을 이용하여 참조확률을 추정하고 이를 사용하여

5) P[C]는 조건 C가 참일 확률을 의미한다.

교체대상을 선택한다. LRU[9]는 마지막 참조시점부터 현재까지의 시간간격을 이용하여 참조확률을 추정하고, LRU-K[13]는 이를 일반화하여 마지막 K번째 참조시점부터 현재까지의 시간간격을 이용하여 참조확률을 추정한다. LFU는 첫 참조시점부터 현재까지의 시간간격 동안의 참조횟수를 이용하여 참조확률을 추정한다.

최근 들어 웹 캐쉬, 객체 버퍼 등에서 크기와 참조비용 등을 참조확률과 함께 사용하는 교체 알고리즘들이 제안되고 있다. 이 방법들은 크게 키기반 전략과 함수기반 전략으로 분류된다[2]. 키기반 전략은 특성키들의 비교 순서를 정하여 교체대상을 선정하는 방법으로서 SIZE[2], HYPER-G[18] 등이 있다. SIZE는 크기가 큰 데이타 아이템을 먼저 교체하고, 크기가 동일한 경우에는 참조확률이 작은 데이타 아이템을 교체한다. 즉, SIZE는 크기를 첫번째 키로, 참조확률을 두번째 키로 사용한다. HYPER-G는 LFU 방법으로 구한 참조확률을 첫번째 키로 사용하고, LRU 방법으로 구한 참조확률을 두번째 키로 사용한다. 함수기반 전략은 크기나 참조비용 등을 이용하여 함수식을 만들고 그 값을 비교하여 교체대상을 선정하는 방법으로서 SLRU[2], PSS[2], LNC-R[7] 등이 있다. SLRU는 (참조확률/크기)이 작은 데이타 아이템을 교체하는 알고리즘이고, PSS는 ((참조확률/크기) 참조비용)이 작은 데이타 아이템을 교체한다. 그리고, LNC-R은 PSS와 유사한 함수식에 예상되는 전송지연시간을 참조비용에서 분리하여 계량화한 식을 추가한 알고리즘이다.

최근에 만기시간을 교체전략에 반영하려는 연구가 진행되고 있으나, 이들 연구에서는 만기시간에 의한 영향을 무시하거나 간단한 휴리스틱을 사용하고 있다. Pitkow-Recker 방법[5]은 단순히 하루 단위로 만기시간의 영향을 반영했고, Harvest 방법[12]은 만기된 데이타 아이템만을 제거하고 LRU를 사용하므로, 만기시간이 지나지 않은 데이타 아이템에 대해서 만기시간이 캐쉬효과에 미치는 영향을 고려하지 않았다. Aggarwal 등[2]은 크기와 참조비용이 가변임을 반영한 PSS (Pyramidal Selection Scheme)에 refresh overhead factor를 통합한 방법[6]을 소개하였다. Refresh overhead factor란 캐시진입시간, 마지막 참조시간, 만기시간에 기반하여 만기시간의 영향을 구한 근사값이다. PSS-ROF는 이 값을 캐쉬진입시점에 계산하여 교체 시점의 위치와 관계없이 고정적으로 사용하므로 교체 시점으로부터 만기시간까지의 거리를 무시한다. 그러나, 교체시점부터 만기시간까지의 거리는 다음 참조시점에서 참조된 데이타 아이템이 유효할 확률을 결정하는 중요한 요소이다. 따라서, 이 방법은 만기시간의 영향을 정확히 반영하지 못한다. 결국, 확률적인 모델 하에 만기시간의 영향을 분석 및 반영한 연구결과가 없으며, 최근의 많은 응용에서는 만기시간이 제공되므로 이에 대한 연구가 필요하다.

3. 연구동기

본 절에서는 만기시간이 참조확률에 미치는 영향에 대해서 기술한다. 데이타 아이템 x의 참조확률 $P[r_{i+1}=x]$를 다른 관점에서 보면, 현시점에서 x가 캐쉬 된다면 다음 참조시점에서 x로 인해 캐쉬한 효과를 볼 수 있는 확률이다. 즉, i번째 참조에서 x가 캐쉬되어, $(i+1)$번째 참조에서 x에 대한 참조비용이 발생하지 않을 확률이다. 데이타 아이템에 만기시간이 없다면(무한대라면), $(i+1)$번째 참조가 발생한 실제시점[7]은 항상 만기시간 이전이므로 참조확률은 유효참조확률과 같다.

반면에, 데이타 아이템에 만기시간이 주어진다면 참조확률과 유효참조확률이 달라질 수 있다. 참조확률 $P[r_{i+1}=x]$은 다음 참조시점에서 x를 참조할 확률이다. 그런데, 다음 참조가 발생한 실제시점이 x의 만기시간 이후라면, 다음에 참조할 데이타 아이템 x는 이미 유효하지 않으므로 캐쉬효과를 얻을 수 없다. 그림 1은 만기시간에 의하여 나누어진 유효구간과 무효구간을 나타낸다. 유효구간은 구간 A와 같이 교체가 발생한 실제시점부터 만기시간까지를 의미하고, 무효구간은 구간 B와 같이 만기시간 이후를 의미한다. 그림에서와 같이 $(i+1)$번째 참조가 발생한 실제 시간이 유효구간에서 발생한다면 x의 캐쉬효과를 얻을 수 있지만, 무효구간에서 발생한다면 이미 만기되었으므로 캐쉬효과를 얻을 수 없다. 따라서, 다음 참조가 발생한 실제시점의 위치를 구분하지 않는 참조확률과 유효참조확률은 서로 다르게 된다.

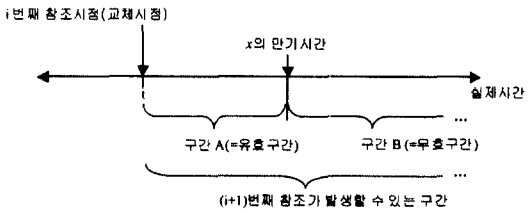


그림 1 만기시간에 의해 나누어진 유효구간과 무효구간의 예

6) 본 논문에서는 이 방법을 PSS-ROF라고 부른다.

7) '실제시점'은 실제 시간에서의 위치를 의미한다.

그림 2는 만기시간을 고려하지 않은 경우와 고려한 경우에 교체대상이 달라질 수 있음을 보여준다. 그림에서 데이타 아이템 x의 참조확률 p_x가 y의 참조확률 p_y보다 크다. 여기에서 만기시간을 고려하지 않는다면, x의 캐쉬효과가 y의 캐쉬효과보다 크므로, y를 교체대상으로 선정한다. 그러나, 만기시간을 고려하면, $t+x$의 전방거리 기대값은 x의 만기시간 이후이고 $t+y$의 전방거리 기대값은 y의 만기시간 이전이다. 그러므로, x보다는 y를 캐쉬하는 것이 캐쉬효과를 얻을 가능성이 더 커져서, x를 교체대상으로 선정한다. 이와 같이 , 만기시간이 반영된 경우와 그렇지 않은 경우의 교체대상이 다르므로, 만기시간이 주어진다면 그 영향이 반영된 유효참조확률을 이용하는 것이 캐쉬효과를 얻을 가능성을 더 크게 한다. 따라서, 본 논문에서는 만기시간을 이론적인 모델에 도입하여 캐쉬효과에 미치는 영향을 반영한 유효참조확률을 유도하고 기존의 교체 알고리즘에 반영하는 방법을 제안한다. 또한, 성능평가를 통해 유효참조확률을 사용한 교체 알고리즘이 참조확률을 사용한 교체 알고리즘보다 우수하다는 것을 보인다.

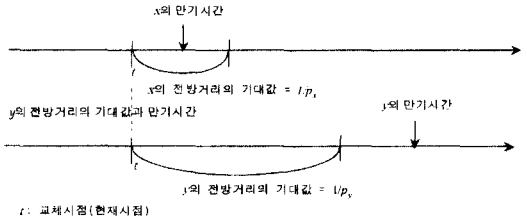


그림 2 참조확률의 크기 순서와 유효참조확률의 크기 순서가 다른 경우의 예.

4. 유효참조확률

본 절에서는 만기시간을 고려하여 캐쉬효과를 얻을 확률인 유효참조확률을 정의하고, 그 값을 확률 모델 하에서 유도한다.

4.1 유효참조확률의 정의

지금까지 정의된 표기법과 용어들을 사용하여 유효참조확률을 정의하면 다음과 같다.

정의 1 데이타 아이템 x의 유효참조확률이란 i번째 참조시점에서 x를 캐쉬한다면, $(i+1)$번째 참조에서 x를 참조하고 그 참조가 x의 유효구간에서 발생할 확률이다.

정의 1에서 알 수 있듯이, 유효참조확률은 참조확률($(i+1)$번째 참조가 x이어야 하므로)과 유효구간의 크기(유효구간에 참조되어야 하므로)에 의해서 결정된다. 참조확률이 클수록 다음 참조에서 참조될 가능성이 크고 전방거리의 기대값이 짧으므로 만기시간 이전에 참조될 가능성이 커진다. 또한, 만기시간까지의 남은 거리가 클수록 참조한 데이타가 유효할 가능성이 크다. 따라서, 유효참조확률은 참조확률에 만기시간까지 남은 거리를 반영한 값이다.

4.2 유효참조확률의 유도

본 절에서는 실제시간 기반으로 유효참조확률을 수식으로 정의하고 유도한다. 실제시간을 사용하는 이유는 제 3절에서 언급한 바와 같이 실제시간으로 주어지는 만기시간과 다음 참조시간의 위치를 비교해야 하기 때문이다. 먼저, 주어진 만기시간을 모델에 반영하기 위하여 실제시간을 기준으로 기호와 용어를 새롭게 정의한다. 데이타 아이템들의 집합은 $N=\{1,...,n\}$으로 표시하고, 데이타 아이템들의 읽기 요청들의 순열을 **참조 스트링**이라 하며, $\omega=r_1r_2...r_i..., r_i\in N$ 같이 표현한다. 여기에서 $r_i=x$는 i번째로 참조한 데이타 아이템이 x임을 의미하고, i번째 참조가 발생한 실제시점을 t로 표시한다. 실제시점 t에서 데이타 아이템 x의 실제시간에 기반한 전방거리를 이산시간 기반 전방거리 $d_t(x)$와 구분하기 위하여 $\delta_t(x)$로 표시하고, t에서는 그 값을 알 수 없으므로 $\delta_t(x)$의 확률변수인 $\Delta_t(x)$로 표시한다. 그리고, 모든 데이타 아이템에 대한 전방거리의 확률변수는 Δ_t로 표시한다. 실제시점 t 이후의 x의 만기시간을 $e_t(x)$로 표시하고 t에서 x의 유효구간 거리 $(e_t(x)-t)$를 x의 **유효거리**(*time to live*)라 부르고 $ttl_t(x)$로 표시한다.

새롭게 정의된 기호와 용어를 이용하여 유효참조확률을 수식으로 나타내면 다음과 같다. 유효참조확률 $p_t(x)$는 i번째 참조가 발생한 교체시점 t에서 예상한 값으로서, $(i+1)$번째 참조에서 x를 참조하고 그 참조시점이 x의 만기시간 이전에 발생할 확률이므로 식 (1)과 같이 나타낼 수 있다. 식 (1)은 식 (2)로 전개되고, 식 (2)에서 $(i+1)$번째 참조된 데이타 아이템이 x인 조건이 이미 있으므로, $\Delta_t(x)=\Delta_t$가 되어 식 (2)는 식 (3)으로 전개된다. 그리고, 조건부 확률의 정의를 이용하여 식 (3)은 식 (4)로 전개된다.

$$p_t(x)=P[r_{i+1}=x,\ t+\Delta_t(x)\le e_t(x)] \quad (1)$$

$$=P[r_{i+1}=x,\ \Delta_t(x)\le e_t(x)-t] \quad (2)$$

$$=P[r_{i+1}=x,\ \Delta_t\le e_t(x)-t] \quad (3)$$

$$=P[\Delta_t\le e_t(x)-t]P[r_{i+1}=x|\ \Delta_t\le e_t(x)-t] \quad (4)$$

식 (4)를 구하기 위해서는 Δ_t의 확률분포가 필요하다. Δ_t는 모든 데이타 아이템의 전방거리의 최소값이므로, Δ_t의 확률분포를 구하기 위하여 각 데이타 아이템의 전

방거리 $\Delta_t(x)(x\in N)$의 확률분포가 필요하다.

본 논문에서는 $\Delta_t(x)$의 확률분포를 유도하기 위해서, 독립참조모델을 연속된 실제시간으로 확장한다. 이 모델에서는 모든 $x\in N$에 대하여 t_x가 다음과 같이 주어진다. 확률변수 t_x는 t 이후에 데이타 아이템 x가 참조된 실제시점을 나타내는 독립적인 확률변수로서, 단위시간당 평균 참조횟수가 λ_x이고, 안정성(stationary property)을 갖는 확률분포를 따른다. 안정성이란 실제시점에 관계없이 동일한 확률분포를 갖는다는 성질이다. 즉, t_x는 모든 실제시점 t에 대하여 $P[t<t_x<t+h]=\lambda_x\times h(h\ll 1)$를 따르는 독립적인 확률변수이다. 이 모델은 모든 시점(참조순번)에서 x가 참조될 확률이 동일하다는 기존의 독립참조모델을 실제시간으로 확장한 것이다. 본 논문에서는 이 모델을 **연속독립참조모델**(*continuous independent reference model*)이라고 정의하고, 이 모델 하에서 $\Delta_t(x)$의 확률분포를 유도한다.

연속독립참조모델을 가정하면, 다음 보조정리 1에 의하여 데이타 아이템의 참조횟수는 포아송 과정임을 보일 수 있다.

보조정리 1 연속독립참조모델을 가정하면 단위시간당 평균 참조횟수가 λ_x인 데이타 아이템 x의 참조횟수는 발생율(rate)이 λ_x인 포아송 과정이다.

증명: 단위시간당 발생하는 평균 사건수가 λ_x인 계수과정 $\{N(t)^{8)}, t\geq 0\}$이 다음 세 가지 성질을 갖고 있으면 발생율이 λ_x인 포아송 과정임이 알려져 있다[17].

- 단수증가(orderliness)[9]
- 독립증분(independent increment)[10]
- 정상증분(stationary increment)[11]

참조 스트링은 일렬로 요구되는 것을 가정했기 때문에 단수증가 성질을 만족한다. 그리고, 연속독립참조모델에서 가정한 안정성을 이용하면 겹치지 않는 두 시간간격에 일어난 참조횟수는 독립적이므로 독립증분 성질을 만족한다. 그리고, 연속독립참조모델에서 실제시점 t에 관계없이 $P[t<t_x<t+h]=\lambda_x\times h$이므로 정상증분 성질을 만족한다. 따라서, 연속독립참조모델을 가정하면, 데이타 아이템 x의 참조횟수는 발생율이 λ_x인 포아송 과정이다. □

각 데이타 아이템의 참조횟수는 포아송 과정을 따르고 서로 독립이므로, 다음 보조정리 2에 의하여 모든 데이타 아이템에 대한 참조횟수 또한 포아송 과정을 따른다.

보조정리 2 연속독립참조모델을 가정하면, 모든 데이타 아이템들에 대한 참조횟수는 발생율이 $\lambda=\sum_{x=1}^{n}\lambda_x$인 포아송 과정이다.

증명: 보조정리 1에 의하여 각 데이타 아이템 x의 참조횟수는 발생율이 λ_x인 포아송 과정이다. 그리고, 연속독립참조모델에서 각 데이타 아이템이 참조되는 실제시점들은 서로 독립이므로 동일한 시간간격동안 발생한 각 데이타 아이템의 참조횟수는 서로 독립이다. 그런데, 복합 포아송 과정에 따르면 서로 독립인 포아송 과정의 확률변수의 합은 각 확률변수가 따르는 포아송 과정의 발생율 합을 새로운 발생율로 하는 포아송 과정이다[17]. 따라서, 모든 데이타 아이템에 대한 참조횟수는 각 데이타 아이템에 대한 참조횟수의 발생율 합을 발생율로 하는 포아송 과정이다. □

보조정리 2를 이용하면 Δ_t에 대한 확률분포를 다음 보조정리 3과 같이 구할 수 있다.

보조정리 3 연속독립참조모델을 가정하면, 모든 데이타 아이템들을 대상으로 한 전방거리에 대한 누적확률분포 $P[\Delta_t\leq s]$는 $(1-e^{-\lambda\times s})$이다.

증명: 보조정리 2에 의하여 집합 N을 대상으로 한 참조횟수는 발생율이 λ인 포아송 과정이다. 그런데, 사건횟수가 발생율 λ인 포아송 과정의 도착간격은 평균이 $1/\lambda$인 지수분포를 따르므로 [17], 첫 번째 참조시점까지의 시간간격, 즉 전방거리 Δ_t도 평균 $1/\lambda$인 지수분포를 따른다. 따라서,

$$P[\Delta_t>s]=P[N(s)=0]=e^{-\lambda\times s}$$

$$P[\Delta_t\leq s]=1-e^{-\lambda\times s}$$ □

보조정리 3은 앞서 구한 유효참조확률 수식 (4)의 전반부를 구하는데 사용된다. 즉, 수식 (4)의 전반부는 보조정리 3에서 s 대신 유효거리 $(e_t(x)-t)$가 사용되어 다음과 같이 구할 수 있다.

$$P[\Delta_t\leq e_t(x)-t]=1-e^{-\lambda\times(e_t(x)-t)} \quad (5)$$

그리고, 유효참조확률 수식 (4)의 뒷부분인 $P[r_{i+1}=x|\Delta_t\leq e_t(x)-t]$는 다음 보조정리 4를 사용하여 구할 수 있다.

보조정리 4 연속독립참조모델을 가정하면, 조건부확률 $P[r_{i+1}=x\,|\,\Delta_t\leq e_t(x)-t]$는 λ_x/λ이다.

증명 : 부록 참조 □

보조정리 4의 결과를 직관적으로 설명하면 다음과 같다. $P[r_{i+1}=x]$는 $(i+1)$번째 참조한 데이타 아이템이 x일 확률이므로, $(i+1)$번째 참조가 이미 발생했다는 조

8) 현재시점으로부터 t시간동안 발생한 사건횟수
9) 사건이 한번에 하나만 발생하는 성질
10) 겹치지 않는 두 시간 간격동안 발생한 사건횟수가 독립인 성질
11) 어떤 시간간격에 발생한 사건횟수는 시작시간과는 독립적으로 시간간격의 길이에 의해서만 결정되는 성질

건 하에 참조된 데이타 아이템이 x일 확률을 의미한다. 그리고, 각 데이타 아이템의 참조가 독립임을 이용하면, Bayes의 정리[17]에 의해 $P[r_{i+1}=x]=\lambda_x/\lambda$이다. 또한, $P[r_{i+1}=x]$는 안정성에 의해 $(i+1)$번째 참조시점의 위치에 무관하므로 조건 $(\Delta_t \le e_t(x)-t)$와 독립이므로, $P[r_{i+1}=x|\Delta_t \le e_t(x)-t]=P[r_{i+1}=x]$이 성립한다. 이것은 결국 $(i+1)$번째 참조될 데이타 아이템이 선택될 확률은 연속시간에 기반한 $(\Delta_t \le e_t(x)-t)$ 조건과 독립임을 의미한다. 따라서, 다음 참조가 발생했을 때 참조된 데이타 아이템이 x일 확률을 의미하는 이산시간 기반의 확률 $P[r_{i+1}=x]$와 동일하게 된다. 결국, 구현과정에서 실제시간에 기반한 λ_x/λ을 사용하지 않고, 참조순번에 기반한 $P[r_{i+1}=x]$을 대신 사용할 수 있다.

지금까지의 결과를 종합하면 다음 정리 1과 같이 유효참조확률을 구할 수 있다.

정리 1 단위시간당 참조횟수가 λ_x인 데이타 아이템 $x(\in N)$가 연속독립참조모델을 따른다면, 교체시점 t에서 x에 대한 유효참조확률 $p_t(x)$는 다음식과 같다.

$$p_t(x)=(1-e^{-\lambda\times(e_t(x)-t)})\times(\lambda_x/\lambda) \quad (6)$$

증명: 보조정리 3과 보조정리 4에 의하여 유효참조확률 $p_t(x)$는 $(1-e^{-\lambda\times(e_t(x)-t)})\times(\lambda_x/\lambda)$이다. □

정리 1을 보면 유효참조확률이 참조확률을 포괄함을 알 수 있다. 즉, 만기시간이 주어지지 않아 무한대라면 식 (6)은 λ_x/λ가 된다. λ_x/λ는 참조가 일어났다는 조건하에 x가 참조될 확률로서 이산시간기반 참조확률과 동일한 값이다. 따라서, 유효참조확률은 만기시간이 있는 경우에는 그 영향을 고려한 캐쉬효과를 얻을 확률이고, 만기시간이 없는 경우에는 이산시간에 기반한 참조확률과 동일하다.

5. 유효참조확률의 적용 및 구현

5.1 유효참조확률의 적용

본 절에서는 유효참조확률을 기존의 알고리즘에 적용하는 방법을 설명하고 구현방법에 대해서 기술한다. 제 2.2절에서 서술한 바와 같이 기존의 많은 교체 알고리즘들은 교체대상을 선정하기 위하여 참조확률을 사용한다. LRU-K, LFU 등과 같은 페이지기반 교체 알고리즘은 구체적인 방법은 다르지만 모두 참조확률의 추정값만을 사용하여 교체대상을 선정한다[9]. 또한, 키기반 교체전략과 함수기반 교체전략에 속하는 많은 교체 알고리즘들도 참조확률을 데이타 아이템의 다른 특성과 함께 사용한다.

기존의 교체 알고리즘에서 참조확률 대신에 유효참조확률을 사용하면 만기시간의 영향이 반영된 교체 알고리즘이 된다. 기존 교체 알고리즘에서 참조확률은 다음 참조에 캐쉬효과를 얻을 확률로서 사용된다. 유효참조확률은 만기시간의 영향을 반영한, 캐쉬효과를 얻을 확률이므로, 참조확률 대신에 유효참조확률을 사용하면 기존의 교체 알고리즘에 만기시간의 영향을 반영할 수 있다. 따라서, 참조확률을 사용하는 모든 교체 알고리즘에 만기시간의 영향을 반영할 수 있다. 예를 들어, LRU의 경우에는 후방거리를 비교하여 교체 대상을 결정하는데, (1/후방거리)을 참조확률의 추정값으로 해석하여 이 값 대신에 유효참조확률을 사용할 수 있다. 그리고, 유효참조확률식에서 λ_x/λ 대신에 $P[r_{i+1}=x]$을 사용할 수 있으므로, 이 값을 LRU 방식으로 추정한 (1/후방거리)을 사용한다. 이렇게 구한 유효참조확률은 LRU에서 희생자 선택의 기본 개념인 최근에 가장 적게 사용된 데이타 아이템이 확장되어 최근에 사용한 정도와 유효거리를 고려하여 캐쉬효과를 얻을 확률이 가장 작은 데이타 아이템으로 확장된다. LRU에서 참조확률 대신 이렇게 구한 유효참조확률을 사용하는 알고리즘은 기존의 LRU 알고리즘과 같이 참조기록에서 마지막 참조시점만의 영향을 받는다. LRU-K와 LFU에서도 (1/x의 후방 K-거리)와 (x의 참조횟수/총참조횟수)를 참조확률의 추정값으로 해석하여 쉽게 확장될 수 있다. 따라서, 기존 알고리즘이 사용한 참조확률을 추정하는 방법의 개념과 특성은 그대로 유지되고 만기시간의 영향은 추가로 고려된다.

최근의 일부 알고리즘에서는 참조확률과 함께 데이타 아이템의 크기나 참조비용 등과 같은 다른 특성을 교체대상 선정에 사용하였다. 이들을 포함하여 기존의 교체 알고리즘들을 일반화하면, (참조확률×$f(x)$) 값을 이용하여 교체대상을 선정한다고 해석할 수 있다. 페이지 기반 알고리즘과 키기반 교체 알고리즘에서 참조확률을 이용하는 키값은 '$f(x)$=상수'인 경우로 해석하여 다른 특성이 동일함을 표시할 수 있고, 함수기반 교체 알고리즘은 크기나 참조비용 등과 같은 페이지에 없는 특성을 $f(x)$에 반영한다[2,19]. 따라서, 이들 알고리즘에서는 참조확률 대신에 유효참조확률을 사용하여 (유효참조확률× $f(x)$)값을 사용하도록 확장하면 만기시간을 고려할 수 있게 된다.

5.2 구 현

본 논문에서 제안한 방법을 구현하기 위해서는 λ, λ_x/λ, $(1-e^{-\lambda\times(e_t(x)-t)})$를 계산해야 한다. 이 중 단위시간당 평균참조횟수 λ는 평균참조간격의 역수를 이용하여 마지막 참조시간에 구할 수 있다. 또한, λ_x/λ는 제 4.2절에서 기술한 바와 같이 이산시간기반 참조확률

을 사용할 수 있으므로 기존 알고리즘과 유사한 방법으로 참조확률을 구할 수 있다. 즉, 데이타 아이템이 처음 캐쉬될 때는 기존 알고리즘의 처리방식에 따라 LRU-2의 경우에는 λ_x/λ값을 0으로 설정해야 한다. 그러나, 처음 참조되는 데이타 아이템도 참조확률을 부여하여 하나의 스택에 몰리는 현상을 피하기 위하여 참조스트링에서 최초 참조시점을 모든 데이타 아이템의 첫 참조시점으로 간주한다. 한편, 식 (6)의 $(1-e^{-\lambda\times(e_t(x)-t)})$는 교체시점마다 캐쉬된 모든 데이타 아이템에 대해서 계산해야 하므로 오버헤드가 될 수 있다. 본 논문에서는 이러한 오버헤드를 줄이기 위해 PSS(Pyramidal Selection Scheme)[2]의 아이디어를 이용한 구현방법을 제시한다. PSS 방법은 지수분포에 따라 범위를 갖는 스택을 이용하여 캐쉬된 데이타 아이템을 관리한다. 즉, 데이타 아이템이 속할 스택은 크기나 참조비용과 같이 교체시점과 무관하게 미리 결정된 값을 이용해 결정되고, 각 스택 안에서는 교체시점마다 바뀌는 값에 따라 순서를 유지하는 구조이다. 이와 같은 아이디어를 사용하여 본 논문에서는 그림 3과 같이 지수분포에 따라 스택의 범위를 정한다. 그리고, 캐쉬되는 데이타 아이템은 $(\lambda_x/\lambda)\times f(x)$에 따라 스택을 결정하고 $(1-e^{-\lambda\times(e_t(x)-t)})$ 값에 따라 스택 안에서 순서가 유지된다. 그런데, 데이타 아이템 x의 참조확률 (λ_x/λ)은 스택을 결정하기 위해 사용되므로 참조시점에 알 수 있어야 한다. 그런데, 참조확률($P[r_{i+1}=x]$)은 전방거리 기대값($E[d_t(x)]$)의 역수이므로[13], 참조시점에 이전 참조시간 간격의 평균값을 이용하여 추정할 수 있다. 이러한 방법은 LRU-K에서도 사용한 것으로 해석할 수 있으므로[20][12], 구현에서는 참조시점에 구할 수 있는 통계적인 방법을 통하여 참조확률을 추정한다.

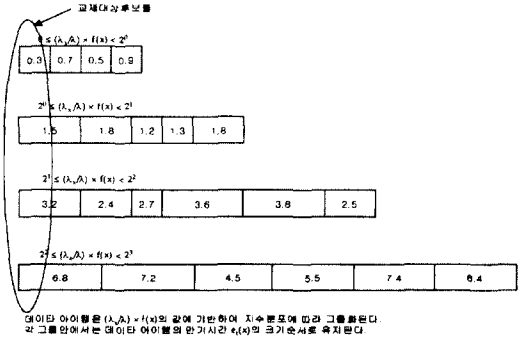


그림 3 PSS방식의 교체대상 선정을 위한 데이타 구조의 예

12) 부록 B 참고

즉, 그림 4에서 참조시점 t_1에 평균 참조간격 I는 다음과 같다.

$$I=\frac{(t_{-1}-t_{-2})+(t_{-2}-t_{-3})+\cdots+(t_{-(k-1)}-t_{-k})}{k}$$

그리고, $(1-e^{-\lambda\times(e_t(x)-t)})$의 $(e_t(x)-t)$값은 교체시점 t마다 변하지만, 특정시점 t에서 $e_t(x)-t$은 $e_t(x)$에 단조증가하므로 $(1-e^{-\lambda\times(e_t(x)-t)})$의 크기 순서와 $e_t(x)$의 크기순서는 동일하다. 따라서, 스택 안의 순서는 $e_t(x)$에 따라 정렬하면 된다. 교체 대상을 선택할 때는 각 스택에서 $e_t(x)$가 가장 작은 데이타 아이템들의 $(1-e^{-\lambda\times(e_t(x)-t)})\times(\lambda_x/\lambda)\times f(x)$ 값을 비교하여 결정한다. 이러한 방법은 교체시점마다 변하는 $(1-e^{-\lambda\times(e_t(x)-t)})$값을 매번 계산하지 않고, 이미 정해진 $e_t(x)$값을 이용해 각 스택에서 $(1-e^{-\lambda\times(e_t(x)-t)})$의 최소값을 갖는 데이타 아이템을 $O(1)$에 찾아 낼 수 있는 장점이 있다.

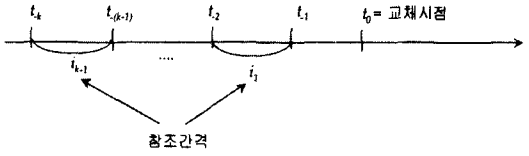


그림 4 참조 간격

6. 성능 평가

본 절에서는 실험을 통하여 유효참조확률을 사용한 교체 방법들의 성능을 평가한다. 제 6.1 절에서는 실험환경을, 제 6.2 절에서는 실험 구현을 설명하고, 제 6.3 절에서는 실험 결과를 제시한다.

6.1 실험 환경

사용된 데이타 아이템의 수는 500으로 하였고,[13] 고정된 크기의 데이타 아이템을 사용하는 실험에서는 크기를 1로 하였고, 가변 크기의 데이타 아이템을 사용하는 실험에서는 크기를 1, 2, …, 500로 하여 데이타 아이템간에 차이가 충분히 크도록 하였다. 그리고, 고정크기를 사용하는 경우에는 참조비용을 1로 하여 만기시간에 의한 영향만을 반영한 실험이 되도록 하였고, 가변크기를 사용하는 경우에는 참고문헌 [2]에 따라 데이타 아이템의 크기와 참조비용의 상관관계를 다음 식 (7)과 같이 모델한다.

$$\text{참조비용}=q\cdot\text{크기}+2\cdot(1-q)\cdot\text{평균크기}\cdot random(0,1) \quad (7)$$

이 모델은 웹 데이타 아이템의 참조비용은 크기뿐만

13) 이 값은 참고문헌 [2]의 예를 따른 것임.

아니라 네트워크 밴드위쓰 등에 의해 결정되므로 크기와 참조비용의 상관관계를 계수 q로 표현한다. 계수 q 값은 0에서 1까지의 값을 갖는데 1에 가까울 수록 참조비용과 크기의 상관관계가 커진다.

데이타 아이템의 참조확률의 분포는 $\theta=0.28$인 Zipf 분포를 사용하고, 참조확률은 데이타 아이템의 크기와 상관관계 없이 설정된다. 이는 웹 데이타 참조기록을 조사한 결과 참조확률은 대체로 $\theta=0.18\sim0.38$인 Zipf 분포를 따르고 참조확률과 크기는 상관관계가 없기 때문이다[21]. 캐쉬비율은 참조되는 모든 데이타 아이템의 크기의 합에 대한 캐쉬 크기의 비율로 정의한다. 본 실험에서는 0.01, 0.02, …, 0.10의 캐쉬비율을 사용한다. 참조 스트링의 길이는 50,000으로 하여 Zipf 분포에 따라 설정된 참조확률이 가장 작은 데이타 아이템도 최소한 25번 이상 참조되도록 하였다. 데이타 아이템은 주기적으로 갱신된다고 가정하고, 그림 5와 같이 그 갱신주기를 (K×참조주기)으로 하여 K값에 따라 갱신주기를 변경할 수 있도록 하였다. K값은 5, 10, …, 50을 사용한다. 왜냐하면, 웹 데이타 아이템의 (갱신횟수/참조횟수)가 0~0.2에 집중하여 분포하기 때문이다[21]. 또한, 그림 5와 같이 데이타가 참조되는 실제시점은 데이타 갱신시점과 무관하다는 가정 하에, 참조되어 캐쉬되는 실제시점은 갱신주기 상의 임의의 실제시점에 주어진다. 참조시점은 반복되는 갱신주기 (0~K×참조간격) 상에 균일하게 분포하므로, 유효거리는 (*random*(0,1)×갱신주기)로 주어진다. 표 1에는 사용된 변수와 그 값들을 정리하였다.

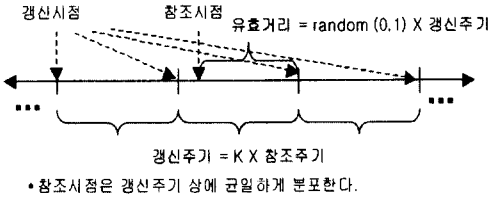


그림 5 참조주기, 갱신주기 및 캐쉬될 때의 유효거리

표 1 실험 환경

항목	값
데이터 아이템의 수	500
참조확률의 분포	$\theta=0.28$인 Zipf 분포
캐쉬비율	0.01, 0.02,0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.10
참조 스트링의 길이	50,000
K	5, 10, 15, 20, 25, 30, 35, 40, 45, 50
상관계수(q)	0~1

실험은 유효참조확률과 기존 알고리즘의 참조확률의 성능비교를 위해 크기와 참조비용이 고정된 경우와, 실제 웹 데이타 아이템과 같이 크기와 참조비용이 가변인 경우로 나누어 수행하였다. 크기와 참조비용이 고정된 실험에서는 성능척도로 **캐쉬 적중율(*hit ratio*)**을 사용한다. 캐쉬 적중율은 전체 참조횟수에 대하여 캐쉬에서 참조한 횟수의 비율로서 크기와 참조비용이 고정인 경우와 같이 캐쉬에서 참조하는 횟수와 절감하는 비용이 비례할 경우에 적합하다. 크기와 참조비용이 가변인 실험에서는 성능척도로는 캐쉬 적중율과 함께 **비용 절감율(*cost savings ratio*)**을 사용한다. 비용절감율이란 캐쉬를 사용하여 절감된 참조비용의 비율로서 식 (8)과 같다. 비용절감율은 크기와 참조비용의 가변성으로 인해 캐쉬에서 참조하는 횟수와 절감하는 비용이 비례하지 않는 경우에 적합한 성능척도다.

$$\text{비용절감률} = \frac{\text{캐쉬로 인해서 절약되는 비용}}{\text{캐쉬가 없을 때 소모되는 참조비용}} \quad (8)$$

실험을 통해 비교되는 교체 알고리즘으로는 만기시간을 고려하지 않은 알고리즘에서 성능이 우수한 것으로 알려진 LRU-2[13]를 사용하고, 만기시간을 고려한 알고리즘에서 가장 성능이 우수한 Aggarwal의 방법(PSS-ROF)[2]을 사용한다.

6.2 실험 구현

실험 시스템은 그림 6과 같이 데이타 아이템 생성기, 참조스트링 생성기, 캐쉬 알고리즘 처리기로 구성된다. 데이타 아이템 생성기는 실험종류(고정크기 실험 혹은 가변크기 실험), 참조확률의 분포(θ), 참조비용과 크기의 상관관계(q)를 입력으로 받아서 데이타 아이템 집합을 파일로 생성한다. 이 파일에 저장된 데이타 아이템은 식별자, 참조확률, 크기, 참조비용 등과 같이 데이타의 고정된 특성을 기록한다. 이 파일을 바탕으로 하여 참조스트링 생성기는 만기시간을 부여하며 참조스트링을 생성하여 캐쉬 알고리즘 처리기에 참조를 요청한다. 캐쉬 알고리즘 처리기는 구현된 알고리즘에 따라 참조스트링을 처리하면서, 성능척도를 구할 수 있도록 처리 결과를 로그 파일에 기록한다.

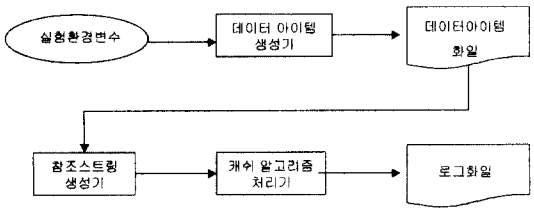


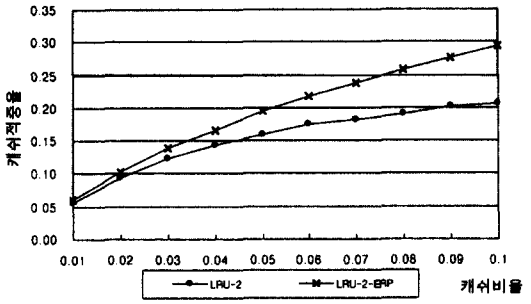
그림 6 실험 시스템 구조

실험을 위한 스택의 수는 $(\lambda_x/\lambda)\times f(x)$의 최대값을 포함하도록 설정해야 한다. 그러나, 스택의 범위값이 작은 스택들에 포함된 데이타 아이템들이 작은 $(\lambda_x/\lambda)\times f(x)$ 값을 갖으므로 교체대상으로 대부분 선정된다. 따라서, 스택의 수를 대부분의 데이타 아이템을 포함하도록 큰 상수값(본 실험에서는 7개의 스택을 사용하였다.)으로 고정하고, 마지막 스택의 값의 범위를 무한대까지로 확장하여 구현할 수 있다.

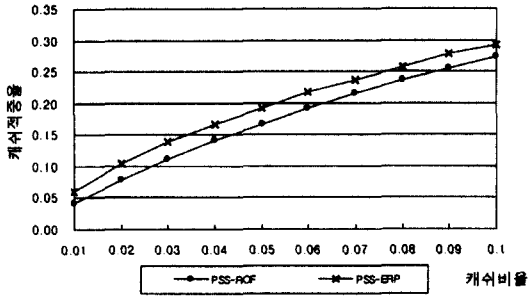
6.3 실험 결과

1) 크기와 참조비용이 고정인 실험결과

그림 7은 K=20일 때, 캐쉬비율에 따른 성능비교를 나타낸다. 그림을 보면, 유효참조확률을 적용한 방법이 그렇지 않은 방법보다 캐쉬비율에 상관없이 항상 우수함을 알 수 있다. 그림 7(a)의 LRU-2의 경우, 캐쉬비율이 커짐에 따라 유효참조확률을 사용하는 LRU-2-ERP[14]가 유효참조확률을 사용하지 않는 LRU-2와 성능차이가 커짐을 알 수 있다. 이는 LRU-2의 경우 캐쉬가 커지더라도 만기된 데이타가 캐쉬에 그대로 남아 있



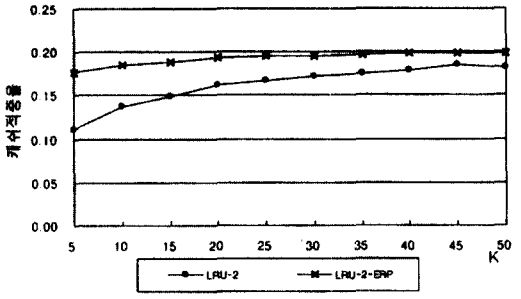
(a) LRU-2



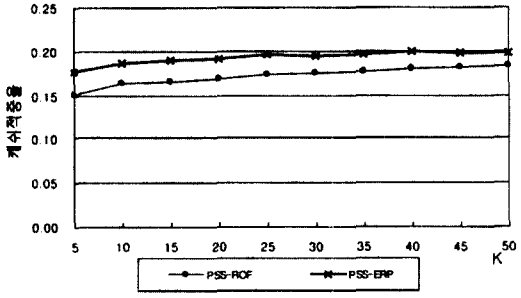
(b) PSS

그림 7 K=20일 때 캐쉬비율의 변화에 따라 유효참조확률을 사용한 경우와 그렇지 않은 경우의 성능비교

14) 유효참조확률이 적용된 알고리즘은 접미사로 -ERP를 붙인다.



(a) LRU-2



(b) PSS

그림 8 캐쉬비율=0.05일 때 K값의 변화에 따라 유효참조확률을 사용한 경우와 그렇지 않은 경우의 성능비교

어 그 효과를 충분히 활용하지 못하기 때문이다. 반면에, 작은 캐쉬비율에서는 최근에 참조한 데이타들만이 캐쉬되므로, 캐쉬된 데이타 아이템들이 대부분 만기 이전이어서 LRU-2-ERP와 성능 차이가 작게 나타났다. 그림 7(b)에서는 PSS-ROF와 PSS-ERP의 성능차이가 캐쉬비율에 관계없이 일정하다. 이는 두 방법의 성능차이가 만기시간의 영향을 반영한 방법의 차이로 인한 것이기 때문이다.

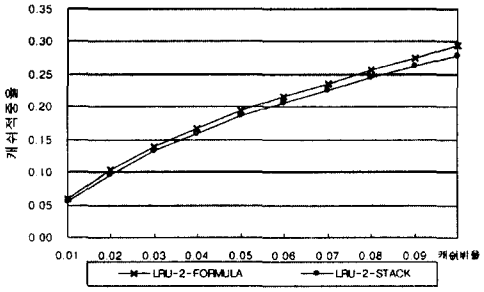
그림 8은 캐쉬비율이 0.05일 때, K값의 변화에 따라 유효참조확률을 사용한 경우와 그렇지 않은 경우의 성능비교를 나타낸다. 그림에서 유효참조확률을 사용하면 K값이 10 이하일 때 LRU-2의 경우 30% 이상, PSS-ROF의 경우 13% 이상 성능이 개선된 것을 알 수 있다. 그리고, 두 가지 경우 모두 K값이 커질수록 유효참조확률을 사용한 방법과 그렇지 않은 방법간의 성능차이가 감소함을 알 수 있다. 이런 현상은 K값이 커져 갱신주기가 길어지면, 만기시간이 길어지므로 만기시간에 의한 영향이 작아지기 때문이다. 또한, 이것은 만기시간이 없으면(만기시간이 무한대이면) 유효참조확률이 기존

의 참조확률과 동일하다는 것을 보여주는 실험 결과다.

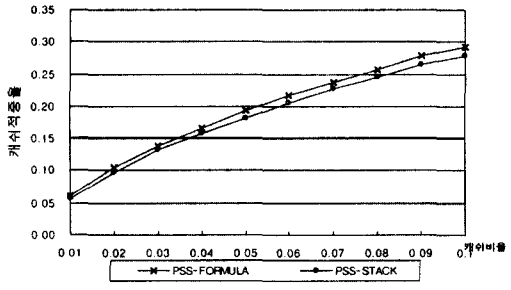
그림 9는 식 (6)의 유효참조확률을 그대로 이용한 방법(FORMULA방법이라 하고 "-FORMULA"을 붙여 표시함)과 PSS와 같이 스택을 이용하여 구현한 방법(STACK방법이라 하고, "-STACK"을 붙여 표시함)의 성능비교를 나타낸다. 그림에서 두 가지 경우 모두 FORMULA방법이 STACK방법보다 약간 우수하거나, 거의 차이가 없는 것으로 나타났다. 이는 FORMULA방법 대신 STACK방법을 사용하는 오차를 무시할 수 있음을 나타낸다. 그러나, FORMULA방법은 교체시점마다 t값의 변화에 따라 $(1-e^{\lambda\times(e_i(x)-t)})$값을 다시 계산해야 하므로 STACK방법이 실용적으로는 우수하다.

2) 크기와 참조비용이 가변인 실험결과

본 실험에서는 유효참조확률을 사용하는 것이 크기와 참조비용이 가변인 웹 데이타에서도 성능이 개선됨을 보인다. 그림 10은 크기와 비용의 상관관계 q가 0.5이고, K=20일 때 캐쉬비율의 변화에 따른 PSS에서의 실험결과로서, 유효참조확률을 사용하는 경우와 그렇지 않은 경우의 성능비교를 나타낸다.[15] 그림 10의 결과는



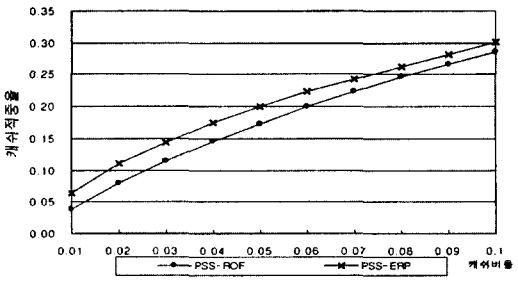
(a) LRU-2-ERP



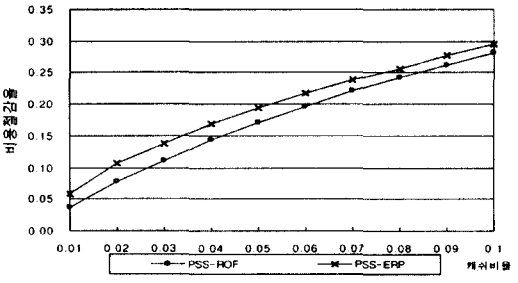
(b) PSS-ERP

그림 9 K=20일 때의 FORMULA방법과 STACK방법의 비교

15) LRU-2는 크기와 참조비용이 가변인 경우를 고려하지 않으므로 실험에서 제외한다.

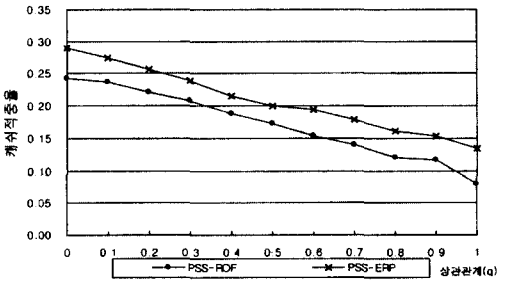


(a)

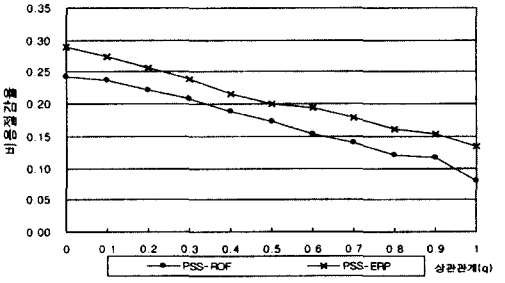


(b) 비용절감율에 의한 성능비교

그림 10 q=0.5, K=20일 때 캐쉬비율에 따른 PSS-ROF와 PSS-ERP의 성능비교



(a) 상관관계에 따른 캐쉬적중률의 비교



(b) 상관관계에 따른 비용절감율의 비교

그림 11 캐쉬비율=0.05, K=20일 때 상관관계(q)에 따른 PSS-ROF와 PSS-ERP의 성능비교

그림 7(b)의 고정 크기와 참조비용을 사용한 경우와 매우 유사하다. 즉, 유효참조확률을 사용하는 것이 그렇지 않은 경우보다 캐쉬적중률과 비용절감율에서 모두 우수한 성능을 보였다. 그림 11의 결과는 크기와 비용의 상관관계에 따른 실험결과를 나타낸다. [2]에서 마찬가지로 상관관계가 커짐에 따라서 크기와 비용이 비례하므로 크기를 고려하는 효과가 줄어들어 성능이 떨어짐을 알 수 있다. 즉, PSS에서 (비용/크기)이 큰 값을 교체대상으로 선정하는 효과가 줄어들어 성능이 저하된다. 그러나, 유효참조확률이 개선한 효과는 유지되므로, PSS-ROF와 PSS-ERP의 성능차이는 유지된다. 이것은 PSS-ERP가 크기와 비용의 영향은 PSS-ROF와 동일하게 반영하고, 만기시간의 영향을 반영하는 방법을 개선했기 때문이다.

7. 결 론

본 논문에서는 웹 데이타에 부여되는 만기시간을 웹 캐쉬의 성능향상에 활용하는 방안을 제안하였다. 이를 위하여 우선 기존 캐쉬 교체 알고리즘에서 사용된 참조확률에 만기시간의 영향을 반영한 유효참조확률의 개념을 제시하였다. 다음으로, 유효참조확률을 정형적으로 정의한 후, 연속독립참조모델 하에서 이론적으로 이를 유도하였다. 제안한 유효참조확률은 만기시간이 없는 경우에 참조확률과 동일하고, 만기시간이 있는 경우에는 그 영향을 반영하므로 참조확률의 진보된 개념이다. 또한, 기존의 교체 알고리즘에서 참조확률 대신 유효참조확률을 사용함으로써 기존 교체 알고리즘에 만기시간의 영향을 반영할 수 있다. 그리고, PSS와 같이 스택을 이용한 방법으로 유효참조확률이 실용적으로 사용될 수 있음을 보였다.

성능평가 결과, 유효참조확률을 사용한 알고리즘이 그렇지 않은 알고리즘보다 항상 우수한 결과를 보였다. 특히, 유효참조확률은 고정된 크기와 참조비용을 사용하는 실험에서 캐쉬비율이 0.05이고 K 값이 10 이하일 때(즉, 갱신이 참조의 1/10 이하의 횟수로 일어날 때), LRU-2의 경우 30% 이상, PSS-ROF의 경우 13% 이상 성능을 개선하였다. 이것은 제안한 방법이 만기시간을 고려하여 캐쉬효과를 얻을 이론적인 확률을 정확히 반영한 방법으로서, 만기시간의 영향을 보다 효과적으로 반영하였기 때문이다. 유효참조확률은 만기시간이 주어지는 여러 분야에 사용될 수 있다. 특히, 네트워크 캐쉬를 사용하는 웹 브라우저, 프락시 서버, FTP 미러링 서버와 이동환경에서의 캐쉬 등에서 이용될 수 있을 것이다.

참 고 문 헌

[1] Abrams, M., Standridge, C., Abdulla, G., Williams, S., and Fox, E., Caching Proxies: Limitations and Potentials, In *Proc. Int'l World Wide Web Conf.*, CERN, Boston, pp. 110-133, Dec. 1995.

[2] Aggarwal, C., Wolf, J. L., and Yu, P. S., Caching on the World Wide Web, *IEEE Trans. on Knowledge and Data Engineering*, Vol. 11, No. 1, pp. 94-107, Feb. 1999.

[3] 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.

[4] Gadde, S., Rabinovich, M., and Chase J., Reduce, Reuse, Recycle: An Approach to Building Large Internet Caches, In *Proc. Workshop on Hot Topics in Operating Systems(HotOS-VI)*, May 1997.

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

[6] Markatos, E., Main Memory Caching of Web Documents, *Computer Networks and ISDN Systems*, Vol. 28, pp. 893-905, 1996.

[7] Scheuermann, P., Shim, J. H., and Vingralek, R., A Case for Delay-Conscious Caching of Web Documents, In *Proc. Int'l WWW Conf.*, CERN, Santa Clara, California, Feb. 1997.

[8] Williams, S., Abrams, M., Standridge, C. R., Abdulla, G., and Fox, E. A., Removal Policies in Network Caches for World-Wide Web Documents, In *Proc. Conf. on Applications,Technologies, Architectures, and Protocols for Computer Communications*, ACM SIGCOMM, Palo Alto, Aug. 1996.

[9] Coffman, E. G. Jr., and Denning, P. J., *Operating Systems Theory*, Prentice Hall, 1975.

[10] Liu, G., and Maguire, G. Q., A Survey of Caching and Prefetching Techniques in Distributed Systems, TR. TRITA-IT R 94-40, Dept. of Teleinformatics, Royal Institute of Technology, Oct. 1994.

[11] Fielding, R., Gettys, J., Mogul, J. C., Frystyk, H., and Berners-Lee., T., Hypertext Transfer Protocol HTTP/1.1. RFC 2068, http://nic.ddn.mil/ftp/rfc/rfc2068.txt, Jan. 1997.

[12] Chankhunthod, A., Danzig, P. B., and Neerdaels, C., A Hierarchical Internet Object Cache, In *Proc. USENIX Technical Conf.*, USENIX Association, San Diego, CA, Jan. 1996.

[13] O'Neil, E. J., O'Neil, P. E., and Weikum, G., The LRU-K Page Replacement Algorithm For Database, In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Washington, D.C., pp. 297-306, May 1993.

[14] Aho, A. V., Denning, P. J., and Ullman, J. D., Priciples of Optimal Page Replacement, *Journal of ACM*, Vol. 18, No. 1, pp. 80-83, Jan. 1971.

[15] Belady, L. A., A Study of Replacement Algorithms for Virtual Storage Computers, *IBM Systems Journal*, Vol. 5, No. 2, pp. 78-101, 1966.

[16] Denning, P. J., The Working Set Model for Program Behavior, *Comm. of the ACM*, Vol. 11, No. 5, pp.323-333, May 1968.

[17] Ross, S. M., *Introduction to Probability Models*, Academic Press, 1993.

[18] Andrews, A., Kappe, F., Maurer, H., and Schmaranz, K., On Second Generation Hypermedia Systems, In *Proc. ED-Media 95, World Conf. on Educational Multimedia and Hypermedia*, June 1995.

[19] Sinnwell, M., and Weikum, G., A Cost-Model-Based Online Method for Distributed Caching, In *Proc. Int'l Conf. on Data Engineering*, IEEE, Birmingham, U.K., pp. 532-541, 1997.

[20] Park, C.M., Whang, K.Y., Han, W.S., and Song, I.Y., A Cost-Based Buffer Replacement Algorithm for Object-Oriented Database Systems, *Information Sciences*, Vol. 138, No. 14, pp. 99-117, 2001.

[21] Breslau, L., Cao, P., Fan, L., Philips, G., and Shenker, S., Web Cachig and Zipf-like Distributions: Evidence and Implications, In *Proc. IEEE Int'l Conf. on Computer Communications (INFOCOM)*, New York, 1999.

부록 A. 보조정리 4의 증명

$P[r_{i+1}=x]$은 안정성에 의해서 조건 "$\Delta_t \le e_t(x)-t$"와 독립이므로 $P[r_{i+1}=x|\Delta_t \le e_t(x)-t]=P[r_{i+1}=x]$이다. 따라서, $P[r_{i+1}=x]=\lambda_x/\lambda$을 증명하면, $P[r_{i+1}=x|\Delta_t \le e_t(x)-t\ ]=\lambda_x/\lambda$이 증명된다. $r_{i+1}=x$는 i번째 참조 이후에 첫 번째 참조한 데이타 아이템이 x임을 의미하므로 $P[r_{i+1}=x]=P[\Delta_t=\Delta_t(x)]$이다. 다음식에서 $P[\Delta_t=\Delta_t(x)]=\lambda_x/\lambda$임을 증명한다.

$$
\begin{aligned}
&P[r_{i+1}=x] = P[\Delta_t = \Delta_t(x)] \\
&= P[\Delta_t(k) > \Delta_t(x)], \text{ for } k \in \{1,\cdots,n\} \text{ and } k \neq x \\
&= \int_0^\infty P[\Delta_t(1)>t,\cdots,\Delta_t(k-1)>t,\Delta_t(k+1)>t,\cdots,\Delta_t(n)>t \,|\, \Delta_t(x)=t] f_x(t)dt \\
&= \int_0^\infty P[\Delta_t(1)>t,\cdots,\Delta_t(k-1)>t,\Delta_t(k+1)>t,\cdots,\Delta_t(n)>t] f_x(t)dt \\
&\quad (\because \Delta_t(1),\cdots,\Delta_t(k-1),\Delta_t(k+1),\cdots,\Delta_t(n) \text{독립}) \\
&= \int_0^\infty \Big(\prod_{k\neq x} P[\Delta_t(k)>t]\Big) f_x(t)dt \\
&\quad (\because \Delta_t(1),\cdots,\Delta_t(k-1),\Delta_t(k+1),\cdots,\Delta_t(n) \text{독립}) \\
&= \int_0^\infty \Big(\prod_{k\neq x} e^{-\lambda_k t}\Big)\lambda_x e^{-\lambda_x t}dt \\
&\quad (\because \text{보조정리 3}) \\
&= \int_0^\infty \lambda_x e^{-(\lambda_1+\lambda_2+\cdots+\lambda_n)t}dt \\
&= \frac{\lambda_x}{\lambda_1+\lambda_2+\cdots+\lambda_n} \\
&= \frac{\lambda_x}{\lambda}
\end{aligned}
$$

부록 B

본 부록 B에서는 LRU-K가 교체대상선택에 전방거리의 기대값을 이용하는 것임을 보인다.

그림 7에서 보인 바와 같이 마지막 K개의 참조시점들과 앞으로 예상되는 참조시점을 이용하여 전방거리를 구할 수 있다. 즉, 어떤 시점 t_0에서 객체 x에 대해 마지막으로 발생된 $k-1$개의 참조간격들인 i_1, i_2, …, i_{k-1}과 앞으로 예상되는 참조간격 i_k의 평균을 I라 하면, I는 시점 t에서 앞으로 예상되는 전방거리와 같아야 하므로,

$$I = \sum_{j=1}^{k} i_j / K = (\text{후방}K\text{거리} + I)/K$$

$$K \times I = \text{후방}K\text{거리} + I$$

$$I = \text{후방}K\text{거리}/(K-1)$$

라고 할 수 있다. 결국, LRU-K는 참조간격의 평균을 이용하여 전방거리 기대값을 추정한 값을 사용한다.

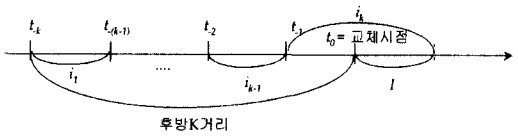


그림 12 LRU-K의 전방거리 기대값의 추정

이 정 준

1992년 2월 서울대학교 계산통계학과 학사. 1994년 2월 한국과학기술원 전산학과 석사. 1994년 1월 ~ 1999년 4월 한국과학기술원 정보시스템연구소 연구원. 1995년 3월 ~ 현재 한국과학기술원 전산학과 박사과정. 2001년 9월 ~ 현재 한국산업기술대학교 컴퓨터공학과 전임강사. 관심분야는 웹 캐쉬, 웹 데이타베이스, XML

문 양 세
정보과학회논문지 : 데이타베이스
제 28 권 제 1 호 참조

황 규 영
정보과학회논문지 : 데이타베이스
제 28 권 제 1 호 참조

홍 의 경
정보과학회논문지 : 데이타베이스
제 28 권 제 3 호 참조