준(準)캐시를 위한 적응적 방송 기법

오창민 °
서울대학교 전기.컴퓨터공학부
redmoon@db.snu.ac.kr

송병호
상명대학교 소프트웨어학부
bhsong@sangmyung.ac.kr

이석호
서울대학교 전기.컴퓨터공학부
shlee@cse.snu.ac.kr

Adaptive Broadcasting Strategy for Quasi Cache

Changmin Oh °
School of Electrical Engineering and Computer Science,
Seoul National University

Byoungho Song
Division of Software Science,
Sangmyung University

Sukho Lee
School of Electrical Engineering and Computer Science,
Seoul National University

요 약

준 캐시(quasi cache)는 데이타 캐시 방법의 일종으로 엄격한 일관성이 아닌 약한 형태의 일관성을 적용함으로써 캐시 일관성 유지 비용을 줄이는 방법이다. 그러나 준 캐시 또한 갱신이 빈번한 환경에서는 기존 캐시와 같이 일관성 유지를 위한 통신 비용이 커질 수 있다. 이 논문에서는 준 캐시의 기본 모델을 살펴보고, 기본 모델에 적응적인 기법을 적용한 모델을 제시한다. 또한 실험을 통해 갱신이 빈번한 환경에서는 적응적 모델이 기본 모델보다 더 작은 통신 비용으로 캐시 일관성을 유지할 수 있음을 보인다.

1. 서 론

캐시(cache)란 클라이언트 컴퓨터에 복사되어 있는 서버 컴퓨터 데이타의 일부분을 말한다. 캐시된 데이타는 클라이언트가 서버로의 접근 없이 데이타를 사용할 수 있게 함으로써 이득을 준다. 특히 무선 이동 환경 같은 경우에는 클라이언트에서 서버로 가는 네트웍의 대역폭이 서버에서 클라이언트로 가는 대역폭보다 상대적으로 훨씬 작다. 또한 에너지 측면에 있어서도 데이타를 내려 받는데 드는 전력이 데이타를 올리는 데 드는 것보다 더 크다. 따라서 클라이언트는 되도록 서버로의 접근 비용을 줄이는 것이 이득이 된다[2].

그러나 캐시되어 있는 데이타는 서버 데이타의 복사본이기 때문에 서버 데이타에 대한 갱신이 발생하게 되면 쓸모가 없어진다. 따라서 캐시를 사용하면 서버와 클라이언트 간의 캐시 일관성(cache consistency)을 보장해줘야 한다. 일관성을 유지하는 방법에는 갱신된 항목의 목록을 보내줌으로써 캐시를 무효화 시키는 방법[3]과 갱신된 값 자체를 전송하는 방식이 있다. 그러나 서버의 데이타 항목에 대한 갱신이 빈번하게 일어나게 되는 경우에는 캐시를 유지함으로써 얻는 이득보다는 캐시 일관성을 보장하기 위해 드는 비용이 더 커질 수 있다. 이는 캐시 일관성을 엄격하게 보장하려고 함으로써 생기는 문제이다.

응용에 따라서는 엄격한 일관성이 필요치 않은 곳도 있는데, 준 캐시(quasi cache)[1]는 이와 같은 환경에서 약한 형태의 일관성을 적용함으로써 캐시 일관성 유지 비용을 줄일 수 있다.

준 캐시에서는 클라이언트들이 캐시된 데이타 항목들에 대해 캐시된 값과 서버의 실제 데이타 값 사이의 허용되는 편차를 명시할 수 있다. 예를 들면, 어떤 회사의 주식을 갖고 있는 고객은 자신이 알고 있는 주식시세와 실제 장내 주식 시세가 5% 미만의 차이를 보이면서 변할 때는 정확한 주식 시세를 모르더라도 상관 없다고 제시할 수 있다. 이 경우, 실제 주식 시세를 알려주는 서버 입장에서는 마지막으로 알려 준 값에서 5% 라는 한도를 넘어서는 변화를 보일 때만 고객에게 알려주면 된다. 따라서 주식 시세가 매번 바뀔 때마다 알려줘야 하는 비용이 줄어든다.

그러나 이 준 캐시에서도 캐시 일관성 유지를 위한 통신 비용이 필요하다. 특히 갱신이 활발한 환경에서는 이 비용이 상당히 크다. 이 논문에서는 데이타의 갱신 정도와 편차 정보를 토대로, 적응적인 방식으로 이러한 통신 비용을 줄이는 기법을 제시한다.

논문의 구성은 다음과 같다. 2절에서는 준 캐시와 관련된 연구들을 살펴보고, 3절에서는 기본적인 준 캐시의 모델을 설명한다. 4절에서 기본 모델에 적응적 기법을 적용하여 통신 비용을 줄이는 방법을 제시한다. 5절에서는 기본 모델과 적응적 모델을 비교·실험하고 6절에서 결론을 맺는다.

2. 관련연구

[1]에서는 준 캐시에서 클라이언트가 서버에 제시할 수 있는 편차의 형태를 정의하였고, 준 캐시가 질의 처리를 하는데 얼마만큼의 이득을 줄 수 있는가에 대한 성능분석을 하고 있다. 하지만 실제 클라이언트가 제시

하는 편차가 동적으로 변함으로써 생길 수 있는 클라이언트에서 서버로의 통신 비용은 고려하고 있지 않다. 또한 적응적인 방식으로 준 캐시를 유지하지도 않는다. [4]에서는 캐시의 정밀도와 정밀도를 유지하기 위한 통신 비용의 상관관계를 분석하고 적응적으로 정밀도를 유지하는 방법을 제시한다. 그러나 [4]에서는 데이타의 정밀도를 결정하는 것이 클라이언트가 아니고 서버이기 때문에, 클라이언트에서 서버로 편차 정보를 보내는 환경에서 생기는 통신 비용에 대한 문제를 고려하고 있지 않다. [5]에서도 [4]와 비슷하게 캐시의 정밀도와 그 유지 비용에 대한 논의를 하고 있다. 그러나 [5]에서 제시되는 편차의 형태는 데이타 항목의 값 자체에 대한 것이 아니고 데이타 항목에 적용된 갱신의 횟수이다. 따라서 편차의 형태가 데이타 항목의 값 자체에 대한 환경에 대해 일반적으로 적용시킬 수 없다.

3. 기본 모델 (Basic Model)

이 절에서는 준 캐시의 기본 동작 환경을 알아본다.

데이타의 원본은 서버에 있으며 데이타에 대한 갱신은 서버에서만 발생 하고, 갱신이 일어난 데이타 항목은 전체 클라이언트로 방송된다고 가정한다. i번째 클라이언트는 일단 서버로부터 데이타 항목 x에 대해 방송된 값을 받고 나서 x에 대한 편차 구간정보를 보낸다. 따라서 x를 캐시하고 있는 클라이언트들은 모두 동일한 값을 가지고 있게 된다. 편차 구간 정보는 방송된 값을 기준으로 한 하한 편차와 상한 편차를 포함한다. i번째 클라이언트의 데이타 x에 대한 편차 구간 정보를 δ_x^i 라 하면, 다음과 같다.

$$\delta_x^i = \{i, x, -lb_i, ub_i\} \text{ (단, } lb_i > 0, ub_i > 0).$$

이는 i번째 클라이언트는 서버에서 x에 대한 갱신이 일어났을 때에 새로운 값이 마지막으로 방송된 값으로부터 lb_i 크기 이내에서 작아지거나, ub_i 이내에서 커지는 경우에는 새로운 값을 받을 필요가 없다는 것을 의미한다.

서버에서는 이러한 각각의 편차 구간의 정보를 모아 항목 x에 대한 전체적인 편차 구간을 만든다. 이때에는 모든 클라이언트의 요구 조건을 만족시킬 수 있어야 하므로, 데이타 항목 x에 대한 클라이언트들의 모든 편차 정보 중 가장 큰 하한 편차를 전체 하한 편차로, 가장 작은 상한 편차를 전체 상한 편차로 잡으면 된다. 데이타 x에 대한 전체 편차 구간 정보를 Δ_x 라 하면, 다음과 같다.

$$\Delta_x = \bigcup_{i}^{k} \delta_x^i$$
$$= \{-lb_{highest\_lower}, ub_{lowest\_upper}\}$$
(i : 클라이언트 아이디, k : 클라이언트 수).

서버에서는 이 전체 편차정보를 이용하여, 갱신이 발생했을 때 갱신된 값의 변화가 $lb_{highest\_lower}$ 보다 작아지거나, ub_{lowest_upper} 보다 커지는 경우에 값을 방송한다.

새로운 값을 받은 클라이언트는 받은 항목이 자신이 가지고 있는 항목인 경우 새로운 편차 구간을 작성하여 서버로 전송할 수도 있고, 편차 구간을 변경시키지 않고 이전 편차 구간을 그대로 사용할 수도 있다.

그러나 이 기본 모델에서의 문제는 서버에서의 데이타 갱신이 빈번하게 일어나고 연관된 전체 편차 구간이 작은 경우 새로운 값이 자주 방송되는 것이다. 그러면 방송된 데이타 값에 대해서 새로운 구간 정보를 보내는 클라이언트들로 인해 통신 비용이 커진다.

4. 적응적 모델 (Adaptive Model)

기본 모델의 문제는 데이타가 갱신되는 정도가 전체 편차 구간의 크기에 비해 크다는 데 있다. 따라서 그 비율을 계산하여 데이타가 갱신되는 추이가 편차 구간의 크기에 비해 크다면, 해당 데이타 항목에 대해서는 준 캐시를 적용하지 않는 것이 비용면에 있어 이득이 된다. 이를 위해 특정한 크기의 윈도를 두고 윈도 안에서 데이타의 갱신 크기와 편차 구간을 모니터링(monitoring)하여 준 캐시 유지 여부를 결정한다. 다음은 그 비율을 계산하는 과정이다. 사용되는 기호는 표 1과 같다.

편차 구간을 만들어 내는 클라이언트 수	k
편차 구간 메세지 크기	C_1
방송 메세지 크기	C_2
데이타 항목의 평균 갱신 크기 변화	s
단위 시간당 갱신 빈도수	μ
전체 편차 구간에서의 상한값 (>0)	i^+
전체 편차 구간에서의 하한값 (<0)	i^-

표 1 사용 기호

준 캐시를 적용하는 경우, 데이타 항목 갱신에 따른 방송으로 인해 발생하는 클라이언트들의 편차 메세지 크기의 합은 $k \times C_1$ 이다. 갱신으로 인해 데이타 항목 값이 전체 편차 구간을 벗어나는데 걸리는 시간을 t라 하면, 다음과 같다.

$$t = \frac{i^+}{\mu \times s}(s > 0), \quad t = \frac{i^-}{\mu \times s}(s < 0)$$

편차 메세지들이 이 시간 동안에 나뉘어져 들어 오는 것이라고 생각하면 준 캐시를 적용할 때 생기는 통신 비용은 다음과 같다.

$$Cost_{quasi} = \frac{(k \times C_1)}{t} = \frac{k \times C_1 \times \mu \times s}{i^*} \quad \cdots\cdots\cdots(1)$$

한편 준 캐시를 하지 않는 경우에는 갱신이 일어날 때 마다 메세지를 방송하므로 그 비용은 다음과 같다.

$$Cost_{non-quasi} = \mu \times C_2 \quad \cdots\cdots\cdots(2)$$

따라서 (1)의 값이 (2)의 값보다 큰 항목에 대해서는 준 캐시를 유지하지 않고 값이 갱신될 때마다 바로 방

송을 하고, (2)가 (1)보다 큰 경우에는 준 캐시를 유지하면 통신 비용면에서 이득이 된다. (1) ≥ (2)가 되는 식을 계산하여 정리하면 다음과 같은 부등식을 얻을 수 있다.

$$\frac{k \times s}{i^{t}} \geq \frac{C_2}{C_1} \quad \cdots\cdots(3)$$

(3)의 왼쪽 항을 보면, i^t가 작으면 분모가 작아져 식을 만족시킬 확률이 커진다. 이때는 작은 구간으로 인해 데이타 방송이 자주 일어나는 경우이기 때문에 준 캐시를 적용하지 않는다. 한편, 분모가 작더라도 데이타의 갱신 정도, 즉 s가 작은 경우에는 식이 만족되지 않으므로 준 캐시를 그대로 유지하게 된다.

5. 실험

실험은 UltraSparc II 333MHz에서 JavaSim[6] 시뮬레이터(simulator)를 사용하여 수행하였다. 서버에서의 데이타 항목에 대한 갱신은 균등 분포를 가정하되, 빈번한 갱신이 이루어지는 항목에 대해서는 평균적으로 값이 증가하는 방향으로 갱신이 이루어지도록 하였다.

전체 데이타 항목 수	200
갱신이 빈번한 항목 수	40
클라이언트 수	20
클라이언트 데이타 스큐(skew)	0.2
클라이언트의 데이타 항목 수	20
편차 구간 변화 항목 비율	0.2
갱신 항목의 전체에 대한 선택 비율	0.3
클라이언트 메세지 크기 (byte)	85
서버 메세지 크기 (byte)	64
기본 윈도 크기	3

표 2 기본 파라미터(parameter) 값

표 2는 실험에 쓰인 기본 파라미터 값들이다. 클라이언트 데이타 스큐(skew)는 클라이언트가 갖고 있는 데이타 항목 중 얼마만큼이 서버에서 갱신이 활발하게 이루어지는 항목인가를 나타내는 비율이다. 편차 구간 변화 항목 비율은 클라이언트에 새로이 갱신된 항목 값이 왔을 때 편차 구간이 변하는 항목의 비율이다. 갱신 항목의 전체에 대한 선택 비율은 한번 갱신이 이루어질 때 선택되는 데이타 항목 수의 전체 항목 수에 대한 비율이다. 기본 윈도 크기가 3이라는 것은 갱신이 3번 이루어질 동안에 데이타 항목에 적용된 갱신의 크기와 편차 구간 변화 측정치를 토대로 (3)의 식을 계산한다는 의미이다.

그림 1은 적응적 모델과 기본 모델의 비교를 나타내는 실험 결과이다. 그림에서 X축은 서버에서 한번 갱신이 발생할 때 갱신되는 항목들에 얼마나 스큐가 일어나는 지를 나타내는 것이다. 오른쪽으로 갈수록 한번 갱신되었던 항목이 다음 번에도 갱신될 확률이 높아짐을 나타낸다. 이때 측정된 총 메세지의 크기를 비교해 보면, 갱신 스큐가 0.3보다 작은 경우에는 두 모델이 별 차이를 보이지 않으나, 데이타 항목에 대한 갱신이 빈번하게 이루어지는 경우 적응적 모델이 기본 모델보다 더 작은 통신 비용을 보임을 알 수 있다.

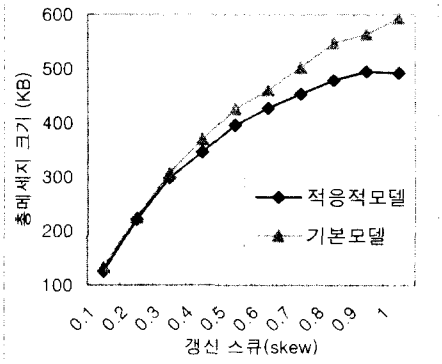

그림 1 갱신 스큐에 따른 통신 비용 비교

6. 결론

이 논문에서는 준 캐시의 기본 모델에서 데이타 항목에 대한 갱신이 빈번히 일어날 때 일관성 유지를 위해 생기는 통신 비용의 문제를 지적하였다. 그리고 적응적 모델을 통해 이러한 기본 모델의 문제를 해결하였다. 적응적 모델에서는 데이타 갱신 정도와 편차 구간의 크기 비율 변화를 측정한다. 그리고 이 비율 변화에 따라 데이타 항목 각각에 대한 준 캐시 적용 여부를 판별하여 선택적으로 준 캐시를 적용한다. 실험을 통해 데이타 갱신이 활발한 환경에서는 적응적 모델이 기본 모델보다 캐시 일관성 유지를 위한 통신 비용이 작음을 확인할 수 있다.

참고문헌

[1] Rafael Alonso, Daniel Barbara, and Hector Garcia Molina, "Data Caching Issues in an Information Retrieval System," *ACM TODS* 15(3), pp. 359-384, 1990.

[2] Daniel Barbara, "Mobile Computing and Databases - A Survey," *IEEE TKDE* 11(1), pp. 108-117, 1999.

[3] Daniel Barbara and Tomasz Imielinski, "Sleepers and Workaholics: Caching Strategies in Mobile Environments," *ACM SIGMOD Record* 23(2), pp. 1-12, 1994.

[4] Chris Olston, Boon Thau and Jennifer Widom, "Adaptive Precision Setting for Cached Approximate Values," *ACM SIGMOD Conference*, pp. 355-366, 2001.

[5] Yixiu Huang, Robert H. Sloan and Ouri Wolfson, "Divergence Caching in Client-Server Architectures," *IEEE PDIS*, pp. 131-139, 1994.

[6] JavaSim manual Ver0.3, http://javasim.ncl.ac.uk/manual/javasim.pdf.