이동 데이터베이스 시스템에서 최근 질의테이블을 이용한 캐쉬 유지 기법

곽도위*, 전홍태*, 김양택*, 윤성대**
*부경대학교 전산정보학과
**부경대학교 전자계산학과
e-mail:dowiduk@baesan.psks.ac.kr

A Schemes for cache Maintenance using Recently Query Table in a Mobile Database System

Do-Wee Kwak*, Hong-Tae Jeun*, Yang-Taek, Kim*, Sung-Dae Youn**
*Dept. of Computer and Information, Pukyong National University
**Dept. of Computer Science, Pukyong National University

요 약

이동 컴퓨팅에서의 배터리 한계, 제한된 대역폭의 단점 보안을 위해서 이동 클라이언트에 캐쉬를 사용한다. 이 경우 빠른 질의 응답과 대역폭 절약면에서는 효과적이지만, 서버와의 캐쉬 일관성을 유지해야 하는 단점이 있다. 통신의 오랜 단절로 인해 서버로부터 무효화보고만으로 유효성을 확인할 수 없을 때, 캐쉬내의 전체 데이터를 버리든지 캐쉬 내의 데이터 유효성에 대해 서버에게 요청해야 한다. 본 논문에서는 가장 최근에 질의된 데이터와 질의 빈도수를 유지하여 유효성 확인 요청시 효율적으로 대역폭을 감소할 수 있는 최근 질의테이블을 이용한 캐쉬 유효성 기법을 제안한다.

1. 서론

이동 통신 기술의 발달로 언제 어디서든 필요한 데이터를 받을 수 있게 되었고 이동 정보 서비스의 기능은 크게 증가하였다.

이동 컴퓨팅 기술은 제한된 대역폭, 이동 단말기의 전기 공급 능력 한계성과 무선통신망 자체의 불안정과 같은 문제점이 있다. 이 문제점을 해결하기 위한 방법으로 클라이언트의 기억장치에 캐쉬를 두는 캐싱기법이 있다[3]. 이 기법은 질의 응답시간이 단축되고 대역폭도 절약할 수 있으나 캐쉬 일관성을 유지해야하는 단점이 있다.

캐쉬 일관성 유지를 위한 기법으로는 서버에서 갱신된 객체에 대해 주기적으로 방송되는 무효화보고를 통해 캐쉬 일관성을 유지하는 방법이 있다[1]. 이 기법은 이동 클라이언트가 무효화보고를 보고 캐쉬내의 유효하지 않은 데이터를 캐쉬에서 삭제하는 기법으로, 오랜 접속 단절로 인해 서버로부터 무효화보고를 수신할 수 없게 되면 캐쉬 내의 모든 데이터를 무효화 해야하는 단점이 있다.

또 다른 방법으로 무효화보고만으로 캐쉬 유효성 확인을 할 수 없을 경우 서버에게 요청하여 유효성 확인을 체크하는 방법이 있다[3]. 이 방법에는 데이터 항목 전부를 보내는 방법과 그룹별로 데이터를 보내는 두 가지 방법이 있다. 이 방법들은 전체 데이터 무효화를 방지할 수는 있으나, 대역폭 소모가 크다는 점, 잘못된 무효화가 발생할 수 있는 단점이 있다.

그러므로, 본 논문에서는 전체 무효화를 방지하면서 대역폭 소모량을 최소화하고, 잘못된 무효화 발생을 최소화할 수 있는 캐쉬 유효화 기법을 제안하

고, 기존의 기법들과 성능을 비교하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구에 대해서 알아보고, 3장에서 본 논문에서 제안하는 기법에 대해 설명하고, 4장에서는 시뮬레이션을 통한 기존 기법들과의 성능 비교를 보여주고, 5장 결론으로 마무리한다.

2. 관련 연구

2.1 단순 체킹 기법 (Simple Checking Caching)

오랜 단절 후에도 캐쉬에 유효한 데이터가 존재하고 그 데이터는 서버에서 방송되는 무효화보고만으로 캐쉬 유효성을 보장받지 못한다. 그 데이터를 계속 사용하기 위해서는 서버에게 데이터 유효성의 확인을 받아야만 한다. 단순 체킹 기법은 캐쉬내의 모든 데이터 식별자와 최근의 타임스탬프를 서버에게 전송하여 캐쉬 유효성 확인을 하는 방법이다[3].

2.2 그룹 체킹 기법 (Simple Grouping Caching)

서버와의 통신시 대역폭 소모량을 감소하기 위해 데이터베이스를 그룹으로 나누어 유효성을 확인하는 방법이다[3]. 그러나, 그룹내 하나의 데이터만 갱신되어도 그룹 전체의 데이터가 무효화되는 단점이 있다.

3. 최근 질의테이블을 이용한 캐쉬 유효화 기법

다시 사용될 가능성이 많은 데이터만 추출하여 서버에게 유효성 확인 요청이 가능하다면 대역폭 소모량도 줄이면서 동시에 잘못된 무효화도 방지할 수 있다. 자주 사용되는 데이터는 캐쉬에 오래 머물러 있고 그 질의빈도도 높다.

본 논문에서는 캐쉬 내에 있는 데이터들 중 자주 질의가 일어나면서 최근의 데이터들만을 모아놓은 최근 질의테이블(Recently Query Table : RQT)을 구성한다. 오랜 단절 후 무효화보고 만으로 캐쉬 내의 데이터 유효성을 판단할 수 없을 때 최근 질의테이블의 데이터만을 전송함으로써 대역폭 소모량도 줄이고 다시 사용될 데이터들의 유효성을 체크함으로써 잘못된 무효화가 일어나는 확률을 감소시켰다.

```
Recently Query Table (RQT)
Data_ID,
Re_Time,
Que_Cnt
```

(그림 1) 최근 질의테이블의 구성

그림 1과 같이 최근 질의테이블은 데이터식별자(Data_ID), 가장 최근 질의시간(Re_Time), 질의빈도수(Que_Cnt)로 구성된다. 여기서 Data_ID는 클라이언트의 캐쉬에 질의된 객체를, Re_Time은 그 객체가 마지막으로 읽은 시간을, Que_Cnt는 그 객체의 질의빈도수를 나타낸다. 최근 질의테이블 유지를 위한 알고리즘은 아래의 그림 2, 그림 3과 같다.

```
IF Data_ID ∈ DRQ
    Update Re_Time of Data_ID
    Add 1 TO Que_Cnt
ELSE
    Insert Data_ID, Re_Time, Que_Cnt
END IF
```

(그림 2) 데이터 질의 후 최근 질의테이블 갱신

그림 2에서 클라이언트는 캐쉬에 데이터를 읽은 후, 최근 질의테이블의 그 객체에 대한 정보를 갱신시킨다.

만약 최근 질의 테이블에 그 객체가 존재하지 않으면, 그 객체의 식별자와 시간, 질의빈도수를 1로 초기화한다. 객체가 존재한다면, 그 객체의 최근 질의시간을 변경하고, 질의빈도수를 증가시킨다.

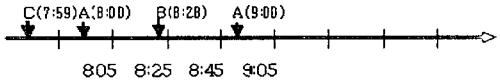
```
IF ( Data_ID is Invalid ) AND ( Data_ID ∈ DRQ )
        Delete the Data_ID of DRQ
END IF

IF ( Re_Time < T_lb-W×L ) AND ( Que_Cnt < Avg_Cnt )
        Delete the Data_ID of DRQ
END IF
```

(그림 3) IR 받은 후 최근 질의테이블 갱신

서버로부터 무효화보고를 받은 후 클라이언트는 최근 질의테이블을 갱신한다. 최근 질의테이블에 존재하는 객체가 무효화보고에 의해 클라이언트의 캐쉬에서 무효화되었다면 그 객체는 최근 질의테이블에 더 이상 존재할 필요가 없다. 최근 질의테이블은 캐쉬 내의 데이터 중 가장 최근에 질의된 history를 나타내 주기 때문이다. 따라서 최근 질의테이블에서도 삭제되어야 한다.

그리고, 최근 질의 객체만을 유지하기 위해서 객체의 Re_Time이 $T_{lb}-w\times L$ 보다 작은 객체는 최근 질의테이블에서 삭제된다. $T_{lb}-w\times L$은 최근 질의테이블을 유지할 수 있는 시간을 나타낸다. 하지만, 질의된 시간이 오래되었다 하더라도 질의빈도가 많

은 객체는 다시 사용될 가능성이 높은 객체이기 때문에 최근 질의테이블 내에서의 평균 질의빈도수 보다 많은 객체는 삭제하지 않는다.



1. T_{lb} = 8:45 일때, 최근 질의 테이블

Data_ID	Re_Time	Que_Cnt
C	7:59	1
A	8:00	1
B	8:28	1

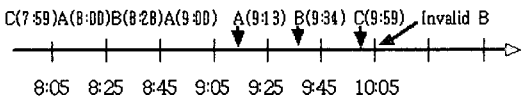
2. T_{lb} = 9:05 일때, 최근 질의 테이블

Data_ID	Re_Time	Que_Cnt
A	9:00	2
B	8:28	1

(그림 4) 데이터 질의 후의 최근 질의 테이블

그림 4는 캐쉬에서 데이터 질의가 발생한 후 최근 질의테이블을 갱신한 예이다. 서버로부터 20분 간격으로 무효화보고를 받는 이동 클라이언트가 있다. 최근 질의테이블을 유지를 위한 무효화 윈도우 간격이 W=3일 경우에 먼저 8:45에 무효화보고를 받은 최근질의테이블을 보면, 각각 C(7:59), A(8:00), B(8:28) 시각에 객체가 질의되었고 최근 질의테이블에 시간과 빈도수가 각각 Insert 되었다.

9:05에는 A(9:00)가 다시 질의되어 Re_Time이 9:00으로 수정되고, 빈도수는 하나 증가했다. 그리고 C(7:59) < $T_{lb} - w \times L$ (8:05 → 9:05 - 3× 20) 이고, 질의빈도수 1 < Avg_Cnt (1.3 → 4/3) 보다 작기 때문에 최근 질의테이블에서 삭제되었다.



3. T_{lb} = 9:45 일때, 최근 질의 테이블

Data_ID	Re_Time	Que_Cnt
A	9:13	3
B	9:34	2

4. T_{lb} = 10:05 일때, 최근 질의 테이블

Data_ID	Re_Time	Que_Cnt
A	9:13	3
C	9:59	1

(그림 5) IR을 받은 후 최근 질의테이블 예

그림 5는 무효화보고를 받은 후의 최근 질의테이블의 예를 보여준다. 9:45에 무효화보고를 받은 후, 최근 질의테이블을 보면 A(9:13)에 추가 질의되어 Update되고 B(9:34)에 질의되어 최근 질의시간과 질의빈도수가 증가했다. 10:05의 예를 보면, 객체 C가 9:59에 질의되어서 새롭게 Insert되고 무효화보고에 의해서 이동 클라이언트 캐쉬 내에서 B가 무효화되었으므로 최근 질의테이블에서도 삭제된다.

4. 시뮬레이션

4.1 시뮬레이션 파라미터

본 논문의 시뮬레이션은 서버와 클라이언트의 환경에서 시행한다. 이 시뮬레이션에서 서버는 전체 데이터의 내용을 가지고 있고, 서버 측에서만 갱신이 일어난다. 서버는 클라이언트 캐쉬와의 데이터 일관성을 위해서 주기적으로 무효화보고를 방송한다. 이동 클라이언트는 판독 연산만을 수행하며, 이동 클라이언트에서 발생된 질의가 클라이언트 캐쉬 내에 존재하지 않을 경우, 서버에 데이터를 요청하고, 서버는 클라이언트에게 데이터를 전송한다. 무효화보고만으로 클라이언트의 캐쉬 유효성을 판단할 수 없을 때 이동 클라이언트는 서버측으로 캐쉬 유효성 확인 요청을 한다.

표 1은 시스템 및 작업 파라미터들을 나타낸 것이다. 대부분은 기존 연구[1]-[3]의 것을 그대로 사용하거나 조금 수정한 것이다.

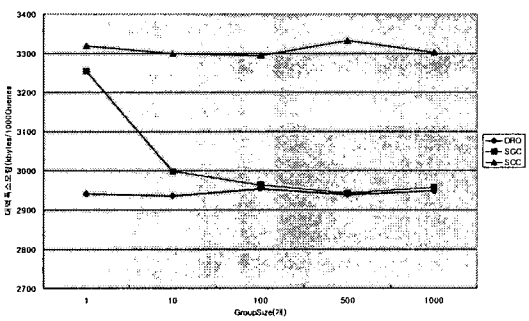
본 시뮬레이션에서 데이터 처리 단위는 객체로 가정한다. DBSize는 서버 데이터베이스의 객체의 수이다. CacheSize는 이동 클라이언트의 캐쉬 내에 포함할 수 있는 객체의 수이다. ObjSize는 하나의 객체 크기를 나타내고, IdSize는 하나의 객체나 한 그룹의 식별자 크기를 나타낸다. groupSize는 한 그룹이 포함할 수 있는 객체 수를 나타낸다. w는 무효화보고 윈도우 크기를 나타낸다. W는 최근 질의테이블에서 유지되는 객체를 나타낸다. BroadInter는 무효화보고가 방송되는 주기를 나타내고, UplinkCapacity는 서버에서 클라이언트로 무효화보고를 방송할 때나 클라이언트가 서버에 데이터 유효성 요청을 할 때의 대역폭 크기를 나타낸다. QueryCntProb는 질의빈도가 높은 데이터가 다시 질의될 확률을 나타낸다. 예를 들어, 1000건의 질의 중 QueryCntProb가 0.02이라면 50건 중에 한 번 자주 질의되는 데이터가 질의됨을 의미한다.

<표 1> 시스템 및 작업 파라미터

파라미터	내 용	기본설정치(단위)
DBSize	서버 데이터베이스 크기	100,000(개)
CacheSize	클라이언트 캐쉬 크기	5000(개)
ObjSize	하나의 객체 크기	256(bytes)
IdSize	한 객체,그룹 식별자 크기	64(bits)
groupSize	그룹이 포함하는 객체수	100(개)
w	무효화보고의 윈도우 크기	10
W	갱신된 객체 유지 크기	60
BroadInter	무효화보고 방송 주기	20(초)
UplinkCapacity	송수신 및 방송을 위한 대역폭 크기	1.2(Kbytes/sec)
Tstamp	타임스탬프 크기	64(bits)
TranRate	갱신 트랜잭션 도착률	0.01(개/초)
UpdateSet	갱신이 자주 발생되는 지역의 크기 비율	0.1
UpdateProb	UpdateSet이 갱신될 확률	0.9
QueryRate	초당 생성되는 질의 수	0.1(개/초)
DisconnetProb	트랜잭션 단절 확률	0.1
DisconnetTime	평균 단절 유지 시간	200(초)
QueryCntProb	질의빈도가 높은 데이터의 비율	0.02

4.2 시뮬레이션 결과

그룹 식별자를 이용하면 캐쉬 일관성 유지를 위한 대역폭은 줄일 수 있으나, 잘못된 무효화 발생으로 인해 해당 데이터를 다시 서버로부터 읽어와야 한다. 따라서 본 시뮬레이션은 성능 평가 척도를 캐쉬 일관성 유지를 위해 소모되는 대역폭과 데이터 검색 시 소모되는 대역폭의 합으로 나타낸다.



(그림 6) 그룹 크기에 따른 대역폭 소모량

그림 6은 그룹 크기 변화에 따른 대역폭 소모량을 나타낸 것이다. 본 논문에서 제안한 기법이 그룹 크기에 상관없이 다른 기법보다 낮은 대역폭을 소모함을 알 수 있다. SGC기법은 그룹의 크기가 커져가면서 대역폭이 줄어들다가 그룹의 크기가 너무 커지면 잘못된 무효화 발생으로 대역폭이 증가한다.

5. 결론

본 논문에서는 최근 질의테이블을 이용하여 유지함으로서 이동 클라이언트의 캐쉬 내에 가장 최근 질의자료를 저장함으로서 캐쉬 유효성 확인 요청시 최근 질의테이블의 데이터를 전송하여 대역폭 소모량을 감소시키는 기법을 제안하였다. 또한 최근 질의테이블에는 다시 사용될 가능성이 높은 데이터만을 유지함으로서 이동 클라이언트에서 다시 질의했을 때 잘못된 무효화 발생을 최소화할 수 있다.

시뮬레이션을 통하여 제안한 최근 질의테이블을 이용한 기법이 다른 기법들보다 대역폭 소모량 측면에서 우수함을 알 수 있었다.

참고문헌

[1] D. Barbara and T.Imielinski, "Sleepers and Workaholics : Caching Strategies in Mobile Environments," Proc. ACM SIGMOD Conf. on Management of Data, pp.1-12, 1994.

[2] J.Cai et al., "On Incremental Cache Coherency Schemes in Mobile Computing Environments," Proc. Int'l Conf. on Data eng., pp.114-123, 1997.

[3] K. Wu et al., "Energy-Efficient Caching for Wireless Mobile Computing," Proc. IEEE Int'l Conf. on Data Eng., pp.336-343, 1996.

[4] T.Imielinski and B.R. Badrinath, "Mobile wireless computing," Communications of the ACM, vol. 37, no. 10, pp.18-28, 1994.

[5] 임상민, 강현철, "이동 데이터베이스 시스템에서 효율적인 캐쉬 일관성 유지 기법," 정보처리학회 논문지 D 제8-D권 제3호, pp.221-232, 2001.