제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003.5)

방송환경에서 효율적인 갱신 거래 스케줄링 기법의 성능분석

박 준*, 황부현*, 정승욱**, 김중배**
*전남대학교 전산학과
**한국전자통신연구원
e-mail:kingrion@sunny.chonnam.ac.kr

The Performance Analysis of Efficient Update Transaction Scheduling Method in Broadcasting Environments

Joon Park*,Buhyun Hwang*,SeungWoog Jung**,JoongBae Kim**
*Dept of Computer Science, Chonnam National University
**ETRI

요 약

방송환경은 서버에서 이동 클라이언트로의 대역폭보다 이동 클라이언트에서 서버로의 대역폭이 상당히 좁은 비대칭 구조이다. 이러한 방송환경을 기반으로 하는 이동 컴퓨팅 응용 시스템은 제한된 대역폭의 효율성과 거래의 수행성능을 향상시키기 위한 비용 효과적인 방법이 요구된다.

본 논문에서는 방송환경에서 이동 클라이언트의 갱신연산이 가능한 경우에 데이터의 현재성과 상호일관성을 만족하기 위한 효율적인 거래 스케줄링 알고리즘을 소개하고 시뮬레이션을 통하여 기존에 제안된 기법과 비교 분석함으로써 제안하는 알고리즘의 우수성을 확인한다.

1. 서론

많은 수의 이동 클라이언트를 갖는 주식 거래 정보, 금융 정보 서비스등과 같은 이동 컴퓨팅 응용 시스템은 방송기법을 이용한 데이터 전달방식이 적합하다. 서버는 주기적인 방송을 통하여 이동 클라이언트에 무효화 보고서를 전달하고 이동 클라이언트는 무효화 보고서를 듣고서 데이터의 상호일관성을 유지할 수 있다[1].

그러나 방송 환경은 비대칭적인 대역폭을 갖는 것이 특징이다. 즉, 서버에서 이동 클라이언트와의 대역폭보다 이동 클라이언트에서 서버로의 대역폭이 상대적으로 좁다. 다수의 이동 클라이언트가 존재하는 방송환경에서는 제한된 대역폭(uplink)의 사용을 최소화하면서 데이터의 상호 일관성을 유지할 수 있는 방법이 필요하다.

본 논문에서는 캐쉬를 사용하고 주기적인 방송으로 캐쉬 데이터의 일관성을 유지하는 것을 기본 가정으로 하고 있다. 제한된 대역폭을 효율적으로 사용할 수 있는 동적인 데이터 갱신 패턴 비율을 적용한 스케줄링 기법을 구현하고 성능 분석을 통하여 기존의 제안된 기법보다 월등한 수행 성능을 나타냄을 보인다.

이 논문의 구성은 다음과 같다. 제2장에서는 이동 컴퓨팅 환경에서 캐쉬 일관성 유지와 동시성 제어 기법등 관련 연구들을 살펴보고, 제 3장에서는 제안하는 동시성 제어 기법인 UTSM-ARM(An Update Transaction Scheduling Method using Adaptive Request Message)을 소개한다. 제 4장에서는 제안한 UTSM-ARM의 구현을 위한 시스템 모델에 대해서 살펴본다. 제 5장에서는 구현한 시스템의 성능을 비교 분석하고, 마지막으로 제 6장에서는 이 논문의 결론 및 향후 연구방향에 대하여 기술한다.

2. 관련 연구

방송환경에서 캐싱기법은 이동 거래의 응답 시간을 향상시킬 수 있고, 이동 클라이어트에서 서버로의 좁은 대역폭을 효율적으로 사용할 수 있다. 그리고 서버는 캐쉬 데이터의 일관성 유지를 위하여 무효화 보고서를 주기적으로 방송한다[2]. 이러한 방송환경에서 캐쉬데이터의 일관성을 유지하기 위한 많은 연구가 제안되었다. Push기법과 Pull기법 그리고 이 두 가지를 혼합한 기법[1], 무효화 보고서를 최적화 하는 기법[3], 캐쉬 데이터의 유효성을 고려한 일관성 유지 기법[4] 등이 제안되었다. 그러나 제한된 기법들은 이동 클라이언트에서 수행하는 대부분의 거래를 읽기 전용으로 제한하였다.

그리고 갱신 거래가 가능한 방송환경에서 캐쉬를 사용하는 동시성 제어 방법으로 OCC-UTS (Optimistic Concurrency Control with Update TimeStamp)가 제안되었다[6]. 그러나 OCC-UTS 기법에서는 읽기 전용 거래와 갱신거래와의 사이클 발생으로 이동거래의 철회율과 철회될 수 있는 이동거래의 완료요청으로 대역폭(uplink) 사용비율이 증가될 수 있는 문제점이 있다.

본 논문에서는 이러한 비대칭적인 대역폭을 갖는 방송환경에서 불필요한 대역폭 사용을 최소화하고 이동 거래 철회율을 줄일 수 있는 새로운 기법을 제안한다.

3. 제안하는 기법

본 논문에서 제한하는 기법은 데이터의 동적인 갱신 패턴 비율을 추가한 무효화 보고서의 방송으로 캐쉬 일관성을 유지한다. 그리고 비대칭적인 대역폭 구조를 갖는 방송환경에서 이동 거래의 철회율을 줄이고 이동 응용 시스템의 성능을 향상시킬 수 있는 UTSM-ARM을 제안한다.

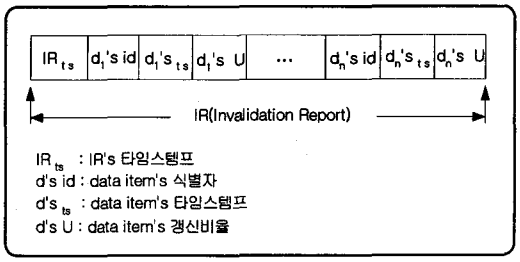
```
/* 데이터의 갱신 패턴 비율 */
Compute_Update_Pattern_rate(){
  if( CommitRequest Upadte Transaction is Commit ){
     Update Transaction DataItem(J)'s U_rate
       = ( DataItem(J)'s Update_count/BC_count ):
     Add to DataItem(J)'s U_rate In Next IR:
  }
}

// U_rate : 한 방송 주기 동안 데이터의 갱신 비율.
// BC_Count : 최근 C(C:상수)방송 횟수.
// DataItem(Di)'s Update_count : BC_Count 내에서의
// 데이터의 갱신 된 카운트
```

[그림 1] UTSM-ARM의 갱신 패턴 비율 알고리즘

위의 그림 1은 캐쉬 데이터의 일관성 유지를 위한 UTSM-ARM의 갱신 패턴비율 적용 알고리즘이다. 각 데이터에 대한 동적인 갱신 패턴비율은 최근 방송 주기횟수(상수:C) 동안에 서버에서 거래 완료된 각 데이터의 확률값이다.

그리고, 각 데이터(*d*)의 갱신 비율(*d'sU*)은 L초마다 $ts_i = iL$인 시간에 방송된다. 그리고 무효화 보고서는 방송 타임스탬프(IR_{ts})와 각 데이터에 대한 가장 최근의 정보($d's_{id}$, $d's_{ts}$, $d's_{valuc}$, $d's_U$)를 리스트로 구성하여 이동 클라이언트에 전달된다. 다음 그림 2는 UTSM-ARM에서는 방송되는 무효화 보고서의 구조를 나타낸다.



[그림 2] UTSM-ARM의 무효화 레포트

여기서 $d's_{ts}$는 $ts_i - wL < d's_{ts} < ts_i$ (w는 무효화 방송 윈도우)인 d의 마지막 갱신 타임스탬프이다. 어떤 하나의 완료된 거래에 의해 갱신된 모든 데이터 항목들은 같은 타임스탬프를 갖는다. 또한, 무효화 보고서 IR_{ts}를 구축하기 위해 다음과 같이 정의되는 갱신 트랜잭션 $U(ts_i)$를 유지한다.

$U(ts_i)$ = $[d, d_{ts}]$ /d는 $ts_i - wL$ 과 ts_i 사이에 완료된 트랜잭션들에 의해 갱신된 데이터이고, d_{ts}는 $ts_i - wL <= d_{ts} <= ts_i$ 인 d가 마지막 갱신된 타임스탬프

그리고 이동 클라이언트의 읽기 전용 거래는 데이터에 대한 동적인 갱신 패턴 비율을 적용하여 캐쉬 데이터의 유효성 유무를 판단한다. 즉, 캐쉬내의 데이터 j에 대한 갱신 패턴비율(j_u)을 한 방송주기 동안에 갱신될 수 있는 판단 기준인 상수값(a)와 비교하여 현재 방송주기 내에 데이터의 무효화 가능성을 결정한다. 다음은 동적인 갱신 패턴 비율을 적용한 이동 클라이언트의 연산 처리 알고리즘이다.

if($j_u \geq a$) : 갱신가능성이 있다고 판단하여 서버의 최신의 데이터를 다시 요구한다.
if($j_u < a$) : 갱신가능성이 없다고 판단하여 현재 캐쉬 데이터의 값을 사용한다.

이동 컴퓨팅 환경에서의 동시성 제어 기법은 낙관적인 방법과 비관적인 방법이 있다. 제안하는 기법

은 이동 거래의 각 연산이 직렬성에 위배되지 않는다는 가정을 두고 수행하며 거래의 수행이 종료되는 시점에서 거래에 대한 직렬성을 검증하는 낙관적인 방법을 채택하였다[5]. 그리고 제안하는 기법은 낙관적 동시성 제어 방법에서 서버나 다른 이동 클라이언트가 실행중인 거래에 대한 정보를 모르는 상황에 적합한 후 방향 유효화 기법을 도입하였다[6].

제안하는 UTSM-ARM은 이동거래의 읽기 연산인 경우에 접근하는 데이터 항목들의 캐쉬 내 타임스탬프를 읽기 연산의 시작 시점에 파악하여 그것들 중 가장 큰 값을 이동 거래의 완료요청 때까지 기억한다. 또한, 각 이동 클라이언트들은 갱신 거래을 위하여 다음과 같은 두 가지 집합을 유지한다.

ReadSet(T_{id}) = { *[j, j^c_{ts}] | j*는 갱신 트랜잭션 T_{id}에 의해 읽혀진 데이터항목, j^c_{ts}는 캐쉬 데이터 *j*의 타임스탬프 }

NewValue(T_{id}) = { *[j, j_v] | j*는 갱신 트랜잭션 T_{id}에 의해 갱신된 항목, j_v는 *j*의 새로운 갱신된 값 }

그리고 이동 클라이언트에서 모든 연산을 완료하고 이동 거래를 서버에 완료요청 할 때, 이동 거래의 구분자 T_{id} 및 *ReadSet(T_{id}), NewValue(T_{id}),* 그리고 *RequestToCommit*로 구성된 메시지를 전달한다. 만약, 갱신 연산이 아닌 읽기 연산으로만 구성된 이동 거래의 *NewValue(T_{id})* 값은 *Null* 값으로 전달된다.

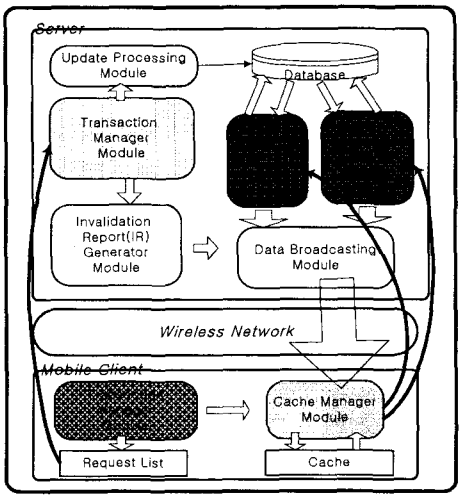
```
/* 이동 클라이언트에서 제출된 트랜잭션 처리*/
Dequeue a message m from Que in the interval[ts_{i-1}, ts_i]{
   if there exists at least one j in m.ReadSet such that t^c_j<t_j{
      put m.T_id in  Abortlist the next IR;
   }else{
      put m.T_id in the next CommitList report;
      /* commit the transaction in the server */
      install the value in the m.newValue into the database;
      recompute U(ts_i) such that all data items in m.Newvalues
      has the same timestamp tj and ts_{i-1} < t_j < ts_i
      /* 갱신이 발생한 데이터에 대한 갱신 비율을 추가 */
      Recompute_update_rate(j);
   }
}
```

[그림 3] UTSM-ARM 이동 거래 직렬성 검증

위의 그림 3은 이동 거래의 직렬성 검증을 위하여 완료 요청된 *RequestToCommit* 메시지들을 받아 큐에 저장한다. 완료 요청된 메시지 m의 *m.ReadSet* 내에 존재하는 데이터 항목 중에서 하나의 데이터라도 서버의 데이터베이스에 존재하는 동일한 데이터의 타임스탬프 보다 작다면 큐에 존재하는 해당 이동 거래는 *AbortList*에 등록되고, 그렇지 않은 경우는 *CommitList*에 등록한다. 그리고 완료된 이동 거래에 함께 완료 요청된 *m.NewValues*가 존재한다면 *m.NewValues*내의 값들을 데이터베이스에 영구 반영된다. 그리고 *m.NewValues* 내의 모든 데이터 항목들이 $ts_{i-1} < j_{ts} < ts_i$ 과 같은 타임스탬프 j_{ts}를 갖도록 $U(ts_i)$를 갱신한다. 또한, 갱신 완료된 데이터에 대한 갱신 비율을 동적으로 유지하기 위하여 데이터에 갱신 비율을 다시 계산한다. 그리고 다음 방송 주기의 무효화 보고서에 완료리스트와 철회리스트를 추가하여 완료 요청된 이동거래에 대한 결과를 이동 클라이언트에 전달한다.

4. 시스템 모델

본 연구에서 제안하는 UTSM-ARM알고리즘의 시시스템 모델은 다음 그림 3과 같다. 방송환경에서 이동 클라이언트와 서버와의 통신을 위한 수단은 무선 네트워크이다. 그리고 이동 클라이언트와 서버에는 각각의 기능을 담당하는 모듈들이 존재한다.



[그림 4] UTSM-ARM 시스템 모델

1) 서버 모듈

- Transaction Manager Module : 이동 클라이언트에서 완료 요청되는 이동 거래의 처리 및 관리 모듈.
- Update Processing Module : 최종 완료된 이동 거래들의 NewValue를 데이터베이스에 반영하는 모듈.
- Adaptive Request Module : 각 캐쉬 데이터의 동적인 갱신 패턴 비율에 의한 데이터요청을 처리하는 모듈.
- Immediately Caching Module : 이동 클라이언트의 캐쉬에 존재하지 않는 데이터를 참조한 연산의 수행에서 서버에 즉시 요청하는 즉시캐싱 처리를 담당한다.
- Invalidation Report Generator Module : 데이터베이스의 갱신된 데이터, 이동 거래 완료 및 철회에 관한 정보를 전달하기 위한 무효화보고서 생성모듈.

■ Data Broadcasting Module : 이동 클라이언트에 데이터를 전달을 위한 방송 모듈.

2) 이동 클라이언트 모듈

■ Cache Manager Module : 주기적인 방송을 듣고, 데이터의 갱신패턴 비율에 따른 캐쉬 데이터의 일관성 유지를 위한 기능을 담당하는 모듈.

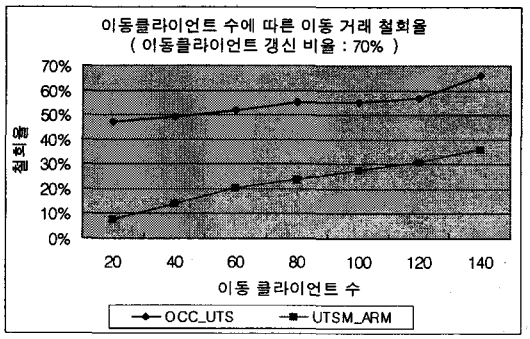
■ Transaction Manager Module : 이동 클라이언트의 캐쉬 데이터를 참조한 연산, 이동 거래의 철회, 완료 및 요청의 기능을 담당하는 모듈.

5. 성능 분석

본 논문에서 제안한 UTSM-ARM은 반송환경에서 갱신 연산이 가능하고 캐쉬를 사용하며 타임스탬프를 이용한 갱신 거래 스케줄링 기법[6]과의 성능을 비교 분석한다. 성능에 대한 비교 분석 기준은 동시에 접근한 이동 클라이언트의 수에 따른 이동 거래의 철회율과 이동 클라이언트에서 서버로의 대역폭(uplink)에 사용비율이다.

■ 트랜잭션 철회율에 따른 성능 분석

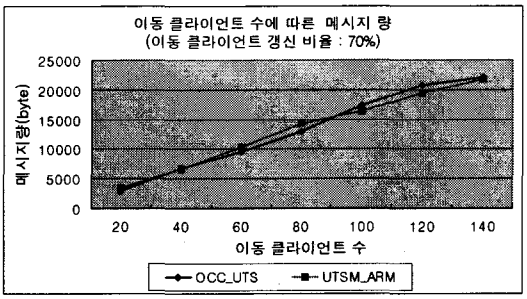
다음 그림 4는 이동 클라이언트의 수에 따른 OCC-UTS기법과 UTSM-ARM의 트랜잭션 철회율에 관한 수행 성능 결과이다. 동시에 수행중인 이동 클라이언트의 수가 20일 경우에 두 기법의 성능 차이가 가장 크게 나타났다. 그리고 이동 클라이언트의 수가 점점 증가할수록 각 기법의 이동 트랜잭션의 철회율은 증가하는 결과를 확인할 수 있다.



[그림 5] 이동 클라이언트 수에 따른 이동 거래 철회율

■ 이동 클라이언트의 수에 따른 메시지량

다음 그림 6의 수행결과에서 나타나는 것처럼 UTSM-ARM에서 이동 거래의 완료요청과 갱신 패턴 비율을 고려하여 요청하는 메시지의 량은 기존에 제안된 UTS-ARM의 완료요청 메시지 량과의 차이를 보이지 않음을 확인할 수 있다.



[그림 6] 이동 클라이언트 수에 따른 메시지 량

6. 결론 및 향후 연구방향

본 논문에서는 비대칭적인 대역폭의 특징을 갖는 방송환경에서 이동 갱신 거래가 가능한 경우, 각 데이터에 대한 동적인 갱신패턴 비율을 적용한 캐쉬 데이터의 일관성을 유지하고 이동거래를 스케줄링하는 UTSM-ARM의 성능을 비교 분석하였다. 제안하는 UTSM-ARM과 OCC-UTS의 성능분석을 통하여 UTSM-ARM기법이 방송환경의 제한된 대역폭 사용량에 따른 트랜잭션의 철회율을 월등하게 줄일 수 있음을 확인하였다.

앞으로의 연구 방향은 이동거래의 사이클을 탐지할 수 있는 방법과 이동 클라이언트와 서버와의 메시지 량을 최적화할 수 있는 방법에 대해 연구하고자 한다.

참고문헌

[1] S.Acharya, M.Franklin and S.Zdonik, "Balancing Push and Pull for Data Broadcasting," Proceedingsof ACM SIGMOD Conference on Management of Data, May, 1997.

[2] D.Barbara and T.Iminelinsky, "Sleepers and Workholics : Caching in Mobile Environment," Proceedings of ACM SIGMOD Conference on Management of Data ENgineering, pp.114-123, April 1997.

[3] J. Jing, A. Helal, A Elmagarmid and R. Alonso, "Bit-Sequences : An Adaptive Cache Invalidation Method in Mobile Client/Server Environment," ACM/Baltzer Mobile Networks and Applications, Vol2, No.2, 1997.

[4] K.L.Wu, P.S.Yu and M.S.Chen, "Energy-Efficient Caching for Wireless Mobile Computing," Proceedings of the 12th International Conference on Data Eng., pp.336-343, 1996.

[5] H.Y. Yun, and B.H. Hwang, "A Pessimistic Concurrency Control Algorithm in Multidatabase Systems," Preceedings of the Third International Symposium on Database Systems for Advanced Applications, Taejon, Korea, April 6-8, 1993.

[6] 이상근, 황종선, 이원규, 유헌창, "이동 클라이언트/서버 환경에서의 캐싱 및 동시성 제어," 한국정보과학회 논문지, Vol. 26, No. 8 August 1999.