이동 컴퓨팅 환경에서 방송기반의 낙관적 캐쉬 프로토콜들에 대한 분석☆

Analysis of Optimistic Cache Protocols based on broadcasting for Mobile Computing Environments

조 성 호*
Cho, Sung Ho

요 약

이동 컴퓨팅 환경에서 캐싱은 통신대역폭의 한계, 자원의 제약과 잦은 접속단절의 영향을 줄일 수 있는 중요한 기술이다. 이동 컴퓨팅의 이러한 제약과 접속단절을 완화하기 위하여 방송 기반의 새로운 캐싱 기법이 소개되었다. 이 논문에서는 새로운 방송기반 캐쉬 프로토콜을 제안하고 다중버전 방식 및 확인보고 방식과 같이 잘 알려진 기법과 비교한다. 모의실험을 통하여 충돌을 검출하고 해결하는 여러 접근방법들의 특징을 보여준다. 또한, 제안하는 기법이 다른 방식에 비하여 공간 및 계산 부하가 적다는 것을 보인다.

Abstract

Caching in mobile computing environments is an important technique that will reduce the effects such as limitation of bandwidth, restriction of resources and frequent disconnection. To reduce limitation and disconnection problems in mobile computing, a new form of caching system have been proposed based on a broadcast approach. This paper proposes a broadcast based cache protocol and examines the behaviors of the proposed scheme and well known schemes such as multiversion based scheme and Certification Reports. By a detailed simulation, we show some characteristics of different approaches to detect and resolve conflicts. We also show proposed scheme out performs other schemes with a low space and operation overhead.

☞ Keyword : Mobile Computing, Cache Protocol, Optimistic Scheme

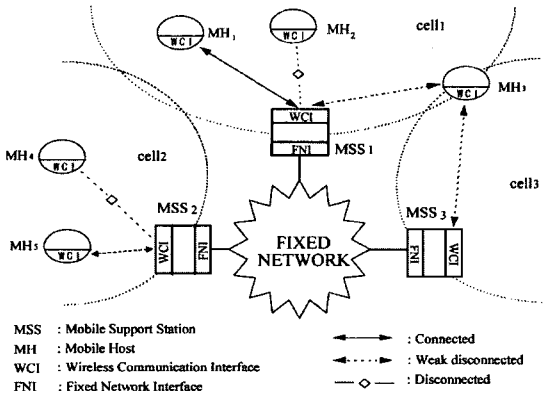
1. 서 론

휴대용 컴퓨터와 무선 통신기술의 발전은 이동 컴퓨팅 환경의 기초가 되었다[1-4]. 이동 컴퓨팅 환경은 통신대역폭의 한계와 자원의 제약[5], 빠르게 변하는 위치정보[1] 및 잦은 접속단절[6]로 특징 지워진다.

그림 1은 본 논문이 참조하는 이동 컴퓨팅 환경을 보여준다. 일반적으로 이동 컴퓨팅 환경은 고정 호스트(fixed host)들과 이동 호스트(Mobile Host)로 나눈다. 각 이동 호스트들은 무선 통신 인터페이스(WCI; Wireless communication Interface)를 통하여 고정 호스트들과 연결된다. 고정 네트워크(Fixed network)에서는 통신연결(connection)과 접속단절(disconnection)의 두 가지 상태가 존재하지만 이동 컴퓨팅 환경에서는 두 가지 상태 이외에 약한 접속단절(Weak disconnection) 상태가 존재한다. 약한 접속단절이란 이동 호스트가 네트워크에 접속하고 있지만 사용 가능한 통신대역폭이 매우 작은 상태를 지칭한다.

이동 컴퓨팅 환경에서 작은 통신대역폭의 문제를 극복하기 위해서 각 클라이언트들은 나중에 다시 사용하기 위한 일정양의 데이터를 자신

* 정 회 원 : 한신대학교 정보통신학과 교수
zoch@hs.ac.kr(제1저자)
[2005/05/20 투고 - 2005/06/20 심사 - 2005/08/22 심사완료]
☆ 본 논문은 2005년 한신대학교 학술연구비 지원에 의하여 연구되었음.



〈그림 1〉 이동컴퓨팅 환경

의 캐시(cache)에 유지하고 있다. 캐시를 사용하게 되면, 트랜잭션의 의미를 이용하여 클라이언트 캐시에 있는 데이터들의 일관성을 유지시켜주는 캐시 관리 프로토콜(management protocol)을 필요로 한다[5-10,12,13]. 일반적으로 캐쉬 관리 프로토콜은 낙관적인 방법(OCC; Optimistic Concurrency Control)과 비관적인 방법(PCC; Pessimistic Concurrency Control)으로 나뉜다. 낙관적 기법에서 트랜잭션은 데이터를 접근할 때 충돌(conflict)에 대한 검사 없이 실행되며, 검사 단계(validation phase)에서 증명(certification) 알고리즘을 이용하여 충돌이 발견되면, 해당 트랜잭션은 철회(abort) 된다[11]. 이러한 특성으로 인하여 낙관적 동시성 제어 기법은 메시지를 전송하는 비용이 높은 환경에서 비관적 기법보다 우수한 성능을 가진다[9-11]. 또한, 낙관적 기법은 일시저적인 접속단절 동안에도 공유된 데이터를 이용할 수 있는 장점을 가진다. 이러한 특징으로 인하여 비관적인 기법보다 낙관적 관리 프로토콜이 이동 컴퓨팅 환경에 적합하다.

본 논문에서는 방송(broadcast)기반의 낙관적 캐쉬 관리 기법을 제안하고 방송기반 기법으로서 잘 알려진 다중버전(MV; Multiversion) 기법[1]과 확인보고(CR; Certification Reports) 기법[5]과 비교를 한다. 모의실험을 통하여 충돌을 검출하고 해결하는 각각 다른 기법들의 특징을 살펴보고 제안하는 기법이 적은 공간과 연산을 사용함에도 좋은 성능을 가진다는 것을 보인다.

2. 이동컴퓨팅환경을 위한 방송기반 낙관적 기법

2.1 기본전략

본 논문에서는 각 이동 호스트들은 자신만의 캐쉬를 가지고 있으며 이 캐쉬는 하드디스크와 같이 휘발성이 없다고 가정한다. 또한, 이동 호스트들 사이에 실행되는 트랜잭션들에게는 제한이 없지만, 한 이동호스트에서 실행되는 트랜잭션들은 순차적으로 실행된다고 가정한다.

이동 컴퓨팅 환경에서 캐쉬의 데이터를 동기화하는 방식으로 CR방식이 처음 소개되었다[5]. 제안하는 기법은 기본적으로 CR방식을 사용한다. 즉, 한 이동호스트의 캐쉬에 데이터가 없다면 이동 호스트는 서버에게 데이터를 요청하고, 데이터의 무결성(consistency)을 유지하기 위하여 서버는 무효화 보고(invalidation reports)를 이동통신 채널을 통하여 매 L초마다 방송한다. 또한, $L * \omega$초 동안 접속단절이 일어난 이동호스트에서 실행중인 트랜잭션들은 철회(aborted)된다(ω는 윈도우 크기를 나타낸다).

제안하는 기법(RaH/w; Run and Hit on wireless)과 CR방식과의 차이점은 다음과 같다. 제안하는 방법에서는 각 데이터 D에 대하여 읽기(read) 타임스탬프(timestamp) D^{RT}와 쓰기(write) 타임스탬프 D^{WT}를 유지한다. 타임스탬프 D^{RT}와 D^{WT}는 가장 최근에 이 데이터를 읽고, 완료(commit)된 트랜잭션의 타임스탬프이다. 서버는 $ts_i = I * L$ 시각마다 주기적으로 무효화 보고를 방송한다. 무효화 보고는 (T^C, S^W)의 리스트로 구성되며 집합 S^W는 업데이트된 데이터의 집합이며 T^C는 해당데이터의 완료시각을 나타낸다. T^C는 S^W의 버전정보로 활용되며 S^W는 $[ts_i - I * L, ts_i]$시간동안에 업데이트된 데이터의 집합이다. 다음번 방송

에 사용할 데이터([ts_i, $ts_i + I * L$])를 보관하기 위하여 서버는 UL이라 불리는 리스트를 저장한다.

각 클라이언트는 자신의 트랜잭션 X를 위하여 표 1과 같은 정보를 유지한다. 일반적인 CR기법과 똑같이, 집합 S^R과 S^W는 검증을 위해서 유지된다. 새로 추가되는 타임스탬프 T^L과 T^U는 재배열을 위해서 유지되며, 집합 S^I는 필요 없는 연산을 줄이기 위해 유지된다. T^L과 T^U의 초기 값은 시스템의 가장 작은 타임스탬프 값이다.

트랜잭션 X가 완료를 요청하면 해당 클라이언트는 서버에게 $X.S^R$, $X.S^W$, $X.T^L$, $X.T^U$를 보낸다. 서버가 이 정보를 받으면 검증을 위해서 트랜잭션 X에게 유일한 타임스탬프 $X.T^C$를 부여한다. 유일한 타임스탬프를 부여한 후, 만약 $X.T^U$의 값이 초기 값이면 트랜잭션 X는 검증 없이 완료된다. 완료단계에서 서버는 $X.S^W$에 있는 데이터를 데이터베이스에 반영하고 $X.S^R$에 있는 각 데이터 D_i에 대하여 $D_i.T$를 $X.T^C$로 변경한다. 타임스탬프를 변경 한 후, 서버는 $X.T^C$와 $X.S^W$가 담긴 메시지를 방송한다. MV기법에서는 트랜잭션이 완료 후에 각 데이터에 다중버전을 유지한다. 즉, 데이터 D_i에 대하여 (D_i, $D_i.T$)값을 복수 개 유지한다. 제안하는 기법에서는 데이터 D_i에 대하여 한 개의 D_i값과 한 개의 $D_i.T$만을 유지한다. 제안하는 기법에서의 데이터 흐름을 그림 2에 도식하였다.

〈표 1〉 트랜잭션 X에 대한 정보 리스트

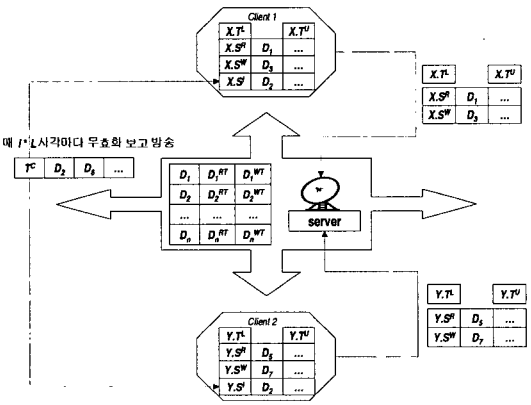
타임스탬프 $X.T^L$: 트랜잭션 X가 읽은 데이터 중 가장 큰 타임스탬프
타임스탬프 $X.T^U$: 트랜잭션 X가 실행중인 동안 완료된 트랜잭션 중 트랜잭션 X와 충돌을 일으킨 트랜잭션의 가장 작은 타임스탬프
집합 $X.S^R$: 트랜잭션 X가 읽은 데이터의 집합
집합 $X.S^W$: 트랜잭션 X가 쓴 데이터의 집합
집합 $X.S^I$: 트랜잭션 X가 실행 중인 동안 서버로부터 무효화(invalidate)된 데이터의 집합

2.2 트랜잭션 처리

제안하는 기법에서는 클라이언트가 서버로부터 무효화 보고를 받았을 때, 만약 트랜잭션 Y가 $X.S^W$에 있는 데이터에 대하여 쓰기 연산을 실행했다면 트랜잭션 Y는 필요 없는 연산을 줄이기 위해 철회된다. 또한, 트랜잭션 Y가 $X.S^W$에 있는 데이터를 읽었다면, 트랜잭션 Y는 재배열되어야만 한다.

MV기법에서는 서버에서 재배열을 위한 다중 버전 정보를 유지하지만, 제안하는 기법에서는 한 개의 버전만을 유지하기 때문에, 재배열을 위해서 각각의 클라이언트가 재배열 타임스탬프 T^R를 위한 유효 공간(valid interval)을 유지한다. 타임스탬프 T^L과 T^U가 유효 공간을 나타내는 데, T^L은 유효 공간의 하한 값을 나타내고 T^U는 상한 값을 나타낸다.

재배열 타임스탬프의 하한 값을 나타내는 타임스탬프 T^L은 다음과 같이 갱신된다. 트랜잭션 Y가 데이터 D_i를 읽을 때마다, $Y.T^L$은 $D_i.T$와 비교된다. 만약 $Y.T^L$이 $D_i.T$보다 작다면 $Y.T^L$은 $D_i.T$로 설정된다. $Y.T^L$이 증가 될 때마다, $Y.T^L$은 $Y.T^U$와 비교되어진다. 만약 $Y.T^L$이 초기 값이 아니고 $Y.T^U$보다 크거나 같다면 재배열 타임스탬프의 값을 결정 할 수 없으므로 트랜잭션 Y는 철회된다. 재배열 타임스탬프의 하한 값을 나타내는 타임스탬프 T^U는 다음과 같이 갱신된다.



〈그림 2〉 제안하는 기법의 데이터 흐름도

○ $X.S^W$에 있는 각 데이터 D_i에 대하여 만약 트랜잭션 Y가 D_i를 썼다면 필요 없는 연산을 줄이기 위해서 트랜잭션 Y를 철회시킨다.

○ $X.S^W$에 있는 각 데이터 D_j에 대하여 만약 트랜잭션 Y가 D_j를 읽었다면, 클라이언트는 $Y.T^U$를 갱신한다. $Y.T^U$는 $X.T^C$와 비교되어진다. 만약 $Y.T^U$가 초기 값이라면 $Y.T^U$는 $X.T^C$로 갱신된다. 아니라면, $Y.T^U$가 $X.T^C$보다 큰 경우에만 $Y.T^U$는 $X.T^C$로 갱신된다.

○ 타임스탬프 $Y.T^U$가 감소될 때마다, 만약 $Y.T^U$가 $Y.T^L$보다 같거나 작다면 트랜잭션 Y의 재배열 타임스탬프 $Y.T^R$의 값을 찾을 수 없기 때문에 트랜잭션 Y는 철회된다.

○ 위에 명시한 조건들에 의해 트랜잭션 Y가 철회되지 않았다면 집합 $X.S^W$와 $Y.S^R$에 속한 데이터 D_k는 필요 없는 연산을 줄이기 위하여 $Y.S^I$에 추가시킨다.

클라이언트가 무효화 보고를 처리한 이후에 트랜잭션 Y가 쓰기-쓰기 충돌이 일어난 데이터를 접근할 수 있다. 이것을 막기 위해서 집합 $Y.S^I$를 유지한다. 트랜잭션 Y가 쓰기 연산을 수행할 때마다, 해당 클라이언트는 그 데이터가 집합 $Y.S^I$에 있는지를 조사한다. 만약 쓰기 연산을 할 데이터가 집합 $Y.S^I$에 있다면 트랜잭션 Y는 철회된다.

2.3 트랜잭션의 완료

표 2에 있는 알고리즘에 의하여, 만약 $Y.T^U$가 초기 값이 아니라면 트랜잭션 Y가 다른 완료된 트랜잭션과 충돌이 있었다는 것을 나타낸다. 뿐만 아니라, $Y.T^U$가 트랜잭션 Y와 충돌을 일으키고 완료된 트랜잭션의 타임스탬프 중 가장 작은 타임스탬프를 나타내므로, 재배열 타임스탬프 값은 $Y.T^U$보다는 작아야만 한다. 그러므로 트랜잭션 Y가 완료를 요청했을 때, 서버는 재배열 타임스탬프 $Y.T^R$를 $Y.T^U$ - δ로 정한다. (δ는 극소 값이다.) 그러나, $Y.T^U$가 직접적으로 충돌이 일어난 트랜잭션들의 타임스탬프 중 가장 작은 타임스탬프를 나타내므로, 트랜잭션 Y는 간접 충돌 검사 없이 완료될 수 없다. 간접 충돌 검사를 위해서 서버는 $Y.S^W$에 들어 있는 데이터 D_i에 대하여 타임스탬프 $D_i.T$가 언제나 $Y.T^R$보다 작은가 검사한다. 만약 이 조건을 만족하지 않으면 트랜잭션 Y는 철회된다. 만약 만족한다면 트랜잭션 Y는 $Y.T^R$로서 완료된다.

완료 단계에서 서버는 $Y.S^W$에 있는 데이터를 데이터베이스에 반영하고, $Y.S^R$에 들어 있는 데이터 D_j에 대하여 타임스탬프 $D_j.T$가 $Y.T^R$보다 작을 때만 $D_j.T$는 $Y.T^R$로 설정된다. 트랜잭션 Y가 완료된 후에 서버는 $Y.T^R$과 $Y.S^W$가 담긴 메시지를 방송한다.

서버와 트랜잭션 X의 클라이언트가 서로에게 동시에 메시지를 보내는 경우에 캐쉬된 데이터의 일관성이 깨질 수 있다. 이러한 경우는 여러 가지 방법으로 처리할 수 있다. 예를 들어, 서버는 방송큐(BQ; Broadcasting Queue)와 시스템에

〈표 2〉 모의실험에 사용된 변수들과 값

parameter	meaning	setting
DB_Size	Number of data	10K, 6K, 4K
Data_Size	Size of each data item	5Kbyte
No_Tr	Number of TRs.	20-200
Tr_Size	Mean size of TRs.	15
Max_Tr	The largest size of TRs.	20
Min_Tr	The smallest size of TRs.	10
L	Broadcast period	1, 2 second
Write_Prob	Prob. of write access	0-100%
Ex_Tr	Mean time between TRs.	0.05 second
Ex_Op	Mean time between OPs.	0.1 second
Ins_Init	Inst. per initial operation	100K
In_Read	Inst. per read operation	50K
Ins_Write	Inst. per write operation	50K
Restart_Delay	Restart Delay	0.1 second
Server_CPU	Server CPU speed in MIPS	50 MIPS
Client_CPU	Client CPU speed in MIPS	5 MIPS
Net_Band	Network bandwidth	0.5 Mbps
Hist_Size	Hist. size for Multiversion	4 per data

서 가장 큰 메시지 지연 시간 보다 조금 큰 값인 임계치(threshold)를 사용 할 수 있다. 어떤 일정 시각 t에서 BQ는 [t - threshold, t]구간에 방송 된 메시지를 유지한다. 서버가 트랜잭션 X로부터 완료 요청을 받으면 BQ를 검사한다. 만약 BQ가 트랜잭션 X가 접근한 데이터를 가지고 있지 않다면 트랜잭션 X는 검사 없이 완료된다. 만약 BQ가 트랜잭션 X가 쓴 데이터를 가지고 있다면 트랜잭션 X는 철회된다. 또한, BQ가 트랜잭션 X가 읽은 데이터를 가지고 있다면, 서버는 타임스탬프 $X.T^U$값을 변화시키고 트랜잭션 X를 재배열될 수 있는 트랜잭션으로서 처리한다.

또 다른 방법은 2-단계 프로토콜(2-phase protocol)을 사용하는 것이다. 완료 요청을 보내기 전에 트랜잭션 X의 클라이언트는 예비 완료 요청 메시지를 서버에게 보낸다. 클라이언트는 예비 완료 요청 메시지에 대한 응답을 받은 후에 서버에게 완료 요청 메시지를 보낸다. 서버가 예비 완료 요청 메시지를 받으면 방송을 중단하고 해당 클라이언트로부터 완료 요청 메시지를 받을 때까지 기다린다. 서버와 클라이언트가 동시에 메시지를 보냄으로서 발생할 수 있는 문제는 위와 같은 방법으로 처리가 가능함으로, 이 논문에서는 이러한 문제에 대해서 더 이상 언급하지 않는다. 그러므로 트랜잭션 X가 완료를 요청할 때, 타임스탬프 $X.T^U$가 초기 값이면 트랜잭션 X는 검사 없이 완료된다.

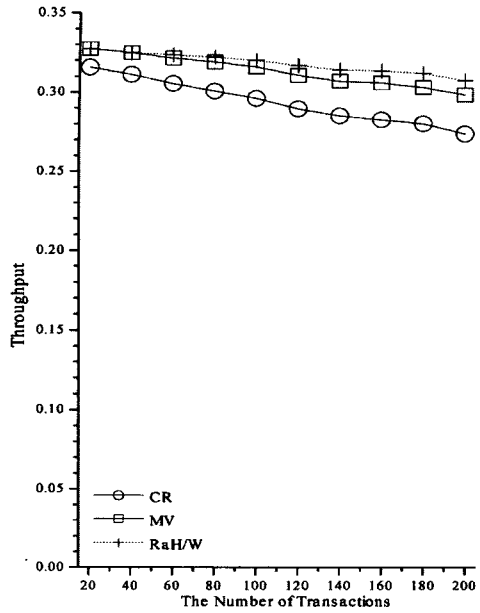
3. 모의실험 및 결과

이 장에서는 제안하는 기법(RaH/w)과 CR기법 및 MV기법을 비교한다. 표 2는 모의실험에 사용된 변수들과 값들을 보여준다. 모의실험은 큐잉(Queueing) 모델을 사용하였으며 CPU의 속도는 MIPS로 나타내었다. 본 모의실험에 사용된 값들은 [4]에 있는 값을 기초로 작성되었다. 본 모의실험에서 트랜잭션 크기(*Tr_size*)는 *Min_TR*과 *Max_Tr*사이의 일정한 균일한 분포(uniform distribution)를 가지도록 하였으며 한 트랜잭션이 접근하는 데이터 중 쓰기연산에 대한 비율은 *Write_prob*에 의해 결정된다. 한 트랜잭션이 철회되면 일정한 시간(*Restart_delay*) 동안 기다린 후 다시 시작하도록 했으며 철회되기 이전에 접근했던 데이터를 다시 접근하도록 하였다. 결국, 각 기법의 성능은 제출된 모든 트랜잭션이 완료되는 시간으로 측정 하였다.

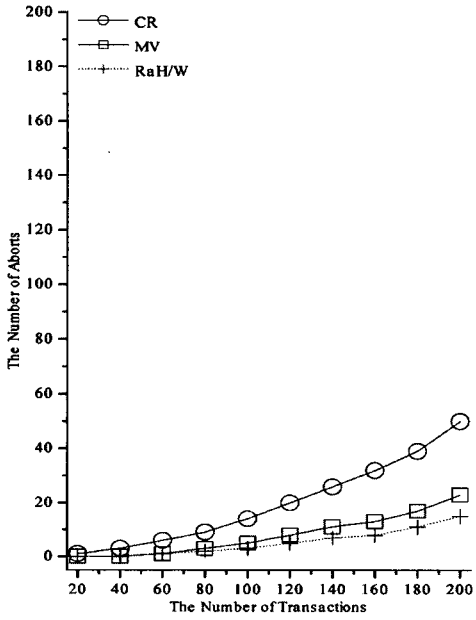
*Ex_Tr*은 트랜잭션들이 생성되는 시간의 분포를 나타내며 *Ex_Op*는 한 트랜잭션 내의 각 연산들이 생성되는 시간을 나타낸다. 두 변수 다 지수분포(exponential distribution)를 따르도록 하였다. 방송 간격을 나타내는 L은 본 실험에서 1초와 2초 두 가지 값을 사용하였다. 또한, MV 방식에 있어서 각 데이터의 멀티버전의 개수는 최대 4개까지(*Hist_Size*) 허용하도록 하였다. 이는 다른 두 가지 기법에 대하여 MV 기법의 저장 공간이 4배 크다는 것을 의미한다. 이 값이 커지면 MV의 성능이 더 좋아지겠지만, 데이터에 대한 추가 공간도 증가하게 된다.

처음의 실험은 적은부하(*DB_Size* = 10,000) 상태에서 3가지의 방식을 비교하였다. 다른 실험에 비하여 데이터베이스의 크기가 크고 트랜잭션의 개수가 상대적으로 적기 때문에 충돌은 적게 일어나며 결론적으로 철회되는 트랜잭션의 수는 매우 적다.

그림 3은 *Write_Prob* = 20%와 L = 1초의 상태의 철회수를 보여준다. 주어진 상태에서, 모든 기법들은 트랜잭션의 수가 증가할수록 철회수가 완만하게 증가하는 모양을 보여준다. 모든 영역에서 CR기법이 다른 두 기법보다 더 많은 철회수를 보여주는데 이는 CR기법이 충돌이 난 트랜잭션들을 재배열 시키지 못하고 철회시키기 때문이다. MV기법과 RaH/w기법 또한 철회되는 트랜잭션 수의 차이를 보이는 데, 이는 재배열되는 트랜잭션의 차이 때문이다. 만약, MV의 History Size를 증가시키면 이 차이가 줄어들 수도 있겠지만, 현재 상태에서도 MV기법은 CR기법보다 4



〈그림 3〉 처리량



〈그림 4〉 철회 수

배, 제안하는 기법보다 2배의 저장 공간을 더 사용하고 있다. MV와 유사 기법인 TSH기법[10] 또한 각 데이터 마다 2개 이상의 타임스탬프를 유지해야하는 부담이 있다. 그림 3이 보여주듯이 RaH/w는 MV보다 적은 공간을 사용하면서도 다른 기법들보다 적은 철회수를 보인다.

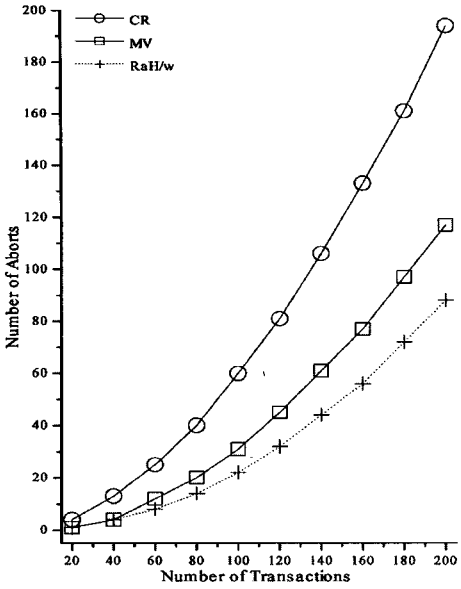
그림 4는 세 가지 기법의 처리량을 보여준다. 일반적으로 철회율과 처리량사이에는 밀접한 관계가 있는 것으로 알려져 있다. 그림 4가 보여주듯이 철회수가 높은 CR기법은 낮은 처리량을 보여주며 철회수가 가장 낮은 제안하는 기법은 가장 높은 처리량을 보여준다.

다음의 실험은 높은 부하(*DB_Size* = 2000) 상태에서 3가지의 방식을 비교하였다. 처음 실험에 비하여 데이터베이스의 크기가 1/5로 줄었기 때문에 충돌이 많이 발생하며 결론적으로 철회되는 트랜잭션의 수는 급격히 증가한다.

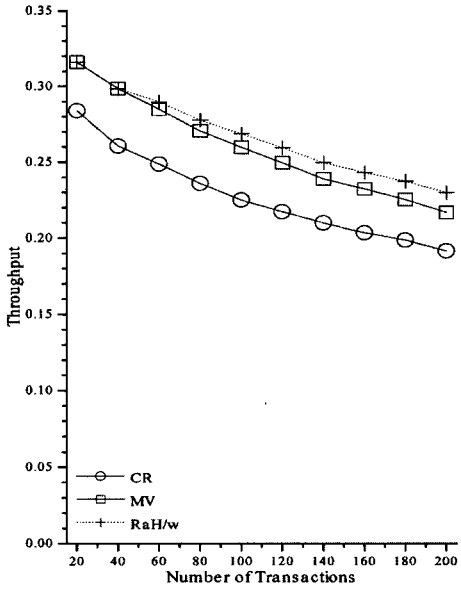
그림 5는 *Write_Prob* = 20%와 *L* = 1초의 상태의 철회수를 보여준다. 그림 3과 비교해보면, 데이터베이스의 크기가 작기 때문에, 트랜잭션의 수가 증가할수록 철회되는 트랜잭션의 수가 급격히 증가한다. 이러한 특징은 세 가지 방식에 동일하게 나타나지만 CR기법이 상대적으로 매우 많은 철회 수를 나타낸다. 제안하는 기법은 재배열되는 트랜잭션의 수가 가장 많기 때문에 여전히 가장 작은 철회수를 보인다. 이러한 특징으로 인하여 그림 6이 보여주는 것처럼 제안하는 기법은 가장 높은 처리율을, CR기법은 가장 낮은 처리율을 보여준다.

세 번째 실험은 중간 부하 상태(*DB_Size* = 6,000)에서 방송주기가 성능에 어떤 영향을 미치는지에 대하여 실험하였다. 그림 7에서처럼 모든 기법들이 방송주기가 길어짐에 따라 성능이 낮아지는 특징을 보여준다. 그러나 RaH/w는 주기와 상관없이 일정한 성능을 보여주는데, 제한하는 기법만이 쓰기-쓰기 충돌이 일어난 트랜잭션들을 그 들이 실행되는 중간에 철회시킴으로서 필요 없는 연산을 줄일 수 있는 특징을 가지고 있기 때문이다. 그러나 그러한 특징이 성능에 미치는 영향은 매우 제한적이다.

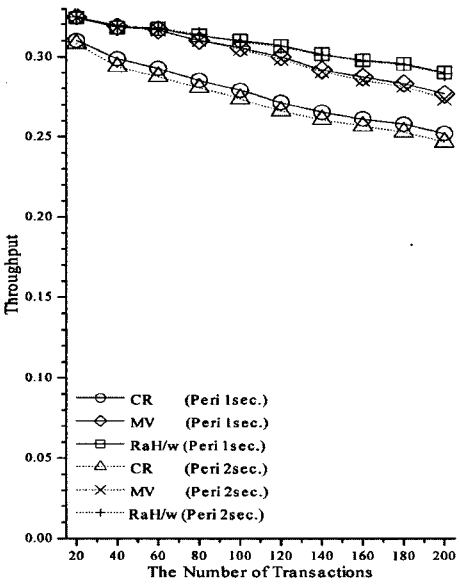
마지막 실험은 중간 부하 상태에서 쓰기확률(*Write_Prob*)이 성능에 어떤 영향을 미치는지에



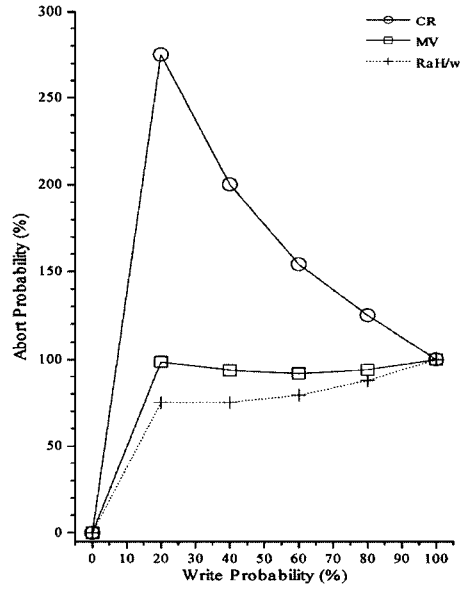
〈그림 5〉 철회 수



〈그림 6〉 처리량



〈그림 7〉 처리량



〈그림 8〉 철회률

대하여 실험 하였다. 쓰기 확률이 증가하게 되면 쓰기-쓰기 충돌이 많아지게 되고 이는 철회율에 영향을 미친다. 쓰기 확률이 0%이면 어떤 트랜잭션도 철회되지 않고 완료가 되며, 쓰기 확률이 100%가 되면 모든 읽기 연산이 없어지기 때문에 모든 트랜잭션이 철회 된 후 순차적으로 실행되게 된다.

그림 8은 트랜잭션의 수(*No_Tr*)를 100으로 고정 시킨 후 실험한 결과이다. CR기법의 경우 철회될 트랜잭션을 재배열 하는 기능이 없기 때

문에 쓰기 확률이 증가함에 따라 철회율이 급격히 증가하는 것을 알 수 있다. MV기법과 RaH/w 기법의 경우에는 철회율의 증가가 완만하지만 여전히 RaH/w기법의 철회율이 MV기법보다 완만하게 증가하는 것을 알 수 있다.

4. 결 론

본 논문에서 이동컴퓨팅환경에서 무효화보고를 이용한 효율적인 재배열 트랜잭션 관리 규약을 제안하였다. 낙관적인 기법의 철회율은 재배열 기법을 사용함으로서 줄일 수 있지만 MV기법과 같은 재배열 기법은 많은 양의 추가 저장 공간을 필요로 하며 저장 공간의 크기에 따라 성능이 달라지는 단점을 가지고 있다. 제안하는 기법은 일반적인 방식과 비교하여 각 데이터 당 한 개의 추가적인 데이터 공간을 요구하면서도 기존의 재배열 기법보다 많은 양의 트랜잭션을 재배열 시키는 특징을 가지고 있다. 또한, 클라이언트에서 유지되는 추가 데이터의 크기는 매우 작으며 이러한 적은 공간적 부하를 가지면서도 쓰기-쓰기 충돌을 미리 막음으로서 성능을 향상 시키는 기능도 가진다. 본 논문에서는 다양한 모의실험을 통하여 제안하는 기법이 일반적인 방식과 재배열 기반 기법과의 차이를 보여주었다. 제안하는 기법은 적은 공간적 부하를 가지면서도 일반적인 기법이나 재배열 기법보다 좋은 성능을 가진다는 것을 보였다.

참 고 문 헌

[1] E. Pitoura and P. K. Chrysanthis, "Multiversion Data Broadcast," *IEEE Transacions on Computers*, Vol.51, No.10, pp 1224-1230, 2002.

[2] Daniel Barbara, "Mobile Computing and Database - a Survey," *IEEE Transactions on Knowledge and Data Engineering*, Vol.11, No.1, pp.108-117, 1999.

[3] E. Pitoura and B. Bhagava, "Revising Transaction Concepts for Mobile Computing," *Proceeding of the IEEE Workshop on Mobile Systems and Applications*, Purdue University, Dept. of Comp. Dec, 1994.

[4] E. Pitoura and B. Bhagava, "Maintaining Consistency of Data in Mobile Distributed Environments," *Proceeding of the 15th International Conference on Distributing Computing System*, Purdue University, Dept. of Comp. 1995.

[5] T. Imielinski and B. R. Badrinath, "Wireless Mobile Computing: Challenges in Data Management", *Communications of the ACM*, pp.18-28, 1994.

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

[7] 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 Engineering*, pp.336-343, Feb. 1996.

[8] C. F. Fong, C. S. Lui and M. H. Wong, "Quantifying Complexity and Performance Gains of Distributed Caching in a Wireless Network Environment", *Proceedings of the 13th International Conference on Data Engineering*, pp.104-113, April 1997.

[9] P. S. Yu and D. M. Dias, "Analysis of Hybrid Concurrency Control Schemes for a High Data Contention Environment," *IEEE Trans. Software Eng.*, 18(2), PP. 118-129, 1992.

[10] P. S. Yu, H. Heiss, and D. M. Dias, "Modeling and Analysis of a Time-Stamp History Based Certification Protocol for Concurrency Control", *IEEE Trans. Know. and Data Eng.*, vol.3, no.4, pp.525-537, 1991.

[11] S. H. Cho, "Transaction Management Protocols using Back-shifting for Mobile Computing," *ISPC 2004*, pp.24-29, 2004.

[12] Jiaxin J. Gao, Dallan Quass, Yiu-Kai Ng, "Selective-Splitting and Cache-Maintenance Algorithms for Associative-Client Caches," *Distributed and Parallel Databases*, pp. 1045-1053, 2004.

[13] Zhifeng Chen, Yan Zhang, Yuanyuan Zhou, Heidi Scott, Berni Schiefer, "Caching & file systems: Empirical evaluation of multi-level buffer cache collaboration for storage systems," *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 205-215, 2005

◐ 저 자 소 개 ◑

조 성 호 (Cho, Sung Ho)
1994년 한국외국어 대학교 컴퓨터학과 졸업(이학사)
1997년 고려대학교 대학원 컴퓨터학과 졸업(석사)
2000년 고려대학교 대학원 컴퓨터학과 졸업(박사)
2002년 ~ 현재 한신대학 정보통신학과 교수
관심분야 : 분산시스템, E-러닝, etc.
E-mail : zoch@hs.ac.kr