이동 데이타베이스 시스템에서 효율적인 캐쉬 일관성 유지 기법

임 상 민[†] · 강 현 철[††]

요 약

이동 통신 기술의 급속한 발전으로, 이동 컴퓨팅 환경에서 데이타 서비스에 대한 수요가 증가하고 있다. 이동 클라이언트 내에 캐쉬가 존재하면, 대역폭의 절약 및 질의에 대한 빠른 응답을 제공할 수 있지만, 캐쉬 일관성을 유지해야 하는 부담이 생긴다. 한 셀 내에 존재하는 이동 클라이언트들의 캐쉬 일관성 유지를 위해서 서버가 캐쉬 무효화 보고를 일정 시간마다 주기적으로 방송하는 방법은 효율적일 수 있다. 그런데, 이동 클라이언트가 오랜 시간 동안의 접속 단절로 인해 무효화 보고만으로 자신의 캐쉬 유효성 여부를 판단하지 못할 경우에는, 서버에게 캐쉬 유효성 여부에 대한 확인을 요청함으로써 캐쉬 일관성을 유지할 수 있다. 이때, 할당 가능한 채널의 수와 이동 클라이언트 수의 관계에 따라서 서로 다른 기법이 각각의 경우에 더 효율적일 수 있다. 본 논문에서는 (1) 할당 가능한 채널의 수가 이동 클라이언트의 수보다 많거나 비슷한 경우와 (2) 채널의 수가 이동 클라이언트의 수보다 훨씬 적을 경우 각각에 대하여 효율적인 새로운 캐쉬 일관성 유지 기법을 제안하고 성능을 평가한다.

Efficient Schemes for Cache Consistency Maintenance in a Mobile Database System

Sang-Min Lim[†] · Hyun-Chul Kang[††]

ABSTRACT

Due to rapid advance of wireless communication technology, demand on data services in mobile computing environment is gradually increasing. Caching at a mobile client could reduce bandwidth consumption and query response time, and yet a mobile client must maintain cache consistency. It could be efficient for the server to broadcast a periodic cache invalidation report for cache consistency in a cell. In case that long period of disconnection prevents a mobile client from checking validity of its cache based solely on the invalidation report received, the mobile client could request the server to check cache validity. In doing so, some schemes may be more efficient than others depending on the number of available channels and the mobile clients involved. In this paper, we propose new cache consistency schemes, efficient especially (1) when channel capacity is enough to deal with the mobile clients involved or (2) when that is not the case, and evaluate their performance.

키워드 : 이동 컴퓨팅(Mobile Computing), 데이타 캐쉬(Data Cache), 캐쉬 일관성 유지(Cache Consistency Maintenance).

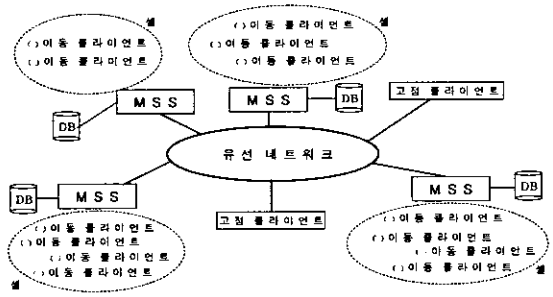
1. 서 론

이동 통신 기술의 발달로 이제 사용자는 유선 네트워크 상의 고정된 위치가 아닌 무선 네트워크 환경 상의 임의의 위치에서 데이타 서비스를 제공받을 수 있게 되었다 [3, 4]. 최근 PCS를 이용한 교통, 증권, 뉴스, 기상, 스포츠, 연예, 오락 등의 데이타 서비스 실시는, 앞으로 계속 확대 제공될 이동 컴퓨팅 환경에서의 데이타 서비스의 예이다.

이동 컴퓨팅 환경은, 기존의 유선 네트워크에 연결된 서버로서 무선 통신 인터페이스를 갖추고 있는 MSS(Mobile Support Station)와 MSS가 관장하는 영역으로 무선 LAN 또는 셀룰라 통신 네트워크 등인 셀(cell) 내의 수많은 이동 클라이언트로 구성된다(그림 1). 이동 컴퓨팅 환경에서는 유선 네트워크 환경에 비하여 서버와 이동 클라이언트들 간의 네트워크 연결 상태가 불안정하며 대역폭이 몹시 낮다. 또한 이동 클라이언트의 배터리 수명에도 제약이 있다. 네트워크 연결 상태의 불안정은 비의도적인 접속 단절(갑작스런 통신 두절로 인한 예측 불가능한 접속 단절)을 일으킬 수 있다. 이에 비해 배터리 수명의 제약은 의도적인 접속 단절(배터리 절약을 위해서 이동 클라이언트의 전원을 끔으로 인한 예측 가능한 접속 단절)을 일으키게 된다. 특

※ 이 논문은 2000학년도 중앙대학교 학술연구비 지원에 의한 것임.
† 준 회 원 : 포스트미디어 연구원
†† 정 회 원 : 중앙대학교 컴퓨터공학과 교수
논문접수 : 2000년 10월 4일, 심사완료 : 2001년 5월 2일

히, 하루의 일과가 끝나면 이동 클라이언트의 전원을 끄고 다음날 아침까지 배터리를 충전하는 것이 보통이다. 대역폭이 유선 환경보다 낮다는 점은 멀티미디어 데이타 서비스와 같은 대역폭 소모가 큰 응용을 지원하는 데 단점으로 작용한다. 이러한 접속 단절과 낮은 대역폭 문제를 해결하기 위한 대표적인 기법으로 방송(broadcast)과 캐싱(caching)이 있다[1, 2, 5-7, 9, 10, 12-14].



(그림 1) 이동 컴퓨팅 환경

불특정 다수에게 데이타 서비스를 제공하는 이동 컴퓨팅 환경의 특성 상, 서버로부터의 방송은 수많은 이동 클라이언트들의 공통된 요구를 충족시키는 데 효율적인 해결책으로 사용될 수 있으며, 자주 접근하는 데이타를 서버로부터 이동 클라이언트의 기억장치로 캐쉬하는 기법은 무선 네트워크를 통한 데이타 검색의 양 및 횟수를 줄일 수 있는 해결책으로 사용될 수 있다.

이 중, 이동 클라이언트 내 캐쉬의 이용은 사용자의 질의 응답 시간의 단축, 대역폭의 절약, 그로 인한 배터리의 절약 등을 가져올 수 있으나, 캐쉬의 일관성을 유지해야 하는 부담이 생기므로 캐쉬 일관성 유지를 위한 효율적인 기법을 필요로 한다. 본 논문은 이 캐쉬 일관성 유지에 관한 것이다.

교통, 증권, 뉴스, 기상, 스포츠, 연예, 오락 등의 대표적인 이동 데이타 서비스 응용에서 데이타 갱신은 서버 측에서 발생한다. 따라서 이동 클라이언트의 캐쉬 일관성을 유지하기 위해서는, 데이타의 갱신 사실 또는 그 내용이 해당 데이타를 캐쉬한 모든 이동 클라이언트로 통지되어야 한다. 이를 위하여 서버는 자신이 관장하는 셀 내에 존재하는 모든 이동 클라이언트들에 대하여, 그들이 캐쉬한 데이타 항목, 캐쉬 시점, 캐쉬의 상태 정보, 그리고 접속 상태 정보 등을 유지해야 하는데, 이는 셀 내에 수많은 이동 클라이언트들이 존재한다는 점을 감안할 때 서버에 부담을 가중시키는 비효율적인 방법이다. 따라서, 서버는 이동 클라이언트들의 캐쉬에 대한 제반 정보를 일체 관리하지 않고, 갱신된 데이타 항목의 식별자들을 주기적으로 모든 이동 클라이언트들에게 방송하는 기법이 제안되었다[5]. 각 이동 클라이언트는 수신된 데이타 식별자들을 보고 자신의 캐쉬에 존재하는 데이타 중 갱신된 것은 캐쉬에서 삭제한다. 이와 같이 갱신되어 더이상 유효하지 않은 데이타를 캐쉬에서 삭제하는 캐쉬 일관성 유지 기법을 캐쉬 무효화(cache invalidation)라 하며[8], 서버가 방송하는 갱신된 데이타 식별자 정보를 무효화 보고(invalidation report, 이하 IR)라 부른다[5].

그런데, 위에서 기술한 것처럼 이동 클라이언트들은 항상 의도적 또는 비의도적 접속 단절 상태에 있을 수 있으므로, 접속 단절 상태 동안에 방송된 IR은 수신할 수 없고 그에 따라 자신의 캐쉬 일관성을 유지할 수 없다. 즉, 이동 클라이언트는 접속 단절 상태에서 활동 상태로 깨어났을 때 자신의 캐쉬의 유효성을 보장할 수 없는 문제가 생긴다.

서버와 이동 클라이언트는 타임스탬프를 이용하여 서버에서의 데이타 갱신 시점과 이동 클라이언트에서의 캐쉬 일관성 유지 시점을 동기화하므로, 이 타임스탬프 정보를 참조하여 이동 클라이언트는 활동 상태로 깨어난 후 처음 수신한 IR로써 자신의 캐쉬의 유효성을 보장할 수 있는지 없는지의 여부를 판별할 수 있다[5, 14]. 따라서, 짧은 시간에 걸쳐 발생한 비의도적인 접속 단절 혹은 기간이 짧은 의도적인 접속 단절인 경우에는 활동 상태로 깨어난 후 처음 수신한 IR의 내용으로 캐쉬 일관성을 유지할 수 있는 것이 보통이다. 하지만, 의도적인 접속 단절과 같이 접속 단절 기간이 길었던 경우에는 활동 상태로 깨어난 후 수신한 IR만으로 캐쉬 유효성을 보장할 수 없는 것이 보통이다. 이 경우, 캐쉬된 데이타 항목 중 어느 것이 유효하고 어느 것이 갱신된 것인지를 판별할 수 없으므로 캐쉬된 데이타 전체를 무효화해야 한다. 이는 질의 응답시간의 단축, 대역폭의 절약, 배터리의 절약이라는 캐싱의 장점을 모두 상실하는 것이다. 특히, 각 이동 클라이언트의 사용자는 자신이 자주 접근하는 데이타를 캐쉬해 두고 있었을 것이며, 응용에 따라 그들 데이타 대부분이 갱신되지 않아 유효한 상태였을 수도 있다. [14]에서는 이와 같은 상황에서 캐쉬 전체를 무효화하는 대신 각 이동 클라이언트가 서버에게 상향(uplink) 채널을 통해 캐쉬 유효성 확인을 요청하여 캐쉬 일관성을 유지하는 기법이 제안되었다. 서버가 방송하는 IR을 통해 갱신된 데이타를 캐쉬에서 삭제함으로써 캐쉬 일관성을 유지하는 것에 비해, 이 기법은 캐쉬된 데이타 전체의 삭제를 피하고 갱신되지 않아 아직 유효한 데이타를 캐쉬에 남기는 점을 강조하여, 본 논문에서는 캐쉬 유효화(cache validation)라고 부른다. 캐쉬 유효화에는 트레이드오프가 존재한다. 캐쉬 전체를 무효화하는 것을 피함으로써 질의 응답시간의 단축, 대역폭의 절약, 배터리의 절약 등의 장점을 확보할 수 있지만, 캐쉬 유효성 확인 과정에서 이동 클라이언트는 서버와 연결하여 데이타를 송수신해야 하는데, 서버가 관장하는 셀 내에 수많은 이동 클라이언트가 존재한다는 점을 감안할 때 대역폭 소모가 크다는 단점이 따른다.

본 논문에서는 의도적인 접속 단절 등으로 인하여 접속 단절 기간이 오랫동안 계속되어 IR만으로 캐쉬 일관성을

유지하지 못하는 경우, 대역폭을 효율적으로 절약할 수 있는 캐쉬 유효화 기법을 제안하고, 기존의 캐쉬 유효화 기법과 성능을 비교 평가한다.

본 논문의 구성은 다음과 같다. 2절에서는 관련 연구를 기술하고, 3절에서는 대역폭을 절약하는 캐쉬 유효화 기법을 제안한다. 4절에서는 성능 평가를 위한 시뮬레이션 모델과 파라미터를 설명하며, 5절에서 성능 평가 결과를 기술하고, 6절에서 결론을 맺는다.

2. 관련 연구

이동 데이타베이스 시스템에서 이동 클라이언트의 캐쉬 일관성을 유지하는 기법은 크게, (1) 서버가 주기적으로 IR을 방송하고 이동 클라이언트는 이를 수신하여 캐쉬 일관성을 유지하는 기법[5, 10]과 (2) 접속 단절 상태에 있다가 활동 상태로 깨어난 각 이동 클라이언트가 자신의 캐쉬에 있는 데이타의 유효성 여부를 서버에게 확인 요청하여 그 응답을 보고 자신의 캐쉬 일관성을 유지하는 캐쉬 유효화 기법[14]으로 나눌 수 있다.

2.1 IR의 방송을 통한 캐쉬 일관성 유지 기법

주기적인 IR의 방송 시 IR의 내용 구성으로는 다음 두가지 방법이 제안되었다.

2.1.1 리스트 기반의 IR

리스트 기반의 IR은 최근에 갱신된 데이타의 식별자와 갱신된 시점의 타임스탬프 쌍의 리스트로 구성된다[5]. 이 때, '최근' 이란, 서버에서 시스템 파라미터로 설정된 시간 동안을 의미하며, 보통 w(≥1)개의 IR 방송 구간으로 정한다. 이동 클라이언트는 갱신된 데이타의 식별자와 타임스탬프로 자신의 캐쉬 내 데이타들의 타임스탬프와 비교하여 갱신된 데이타를 캐쉬로부터 삭제함으로써, 캐쉬 일관성을 유지한다.

2.1.2 비트열(bit sequences) 기반의 IR

비트열 기반의 IR은 데이타베이스 내 데이타 항목 수 만큼의 비트로 구성된 비트열과 타임스탬프의 집합으로 구성된다[10]. 각 비트는 특정 데이타의 갱신 여부를 알려 주게 되므로, 이동 클라이언트는 이들 비트 정보를 이용하여 갱신된 데이타를 캐쉬로부터 삭제함으로써 캐쉬 일관성을 유지한다.

이들 기법들은 접속 단절의 대부분이 짧은 시간 동안 일어나는 것일 경우에는 효율적인 기법일 수 있으나, 접속 단절 상태가 오랜 기간 계속된 경우, 곧 IR만으로 캐쉬 일관성 유지가 불가능한 경우에는 각 이동 클라이언트들이 캐쉬 전체를 무효화하기 때문에 좋지 못한 성능을 나타내게 된다.

2.2 서버로의 요청을 통한 캐쉬 유효화 기법

오랜 시간 동안 단절된 이동 클라이언트, 곧 w개의 IR 방송 구간 보다 오랜 시간 동안 단절된 이동 클라이언트는 활동 상태로 깨어난 이후부터 수신하는 IR로써 자신의 캐쉬 유효성을 보장하지 못한다. 이 경우, 캐쉬 내의 데이타를 단순히 모두 무효화시키는 것보다는 서버로 캐쉬 유효성 확인을 요청하여 캐쉬를 유효화하는 것이 더 효율적일 수 있다. 이와 같은 방법으로 SCC(Simple Checking Caching) 기법과 SGC(Simple Grouping Caching) 기법 등이 있다[14].

SCC 기법에서는 캐쉬 내 데이타들의 식별자와 이들의 유효성을 가장 최근에 확인했던 타임스탬프를 서버에게 보내어 캐쉬 유효성 확인을 요청한다. 서버는 수신된 데이타 식별자 중 갱신된 것들이 어느 것인지 알려주며 이동 클라이언트는 해당 데이타를 캐쉬에서 삭제한다. 이 기법은 캐쉬 내 데이타 각각의 식별자 모두를 전송하기 때문에, 캐쉬 일관성 유지에 소모되는 대역폭량이 크다는 단점이 있다.

SGC 기법에서는 데이타 각각의 식별자 모두를 서버에게 보내지 않고, 데이타를 미리 정해진 규칙에 의해 그룹핑한 다음 각 그룹의 식별자를 서버에 보내어, 그룹별로 유효성 확인을 요청한다. 따라서, 캐쉬 일관성 유지에 소모되는 대역폭의 양은 줄일 수 있지만, 갱신되지 않은 데이타도 무효화 대상 그룹에 포함된다는 이유만으로 캐쉬에서 삭제되는 잘못된 무효화(false invalidation)가 발생하는 단점이 있다. 이는 향후 해당 데이타를 검색할 때 서버로 다시 요청해야 하는 결과를 초래하므로 결국은 데이타 검색에 소모되는 대역폭의 양을 늘리게 된다.

3. 대역폭을 절약하는 캐쉬 유효화 기법

전절에서 살펴본 것처럼 서버로의 요청을 통한 캐쉬 유효화 기법은 대역폭 소모가 크다는 단점이 있다. 이 문제는 이동 클라이언트로 캐쉬되어 있는 데이타의 양이 많을수록 심화된다. 본 절에서는 캐쉬 유효화 과정과 그 이후 캐쉬를 이용한 질의 처리 과정에 걸쳐 대역폭을 효율적으로 절약할 수 있는 캐쉬 유효화 기법을 제안한다. 이동 클라이언트들이 서버에게 캐쉬 유효성 확인을 요청할 경우, 할당 가능한 채널의 수 면에서 다음의 두가지 경우가 발생할 수 있다.

① 서버에게 요청하는 이동 클라이언트의 수가 할당 가능한 채널의 수 보다 적거나 채널 경쟁이 심하지 않은 경우

② 서버에게 요청하는 이동 클라이언트의 수가 할당 가능한 채널의 수 보다 훨씬 많아 채널 경쟁이 극심한 경우

전자의 경우에는, 채널이 충분히 존재하거나 경쟁이 심하지 않으므로, 제한된 대역폭으로 인한 캐쉬 유효화 과정의 지연은 심하게 발생하지는 않는다. 그렇더라도 [14]의 SCC

기법을 쓸 경우 대역폭의 소모가 너무 크므로 SGC 기반의 기법을 사용해야 한다. 그런데 SGC는 잘못된 무효화를 초래하므로 향후 질의 처리시 대역폭의 소모를 가져오게 된다. 따라서, SGC 기법을 바탕으로 하되 잘못된 무효화를 방지하여 대역폭을 절약하는 캐쉬 유효화 기법이 필요하다.

한편 후자의 경우는, 예를 들어, 아침 시간 하루 업무의 시작 시점이나 점심 시간 이후 오후 업무의 시작 시점 등에 발생하는데, 그 이전의 의도적인 접속 단절로 인하여, 캐쉬 유효성 확인을 요청하는 이동 클라이언트의 수가 동일한 시간대에 아주 많아지게 된다. 따라서 극심한 채널 경쟁과 대역폭의 제한으로 캐쉬 유효화 과정의 지연이 심해지며, 서버가 각 이동 클라이언트에게 개별적으로 캐쉬 유효화를 위한 응답을 전송하기 위한 대역폭 소모가 심하다. 이와 같은 상황에서는 채널 수가 절대 부족하므로 서버가 방송 채널을 통하여 셀 내의 모든 이동 클라이언트를 대상으로 캐쉬 유효화에 필요한 정보를 방송하는 것이 효율적일 수 있다. 즉, 방송 기법이 적절히 결합된 캐쉬 유효화 기법이 필요하다.

본 절의 구성은 다음과 같다. 3.1절에서는 이동 데이타 서비스 시스템 환경을 설명하고, 3.2절에서는 제안하는 캐쉬 유효화 기법들이 사용하는 메시지의 종류와 자료구조를 설명한다. 이들을 바탕으로 3.3절에서는 잘못된 무효화를 방지하는 두가지 캐쉬 유효화 기법 2PCV(2 Phase Cache Validation)와 1PCV(1 Phase Cache Validation)를 각각 제안하며, 3.4절에서는 방송을 결합한 캐쉬 유효화 기법 BCV(Broadcast Cache Validation)를 제안한다.

3.1 시스템 환경

본 논문에서 가정하는 시스템 환경은 (그림 1)과 같다. 이동 클라이언트는 서버(MSS)와 무선연결을 통하여, 기존 유선 네트워크에 연결할 수 있다. 서버는 전체 데이타베이스(DB)의 내용을 복제하여 저장하고 있으며, DB 내 각 데이타들의 갱신 시점을 유지하고 있다. 이동 클라이언트 내의 캐쉬 일관성 유지를 위하여, 서버는 자신이 관장하는 영역, 곧 셀 내의 이동 클라이언트들을 대상으로 IR을 주기적으로 방송하게 되며, DB의 갱신은 서버측에서만 일어난다. 또한, 이동 클라이언트는 판독 연산만(read-only)을 수행한다고 가정한다. 이동 클라이언트에서 발생된 질의로 인하여 요청되는 데이타가 캐쉬 내에 유효한 상태로 남아 있지 않을 경우, 이동 클라이언트는 서버측으로 해당 데이타를 요청하게 되고 이 데이타는 이동 클라이언트로 캐쉬된다. 만일 이동 클라이언트가 오랜 단절로 인하여, IR만으로 캐쉬의 유효성 여부 확인이 불가능할 경우, 이동 클라이언트는 서버측으로 캐쉬 유효성 여부에 대한 확인 요청을 한다.

〈표 1〉 전송되는 메시지의 종류와 자료구조

종 류		자 료 구 조
무효화 보고(IR)		< Flag, {TS1, data_ID, data_ID, …}, … {TSw, data_ID, data_ID, …} >
무효화 보고 요청 (Req_IR)	Req_IR_d	< Flag, Last_TS,{data_ID, data_ID, …} >
	Req_IR_g	< Flag, Last_TS,{group_ID, …} >
무효화 보고 응답 (Res_IR)	Res_IR_d	< Flag, Last_TS,{data_ID, data_ID, …} >
	Res_IR_g	< Flag, Last_TS,{group_ID, …} >
유효화 보고 요청 (Req_VR)	Req_VR_d	< Flag, Last_TS, {data_ID, data_ID, …} >
	Req_VR_g	< Flag, Last_TS, {group_ID, group_ID, …} >
유효화 보고 응답 (Res_VR)	Res_VR_d	< Flag, Client_ID, Last_TS, {data_ID, data_ID, …} >
	Res_VR_g	<Flag, Client_ID, Last_TS, {group_ID, group_ID, …} >
데이타 요청 (Req_D)		< Flag, data_ID >
데이타(D)		< Flag, data >

3.2 메시지의 종류와 자료구조

캐쉬 유효화를 위하여 이동 클라이언트와 서버 간에 전송되는 메시지의 종류와 자료구조는 <표 1>과 같다. 이들 메시지에서 Flag, data-ID, group_ID는 각각 메시지의 종류, 데이타 식별자, 그룹 식별자를 나타낸다.

'데이타 요청(Req_D)'은 이동 클라이언트가 데이타를 요청하기 위해 서버에게 보내는 메시지이며, '데이타(D)'는 그에 대한 서버의 응답으로 이동 클라이언트에게 요청된 데이타를 전송하기 위한 것이다.

'무효화 보고(IR)'는 이동 클라이언트의 캐쉬 일관성 유지를 위하여 서버가 주기적으로 방송하는 메시지이다. {TS_i, data_ID, data_ID, …}에서 TS_i (i = 1, 2, …, w, 단 w≥1)는 해당 식별자의 데이타들이 현재 시점에서 i번째 이전 IR 방송 구간에 갱신되었음을 알려주는 타임스탬프 값을 나타낸다[5].

'무효화 보고 요청(Req_IR)'과 '유효화 보고 요청(Req_VR)'은 둘 다 이동 클라이언트가 자신의 캐쉬 유효성 확인을 요청하기 위해 서버에게 보내는 메시지이다. 이때, 유효성 여부를 확인하고자 하는 대상 데이타는 데이타 식별자 리스트로 전달할 수도 있고, 데이타들 간의 그룹핑을 통한 그룹 식별자 리스트로 전달할 수도 있다. 전자를 Req_IR_d (Req_VR_d)로 나타내고, 후자를 Req_IR_g (Req_VR_g)로 나타낸다. Req_IR과 Req_VR은 그 목적과 내용면에서 서로 동일한 메시지이지만 각자에 대한 서버의 응답 메시지가 서로 다르다 (아래 참조). Last_TS는 이동 클라이언트가 IR을 수신한 마지막 시점의 타임스탬프 값을 나타내는데, 서버는 이 값으로 해당 이동 클라이언트가 언제부터 단절되었는지를 판단할 수 있다.

'무효화 보고 응답(Res_IR)'과 '유효화 보고 응답(Res_VR)'은 각각 '무효화 보고 요청(Req_IR)'과 '유효화 보고 요청(Req_VR)'에 대한 서버의 응답 메시지이다. '무효화 보고 응답(Res_IR)'은 이동 클라이언트의 캐쉬에서 삭제되

어야 할 대상 데이타를 알려주는 것인데 반하여 '유효화 보고 응답(Res_VR)'은 그 반대로 이동 클라이언트의 캐쉬에서 삭제되지 않고 남아 있어도 되는 아직 유효한 데이타가 어느 것들인지를 알려주기 위한 것이다. 또 한가지 두 응답 메시지 간의 중요한 차이점은, '무효화 보고 응답(Res_IR)'은 서버가 '무효화 보고 요청(Req_IR)' 메시지를 보내온 해당 이동 클라이언트에게만 송신하는 것에 반하여, '유효화 보고 응답(Res_VR)'은 '유효화 보고 요청(Req_IR)' 메시지를 보내온 해당 이동 클라이언트 뿐만 아니라 셀 내의 모든 이동 클라이언트를 대상으로 방송된다는 것이다. 따라서 '유효화 보고 응답(Res_VR)'의 경우에는 어느 이동 클라이언트의 요청에 대한 응답인지를 명확히 하기 위해서 클라이언트 식별자, Clinet_ID가 포함된다. '무효화 보고 요청(Req_IR)'과 '유효화 보고 요청(Req_VR)'에서와 마찬가지로 캐쉬에서 삭제될 또는 캐쉬에 남을 대상 데이타는 데이타 식별자 리스트로 전달할 수도 있고, 그룹 식별자 리스트로 전달할 수도 있다. 전자를 Res_IR_d (Res_VR_d)로 나타내고, 후자를 Res_IR_g (Res_VR_g)로 나타낸다. Last_TS는 캐쉬 유효성 확인을 요청한 이동 클라이언트가 보낸 타임스탬프 값을 나타내는데, '유효화 보고 응답(Res_VR)'의 경우에 이는 현재의 '유효화 보고 응답(Res_VR)'에 열거된 데이타 또는 그룹들이 Last_TS 이후에는 갱신되지 않았음을 나타낸다. 따라서 캐쉬 유효성 확인 요청을 하지 않은 이동 클라이언트들도 방송되는 '유효화 보고 응답(Res_VR)'을 수신하였을 경우 이 정보를 이용하여 자신의 캐쉬 내 일부 데이타 (또는 그룹)들의 유효성 여부를 확인할 수 있다.

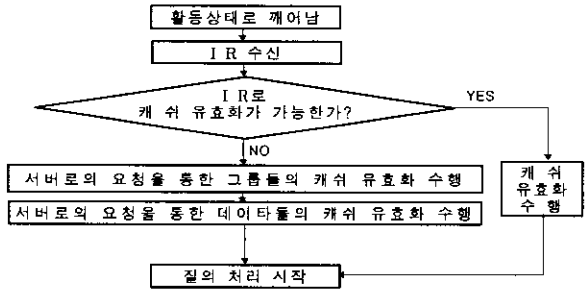
3.3 잘못된 무효화를 방지하는 캐쉬 유효화 기법

본 절에서는 이동 클라이언트의 수에 비해 채널이 충분히 존재할 경우, SGC 기반 기법을 사용할 때 발생하는 잘못된 무효화를 막을 수 있는 두가지 캐쉬 유효화 기법을 제안한다.

3.3.1 2단계 캐쉬 유효화 기법

2단계 캐쉬 유효화 (2 Phase Cache Validation, 이하 2PCV) 기법은 이동 클라이언트가 SCC 기법에서와 동일한 캐쉬 유효화 결과를 가지도록 할 뿐만 아니라, 모든 캐쉬 내의 데이타 식별자들을 전송하지 않기 때문에 SCC 기법보다 효율적인 기법이 될 수 있다. 또한, 잘못된 무효화를 하지 않기 때문에 SGC 기법보다 성능이 우수할 수 있다.

(그림 2)는 2PCV 기법에서의 캐쉬 유효화 과정을 나타낸 것이다. 이동 클라이언트는 자신의 캐쉬 유효성 여부 확인을 위해 서버에게 두 단계에 걸쳐 요청한다. 첫 번째 단계에서는 자신의 캐쉬에 존재하는 그룹의 식별자들을 서버에게 전송한다. 서버는 이에 대해 이동 클라이언트가 요청한 그룹들의 갱신 여부를 이동 클라이언트에게 알려 준다. 이로 인해, 이동 클라이언트는 SGC 기법에서와 동일한 캐쉬의 상태를 얻을 수 있다. 두 번째 단계에서는 첫 번째 단계에서 수신한 서버의 응답에 의해, 무효화 대상으로 판별된 그룹 내의 데이타 식별자들만을 서버에게 전송한다. 서버는 이에 대해 문의된 데이타들의 갱신 여부를 알려준다. 이로 인해 이동 클라이언트는 SCC 기법에서와 같은 캐쉬의 상태를 얻을 수 있다.



(그림 2) 2PCV에 의한 캐쉬 유효화

2PCV 기법에서 이동 클라이언트의 동작을 단계별로 기술하면 다음과 같다.

(1) 활동상태로 깨어남
(2) 서버로부터 방송되는 IR을 수신
(3) IR만으로 캐쉬 유효성 확인이 가능할 경우, 캐쉬 유효화 후 단계 (5)로 분기
(4) IR만으로 캐쉬 유효성 확인이 불가능할 경우,
  ① 서버에게 그룹들의 캐쉬 유효성 확인 요청(Req_IR_g)
  ② 그룹들에 대한 캐쉬 유효성 확인 요청에 대한 응답(Res_IR_g)을 수신하여 그룹들의 캐쉬 유효화 수행
  ③ 무효화 대상 그룹들 내에 속한 데이타들의 캐쉬 유효성 확인을 서버에게 요청(Req_IR_d)
  ④ 데이타들의 캐쉬 유효성 확인 요청에 대한 응답(Res_IR_d)을 수신하여, 무효화 대상 그룹들 내에 속한 데이타들의 캐쉬 유효화 수행
(5) 질의처리 시작

3.3.2 1단계 캐쉬 유효화 기법

2PCV 기법에서는 캐쉬 유효성 확인을 위하여 이동 클라이언트는 서버로 두단계에 걸쳐 메시지를 송수신해야 하는 단점이 있다. 또한, 무효화 대상 그룹의 수가 많거나 이동 클라이언트 내의 갱신된 그룹들에 포함된 데이타의 개수가 많을 경우에는 서버로 유효성 여부를 재확인해야 하는 데이타 식별자의 수가 많기 때문에 비효율적일 수 있다. 따라서, 한번의 접속으로 캐쉬 유효화를 수행할 수 있는 새로운 기법이 바람직하다. 이와 같은 1단계 캐쉬 유효화 (1 Phase Cache Validation, 이하 1PCV) 기법으로서 서버가 캐쉬 유효성 확인 요청을 받은 그룹들에 포함되어 있는 갱신된 데이타들의 식별자 모두를 이동 클라이언트로 전송하는 방안

을 생각할 수 있다. 이와 같이 하면 이동 클라이언트의 캐쉬 내에 존재하지 않는 데이타에 대해서도 갱신 사실을 이동 클라이언트에게 전송하는 결과를 초래할 수 있지만, 갱신이 비교적 많이 일어나지 않는 경우라면 1PCV 기법은 효율적인 기법이 될 수 있다.

<표 2>는 SCC, SGC 기법과 1PCV 기법의 차이를 나타낸 것이다. <표 2>에서 보는 것과 같이 1PCV 기법은 이동 클라이언트와 서버 간에 메시지 교환 시 식별자 단위(granularity)를 변화시키는 기법임을 알 수 있다.

〈표 2〉 캐쉬 유효화를 위해 전송되는 식별자들의 단위 비교

캐쉬 유효화 기법	캐쉬 유효성 확인 요청 시 이동 클라이언트가 전송하는 데이타	캐쉬 유효성 확인 응답 시 무효화 보고에 포함되는 데이타
SCC	데이타 식별자	데이타 식별자
SGC	그룹 식별자	그룹 식별자
1PCV	그룹 식별자	데이타 식별자

1PCV 기법에서 이동 클라이언트의 동작을 단계별로 기술하면 다음과 같다.

(1) 활동상태로 깨어남
(2) 서버로부터 방송되는 IR을 수신
(3) IR만으로 캐쉬 유효성 확인이 가능한 경우, 캐쉬 유효화 후 단계 (5)로 분기
(4) IR만으로 캐쉬 유효성 확인이 불가능한 경우,
  ① 서버에게 그룹들의 캐쉬 유효성 확인 요청(Req_IR_g)
  ② 요청한 그룹에 포함된 갱신된 모든 데이타들의 식별자를 수신(Res_IR_d)하여 데이타들의 캐쉬 유효화 수행
(5) 질의처리 시작

3.4 방송을 결합한 캐쉬 유효화 기법

본 절에서는 캐쉬 유효성 확인을 요청하는 이동 클라이언트의 수가 할당 가능한 채널의 수에 비해 훨씬 많아 채널 경쟁이 심한 경우, 서버가 방송을 이용하여 응답하는 캐쉬 유효화(Broadcast Cache Validation, 이하 BCV) 기법을 제안한다. BCV 기법에서 캐쉬 유효성 확인을 요청해야 하는 이동 클라이언트는 채널 경쟁이 심한 관계로 서버와의 접속에 있어 지연을 겪게 된다. 이 기간 동안 다른 이동 클라이언트의 요청에 의해 서버로부터 방송되는 유효화 보고 응답(Res_VR) 메시지를 수신하여, 3.2절에서 설명한 것처럼 자신의 캐쉬 내 일부 해당 데이타 (또는 그룹)의 유효성을 확인할 수 있다. 그렇게 유효성이 확인된 데이타 (또는 그룹)에 대해서는 서버에게 유효성 여부를 문의할 필요가 없어졌으므로 자신의 유효화 보고 요청 메시지(Req_VR)로부터 해당 데이타 식별자 (또는 그룹 식별자)를 삭제할 수 있다. 이로 인해 채널 대기 상태에 있는 각 이동 클라이언트는 Req_VR의 크기를 줄일 수 있으며, 비슷한 시간대에 활동 상태로 깨어난 이동 클라이언트 간에 중복 캐쉬되어 있던 데이타에 대한 유효성 확인이 반복적으로 서버에 요청되는 것도 피할 수 있다. BCV 기법에서 이동 클라이언트의 동작을 단계별로 기술하면 다음과 같다.

- 이동 클라이언트의 동작
  (1) 활동상태로 깨어남
  (2) 서버로부터 방송되는 IR을 수신
  (3) IR만으로 캐쉬 유효성 확인이 가능한 경우, 캐쉬 유효화 후 단계 (5)로 분기
  (4) IR만으로 캐쉬 유효성 확인이 불가능한 경우,
    ① 서버에게 송신할 캐쉬 유효성 확인 요청 메시지 (Req_VR)를 구성
    ② 서버와 연결을 시도하여 연결되지 않는 동안 : 서버로부터 방송되는 유효화 보고 응답(Res_VR)을 수신하여 자신의 Req_VR을 수정
    ③ 서버에 Req_VR을 송신
    ④ 클라이언트 식별자가 자신을 지칭하는 유효화 보고 응답(Res_VR)을 수신하여 캐쉬 유효화 수행
  (5) 질의처리 시작

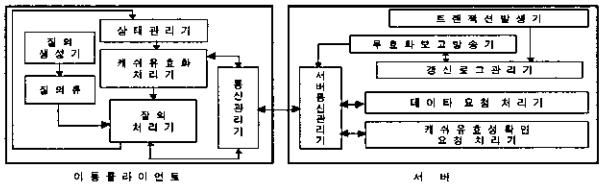
4. 시뮬레이션 모델

본 논문에서 제안한 기법들의 성능은 시뮬레이션을 통하여 평가하였다. 시뮬레이션을 위해 시뮬레이션용 C++ 라이브러리인 C++SIM [11]을 이용하였으며, Windows NT 4.0 환경 하에서 Visual C++ 6.0으로 시뮬레이션 시스템을 작성하였다. 본 절에서는 시뮬레이터 시스템 모델 및 파라미터들을 설명한다.

4.1 시뮬레이터 모델링

시뮬레이터는 (그림 3)과 같이 모델링하였다. 시뮬레이션이 시작되면, 이동 클라이언트는 서버와 접속이 되며, 방송되는 IR을 수신할 수 있는 활동 상태로 들어가게 되고, 캐쉬 일관성 유지를 위해서 서버의 '무효화 보고 방송기'가 IR을 방송할 때까지 기다리게 된다. 이동 클라이언트는 IR을 수신하게 되면 '캐쉬 유효화 처리기'에 의해서 자신의 캐쉬 내 데이타가 모두 유효한지를 점검하게 된다. 만일 IR만으로 이동 클라이언트의 캐쉬 유효성 여부 확인이 불가능하다면, 서버측의 '캐쉬 유효성 확인 요청 처리기'에게 캐쉬 유효성 확인을 요청하여 캐쉬 유효화를 수행하게 된다. 이 부분에서 다양한 캐쉬 일관성 유지 기법들이 사용 가능하다. 만일 IR만으로 캐쉬 유효화가 가능하다면 캐쉬 유효화 수행 후 질의가 바로 처리될 수 있다. 이동 클라이언트는 자신이 초기에 캐쉬했던 데이타만을 다시 요구한다고 가정한다[5, 14]. 또한, 시뮬레이션의 시작 시에는 이동 클라이언트 캐쉬 내의 데이타가 모

두 유효하다고 가정한다. 한편, 이동 클라이언트가 활동 상태에 있게 되면, '질의 생성기'에 의해서 질의가 발생되는데, 이 질의는 모두 '질의 큐'에 쌓이게 된다. '질의 큐'에 쌓인 질의는 캐쉬 유효성 확인이 모두 끝난 후, '질의 처리기'에 의해서 발생된 순서대로 처리된다. 질의가 처리되는 과정에서 이동 클라이언트가 캐쉬에 없는 데이타를 요구하면, 서버측 '데이타 요청 처리기'에 요청하여 데이타를 얻어 오게 된다. 이동 클라이언트의 '상태 관리기'는 이동 클라이언트가 한 개의 질의를 처리한 후, 서버와 단절될지 여부를 결정한다.



(그림 3) 시뮬레이터 구성도

4.2 시뮬레이션 파라미터

본 절에서는 시뮬레이션 파라미터를 설명한다. 파라미터들은 데이타베이스 및 작업부하 파라미터와 시스템 및 자원 파라미터로 크게 나눌 수 있으며, 이들 대부분은 이동 컴퓨팅 환경에서의 캐쉬에 대한 기존 연구[5, 6, 14]의 것들을 그대로 사용하거나 확장한 것들이다.

4.2.1 데이타베이스 및 작업부하 파라미터

본 시뮬레이션 시스템에서 데이타 처리의 단위 및 캐쉬의 단위는 객체로 가정한다. <표 3>은 데이타베이스 및 작업부하 관련 파라미터들을 나타낸 것이다. databaseSize는 서버 내의 데이타베이스에 존재하는 객체의 개수이다. cacheSize는 이동 클라이언트가 자신의 캐쉬에 포함할 수 있는 객체의 개수이다. objSize는 각 객체의 크기이다. idSize는 객체, 그룹, 또는 이동 클라이언트의 식별자 크기이며, groupSize는 데이타베이스 내의 객체를 그룹 단위로 나누어 관리할 경우, 한개의 그룹에 포함되는 객체의 개수이다. timestampSize는 타임스탬프의 크기이다.

tranArrRate는 서버 측에서 초당 생성되는 트랜잭션의 개수이며, updateNumPerTrans은 한 개의 트랜잭션에서 갱신되는 객체의 개수 평균값이다. hotUpdateSet은 전체 데이타베이스 중 갱신이 자주 발생하는 지역을 의미하는 것으로 전체 데이타베이스 크기에 대한 크기 비율로 나타낸다. hotUpdateProb는 전체 갱신에 대해 hotUpdateSet이 갱신될 확률을 나타낸다. 예를 들어, hotUpdateSet이 0.1이고 hotUpdateProb가 0.9라면, 전체 갱신의 90%가 전체 데이타베이스의 10%에 집중됨을 의미한다. refNumPerQuery는 한개의 질의가 접근하는 객체의 평균 개수이다. queryArrivalRate는 이동 클라이언트에서 초당 생성되는 질의의 개수이다. 한편, disconnectionProb는 이동 클라이언트가 한개의 질의를 처리한 후, 단절될 확률로 만일 disconnectionProb가 0.1이라면, 10번의 질의 처리 동안 한번은 단절됨을 의미한다. disconnectionTime은 이동 클라이언트가 단절된 뒤 단절 상태를 유지하는 평균 시간을 의미한다. 평균 단절 시간과 차이가 심한 단절 시간의 생성을 막기 위해 본 시뮬레이션에서는 평균 단절 유지 시간의 1/2에서 3/2 사이의 값만이 생성되도록 조정하였다.

4.2.2 시스템 및 자원 파라미터

<표 4>는 시스템 및 자원 관련 파라미터들을 나타낸 것이다. w는 IR의 윈도우[5] 크기로서, 한 IR에 포함되는 갱신 데이타의 정보가 현재의 타임스탬프로부터 과거 w번의 IR 방송 구간 동안 갱신된 데이타를 포괄함을 의미하는 것으로, 이동 클라이언트가 IR만으로 갱신된 객체들을 파악알 수 있는 과거의 시점을 한정해 준다. broadcastInterval은 서버가 IR을 방송하는 주기이다. uplinkCapacity는 이동 클라이언트들이 서버와 연결되어 통신할 경우 할당하는 무선 네트워크의 대역폭이다. broadcastCapacity는 방송 채널의 대역폭을 나타낸다.

〈표 3〉 데이타베이스 및 작업부하 파라미터

파라미터	내 용	기본설정치 (단위)
databaseSize	서버 내의 데이타베이스 크기	100,000(개)
cacheSize	이동 클라이언트 내의 캐쉬의 크기	5,000(개)
objSize	하나의 객체의 크기	256(bytes)
idSize	한 객체, 한 그룹, 또는 한 이동 클라이언트의 식별자 크기	64(bits)
groupSize	데이타베이스 내의 한 개의 그룹이 포함하는 객체의 개수	100(개)
timestampSize	타임스탬프의 크기	64(bits)
tranArrRate	갱신 트랜잭션 도착률	0.01(개/초)
updateNumPerTrans	transaction 한 개당 update되는 object의 평균개수	5(개)
hotUpdateSet	데이타베이스의 전체 크기 대비 갱신이 많이 일어나는 지역의 크기 비율	0.1
hotUpdateProb	서버 측에서 hotUpdateSet을 갱신할 확률	0.9
refNumPerQuery	질의 한개당 참조되는 객체의 평균 개수	20(개)
queryArrivalRate	초당 생성되는 질의의 개수	0.1(개/초)
disconnectionProb	한 트랜잭션을 처리하고 단절될 확률	0.1
disconnectionTime	평균 단절 유지시간	700(초)

4.2.2 파라미터 설정

실험에 사용될 파라미터의 기본 설정값은 <표 3>과 <표 4>에 제시된 것과 같다. 앞으로의 실험에서 특별한 명시가 없는 경우 각 파라미터의 값 설정은 이들 기본 설정을 따른다. 즉, 각 실험에서 그 값이 변화되는 파라미터 (예를 들어, (그림 4)의 x축을 나타내는 그룹의 크기(groupSize))를 제외한 나머지 파라미터들은 따로 명시가 없는 경우 그 설정값은 <표 3>과 <표 4>를 따른다.

5. 성능 평가

본 절에서는 제안한 기법들의 성능 평가 결과를 기술한다. 5.1절에서는 성능 평가 척도 및 파라미터 설정에 관한 내용을 언급하고, 5.2절에서는 시뮬레이션 실험 내용 개요를 설명한다. 5.3~5.5절은 실험 결과를 기술한다.

5.1 성능 평가 척도

캐쉬 일관성 유지를 위하여 그룹 식별자를 이용할 경우에는 데이타 식별자를 이용하는 경우보다 캐쉬 일관성 유지에 필요한 대역폭의 양은 줄일 수 있지만, 잘못된 무효화가 발생될 경우 해당 데이타를 서버로부터 검색하기 위한 대역폭을 대신 소모해야 한다. 따라서, 본 시뮬레이션에서는 각 이동 클라이언트 당 캐쉬 일관성 유지를 위해서 소모되는 대역폭과 데이타 검색 시 소모되는 대역폭의 합을 성능 평가의 척도로 삼았다.

〈표 4〉 시스템 및 자원 파라미터

파라미터	내 용	기본설정치 (단위)
uplinkCapacity	이동 클라이언트들이 서버와 송수신을 하는 대역폭의 크기	1.2 (kbytes/sec)
broadcastCapacity	방송용 채널의 대역폭의 크기	1.2 (kbytes/sec)
w	IR의 윈도우 크기	10
broadcastInterval	IR이 방송되는 주기	20(초)

5.2 시뮬레이션 실험 내용 개요

본 논문이 제안하는 캐쉬 일관성 유지 기법들의 성능을 평가하기 위해서, 다음 실험을 수행하였다.

5.2.1 SGC와 2PCV/1PCV 기법의 성능 비교

채널 경쟁이 심하지 않은 경우에, 본 논문에서 제안한 두 가지 새로운 캐쉬 유효화 기법(2PCV, 1PCV)이 잘못된 무효화가 발생할 수 있는 SGC 기법보다 더 좋은 성능을 나타낼 수 있는지의 여부를 살펴본다.

5.2.2 2PCV 기법과 1PCV 기법의 성능 비교

채널 경쟁이 심하지 않은 경우에, 본 논문에서 제안한 2PCV 기법과 1PCV 기법 상호 간의 성능을 비교하여 어떤 환경에서 어느 기법이 더 좋은 성능을 나타낼 수 있는지를 살펴본다.

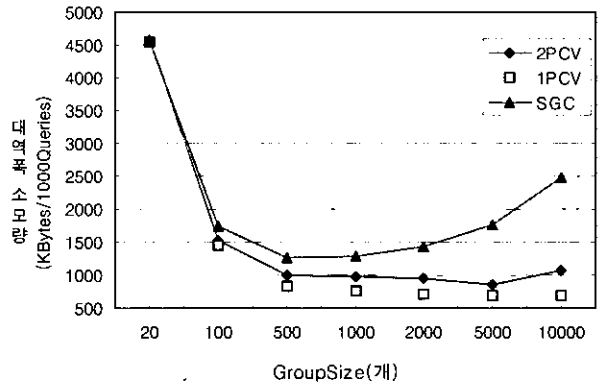
5.2.3 SGC, 2PCV, 1PCV, BCV 기법의 성능 비교

채널 경쟁이 심한 경우에, 본 논문에서 제안한 BCV 기법이 다른 기법들보다 더 좋은 성능을 나타낼 수 있는지의 여부를 살펴본다.

5.3 SGC 기법과 2PCV/1PCV 기법의 성능 비교

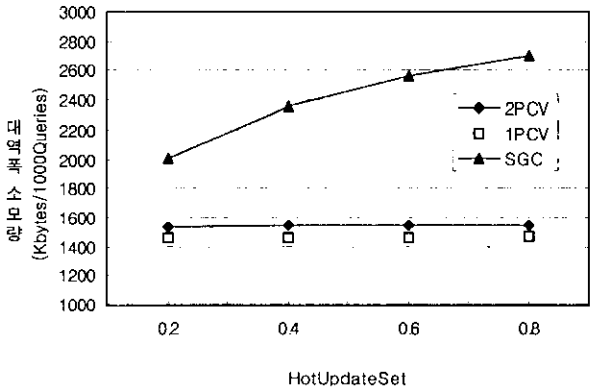
본 절에서는 SGC 기법과 2PCV/1PCV 기법의 성능을 비교한다. 그룹의 크기(groupSize), 자주 갱신되는 데이타베이스 지역의 크기(hotUpdateSet), 객체의 크기(objSize)의 변화에 따른 각 기법의 성능을 비교한 결과는 다음과 같다.

(그림 4)는 groupSize 변화에 따른 대역폭 소모량을 나타낸 것이다. groupSize가 작을 때는 모든 기법이 성능 향상을 보이는데, 그 이유는 그룹핑으로 인한 대역폭 절약 효과 때문이다. 하지만 groupSize가 커져감에 따라서 SGC 기법은 다른 제안한 기법들과는 달리 성능이 점차 나빠지는데 이는 잘못된 무효화 때문이다. 제안된 2PCV 기법과 1PCV 기법은 잘못된 무효화가 발생하지 않아 groupSize가 커지더라도 성능 저하가 나타나지 않았다.



(그림 4) 그룹의 크기 변화에 따른 대역폭 소모량

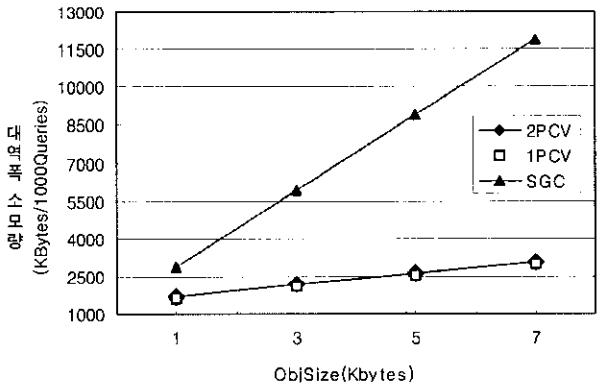
(그림 5)는 hotUpdateSet의 변화에 따른 대역폭 소모량을 나타낸 것이다. hotUpdateSet의 변화에 상관없이 2PCV 기법과 1PCV 기법은 거의 일정한 대역폭 소모량을 보이는 반면, SGC 기법은 hotUpdateSet이 커짐에 따라 점차 많은 대역폭의 소모를 나타냈다. 제안한 두가지 기법의 경우는 잘못된 무효화를 하지 않기 때문에 대역폭의 소모가 거의 일정한 것이며, SGC 기법의 경우에는 hotUpdateSet가 커질수록 잘못된 무효화의 가능성이 높아지기 때문에 더욱 많은 대역폭의 소모를 가져오게 된 것이다.



(그림 5) 자주 갱신되는 데이타베이스 지역 크기의 변화에 따른 대역폭 소모량

(그림 6)은 objSize 변화에 따른 대역폭 소모량을 나타낸

것이다. 객체의 크기 증가는 캐쉬 미스의 경우 서버와 이동 클라이언트 간의 데이타 전송량이 증가함을 의미한다. 따라서, 객체의 크기가 커질수록 세가지 기법 모두 대역폭 소모량이 늘어남을 알 수 있다. 그러나 그 증가폭이 2PCV 기법과 1PCV 기법의 경우에는 아주 완만한데 반하여 SGC 기법의 경우에는 그 증가율이 급격하다. 이는 역시 SGC 기법의 잘못된 무효화 때문이다. 한편, 그래프 표현 상 1PCV와 2PCV 간의 성능 차이는 거의 없는 것처럼 보이지만 (그림 4)와 (그림 5)에서 나타난 것과 비슷한 정도로 1PCV가 2PCV에 비해 더 우수하다. ((그림 4)의 groupSize = 100의 경우에 두 기법 간의 성능 비교 참조.)



(그림 6) 객체의 크기 변화에 따른 대역폭 소모량

이상의 실험들에서 나타난 2PCV/1PCV 기법의 SGC 기법에 대한 성능 향상의 이유는 2PCV/1PCV 기법은 잘못된 무효화를 방지하기 때문이다. 잘못된 무효화의 허용과 방지에 따른 이들 기법 간의 메시지 전송 부담을 비교하면 다음과 같다. 먼저 2PCV 기법과 SGC 기법을 비교하면, 캐쉬 유효화 과정에서 Req_IR_g 및 Res_IR_g의 전송은 양쪽 모두 동일하게 수행된다. 그러나 2PCV의 경우에는 추가적으로 Req_IR_d 및 Res_IR_d의 전송이 수행된다. 이후 질의 처리 과정에서는, SGC의 경우 잘못된 무효화로 인하여 2PCV에 비해 Req_D 및 그에 따른 D의 전송이 추가로 수행된다. SGC에서 이동 클라이언트가 Res_IR_g의 수신으로 무효화시키는 캐쉬 내 그룹의 수를 g, 캐쉬 내 각 그룹의 평균 데이타 객체의 수를 d, 이들 중 잘못 무효화가 되는 비율을 f, 이들 중 질의 대상 객체가 될 확률을 q라 하면, 잘못된 무효화로 인해 SGC가 2PCV에 비해 추가로 부담하는 메시지 전송량 M_{Δ}(SGC,2PCV)는 다음 식 (1)으로 나타낼 수 있다.

$$M_{\Delta}(SGC,2PCV) = g\times\{d\times f\times q\times(Size(Req\_D)+Size(D)) - (Size(Req\_IR\_d)+Size(Res\_IR\_d))\} \quad (1)$$

여기서 Size(*)는 해당 메시지의 크기를 나타낸 것이다. <표 1>에 정리된 각 메시지의 Flag 및 타임스태프의 크기를 무시하고 <표 3>의 파라미터 값으로 식 (1)에 나타난 메시지들의 크기를 구하면 다음과 같다.

$$Size(Req\_D) = 8 \text{ bytes} \quad (2-1)$$
$$Size(D) = 256 \text{ bytes} \quad (2-2)$$
$$Size(Req\_IR\_d) = 8\times d \text{ bytes} \quad (2-3)$$
$$Size(Res\_IR\_d) = 8\times d\times(1-f) \text{ bytes} \quad (2-4)$$

이들을 식 (1)에 대입하면 다음과 같다.

$$M_{\Delta}(SGC,2PCV) = g\times(264\times d\times f\times q + 8\times d\times f - 16\times d) \text{ bytes} \quad (3)$$

한편, 1PCV 기법과 SGC 기법을 비교하면, 캐쉬 유효화 과정에서 Req_IR_g의 전송은 동일하다. 그러나 SGC에서는 그에 대한 응답으로 Res_IR_g를 전송하는 데 반해 1PCV에서는 Res_IR_d를 전송한다. 따라서 잘못된 무효화로 인해 SGC가 1PCV에 비해 추가로 부담하는 메시지 전송량 M_{Δ}(SGC,1PCV)는 다음 식으로 나타낼 수 있다.

$$M_{\Delta}(SGC,1PCV) = \{Size(Res\_IR\_g)+g\times d\times f\times q\times(Size(Req\_D)+Size(D))\} - g\times Size(Res\_IR\_d) \quad (4)$$

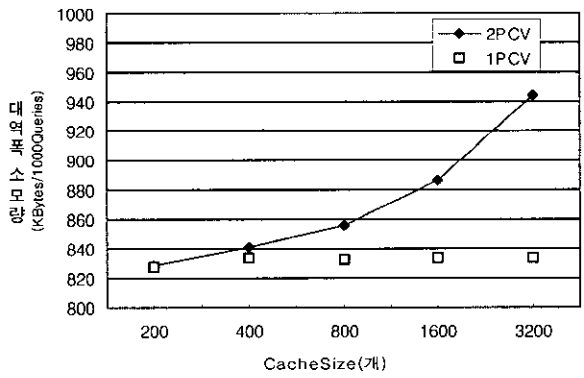
$Size(Res\_IR\_g) = 8\times g$ bytes이므로, 식 (2-1), 식 (2-2), 그리고 식 (2-4)를 식 (4)에 대입하면 다음과 같다.

$$M_{\Delta}(SGC,1PCV) = g\times(264\times d\times f\times q + 8\times d\times f - 8\times(d-1)) \text{ bytes} \quad (5)$$

5.4 2PCV 기법과 1PCV 기법의 성능 비교

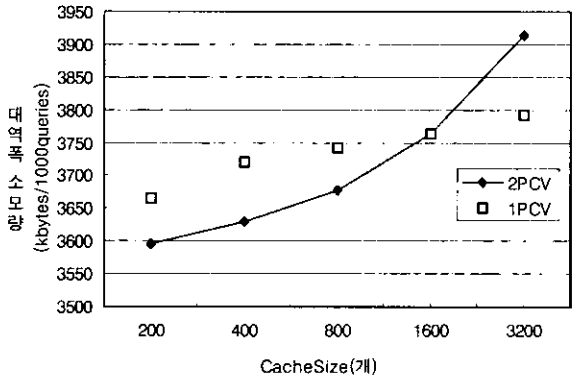
본 절에서는 본 논문에서 제안한 2PCV 기법과 1PCV 기법 상호 간의 성능 비교 결과를 기술한다. 시뮬레이션 파라미터 중 그룹의 크기(groupSize)를 500개로 설정한 상태에서 캐쉬의 크기(cacheSize) 변화에 따른 실험을 수행하였다.

(그림 7)은 트랜잭션 도착률(tranArrRate)을 0.01로 설정하였을 경우, 이동 클라이언트 내의 cacheSize 변화에 따른 대역폭 소모량을 나타낸 것이며, (그림 8)은 tranArrRate를 0.1로 설정하였을 경우의 cacheSize 변화에 따른 대역폭 소모량을 나타낸 것이다.

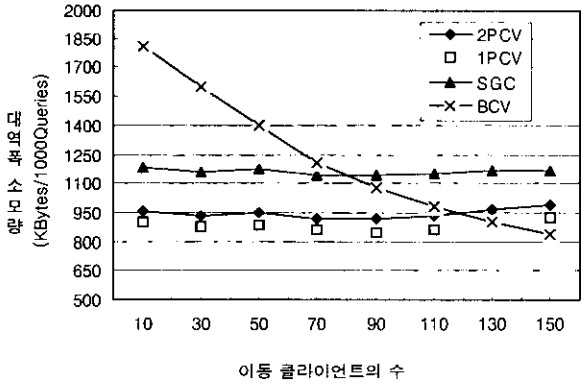


(그림 7) 캐쉬의 크기 변화에 따른 대역폭 소모량 (groupSize=500, tranArrRate=0.01)

(그림 7)에서 2PCV 기법이 cacheSize가 커짐에 따라서 1PCV 기법보다 더욱 많은 대역폭 소모량을 보이고 있다. 이는 2PCV 기법의 두 번째 단계에서 캐쉬 유효성 확인을 위하여 이동 클라이언트가 서버에게 객체 식별자를 전송할 때 전송되는 객체 식별자의 개수가 cacheSize가 커짐에 따라 증가하기 때문이다. 서버측으로 전송되는 객체 식별자의 개수가 증가하는 이유는 cacheSize가 커져감에 따라서 이동 클라이언트 내 캐쉬에서 하나의 그룹당 포함되는 객체의 개수가 증가하기 때문이다. 그러나, 1PCV 기법의 경우에는 캐쉬 유효성 확인 요청에 대해서 서버가 요청된 그룹 내의 갱신된 객체 식별자만 이동 클라이언트로 전송하게 되므로 cacheSize가 증가하더라도 그 전송량이 크게 증가하지 않는다.



(그림 8) 캐쉬의 크기 변화에 따른 대역폭의 소모량 (groupSize=500, tranArrRate=0.1)



(그림 9) 이동 클라이언트 수의 변화에 따른 대역폭 소모량

(그림 8)에서는 (그림 7)과 달리 cacheSize가 작을 경우에는 2PCV 기법이 더 좋은 성능을 보이고 있다. 이와 같은 결과가 나타나는 첫 번째 이유는 (그림 7)의 경우보다 갱신이 자주 일어나는 환경이기 때문이다 (transArrRate = 0.1). 두 번째 이유는 cacheSize가 작아지면 이동 클라이언트 캐쉬 내의 한 그룹당 객체의 개수가 줄어들게 되어, 캐쉬 유효성 확인을 위해 서버로 두단계에 걸쳐 요청을 해야하는 2PCV 기법의 경우, 서버측으로 전송되는 객체 식별자의 개수가 적어지기 때문이다. 그러나 cacheSize가 약 1600 이상을 넘어서면서부터는 2PCV 기법에서는 서버로 전송되는 객체의 개수가 많아지므로 1PCV 기법에 비해 점점 더 성능이 나빠졌다.

5.5 SGC, 2PCV, 1PCV, BCV 기법의 성능 비교

본 절에서는 캐쉬 유효화를 수행하려는 이동 클라이트들 간에 채널 경쟁이 심한 경우 방송 기법을 이용하는 BCV 기법의 성능이 다른 기법들에 비해 어떻게 나타났는지 그 결과를 기술한다. (그림 9)는 하나의 채널을 경쟁하는 이동 클라이언트 수의 변화에 따른 SGC, 2PCV, 1PCV, BCV 기법들의 대역폭 소모량을 나타낸 것이다. 채널을 경쟁하는 이동 클라이언트의 수가 적을 경우에는 2PCV 기법 및 1PCV 기법이 가장 좋은 성능을 나타냈으며, BCV는 SGC보다도 나쁜 성능을 나타냈다. 그러나 채널을 경쟁하는 이동 클라이언트의 수가 약 120개 이상을 넘어서면서부터는 BCV 기법이 2PCV 기법 및 1PCV 기법보다 더 좋은 성능을 나타내었다. 이는 3.4절에서 기술한 것처럼 채널을 기다리고 있는 이동 클라이언트의 수가 많아질수록, 다른 이동 클라이언트에 의해 요청되어 방송되는 캐쉬 유효화 보고를 수신하고 참조하여 각 이동 클라이언트는 자신이 준비하고 있던 캐쉬 유효화 보고 요청 메시지의 크기를 계속 줄여 나갈 수 있기 때문이다.

6. 결 론

이동 통신 기술의 급속한 발전으로 교통, 증권, 뉴스, 기상, 스포츠, 연예, 오락 등 이동 컴퓨팅 환경에서의 데이타 서비스에 대한 수요가 증가하고 있다. 이동 데이타 서비스를 제공하는 데 있어 자주 접근되는 데이타를 서버로부터 이동 클라이언트의 기억 장치로 캐쉬하는 기법은 사용자 질의 응답 시간의 단축, 대역폭의 절약, 그리고 그로 인한 배터리의 절약 등을 가져올 수 있지만, 캐쉬의 일관성을 유지해야 하는 부담이 생긴다. 캐쉬 일관성을 유지하기 위해서 한 셀 내에서 서버(Mobile Support Station)가 무효화 보고(invalidation report)를 일정 시간마다 주기적으로 방송하는 기법은 효율적일 수 있다. 그러나, 이동 클라이언트들은 의도적 또는 비의도적인 접속 단절을 겪을 수 있기 때문에 다시 활동 상태로 깨어난 후 수신하는 무효화 보고만으로 자신의 캐쉬 유효성 여부를 판단하지 못할 수 있다. 이 경우에는, 캐쉬 전체를 무효화하든지 아니면 서버에게 자신의 캐쉬 유효성 확인을 요청하여 캐쉬 내의 데이타 중 유효한 것만 남기는 캐쉬 유효화를 수행할 수 있다.

본 논문에서는 한 셀 내에서 할당 가능한 채널의 수와 이동 클라이언트 수의 관계에 따라서 대역폭 소모량을 효율적으로 줄일 수 있는 세가지 새로운 캐쉬 유효화 기법, 2PCV(2 Phase Cache Validation), 1PCV(1 Phase Cache Validation), BCV(Broadcast Cache Validation)을 제안하였

다. 시뮬레이션을 통하여 이들의 성능을 평가한 결과, (1) SGC 기법과 2PCV/1PCV 기법의 비교에서는, 잘못된 무효화를 방지하는 본 논문의 2PCV 및 1PCV 기법의 성능이 그룹의 크기, 자주 갱신되는 데이타베이스 지역의 크기, 객체의 크기가 커짐에 따라서 SGC 기법에서처럼 잘못된 무효화로 인한 급격한 성능 저하를 나타내지 않고, 일정한 성능 수준을 유지하였다. (2) 2PCV 기법과 1PCV 기법의 성능 비교에서는 캐쉬의 크기가 작고, 서버에서 갱신이 자주 일어날 경우에는 2PCV 기법이 PCV 기법보다 좋은 성능을 나타내었고, 그렇지 않은 경우에는 1PCV 기법이 더 우수하였다. (3) BCV 기법과 다른 기법들의 비교에서는, 셀 내에서 채널 경쟁이 심하지 않은 경우에는 BCV 기법의 성능은 좋지 못했지만, 채널 경쟁이 심해지면 BCV 기법의 성능이 가장 우수한 것으로 나타났다.

본 논문의 성능 평가에서는 방송 채널과 상향 채널의 대역폭이 동일한 대칭적 구조를 대상으로 하였다. [1] 등의 방송 디스크(broadcast disk) 연구에서는 상하향 채널 간의 대역폭이 다른 비대칭적인 구조를 요하는 응용들을 대상으로 하였는데 이러한 응용들이 많이 제안되고 있다. 본 논문에서 제시한 캐쉬 유효화 기법을 비대칭적 환경에서 사용할 경우, 대칭적 환경에서의 성능과 비교하여 성능 향상이 기대된다. 1PCV와 2PCV 기법의 경우 방송 채널을 사용하지 않으므로 대역폭 소모량에 변화가 없다. 그러나 BCV 기법의 경우, 대역폭 소모량은 줄어들 것이다. 상향 채널의 대역폭이 작아지고 상대적으로 방송 채널의 대역폭이 커지면 Res_VR이 방송될 때 더 많은 정보를 전송할 수 있고, 이는 상향 채널을 기다리고 있는 이동 클라이언트들의 Req_VR 크기를 더 줄일 수 있는 효과를 가져온다. 따라서, 대역폭 소모량을 더 줄이는 효과를 가져온다.

향후 연구로는 제시된 기법들의 최적화와 질의 응답시간 단축을 위한 기법을 들 수 있다.

첫째, Req_IR(VR)에 대한 서버의 응답인 Res_IR(VR)로는 [14]에서 제시된 것처럼 비트 벡터를 이용하여 대역폭 소모량을 더 줄일 수 있다. 뿐만 아니라, Req_IR(VR)에 대해서 Res_IR(VR) 대신 역으로 Res_VR(IR)로 답하는 방법도 생각할 수 있다. Res_IR과 Res_VR은 주어진 Req_IR 또는 Req_VR에 대해 서로 보집합(complement) 관계에 있으므로, 양자 간의 크기를 비교하여 작은 쪽을 전송할 수 있다.

둘째, 이동 클라이언트는 Res_IR(VR)을 수신하면 더 이상 유효하지 않은 데이타를 캐쉬에서 삭제하는데, 대신 서버에서 최신 버전을 가져와 캐쉬하는 방법 즉, 변경 전파(update propagation)를 고려할 수 있다. 물론, 변경 전파의 대상 데이타는 자주 질의되는 것이어야 한다. 어느 데이타를 변경 전파시킬 것인가에 대한 정보는 Req_IR(VR)의 전송시에, 평소 이동 클라이언트가 유지하고 있는 질의 빈도(query frequency) 정보를 함께 전송(piggybacking)함으로써 서버에 전달할 수 있다. 또한 서버는 변경 전파되는 데이타를 해당 Res_IR(VR)에 함께 전송하면 된다. 이와 같은 Req/Res_IR(VR)의 확장은 질의 응답시간의 단축을 위한 것이다.

셋째, 캐쉬 유효화 과정과 질의 처리 과정을 병행(interleaving)하는 기법도 가능하다. 단절 후 재접속된 이동 클라이언트는 초기에 캐쉬 전체를 무효한 것으로 간주한 상태에서 곧바로 질의 처리를 시작한다. 원하는 데이타가 캐쉬에 존재하지 않을 경우 Req_D를 전송하는데 이때 전체 Req_IR의 일부를 함께 전송한다. 서버는 요청된 데이타와 함께 해당 Res_IR을 전송한다. 원하는 데이타가 캐쉬에 존재할 경우에도 그 유효성을 아직 확인하지 않았으므로 확인을 요청한다. 이때에도 아직 처리가 안된 Req_IR의 일부를 함께 전송한다. 해당 데이타가 유효하지 않다고 판단될 경우 서버는 Res_IR 전송시 해당 데이타를 함께 전송한다. 이와 같은 확장 역시 캐쉬 유효화 과정으로 인한 질의 처리의 지연을 줄여 질의 응답시간을 단축하기 위한 것이다.

참 고 문 헌

[1] S. Acharya et al., "Broadcast Disks : Data Management for Asymmetric Communication Environments," Proc. ACM SIGMOD Conf. on Management of Data, pp.199-210, 1995.

[2] S. Acharya et al., "Balancing Push and Pull for Data Broadcast," Proc. ACM SIGMOD Conf. Proc. on Management of Data, pp.183-194, 1997.

[3] R. Alonso and H. Korth, "Database System Issues in Nomadic Computing," Proc. ACM SIGMOD Conf. on Management of Data, pp.388-392, 1993.

[4] D. Barbara, "Mobile Computing and Databases - A Survey," IEEE Transactions on Knowledge and Data Engineering, 11(1), 1999, pp.108-117.

[5] 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.

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

[7] S. Cuce and A. Zaslavsky, "Adaptive Cache Validation for Mobile File Systems," ER Workshops, pp.181-192, 1998.

[8] M. Franklin, "Client Data Caching : A Foundation for High Performance Object Database Systems," Kluwer Academic Publishers, 1996.

[9] T. Imielinski et al., "Energy Efficient Indexing On Air," Proc. ACM SIGMOD Conf. on Management of Data, pp.25-36, 1994.

[10] J. Jing et al., "Bit-Sequences : An Adaptive Cache Invalidation Method in Mobile Client/Server Environments," ACM/Baltzer Mobile Networks and Applications, Vol.2, No.2, 1997.

[11] M. Little and D. McCue, "Construction and Use of a Simulation Package in C++," Technical Report, Dept. of Computer Science, University of Newcastle upon Tyne, March 1994.

[12] L. Mummert and M. Satyanarayanan, "Large Granularity Cache Coherence for Intermittent Connectivity," Proc. the 1994 Summer USENIX Conf., Jun. 1994.

[13] M. Wong and W. Leung, "A Caching Policy to Support Read-only Transactions in a Mobile Computing Environment," Technical Report CS-TR-95-07, Dept. of Computer Sciences, The Chinese Univ., 1995.

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

임 상 민

e-mail : smlim@postmedia.co.kr

1998년 중앙대학교 컴퓨터공학과 졸업 (공학사)

2000년 중앙대학교 대학원 컴퓨터공학과 (공학석사)

현재 포스트미디어 연구원 재직 중

관심분야 : Java, XML, WML 등

강 현 철

e-mail : hckang@cau.ac.kr

1983년 서울대학교 컴퓨터공학과 졸업 (공학사)

1985년 U. of Maryland at College Park, Computer Science(M.S.)

1987년 U. of Maryland at College Park, Computer Science(Ph.D.)

1988년~현재 중앙대학교 컴퓨터공학과 교수

관심분야 : 이동 데이타베이스, 웹 데이타베이스, DBMS 저장 시스템 등