이동 데이타베이스 시스템에서 방송을 이용한 캐쉬 유효화 기법

임상민, 강현철
중앙대학교 컴퓨터공학과

A Broadcast Cache Validation Scheme in a Mobile Database System

Sangmin Lim, Hyunchul Kang
Dept. of Computer Science and Engineering, Chung-Ang University

요 약

이동 통신 기술의 급속한 발전으로, 이동컴퓨팅 환경에서의 데이타 서비스에 대한 수요가 급증하고 있다. 이동호스트 내에 캐쉬가 존재할 경우, 대역폭의 절약 및 빠른 응답시간을 제공할 수 있지만, 캐쉬 일관성을 유지해야하는 부담이 생긴다. 한 셀 내에 존재하는 수 많은 이동호스트들의 캐쉬 일관성 유지를 위해서 서버가 캐쉬 무효화 보고를 일정시간마다 주기적으로 보내는 방법은 효과적일 수 있다. 그런데, 이동호스트가 오랜 시간 동안의 접속 단절로 인해 무효화 보고만으로 자신의 캐쉬 유효성 여부를 판단하지 못할 경우에는, 서버에게 캐쉬 유효성 여부에 대한 확인을 요청함으로써 캐쉬를 유효화 할 수 있다. 만일 유효성 여부 확인을 요청하는 이동호스트의 수가 많을 경우, 서버는 효율적인 방법으로 응답을 해야 한다. 본 논문에서는 유효성 확인을 요청하는 이동호스트들의 수가 할당 가능한 채널 수에 비해 상대적으로 많은 경우, 이를 방송을 이용하여 응답하는 방법을 제안한다. 이동호스트는 방송되는 내용을 계속 듣게 됨으로써, 이미 유효성 여부 확인이 이루어진 데이타에 대한 반복된 요청을 피할 수 있다.

1. 서론

이동 통신 기술의 급속한 발전으로, 이동 통신 환경에서 데이타 서비스가 제공되고 있다. 최근 PCS를 이용한 증권, 뉴스, 기상, 스포츠 등의 데이타 서비스 실시는, 앞으로 계속 확대 제공될 이동 데이타 서비스의 예이다.

이동컴퓨팅 환경의 가장 큰 특성은, 기존의 유선 네트워크에 연결된 호스트 컴퓨터로서 무선 통신 능력을 가지고 있는 MSS(Mobile Support Station)와 이동호스트 간에 네트워크 연결상태가 불안정하다는 것과 의도적 혹은 비의도적인 접속단절이 빈번하다는 것이다. 또한, 유선환경보다 낮은 대역폭을 가지고 있으며, 이동호스트의 배터리 수명이 충분히 길지도 못하다. 이런 문제를 해결하기 위한 대표적인 방법으로 방송과 캐쉬기법이 있다[1][2][3].

이동호스트 내의 캐쉬의 이용은 질의 응답시간의 단축, 대역폭의 절약, 그로 인한 배터리의 절약을 가져올 수 있지만, 캐쉬의 일관성을 유지 해야 하는 부담이 생긴다. 따라서, 캐쉬 일관성 유지를 위한 효율적인 기법이 필요하다.

[2]에서는 MSS가 주기적인 무효화 보고(invalidation report, 이하 IR)를 방송하여 캐쉬 일관성을 유지하는 방법을 제안하였고, [5]에서는 오랜 시간동안 MSS와의 단절로 인하여, 무효화 보고만으로 캐쉬의 유효성을 확인하지 못할 경우, 이동호스트 각각이 MSS 측으로 유효성 여부 확인을 요청하여 캐쉬를 유효화하는 방법을 제안하였다.

그러나, 유효성 여부 확인을 요청하는 이동호스트의 숫자가 할당 가능한 채널의 수보다 많은 경우, 효율적인 캐쉬 유효화 기법이 필요하다. 예를 들어, 하루의 업무가 시작되는 아침 시간이나 오후의 업무가 시작되는 점심 식사시간 직후에는, 수 많은 사용자가 이동호스트를 단절 상태에서 재접속하게 되는데, 이 때 효율적인 캐쉬 유효화 기법이 필요하다. 특히, 셀(cell)내에 이동호스트의 수가 많아질수록 이동호스트들 간에 동일한 데이타가 중복 캐쉬되어 있을 가능성이 높은데, 이들 동일한 데이타에 대한 유효성 확인 요청이 여러 이동호스트들로부터 여러 차례 반복적으로 MSS로 보내질 수 있다.

본 논문에서는 캐쉬 유효성 확인 요청에 대한 응답으로 서버는 방송을 이용하되, 이동호스트는 방송되는 정보를 이용하여, 이미 유효성 여부 확인이 이루어진 데이타에 대한 반복된 요청은 피할 수 있는 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2절에서는 관련연구를 소개하고, 3절에서는 방송을 이용한 캐쉬 유효화 기법을 제안한다. 4절에서는 모의 실험을 통하여 성능을 평가하고, 5절에서 결론을 맺는다.

2. 관련연구

이동컴퓨팅 환경에서 캐쉬 일관성 유지를 위한 방법으로 정기적인 IR의 방송을 이용할 수 있다[2][4]. IR의 내용은 최근에 갱신된 데이타의 식별자와 갱신된 시점의 타임스탬프 쌍의 리스트로 이루어진다.

[2] 이때, '최근' 이란, MSS에서 시스템 파라미터로 설정된 시간 동안을 의미하며, 보통 w(≥1)개의 IR 구간으로 정한다 [4]에서는 이진 비트열(bit sequences)과 타임스탬프의 집합으로 IR을 구성함으로써 그 양을 줄이는 방법을 제안하였다. IR을 받은 이동호스트는 자신의 캐쉬 내 데이타별로 유효성 여부를 판단할 수 있으므로, 캐쉬 일관성을 유지해 나가게 된다 그러나, w개의 IR구간 보다 오랜 시간 동안 단절된 이동호스트는 재접속 시, 방송되는 IR로써 자신의 캐쉬 유효성 여부를 판단하지 못한다. 이 경우, 단순히 캐쉬 내의 데이타를 모두 무효화시키는 Broadcasting Timstamps, Amnesic Terminal 등의 기법이 있으며[2], 캐쉬 내 데이타의 식별자와 이들의 유효성을 가장 최근에 확인했던 타임스탬프를 MSS에게 보내 캐쉬 유효성 여부를 묻는 Simple-Checking Caching 기법과 데이타 각각의 식별자 모두를 MSS에게 보내지 않고, 데이타를 미리 정해진 규칙에 의해 그룹핑 한 다음 각 그룹의 식별자를 MSS에 보내어 그룹별로 유효성 여부를 묻는 Simple-Grouping Caching 기법 또는 GCORE와 같은 기법도 있다[5].

3. 방송을 이용한 캐쉬 유효화 기법

본 논문에서 가정하는 시스템 환경은 그림 1과 같다. 이동호스트는 MSS와 무선연결을 통하여, 기존 유선 네트워크에 연결할 수 있다 MSS는 전체 데이타베이스(DB)의 내용을 복제하여 저장하고 있으며, DB 내 각 데이타들의 갱신 시점을 유지하고 있다. 이동호스트 내의 캐쉬 일관성 유지를 위하여, MSS는 자신이 관장하는 영역, 곧 셀 내의 이동호스트를 대상으로 IR을 주기적으로 방송하게 되며, DB의 갱신은 MSS측에서만 일어난다 갱신이 DB에 반영되는 시점은 IR 시점이다. 따라서 IR 이후 다음 IR까지는 캐쉬 일관성이 보장된다. 또한, 이동호스트는 판독(read-only) 연산만을 한다고 가정한다. 이동호스트에서 발생된 질의로 인하여 요청되는 데이타가 캐쉬 내에 유효한 상태로 남아 있지 않을 경우, MSS측으로 데이타에 대한 요청을 하게 되고 이 데이타는 이동호스트 내에 캐쉬된다. 만일 이동호스트가 오랜 단절로 인하여, IR만으로 캐쉬의 유효성 여부 확인이 불가능할 경우, 이동호스트는 [5]에서처럼 MSS측으로 캐쉬 유효성 여부에 대한 확인 요청을 한다

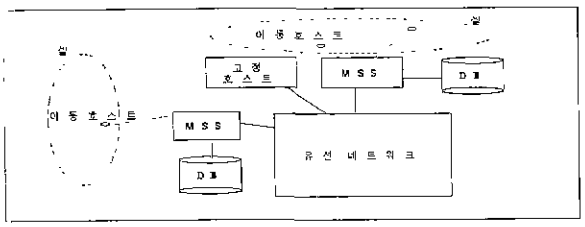

그림 1 이동컴퓨팅 환경

[5]에서 제안한 Simple-Checking Caching 기법에서는 캐쉬 유효성 여부 확인이 요청되면, MSS는 해당 이동호스트만을 위한 IR을 생성하고 이를 그 이동호스트 측으로 전송한다. 그러나, 본 논문에서는 유효성 확인 요청에 대한 응답으로 개별 IR 대신 캐쉬 유효화 보고(validation report, 이하 VR)를 생성하여 방송한다. 따라서, 캐쉬 유효성 여부 확인 요청에 대한 응답이, 그 요청을 한 이동호스트 만이 아닌 셀 내의 다른 이동호스트들에게도 전달된다.

MSS가 주기적으로 방송하는 IR의 형태는 다음과 같다.

< Flag, {TS_1, data_ID, data_ID, ...},{TS_2, data_ID, data_ID, ...}, ..., {TS_w, data_ID, data_ID, ...} >

Flag는 현재 방송되는 보고가 IR인지 VR인지 구별하기 위한 것이다 TS_i (i=1, 2, , w)는 현재 시점을 기준으로 최근 i번째 IR 구간에 갱신된 데이타가 DB에 반영된 시점을 나타내는 것으로 본 논문에서는 해당 IR이 방송된 시점과 같다고 가정한다 TS_1 뒤에 열거된 data_ID는 해당 IR 구간에서 갱신된 데이타들의 식별자이다.

본 논문에서 제안하는 방송을 이용한 캐쉬 유효화 기법(Broadcast Cache Validation, 이하 BCV)을 이동호스트 및 MSS에 대해 단계별로 기술하면 다음과 같다.

● 이동호스트의 동작

1) 활동상태로 깨어남.
2) 주기적으로 방송되는 다음 IR을 기다림.
3) IR만으로 캐쉬 일관성 유지가 가능하다면, 단계 8로 분기.
4) MSS로 승신할 캐쉬 유효성 여부 확인 요청 메시지, Req_VR을 구성.
5) MSS와 연결을 시도하여 연결되지 않는 동안, 방송되는 VR을 수신하여서 Req_VR을 수정.
6) MSS에 Req_VR을 송신
7) VR을 수신.
8) 질의를 처리.

● MSS의 동작

아래를 계속 반복 수행.

1) 일정 시간이 되면, 그 동안의 데이타 갱신을 DB에 반영하고, IR을 생성 및 방송
2) 이동호스트로부터 Req_VR을 받으면, VR을 생성 및 방송.

Req_VR의 형태는 다음과 같다

< Flag, Last_TS, {data_ID, data_ID, ... } >

Flag는 이동호스트의 요청이 데이타 요청인지, 캐쉬 유효성 여부 확인 요청인지 구별하기 위한 것이다. Last_TS는 이동호스트가 마지막으로 받은 IR의 타임스탬프 값이며, data_ID는 유효성 여부 확인을 원하는 데이타들의 식별자이다.

VR의 형태는 다음과 같다.

< Flag, Client_ID, Last_TS, {data_ID, data_ID, ... } >

Flag는 현재 방송되는 보고가 IR인지, VR인지 구별하기 위한 것이다. Client_ID는 어느 이동호스트의 Req_VR에 대한 응답인지를 명시한다. data_ID는 유효하다고 확인된 데이타들의 식별자이다. Last_TS는 Req_VR을 보내온 이동호스트가 Req_VR에 명시했던 Last_TS이다. 이것과 데이타 식별자 리스트 정보는 채널을 할당받지 못해 아직 MSS로 연결하지 못하여 Req_VR을 보내지 못하고 있는 이동호스트들이 자신의 *Req_VR*을 수정하는 데 사용된다(이동호스트의 동작 단계 5 참조). 즉, VR에 명시된 데이타들은 Last_TS 이후 갱신되지 않았음을 의미하므로, 마지막으로 받은 IR의 타임스탬프 값이 Last_TS보다 큰 이동호스트들은 이 VR 내에 명시된 데이타 중 자신의 캐쉬에 있는 것에 대해서 그들이 유효하다는 것을 알 수 있게 된다. 따라서, 자신의 Req_VR로부터 해당 데이타 식별자를 삭제하여, Req_VR의 양을 줄일 수 있다.

4. 모의실험

모의실험은 다음과 같은 환경 아래 수행되었다. 각 MSS는 DB의 모든 내용을 복제하여 가지고 있는데, 100,000 개의 데이터로 구성된다.각 이동호스트의 캐쉬 용량은 5,000 개의 데이터를 저장할 수 있는 크기이다. 데이터 하나의 크기는 256 bytes 이다. Client_ID, data_ID, TS의 크기는 모두 64 bits이다. DB 내 데이터가 자주 갱신되는 지역(hot update region)의 크기는 DB 크기의 10%이며 전체 갱신의 90%가 hot update region에 집중된다 갱신은 hot update region(cold update region) 내에서는 균일한 분포로 일어난다. 이동호스트에 캐쉬된 데이타의 10%는 hot update region으로부터 캐쉬한 것이며, 이동호스트는 초기에 자신의 캐쉬를 가득 채운 데이터들에 대해서만 질의를 제기한다. 질의 하나 당 참조하는 데이타의 개수는 평균 20개이고, 질의는 초당 0.1개가 생성된다. MSS에서 일어나는 트랜잭션 하나 당 평균 갱신되는 데이터 개수는 5개이며, 트랜잭션은 초당 0.01 개가 생성된다. 셀 내 채널 한 개의 대역폭은 1.2 kbytes/sec이다. 본 실험에서의 주요 파라미터로는 중복도와 단절확률이 있다. 중복도는 한 데이타의 중복 캐쉬 정도를 나타내는 것으로, 만일 중복도가 0.5라면, 각 데이타는 총 이동호스트 수의 50%에 중복 캐쉬되어 있음을 의미한다 또한, 단절확률은 한 개의 질의를 처리한 후, 이동호스트가 단절될 확률을 의미하며, 단절이 될 경우, 평균 700초 동안 단절이 일어난다. w는 10이고, IR의 주기는 20초이다.

성능의 척도는 셀 내에서 IR 및 VR의 방송을 비롯하여, Req_VR, VR, 이동호스트로부터 MSS 측으로의 데이타 요청, 데이타 요청에 대한 응답 등을 위해, 한 이동호스트 당 총 1,000 개의 질의가 처리되는 동안 소모된 대역폭의 총합을 전체 이동호스트의 수로 나눈 값이다.

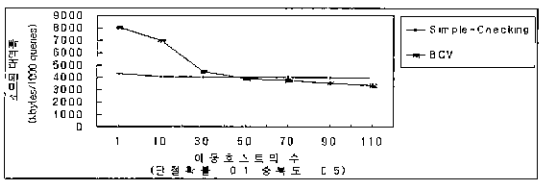

그림 2 이동호스트 수의 변화에 따른 대역폭의 소모량

그림 2는 셀 내 이동호스트 수의 증가에 따른 BCV와 Simple-Checking Caching 기법 간의 성능을 비교한 것이다. 한 개의 채널을 경쟁하는 이동호스트의 수가 50보다 적은 경우에는 BCV가 훨씬 많은 대역폭 소모량을 보이고 있다. 이유는 VR을 방송하기 때문으로, VR은 IR보다 상대적으로 큰 데이타량을 가진다 그러나, 이동호스트의 수가 많아질수록 각 이동호스트는 VR을 수신하여 더욱 더 많은 수의 데이타를 자신의 Req_VR로부터 제거할 수 있는 효과를 얻을 수 있으므로 대역폭 소모가 점차 줄어들고 있다.

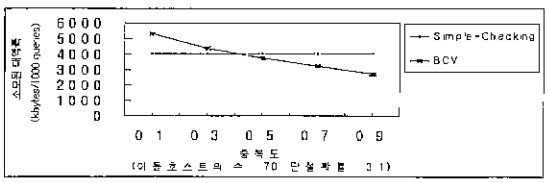

그림 3 중복도의 변화에 따른 대역폭의 소모량

그림 3은 중복도 변화에 따른 성능 비교를 나타낸 것이다 이동호스트의 수가 고정되어 있어도 중복도가 증가되면 BVC는 Req_VR내의 데이타량을 더욱 많이 줄일 수 있음을 확인할 수 있다.

그림 4는 단절확률 변화에 따른 성능 비교를 나타낸 것이다 단절확률이 커질수록 BCV의 성능이 더욱 우수해진다 이유는 새로 활동상태로 깨어나는 이동호스트의 수가 많아짐으로 인해 VR을 방송할 경우, 더욱 많은 수의 이동호스트들의 Req_VR 량을 줄일 수 있는 효과를 가져오기 때문이다.

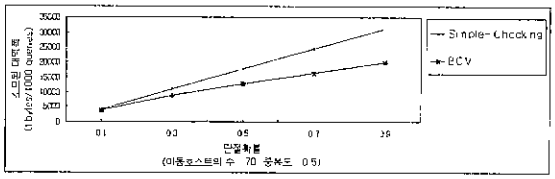

그림 4 단절확률의 변화에 따른 대역폭의 소모량

5. 결론

본 논문에서는 이동 데이타베이스 시스템에서, 접속 단절 이후 재접속된 이동호스트가 정기적인 무효화 보고로 캐쉬 유효성 확인을 할 수 없는 경우 MSS에 요청한 유효성 확인에 대한 응답으로 방송을 이용하는 BCV 기법을 제안하였다. BCV는 셀 내 이동호스트의 수가 많아지게 되면, 이동호스트 각각에 캐쉬되어 있는 데이타 간의 중복도가 높아질 수 밖에 없다는 사실에 착안한 것이다 모의실험 결과 이동호스트의 수가 많아질수록, 중복도가 클수록, 단절확률이 높을수록 BCV의 성능이 Simple-Checking Caching 기법에 비해 좋게 나타났다 세 가지 조건 모두 단절 후 재접속된 이동호스트의 캐쉬 유효성 여부 확인 요청 메시지의 양을 많이 줄일 수 있는 여건을 제공하기 때문이다.

본 논문을 통해, [2]에서와 같은 이동호스트의 캐쉬 일관성 유지를 위한 정기적인 무효화 보고 방송에, 캐쉬 유효화를 위한 방송의 적절한 결합은 성능향상에 기여할 수 있음을 알 수 있다.

참고문헌

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

[2] D. Barbara and T. Imielinski. "Sleepers and Workaholics : Caching Strategies in Mobile Environments," Proc. ACM SIGMOD Int'l Conf. on Management of Data, 1994, pp. 1-12.

[3] T. Imielinski et al., "Energy Efficient Indexing On Air," Proc. ACM SIGMOD Int'l Conf on Management of Data, 1994, pp. 25-36.

[4] 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. II, 1997.

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