장기간 접속 단절된 이동 클라이언트를 위한 효과적 캐시 유지 기법

박 광진 ° 김 성석, 윤 태명, 황 종선

고려대학교 컴퓨터학과 분산시스템 연구실

{kjpark, sskim, tmyoon, hwang}@disys.korea.ac.kr

Efficient Cache Maintenance Scheme for Long Disconnection Mobile Clients

Kwang-Jin Park ° , Sung Suk Kim , Tae-Myung Yun , Chong-Sun Hwang

Distributed Systems Lab., Dept. of Computer Science and Engineering, Korea Univ.

요 약

현재 이동 컴퓨팅 환경의 여러 제약조건을 고려한 클라이언트 캐시에 대한 다양한 연구가 진행되고 있다. 최근에는 캐시 데이터의 일관성 검사를 위하여 서버의 주기적인 무효화보고(Invalidation Report, *IR*) 기법을 이용한 연구가 활발하게 진행되고 있다. 즉 서버로부터 전송되는 이전 일정 주기(ω)동안 수정된 데이터 항목에 대한 정보를 이용하여, 클라이언트는 자신의 캐시 데이터의 일관성을 검사한다. 하지만 클라이언트가 오랜 접속 단절이 발생할 경우 모든 캐시 데이터를 버려야 하므로, 이 경우 성능에 큰 장점을 얻을 수 없게 된다. 이에 본 논문에서는 이동 컴퓨팅 환경에서 빈번한 접속 단절로 인하여 오랫동안 무효화 보고를 받지 못하더라도 유효한 캐시 데이터를 최대한 유지시킬 수 있는 기법들을 제안한다. 먼저 클라이언트가 접속 단절 후 첫 재접속이 되었을 경우, 자신의 이전 접속 단절 시간을 전송하도록 하여 서버가 동적으로 *IR*에 포함될 이전 일정 주기를 결정하도록 하였다. 이에 반하여 두 번째 기법에서는 서버가 특정 기간 동안의 *IR*을 저장하도록 하여, 오랜 접속단절 후에도 클라이언트가 직접 캐쉬 데이터의 일관성 검사를 요청할 수 있도록 하였다.

1. 서 론

이동 컴퓨팅 환경에서 사용자는 자신의 물리적인 위치와 관계없이 정보를 액세스할 수 있게되었다. 특히, 비대칭 통신 환경, 빈번한 접속 단절, 제한된 자원 등의 속성을 가진 이동컴퓨팅 환경에서 캐시 데이터는 매우 중요한 비중을 차지하고 있다. 즉, 캐시의 사용은 클라이언트의 서버에 대한 의존도를 낮춰주고 클라이언트 자원을 효율적으로 사용할 수 있게 한다. 그러나 캐시 데이터의 일관성에 대한 문제를 해결해야하며, 일반적으로 이동 컴퓨팅 환경에서 캐시 일관성 유지 기법은 다음과 같이 크게 세 가지로 분류되어질 수 있다. 첫째, 서버가 클라이언트의 상태를 유지하고 있어서, 데이터 아이템이 변경되면 클라이언트에게 즉시 그 사실을 전달해 준다. 두 번째는 클라이언트가 캐시 데이터에 접근하기 전에 서버에게 그 데이터의 유효성 검사를 요청하는 방식이다. 마지막으로, 서버는 *IR* 메시지를 보내주며, 클라이언트는 이 메시지를 이용하여 캐시의 일관성을 유지한다. 접속 단절이 빈번하고 클라이언트의 수가 유동적인 이동 컴퓨팅 환경에서는 세 번째 방식이 주로 사용되어지고 있다. 그러나 주기적인 캐시 *IR* 방식은 다음과 같은 세 가지 문제점을 안고 있다.

1) 클라이언트는 자신의 캐시 데이터를 사용하기 전 데이터의 유효성 여부를 확인하기 위해 다음 *IR*이 올 때까지 기다려야 한다.

2) 오랫동안 접속단절된 클라이언트의 경우, 무조건 모든 캐시 데이터를 버려야한다.

3) *IR* 메시지를 받은 후 클라이언트들로부터 질의가 집중적으로 발생할 수 있다.

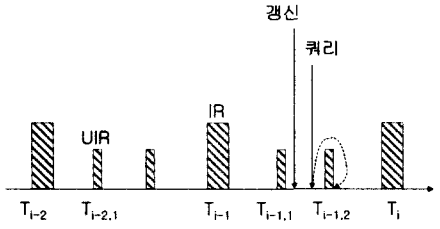


그림 1

위와 같은 문제점들을 극복하기 위해 Bit Sequence (*BS*) 기법[1]과 Updated Invalidation Report (*UIR*) 기법[2] 등이 제안되었다. BS 기법은 오랜 접속단절에 대처하기 위해 계층화된 비트열을 전송하였다. 이에 반하여 *UIR* 기법은 그림 1과 같이 *IR* 메시지 주기 사이에 적절한 정보(*UIR*)를 보냄으로써 질의 처리 시간을 줄여주도록 하였다. 하지만 이 기법들은 정보의 양이 많아지거나, 오랜 접속단절에 효율적으로 대처할 수 없다.

이에 대하여 본 논문에서는 클라이언트로부터 서버로의 데이터 요구 메시지의 회수를 줄이고, 또한 오랜 접속단절에 대처할 수 있는 캐시 유지 기법을 제안한다. 이를 위하여 먼저 서버가 관리하는 데이터를 접근 빈도에 따라 브로드캐스트 전용 및 요구기반 처리 데이터로 나눈다. 이후 일정 기간 이상 접속단절이 발생하면, 다음 두 가지 기법 중 하나를 적용한다. 먼저 클라이언트가 접속단절 후 첫 재접속을 시도할 때 자신의 접속단절 시간을 보내줌으로써, 서버가 *IR*에 대한 w크기를 동적으로 결정하도록 한다. 두 번째 방식은, 서버가 이전 일정 주기동안의 *IR*에 대한 정보를 별도의 공간에 저장해두도록 한다. 이후 클라이언트가 자신의 캐시 데이터에 대하여 명시적으로 유효성을 검사하도록 한다.

2. 기존 연구

서버는 일정 주기(w) 동안 수정된 데이터 항목을 *IR* 메시지를 통해 주기적(L)으로 전송한다. 이때 클라이언트의 접속 단절 시간과 w크기 그리고 주기 L 의 변화는 캐시를 사용하는 이동컴퓨팅 환경에서 매우 중요한 변수라 할 수 있다. 그러므로 이동컴퓨팅 환경에서 발생할 수 있는 여러 가지 상황들 즉, 클라이언트의 대기 시간, 서버의 작업 부하, 네트워크 대역폭의 효과적인 사용 등을 최적화하기 위해 앞서 언급했던 여러 요소들을 고려한 적절한 구성이 필요하다.

- w와 L, 그리고 D_T(접속단절시간)와의 관계는 다음과 같다.

1) $w>L$일 때 a: $D_T \geq w$: 캐시데이터 보장 못함
b: $D_T < w$: 캐시데이터 보장

2) $w=L$일 때 a: $D_T \geq w$: 캐시데이터 보장 못함
b: $D_T < w$: 캐시데이터 보장

3. 시스템 모델

시스템 환경을 구성하기 위해서 여러 다양한 모델을 사용할 수 있다. 일반적인 이동컴퓨팅 환경은 업링크와 다운링크 통신채널로 구성되어지며, 클라이언트는 다음과 같은 세 가지 기본적인 정보 전달 방식중 하나를 이용하여 필요한 정보를 얻을 수 있다.

1. 브로드캐스트 방식

2. 요구기반(on demand)

3. 혼합방식(hybrid)

브로드캐스트 방식은 클라이언트의 업링크를 허용하지 않기 때문에 장기간 접속 단절된 클라이언트는 모든 캐시 데이터를 버려야만 하고 요구기반 방식은 클라이언트가 캐시 데이터에 접근할 때마다 서버에게 쿼리 메시지를 보내기 때문에 클라이언트 숫자가 증가할 경우 서버의 오버헤드를 야기할 수 있다. 따라서 본 논문은 3번 혼합 방식을 사용함으로써 브로드캐스트 방식과 요구기반 방식이 갖는 단점을 보완하고 캐시 가용성을 높일 수 있는 캐시 유지기법을 제안한다.

3.1 기본 모델

본 논문에서는 전달할 데이터를 핫 데이터와 콜드 데이터로 분류한다. 핫 데이터는 브로드캐스트방식을 사용하고, 콜드 데이터는 클라이언트의 요청에 따라 요구기반 방식을 사용하여 클라이언트에 전달하게 된다. 데이터 분류를 위한 기준은 [3]에서 제시한 방식을 사용하였다. 본 논문에서 제안하는 방식에서는 핫 데이터에 대해서는 브로드캐스트 방식을 사용하여 서버로의 메시지 수를 줄이고 콜드데이터에 대해서는 요구기반 방식을 사용함으로써 쿼리에 대한 응답시간을 줄여 전체적인 시스템의 성능을 향상시킬 수 있도록 하였다.

서버 측 모델

서버가 관리하는 데이터는 크게 세 가지로 핫 데이터, *IR* 메시지, 요구기반 데이터로 분류할 수 있다. D를 데이터 아이템들의 집합이라 하고 T_i를 브로드캐스트 되어지는 *IR* 시간이라 가정할 때 서버는 *IR* 메시지 구성을 위해 다음과 같은 데이터 목록 U_i를 유지한다. 갱신된 데이터 아이템 j에 대하여

$$U_i = \{[j, t_j] | \ j \in D \text{ 이고 } T_i - w \le t_j \le T_i\}$$

- 단일 네트워크 채널을 사용하며 *IR*과 핫(데이터 객체)데이터를 *IR* 단계와 데이터 전달(Data Delivery) 단계로 나누어 전달한다.

- 서버는 브로드캐스트 데이터를 주기적(L)으로 클라이언트에게 전달한다. 핫 데이터의 구조는 플랫(flat)한 구조이며, 모든 데이터들은 각 주기의 시작 시점의 일관된 상태라고 가정한다. 매 주기의 시작 시점에서 먼저 *IR*를 전송한다.

- 서버는 업링크 채널을 통해 얻은 클라이언트의 쿼리 정보를 관리한다.

클라이언트 측 모델

클라이언트는 서버로부터 *IR* 정보를 받아 캐시의 유효성 여부를 판단한다. 클라이언트는 IR 메시지 구성을 위해 다음과 같은 Q_i 목록을 유지한다.

Q_i = {j|j 는 $[T_{i-1}, T_i]$의 주기동안 받은 요구기반 데이터에 대한 쿼리임}

- 핫 데이터에 대한 쿼리는 브로드캐스트 데이터를 이용하여 처리한다.

- 클라이언트는 무선 네트워크 환경을 통해 구성된다.

- 클라이언트는 브로드캐스트 채널을 통해 서버의 핫 데이터를 청취하며 업링크 채널을 통해 서버에게 데이터를 요청한다.

- 클라이언트는 캐시를 유지하며 캐시 데이터를 사용하기 전에 서버의 *IR* 메시지를 통해 캐시의 유효성 여부를 판단한다.

구체적인 알고리즘은 다음과 같다.

(i) 데이터 항목 x에 대한 쿼리를 받을 경우

```
if (x ∈ 핫 데이터) { // x가 핫 데이터인 경우
   if (x ∈ 캐시 && 현 주기의 IR을 받은 경우) 캐시데이터 사용
   else 다음 브로드캐스트 채널을 기다림
}
else { // x 가 콜드 데이터인 경우 - 서버에게 요청
   if (x ∈ 캐시 && 현 주기의 IR을 받은 경우) 캐시데이터 사용
   else if (x ∈ 캐시 && 현 주기의 IR을 받지 않은 경우) 다음 IR 기다림
   else 서버에게 데이터 요청
}
```

(ii) 주기적인 *IR*을 받은 경우

```
if (Ti - Tl > w) 모든 캐시 데이터 버리기 --- ① // 3.2 절 아이디어 적용
else {
   - 캐시 데이터 일관성 검사
   - if (연산이 대기중인 경우) (i) 알고리즘 수행// 쿼리에 대한 응답
}
Tl := Ti
```

3.2 장기간 접속 단절된 모빌 클라이언트의 캐시 유지 기법

이 모델은 기존에 클라이언트가 w 주기 이상 *IR*메시지를 놓쳤을 경우 자신의 캐시 데이터를 모두 버려야하는 문제점을 극복하고 캐시내의 유효한 데이터를 최대한 살리는데 목적이 있다. 클라이언트의 캐시 데이터를 표 1과 같이 유효(valid), 불분명(unknown), 스테일(stale) 데이터로 분류하였으며, 본 논문에서는 불분명(unknown)데이터를 버리지 않고 최대한 살리는데 초점을 맞추었다. 따라서 3.2.1절에서는 장기 접속 단절된 클라이언트가 자신의 접속 단절 시간을 서버에게 보내주어 w 크기를 조정하여 줌으로써 *IR*을 놓쳐 캐시 데이터를 버리게되는 클라이언트의 숫자를 최소화할 수 있는 방식을 소개하며 3.2.2절에서는 *IR*메시지를 놓친 클라이언트가 개별적으로 캐시소생 컴포넌트(Cache resuscitation c omponent)에게서 받은 백업 데이터를 통해 자신의 캐시 데이터를 최대한 유지할 수 있는 기법을 소개하고자 한다.

유효(valid)	*IR* 메시지를 통해 갱신되지 않았음을 확인한 데이터
불분명(unknown)	접속 단절로 인해 *IR* 메시지를 받지 못해 갱신 여부를 확인할 수 없는 데이터
스테일(stale)	*IR* 메시지를 통해 갱신됐음을 확인한 데이터

표 1 *IR* 메시지 확인에 따른 데이터 분류

3.2.1 적응적 장기간 접속단절 대처 기법

위와 같은 시스템 모델에서 장기간 접속 단절이 발생하는 클라이언트의 수가 증가하여 업링크 채널을 통한 클라이언트의 쿼리 요청이 빈번하게 발생하게 되면 다음과 같은 문제점이 발생할 수 있다.

1) 업링크 채널할당에 따른 대역폭 문제

2) 쿼리 요청에 따른 서버의 오버헤드

3) 클라이언트의 수가 증가할 경우의 확장성 문제

위와 같은 문제점을 극복하기 위해 다음과 같은 방식을 제안한다.

장기간 접속 단절된 클라이언트가 재접속 이후 서버에게서 받은 *IR* 정보를 통해 자신이 w 크기 이상으로 단절되었음을 확인하면, 기존 방식에서는 클라이언트가 가지고 있는 모든 캐시 데이터를 버렸으나, 본 논문에서는 쿼리 요청 시 업링크 채널을 통해 자신의 접속 단절 시간 정보(D_T)를 함께 첨부하여 보내준다. 서버는 일정주기 동안 클라이언트가 보내준 단절시간 정보를 통해 w와 L 그리고 D_T와의 관계를 고려하여 다음 번 브로드캐스트 데이터의 w를 결정함으로써 다음 주기에서의 클라이언트 캐시 손실을 최소화할 수 있다.

클라이언트는 다음과 같은 알고리즘을 실행한다.

클라이언트가 w 주기 이상 *IR*을 놓쳤다면 다음 작업을 수행한다.

- 먼저 서버에게 자신의 접속 단절 시간 정보를 보낸다..

- 자신이 유지하고 있는 모든 캐시 데이터를 불분명 상태(M 주기동

안)로 전환한다.

- *IR*을 받은 클라이언트는 다음과 같이 두 가지로 분류된다.

  첫 째: 조정된 w 크기로 인해 놓친 *IR* 정보를 다시 얻는 경우

  둘 째: 조정된 w 크기에도 놓친 *IR* 정보가 포함되지 않는 경우

- 첫 번째 경우는 불분명 데이터 중 갱신되지 않은 데이터는 유효로 전환하고 갱신된 데이터는 캐시에서 버린다. 두 번째 경우는 모든 캐시 데이터를 버린다.

3.2.2 캐시 소생 컴포넌트를 사용한 기법

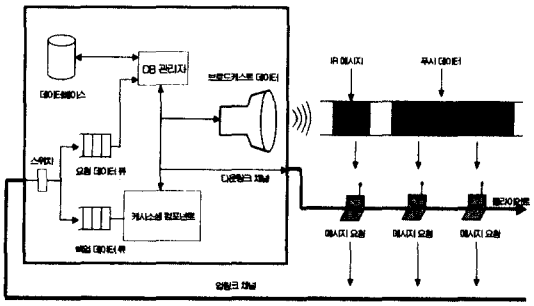


그림 2 혼합 데이터 전달 방식 모델

앞서 장기간 접속 단절 대처기법에 의해 클라이언트의 평균 단절 시간이 길어질 경우 w와 L을 조정하여 클라이언트의 캐시 손실에 대한 문제점을 극복하는 것을 보았다. 그러나 이 경우 무리하게 w와 L을 늘림으로써 다음과 같은 문제점이 발생 할 수 있다.

1) w가 너무 커질 경우: 브로드캐스트 해줄 데이터 양이 증가하여 불필요한 정보를 받게되는 클라이언트가 증가할 수 있다.

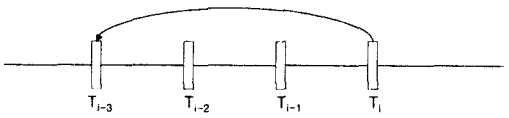
2) L이 너무 커질 경우: 브로드캐스트 데이터를 기다리는 클라이언트의 대기 시간이 무리하게 길어질 수 있다.

따라서 이 경우 N을 캐시소생 컴포넌트가 유지하는 이전 주기 범위라 할 때 캐시 소생 컴포넌트에 w*N주기의 데이터를 유지하고 있다가 요청이 들어올 경우 해당 클라이언트에게 전달해주는 방식을 취한다. 즉 장기간의 접속 단절로 인해 자신의 캐시에 유지하고 있는 데이터의 유효성 여부를 확인할 수 없는 클라이언트는 캐시 소생 컴포넌트의 정보를 사용함으로써 기존의 *IR*을 놓친 클라이언트가 자신의 캐시 데이터를 모두 버려야하는 문제점을 극복할 수 있으며 그에 따른 클라이언트의 업링크 시도로 인한 서버의 작업 부하를 최소로 줄여줄 수 있다. 또한 백업 데이터는 그림 3의 (b)와 같이 *IR* 주기마다 갱신된 데이터 아이템의 이름만으로 구성하여 전달하게 될 전체 메시지의 크기를 줄여줌으로써 캐시 소생 컴포넌트 사용에 따른 부가적인 메시지 부담을 줄여주었다. 그림 2와 같이 데이터와 백업(B) 정보를 얻기 위한 클라이언트의 쿼리 요청은 업링크 채널을 통해 서버에게 전달되며 전달된 정보는 스위치를 통해 각각의 큐로 보내진 뒤 해당 정보를 다운링크 채널을 통해 얻게된다.

서버 프로토콜

- 캐시소생 컴포넌트에는 다음과 같은 특성을 갖는 백업 데이터로 구성된다

$$T_{i-n} \le B \le T_i$$



(a) B 데이터의 N 값이 3인 경우

N 값	갱신된 데이터 아이템 이름									
0 (T_i)	a	d	f	g	h	j	k	o	y	x
1 (T_{i-1})	a	d	g							
2 (T_{i-2})	k	n	p	s	t	y				
3 (T_{i-3})	v									

(b) N 값과 데이터의 갱신 목록 정보

그림 3 캐시 소생 컴포넌트 구성 요소

- 푸시 채널을 통해 전달되는 w 주기의 데이터의 N 크기의 백업 데이터를 캐시소생 컴포넌트에 유지한다. 그림 3의 (a)는 T_i를 현재 브로드캐스트 되는 *IR* 시간이라 할 때 T_{i-3} 주기의 백업 데이터를 유지하고 있는 서버를 보여주고 있다.

- 백업 데이터는 각각의 *IR* 주기에 갱신된 데이터 아이템 이름으로 구성된다.

- 캐시 소생 컴포넌트의 데이터는 현재 서버의 브로드캐스트 데이터 목록이 전송된 뒤 갱신된다.

4. 결론

앞서 서론 부분에서 언급한 바와 같이 본 논문은 모빌 네트워크 환경에서 기존 캐시 유지 기법들이 갖는 단점인 장기간 접속 단절된 클라이언트의 캐시 손실을 최대한 막는데 그 초점을 맞추었다. 즉 적응적 장기간 접속 단절 대처 기법을 통해 w 주기를 넘어 접속 단절된 클라이언트의 수를 줄여줌으로써 자신의 캐시 데이터가 유효함에도 버려야하는 문제점을 최소화하였다. 또한 캐시 소생 컴포넌트 기법을 사용함으로써 개별적인 클라이언트에 대한 캐시 유효성을 보장해줌으로써 잦은 접속단절과 같은 열악한 무선환경에서의 클라이언트 캐시를 최대한 살리고자 하였다. 따라서 캐시를 버려야하는 클라이언트의 숫자를 최소화하고 개별적인 클라이언트의 캐시 유효성을 증명해줌으로써 캐시 손실로 인한 업링크 발생을 최소화하고 그에 따른 서버의 작업 부하를 줄여 줄 수 있다.

참고문헌

[1] Jin. Jing, A. Elmagarmid, A.S. Helal and R. Alonso. "Bit-Sequence: An Adaptive Cache Invalidation Method in Mobile Client/Server Environments," *ACM/Baltzer Mobile Networks and Application*, Vol. 2, No. 3, pp. 115-127, 1997.

[2] Guohong Cao, "Scalable Low-latency Cache Invalidation Strategy for Mobile Environment," *International conference on Mobile computing and networking*, pp. 200 - 209, 2000.

[3] Swarup Acharya and Michael Franklin, "Broadcast Disks: Data Management for asymmetric communication environments," *ACM SIGMOD*, pp. 199-210, 1995.

[4] Daniel Barbara, "Sleepers and Workaholics: Cashing Strategies in Mobile Environments," *ACM SIGMOD international conference on Management of data*, pp. 1994.

[5] Qinglong Hu and Dik Lun Lee "Adaptive Cache Invalidation Method in Mobile Environments," *IEEE International Symposium on High Performance Distributed Computing*, pp. 264-273, 1997.