혼합 데이터 방송 시스템에서 방송 알고리즘의 특성을 고려한 캐싱 전략

신 동 천†

요 약

낮은 대역폭이나 잦은 단절 같은 무선 통신의 근본적인 제약으로 인한 방송 시스템의 성능 저하를 완화할 수 있는 방법 중의 하나는 클라이언트에 캐시를 도입하는 것이다. 본 논문에서는 비트 벡터를 이용한 혼합 데이터 방송 시스템에서 클라이언트가 최근에 요구하는 데이터를 효과적으로 방송할 수 있도록 pull 기반 방송전략을 제안한다. 그리고, 방송 알고리즘의 특성을 고려한 효율적인 캐싱 전략을 제안하고 시스템의 성능을 평가한다. 평가 결과에 따르면, 제안한 전략들을 수용한 시스템이 응답 시간 측면에서 전반적으로 좋은 성능을 보여준다.

A Caching Strategy Considering Characteristics of Broadcast Algorithm in Hybrid-based Data Broadcast Systems

Dong Cheon Shin†

ABSTRACT

To introduce the cache in a client is one of the methods to migrate the performance degradation of broadcast systems due to the inherent restrictions of wireless communication environments such as low bandwidth or frequent disconnections. In this paper, we propose a pull-based broadcast strategy in hybrid-based data broadcast systems using bit vectors in order to effectively broadcast data recently requested by clients. Then, we propose a caching strategy considering the characteristics of data broadcast algorithm and then evaluate the performance of the system. According to the result of evaluation, the system employing the proposed strategies shows the better performance in terms of response time.

키워드 : 데이터 방송(Data Broadcast), 캐싱 전략(Caching Strategy), 이동 컴퓨팅(Mobile Computing)

1. 서 론

최근 들어, 무선통신 기술의 발달로 인하여 이동 컴퓨팅이라는 새로운 컴퓨팅 패러다임에 대한 관심이 고조되고 있다. 이동 컴퓨팅 환경의 특징은 크게 무선과 이동성으로 나누어 볼 수 있다[1, 2]. 기존의 유선망과 비교하여 무선망은 상대적으로 낮은 대역폭(bandwidth), 잦은 단절 등 여러 가지 환경적인 제약요인이 있다. 이러한 이동 컴퓨팅 환경의 특징 때문에 기존의 컴퓨팅 환경에서 제안된 여러 가지 기법이나 알고리즘들은 적용될 수 없거나 낮은 성능을 보이게 된다. 예를 들어, 캐시 관리, 질의처리, 트랜잭션 처리 문제에 무선과 이동성이라는 요인을 고려한 기법들이 효율적이게 된다[3].

제한된 대역폭에서 다수의 클라이언트에게 데이터를 전송하는 방법으로 데이터 방송(data broadcast) 기법이 관심을 끌어 왔다. 데이터 방송 시스템은 방송되는 데이터 항목(페이지)의 선정과 관련하여 크게 3가지 형태로 구분 할 수 있다. 서버가 클라이언트의 데이터에 대한 요청 없이 자체 방송 알고리즘에 따라 필요한 데이터를 주기적으로 혹은 비주기적으로 방송하는 push 기반의 방송 형태가 있다[4, 5]. 반면에 필요한 데이터를 클라이언트가 서버에 요청하여 서버가 데이터를 방송해 주는 pull 기반의 형태가 있다[6]. 그리고 push와 pull 기반을 혼합하여 방송하는 혼합(hybrid) 기반의 형태가 있다[7, 8].

지금까지 제안된 대부분의 push 기반 및 혼합 형태의 방송 알고리즘들은 클라이언트의 데이터 요구에 대한 변화를 동적으로 반영하여 방송 프로그램을 생성하지 못하였다. [8]에서는 비트 벡터를 이용하여 클라이언트의 요구 변화를 동적으로 반영하여 방송하는 혼합형태 기반의 방송 알고리즘을 제안하였다. 그러나 최근에 요청된 데이터와 과거에 요청된 데이터를 차별화하지 않아 과거에 클라이언트들이 자주 액세스 했던 데이터들이 여전히 방송될 데이터에 포함되는 비효율성을 초래할 수 있게 된다. 즉, 최근에 액세스가

※ 이 논문은 2003학년도 중앙대학교 학술연구비 지원에 의한 것임.
† 종신회원 : 중앙대학교 정보시스템학과 교수
논문접수 : 2003년 4월 28일, 심사완료 : 2005년 1월 10일

증가하고 있는 데이터에 대한 클라이언트의 요구에 효율적으로 대처할 수 없게 된다.

한편, 데이터 방송 환경에서는 낮은 대역폭으로 인하여 클라이언트에 캐시를 두어 필요한 데이터가 방송되기를 기다리지 않고 캐시를 액세스하여 효율성을 높일 수 있다. 캐시에 있는 데이터는 서버에 있는 데이터의 사본으로 간주될 수 있으므로 일관성 유지 문제가 대두되고 이러한 일관성 문제를 해결하고자 하는 여러 노력이 있어 왔다[9, 10, 11]. 캐시를 고려하는 방법에서 해결해야 될 또 다른 중요한 문제는 캐싱 전략, 즉 캐시내의 데이터 페이지를 새로운 페이지와 대체시키는 대체 전략이다.

전통적인 시스템 환경을 위한 여러 가지 대체 전략들이 제안되었다[5, 12]. 그러나, 이러한 대체 전략들은 이동 컴퓨팅 환경의 특징을 고려하지 않은 전략들이기 때문에 좋은 성능을 보이기 어렵다. 따라서, 이동 컴퓨팅 시스템에서 데이터 방송 환경을 고려한 많은 대체 전략들의 대부분은 캐시의 히트율을 높이는 것뿐만 아니라 캐시 미스가 발생하는 경우 서버로부터 요구하는 데이터가 방송될 때까지 기다려야 하므로 미스에 따른 비용까지 고려해야 한다는 관점에서 제안되었다 <표 1>.

〈표 1〉 캐싱 전략들의 고려 사항

환경	전략	고려 사항	
		히트율	미스 비용
기존	LFU[12]	o	x
	LRU[12]	o	x
	LRU-k[13]	o	x
데이터 방송	PIX[4]	o	o
	PT[4]	o	o
	Gray[5]	o	o

데이터 방송 시스템에서 궁극적으로 캐시 전략은 방송 알고리즘과 긴밀한 상호 의존성을 갖는다. 예를 들어, 자주 방송되는 데이터는 상대적으로 드물게 방송되는 데이터보다 캐시에 존재할 가치가 떨어지며 캐시에 존재하지 않는 데이터는 존재하는 데이터보다 상대적으로 자주 방송해야 대기 시간을 줄여서 미스 비용을 최소화할 수 있다. 그러나, 지금까지 제안된 캐시 전략의 대부분은 방송 알고리즘과 독립적으로 제안되었거나 방송 알고리즘의 기본 원칙을 효율적으로 반영하지 못하였다고 할 수 있다.

본 논문에서는 비트 벡터를 사용하여 모든 클라이언트의 요구패턴을 반영하고 동시에 클라이언트의 데이터에 대한 요구의 최신성에 비중을 두어 방송된 데이터 중에서 방송 효과가 큰 데이터를 지속적으로 방송할 수 있도록 혼합 데이터 방송 시스템에서 pull 방송을 위한 전략을 제안한다. 그리고, 방송 알고리즘의 기본 특성을 고려한 새로운 캐싱 전략을 제안하고 전통적인 LRU 전략을 사용하는 경우와의 성능 평가를 통해 방송 알고리즘의 특성을 고려한 효과를 평가한다.

본 논문의 구성은 다음과 같다. 2장에서는 데이터 방송 전략에 대해 기술하고 3장에서는 방송 알고리즘의 특성을 고려하여 새로운 캐싱 전략을 제시한다. 4장에서는 성능 평가에 대해 논의한다. 끝으로, 5장에서는 결론을 맺는다.

2. 데이터 방송 전략

이 절에서는 [8]에서 제안한 pull 데이터 방송 전략과 달리 클라이언트가 최근에 요청한 데이터일수록 더 많이 방송되도록 하여 최근에 증가하고 있는 클라이언트의 데이터 요구에 효율적으로 대처할 수 있는 방송 전략을 제시한다.

논의하는 데이터 방송 시스템은 혼합형태의 시스템이다. 즉, push 형태로 방송되는 데이터와 pull 형태로 방송되는 데이터 사이에 대역폭을 효율적으로 할당하여 두 형태의 방송 데이터를 모두 지원한다. 따라서, 클라이언트의 데이터 요청은 큐에 저장되어 FIFO 형식으로 일반 방송 프로그램의 데이터와 할당된 대역폭 범위 내에서 번갈아 가며 방송된다. 큐가 만원일 때 클라이언트의 요청은 거절되며 이미 큐에 있는 데이터에 대한 요청은 앞서 요청된 데이터 방송으로 만족될 수 있으므로 큐에 저장되지 않는다.

각 방송 주기 초에 방송 스케줄 즉, 데이터 식별자와 방송 주기 버전번호를 먼저 방송한다. 각 클라이언트는 수신한 방송 스케줄 정보를 토대로 필요한 데이터가 캐시에 없는 경우 이번 방송 주기에 방송되면 기다리고 그렇지 않으면 서버에게 해당 데이터를 요청한다.

방송된 데이터에 대한 클라이언트의 액세스 정보를 얻기 위해 각 클라이언트는 비트 벡터를 유지한다. 비트 벡터의 각 비트는 현재 방송 프로그램의 각 데이터 페이지를 나타낸다. 새로운 방송 프로그램을 수신하면 클라이언트는 버전번호를 갱신하고 비트 벡터를 모두 '0'으로 클리어 한다. 요구한 데이터가 방송되는 데이터에 있으면(air hit) 해당 데이터 비트는 '1'로 변경된다. 캐시에 있는 경우나 요청하여 방송된 데이터인 경우에는 비트 벡터에 영향을 주지 않는다. 필요한 데이터가 캐시나 방송 프로그램에 없는 경우 클라이언트는 서버에게 데이터 요청을 한다. 이때, 비트 벡터도 함께 피기배킹(piggybacking)된다. 즉, 데이터 요청 메시지는 데이터 식별자, 주기 버전번호, 비트 벡터로 구성된다.

서버는 각 데이터에 대해 MFA(most frequently accessed) 값을 유지한다. 서버는 버전번호를 확인하고 데이터가 클라이언트에 의해 요청되었거나 데이터의 해당 비트 벡터가 '1'인 경우 해당 데이터의 MFA 값을 '1' 증가시킨다. MFA 값이 높은 데이터들을 방송 디스크 기법[8]에 따라 방송하게 된다. 이렇게 함으로써 클라이언트의 데이터 액세스 정보를 다음 방송 프로그램 생성에 반영한다.

그러나, MFA 값을 단순하게 증가시켜 높은 값을 갖는 데이터들을 방송하게 되면 이전에 클라이언트들에게 인기가 높았지만 현재는 인기가 떨어진 데이터들도 당분간은 높은

값을 유지할 수 있어 방송될 가능성이 높아진다. 특히, 새로이 인기가 올라가는 데이터에 대한 방송이 당분간 지연되거나 방송 빈도수가 저하될 가능성이 있다. 이러한 문제를 줄이기 위해 다음과 같은 고찰을 할 수 있다.

비트 벡터에 '0'을 갖는 데이터는 클라이언트의 캐시에 있거나 불필요한 데이터임을 의미한다. 캐시에 있는 데이터는 점차 상대적으로 MFA 값이 낮아지게 되지만 캐시에 있으므로(캐시 대체 전략이 이상적이라면) 방송여부나 방송 빈도수의 변화가 문제되지 않는다. 따라서, 불필요한 데이터라면 MFA 값을 상대적으로 빠르게 낮추어 줄 필요가 있다. 한편, 비트 벡터에 '1'을 갖는 데이터는 향후에 자주 요구될 가능성이 있는 데이터이므로 방송하거나 혹은 방송 빈도수를 더 늘려 줄 필요가 있다. 그러므로, 최근에 요구되는 데이터가 그렇지 않은 데이터에 비해 상대적으로 자주 방송되는 것을 가속화하기 위하여 비트 벡터가 '1'인 데이터만 고려하지 않고 '0'인 데이터도 다음과 같이 고려하여 MFA 값을 유지한다.

(R1) 현재 방송 주기동안 수신한 각 비트 벡터에서 각 데이터별로 비트가 '1'인 개수의 합에서 '0'인 개수의 합을 뺀 값을 해당 데이터의 MFA 값에 더한다.

(R2) 클라이언트로부터 요청된 데이터에 대해 데이터별로 요청된 횟수의 합을 해당 데이터의 MFA 값에 더한다.

따라서 최근에 air 히트가 많은 데이터(비트가 '1'인 데이터) 즉, 방송 효과가 큰 데이터일수록 높은 MFA 값을 갖게 되며 그 반대일수록 상대적으로 더욱 낮은 MFA 값을 갖게 되어 방송 효과가 낮은 데이터의 방송 중단이나 방송 빈도수를 빠르게 낮출 수 있게 된다. <표 2>로부터 인기가 낮아지고 있는 데이터에 대해 MFA 값을 유지하는 전략의 차이를 쉽게 알 수 있다.

〈표 2〉 방송 전략의 MFA 유지 예시

데이터	MFA	1차 방송 주기		2차 방송 주기		3차 방송 주기	
		[8]의 전략	제안한 전략	[8]의 전략	제안한 전략	[8]의 전략	제안한 전략
a	8	9	9	10	10	11	11
b	6	7	7	8	8	9	9
c	4	4	3	4	2	4	1
d	1	2	2	3	3	4	4

<표 2>에서 데이터 a와 b는 계속 인기를 유지하고 있으며 c는 인기가 낮아지고 있는 반면에 d는 새로이 인기가 높아지고 있음을 알 수 있다. 한번에 방송될 방송 프로그램의 크기는 3 즉, 3개의 데이터만 방송한다고 하면 [8]의 전략에서는 3차 방송 이후에나 d가 방송될 수 있으나 제안한 전략에서는 2차 방송에서 d가 방송될 수 있다. 즉, 제안한 전략이 새로이 인기가 높아지는 데이터에 대한 방송을 더 이르게 할 수 있음을 알 수 있다.

3. 캐싱 전략

데이터 방송 환경에서는 캐시 미스가 발생할 경우 전통적인 시스템 환경과는 달리 상대적으로 낮은 대역폭을 통하여 필요한 데이터를 서버로부터 제공받기 때문에 방송 알고리즘에 따라 미스에 따른 비용이 커지며 아울러 그 비용을 예측하기 어렵게 된다. 방송 프로그램의 내용을 완전히 아는 경우에는 미스 비용을 최소화할 수 있음은 당연한 사실이다. 그러나 클라이언트의 데이터 요구 변화를 반영하는 동적인 방송 환경에서는 방송 주기마다 방송 프로그램이 다를 수 있으므로 방송 프로그램을 생성하는 알고리즘의 기본 원칙을 고려하여 대체 전략을 수립하여야 미스 비용을 가능한 최소화 할 수 있게 된다. 따라서, 제안하는 효율적인 캐시 대체 전략은 히트율을 높이고 미스 비용을 최소화하기 위해 다음과 같은 직관을 기반으로 한다.

• 히트율을 높이기 위한 직관

H1) 클라이언트가 자주 요구하는 데이터 페이지는 캐시에 있어야 한다.

클라이언트가 필요로 하는 데이터는 캐시에 있거나(cache hit) 방송중이거나(air hit), 혹은 서버에게 요청을 하는(pull request) 3 가지이다. air 히트인 경우 원하는 데이터가 방송되기를 기다려야 하며 요청하는 경우도 최소한 다음 방송까지 기다려야 한다. 따라서, 자주 요구되는 데이터는 캐시에 있어 히트율을 높임으로써 가장 빠른 응답시간을 얻을 수 있다.

H2) 클라이언트가 최근에 요구한 페이지일수록 캐시에 있어야 한다.

전통적인 시스템에서 가장 많이 고려한 것으로 프로그램의 속성상 참조 지역성에 따라 최근에 액세스된 데이터가 향후에도 많이 액세스될 것이라는 직관을 기반으로 하고 있다. 따라서, 이러한 데이터는 캐시에 있게 함으로써 히트율을 높일 수 있게 된다.

• 미스비용을 최소화하기 위한 직관

M1) 캐시에 오래 존재한 데이터일수록 캐시에 있어야 한다.

방송 알고리즘에 따르면 캐시에 존재하는 데이터가 방송되는 경우에 비트 벡터의 해당 데이터 비트는 '0'으로 서버에게 전달된다. 따라서, 서버가 이 데이터를 방송할 가능성이 줄거나 자주 방송하지 않을 가능성이 증가하게 되어 미스가 발생하는 경우 클라이언트의 대기 시간이 길어지게 되어 미스비용이 증가하게 되므로 이러한 데이터는 캐시에 존재하여야 대기 시간을 줄일 수 있게 된다.

M2) 방송이 적게 되는 페이지일수록 캐시에 있어야 한다.

방송 알고리즘에 따르면 캐시에 데이터가 오래 있을수록 MFA 값이 서버측에서 감소되어 방송의 횟수가 줄어들게 된다. 더욱이 방송 횟수는 자신뿐만 아니라 다른 클라이언트들의 액세스 요청 상황에 따라 영향을 받게 된다. 방송의 횟수가 줄어들게 되면 클라이언트의 대기시간이 길어지게 되어 미스 비용이 증가하게 된다. 따라서, 방송 횟수가 적은 페이지일수록 미스 비용을 줄이기 위해 캐시에 존재하는 것이 바람직하다.

H1)을 반영하기 위해 각 데이터마다 클라이언트가 요청한 횟수(R_n)를 유지한다. 클라이언트가 데이터를 요구할 때마다 해당 데이터의 R_n 값을 1씩 증가시킨다. R_n 값은 초기에 0으로 설정된다.

H2)를 반영하기 위해 데이터 방송 주기 초에 서버로부터 받은 버전번호와 방송 데이터 식별자 정보를 이용하여 각 데이터마다 액세스 요구가 충족되는 시점의 방송 버전번호(R_v)를 유지한다. 클라이언트가 데이터에 대한 액세스 요구를 할 때 해당 데이터의 R_v 값은 요구한 데이터가 서비스되는 시점의 방송 버전번호로 설정된다. 클라이언트가 서버에게 요청하여 방송되는 데이터의 R_v 값은 요구한 시점의 버전보다 이후 버전번호를 갖게 됨을 주목하라.

M1)을 반영하기 위해 데이터 방송 주기 초에 서버로부터 받은 버전번호와 방송 데이터 식별자 정보를 이용하여 각 데이터마다 캐시에 존재하기 시작한 시점의 방송 버전번호(F_v)를 유지한다. 데이터가 캐시에 존재하는 동안 F_v는 변하지 않으며 캐시에서 희생자로 선택되어 더 이상 캐시에 존재하지 않게 되면 기본값 1로 다시 초기화된다. 또한 최근에 요구되지 않았으면서 상대적으로 오랜 동안 캐시에 있는 데이터 페이지가 같은 기간동안 캐시에 존재하면서 상대적으로 최근에 요구된 데이터 페이지 보다 캐시에 존재하게 되는 경우를 방지하기 위해 최근에 요구한 시점도 같이 고려한다. 즉, 캐시에 존재한 기간이 동일하여도 최근에 참조되었으면 캐시에 존재할 가치가 더 있도록 한다. 이를 위해 F_v만 단독으로 고려하지 않고 R_v를 함께 고려하여 R_v에 대한 존재 기간을($F_v \div R_v$) 반영한다.

M2)를 반영하기 위해 데이터 방송 주기 초에 서버로부터 받은 버전번호와 방송 데이터 식별자 정보를 이용하여 각 데이터마다 방송된 횟수(B_f)를 유지한다. <표 3>은 제안한 전략이 고려하는 내용을 보여 주고 있다.

〈표 3〉 제안한 전략의 고려 요소

목적	고려 요소
히트율 향상	• 요청 횟수(Rn) • 방송버전번호(Rv)
미스 비용 감소	• 캐시존재 시작시 방송버전번호(Fv) • 방송 횟수(Bf)

이상으로부터 캐시 대체를 위한 전략은 식 (1) 로부터 유도된 값(Value for Victim: VV)이 제일 작은 데이터 페이지를 희생자로 선택한다.

$$VV = R_n \cdot R_v \div (F_v \div R_v) \div B_f = \mathbf{R_n \cdot R_v^2 / (F_v \cdot B_f)} \quad (1)$$

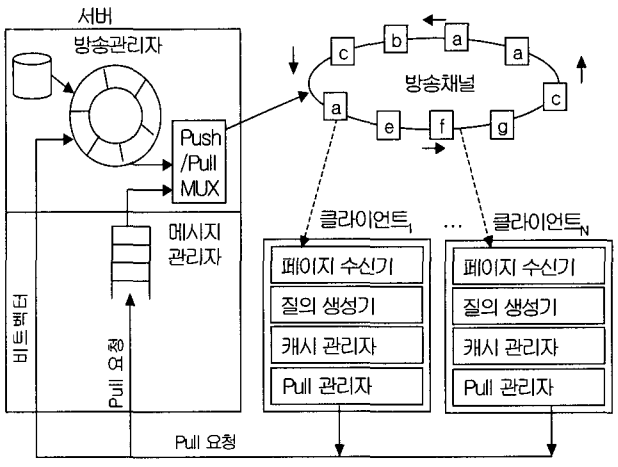
식 (1)에서 요구 회수가 많을수록, 최근에 요구할수록, 그리고 캐시에 오래 머무르고 방송횟수가 적을수록 희생자로 선택될 가능성이 줄어들게 됨을 쉽게 알 수 있다.

한편, 클라이언트가 요청하여 방송된 데이터 페이지는 비록 다른 클라이언트들도 요구하여 점차로 방송될 가능성이 높아질 수 도 있지만 필요성이 자신 혹은 소수 클라이언트에 국한될 가능성이 있어 다시 방송될 가능성이 줄어들 수 있으므로 장기 인기도를 반영하기 위해서 무조건 캐시에 넣는 전략을 취한다.

4. 성능 평가

4.1 성능 평가 환경

개선한 방송 알고리즘과 제안한 캐싱 전략을 사용하는 데이터 방송 시스템(이후 Adaptive Bit Vector, 약칭 ABV라고 함)의 성능을 평가하기 위하여 역시 비트 벡터를 사용하는 기존에 제안된 시스템[8](이후 Bit Vector, 약칭 BV라고 함)과 push 기반의 방송 디스크 기법[4](이후 BroadCast disk, 약칭 BC라고 함)을 시뮬레이션을 통하여 비교하였다. 시뮬레이션 모델은 [8]에서 제시된 것을 사용하여 ((그림 1) 참조) CSIM[14]을 이용하여 구현하였다. 다만, [8]의 시뮬레이션 모델에서 클라이언트가 자신이 필요로 하는 페이지를 얻을 때까지 방송채널을 감시하기 위해서 시스템 전체에 하나의 페이지 수신기 모듈을 이용하고 있지만, 본 연구에서는 각 클라이언트마다 하나의 페이지 수신기 모듈을 갖도록 수정함으로써 [8]의 연구에 비해 실세계의 작업부하를 더 현실적으로 구현하고자 노력하였다.



(그림 1) 시뮬레이션 모델

각 클라이언트는 페이지 수신기, 질의 생성기, 캐시 관리자, Pull 관리자의 4개의 프로세스로 구성된다. 먼저 질의 생성기에 의해 하나의 읽기 연산이 제기되면 캐시 관리자를 통해 해당 데이터가 캐시에 존재하는지 확인하게 된다. 만일 캐시에서 해당 데이터를 찾을 수 없을 경우에는 현재 방송 프로그램에 해당 데이터가 포함되어 있는지 확인하게 되고 여기에도 포함되어 있지 않을 경우에 서버에 pull 요청을 제기하게 된다. 이때 해당 클라이언트에 있는 비트 벡터도 함께 보내짐을 주목하라. 이렇게 생성된 pull 요청과 비트 벡터는 본 논문에서 제안된 기법에 따라 적절히 스케쥴링되어 방송되게 되고 자신이 필요로 하는 데이터를 획득한 클라이언트는 또 새로운 데이터 읽기 연산을 제기할 준비를 하게 된다. 각 클라이언트는 하나의 페이지를 방송 또는 요청을 통해 받을 때까지 새로운 페이지에 대한 연산을 제기하지 않기 때문에 시뮬레이션 중 작업의 부하는 클라이언트의 수로 일정하게 유지될 수 있으므로 폐쇄된 큐잉 모형(closed queueing model)을 채택한 것으로 볼 수 있다. <표 4>는 시뮬레이션 모델과 관련된 시스템 및 응용 매개변수를 보여준다.

〈표 4〉 시스템 및 응용 매개변수

매개변수	설명	설정값
db_size	데이터 페이지의 수	3000으로 고정
num_clients	클라이언트의 수	90으로 고정
cache_size	클라이언트 캐시의 크기	db_size의 0.5%
queue_size	클라이언트 요청들의 대기 큐의 크기	num_clients의 10%
think_time	클라이언트의 각 요청 간 대기시간	10초
down_time	서버가 한 페이지를 방송하면 클라이언트에 도달될 때까지 소요되는 시간	0.2초
upload_time	클라이언트가 페이지를 요청하는데 소요되는 시간	0.01초
cache_acc_time	캐시에 있는 한 페이지를 액세스하는데 소요되는 시간	0.001초
bc_rate	방송 및 요청 중 방송에 할당된 채널의 비율	40%~90%, 10%씩 증가
num_bcdisk	방송 디스크의 수. 전체 데이터 페이지를 num_bcdisk 개의 그룹으로 나눔	3으로 고정
$bcdisk\_rate_i$	i번째 방송 디스크의 방송 비율. 이 비율의 합이 100이어야 함	0~100사이의 값
$bcdisk\_acc\_rate_i$	클라이언트가 i번째 방송 디스크에 있는 페이지를 액세스할 비율. 이 비율의 합이 100이어야 함	0~100사이의 값

데이터베이스의 데이터 항목의 수를 실제 이동 응용에서 이용되는 크기에 비해 적은 3000으로 설정하였는데 이는 상대적으로 적은 수의 클라이언트 간에 적정수준의 방송 히트율을 확보하기 위한 것이다. 반면에 캐시의 영향을 조금 더 세밀하게 관찰하기 위하여 클라이언트 캐시의 크기는 전체 데이터 항목의 0.5%로 비교적 큰 값을 선택하였다. 클라이언트는 자신이 필요로 하는 데이터가 차기 방송프로그램에 포함되어 있지 않을 경우 서버에 해당 데이터를 직접 요청하여 얻게 되는데 이러한 요청이 너무 많을 경우 더 이상 요청을 받을 수 없도록 하기 위해서 서버의 요청대기 큐의 크기를 클라이언트 수의 10%로 제한하였다. 클라이언트가 제기한 읽기연산에 대해 방송 혹은 요청에 의해 해당 데이터를 얻은 후에 일정한 시간이 경과하는 동안 새로운 연산을 제기하지 않도록 하기 위해서 think_time을 10초로 설정하였다. 서버가 방송 혹은 요청된 데이터를 클라이언트로 전송하면 각 클라이언트는 적어도 down_time 만큼의 시간, 즉 0.2초가 지나야 해당 데이터를 읽을 수 있으나 상대적으로 크기가 작은 데이터 요청을 서버로 전송하기 위해서는 upload_time 만큼의 시간, 즉 0.01초가 필요하다. 클라이언트와 서버간의 업로드 및 다운로드에 필요한 시간에 비하면 작은 값이지만 캐시로부터 데이터를 읽는데 소요되는 시간도 캐시 히트율의 차이를 정확히 모델링하기 위해 필요하다. 왜냐하면, 캐시 히트율이 상대적으로 높은 기법은 캐시 액세스 수가 늘어나는 반면 서버의 방송 혹은 요청 횟수가 줄어들지만 그 반대의 경우도 있을 수 있기 때문이다. 이러한 비용을 적절히 반영하기 위해서 [8]에서와는 달리 캐시로부터 하나의 데이터 페이지를 액세스하는 시간, 즉 cache_acc_time을 0.001초로 설정하였다.

한편, 다운로드 채널을 방송에 할당하는 비율에 따라 성능에 커다란 영향을 줄 수 있으므로 방송비율, 즉 bc_rate를 40%에서부터 10%씩 증가시키면서 실험하였다. 그리고 모든 데이터를 num_bcdisk개의 논리적인 방송디스크로 나누고 각 디스크에 있는 데이터의 방송빈도를 지정하기 위해서 $bcdisk\_rate_1$, $bcdisk\_rate_2$, $bcdisk_rate_3$를 도입하였으며 당연히 3개의 합은 100이 된다. 또한, 클라이언트 요청의 집중성(skewness)을 구현하기 위해 각 방송 디스크별 액세스 비율인 $bcdisk\_acc\_rate_1$, $bcdisk\_acc\_rate_2$, $bcdisk_acc_rate_3$를 도입하였으며 역시 3개의 합은 100이 된다. 예를 들어, $bcdisk_acc_rate_1$이 70이라면 각 클라이언트의 100개의 읽기 연산 중 70개는 방송디스크 1에 집중됨을 의미한다.

4.2 성능 평가 결과

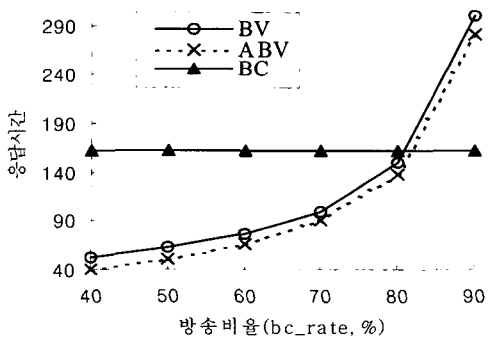
시뮬레이션 결과의 정확성을 제고시키기 위해서 각 시뮬레이션의 초기단계에 수집된 자료는 결과에 반영하지 않았으며, 또한 각 시뮬레이션에 대해 난수생성기의 시드 값을 변경시키면서 수차례 반복 실험한 결과에 대한 평균을 이용하여 최종결과를 산출하였다. 성능 척도로는 평균 응답 시간을 사용하였으며 데이터 방송 환경에서 대기 시간의 고려가 필요함을 입증하기 위해 캐시 히트율에 대한 결과도 도

출하였다. 한편, 기본적으로 혼합 기반의 데이터 방송 시스템에서는 push와 pull의 대역폭 할당과 방송 디스크간의 액세스율 등이 전체적으로 성능에 커다란 영향을 줄 수 있는 주요 매개변수들이므로 이들 변화에 대한 성능 변화를 비교한다.

4.2.1 Push와 Pull 비율의 변화에 따른 성능

혼합 기반의 데이터 방송 시스템은 정해진 방송 알고리즘에 따라 데이터를 방송하면서 동시에 대역폭의 일정 부분을 할당받아 클라이언트로부터 직접 요청된 데이터를 방송하게 된다. 따라서, 방송채널 할당 비율에 따른 성능의 변화를 보이기 위한 실험을 수행하였다.

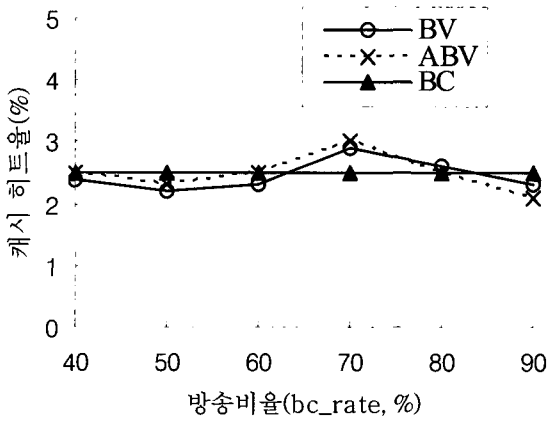
(그림 2)는 방송비율(bc_rate) 변화에 따른 응답시간의 변화를 보여주고 있다. 여기에서 방송·디스크의 방송 비율은 $bcdisk\_rate_1$ = 60%, $bcdisk\_rate_2$ = 30%, $bcdisk\_rate_3$ = 10%로 고정하였으며 클라이언트가 방송 디스크의 페이지를 액세스 할 비율도 $bcdisk\_acc\_rate_1$ = 60%, $bcdisk\_acc\_rate_2$ = 30%, $bcdisk\_acc\_rate_3$ = 10%로 고정하였다. BC는 클라이언트의 요구에 따른 방송을 하지 않으므로 일정한 응답시간을 보이고 있으나, BV와 ABV는 방송 비율에 따라 응답시간의 변화를 보이고 있다. 전반적으로 ABV가 BV보다 방송 비율에 상관없이 좋은 성능을 보이고 있음을 볼 수 있다. 이는 ABV가 클라이언트의 요구를 반영한 캐시 대체 전략을 사용한 결과로 판단된다. 그리고 BV와 ABV 가 방송 비율이 낮을수록 BC보다 좋은 성능을 보이고 있으나 방송 비율이 증가할수록 성능 차이가 줄어들다가 방송 비율 80 부근부터는 오히려 성능이 떨어지고 있음을 알 수 있다. 이는 클라이언트의 데이터 요구 패턴 변화에 능동적으로 대처할 수 있는 대역폭의 할당이 절대적으로 부족해짐에 따라 극심한 경쟁으로 인해 pull을 위해 할당된 대역폭의 효용성은 없으면서 결과적으로 push를 위한 대역폭의 감소만 초래하여 필요한 데이터를 다음 방송 시까지 기다려야 하므로 전체적으로 응답시간의 증가를 초래하는 것으로 판단할 수 있다.



(그림 2) 방송비율 변화에 따른 응답시간

(그림 3)은 방송비율 변화에 따른 캐시 히트율의 변화를 보여주고 있다. 캐시 히트율의 변화에 대해 일관성 있는 분석이 어려움을 알 수 있다. 즉, 히트율이 낮아도 응답시간은 적게 나타날 수 있으며 반대로 히트율이 높아도 응답시간은 크게 나타날 수 있음을 볼 수 있다. 이는 데이터 방송 시스템에서는 히트율뿐만 아니라 미스에 따른 대기시간이 성능에 영향을 미치고 있음을 입증해고 있다. 예를 들어, 히트율이 높아도(낮아도) 대기시간이 길어(짧아)짐으로서 전체적으로 응답시간이 길어(짧아)질 수 있음을 확인할 수 있다.

4.2.2 방송 디스크 액세스율의 변화에 따른 성능

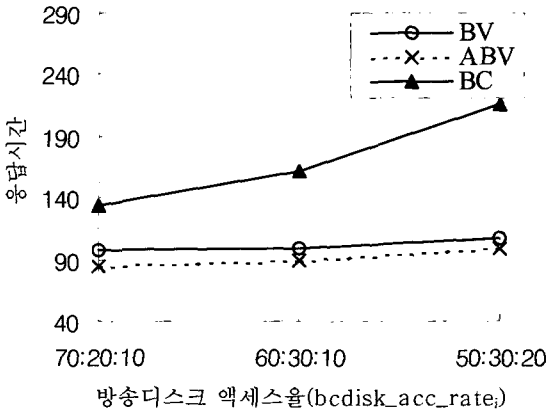


(그림 3) 방송비율 변화에 따른 캐시 히트율

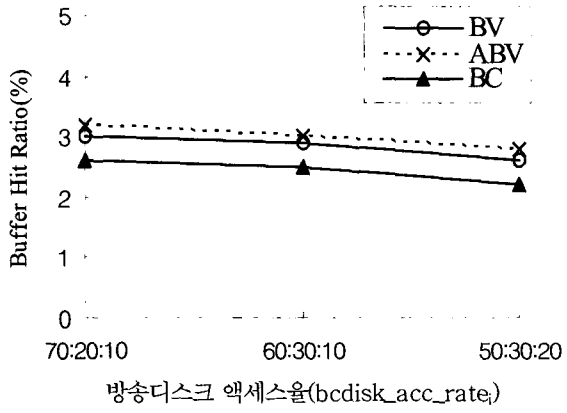
방송 디스크 기법을 도입하는 가장 큰 목적은 클라이언트의 데이터 액세스 빈도에 따라 방송 디스크간의 방송 횟수를 차별화함으로써 성능 향상을 얻고자 하는 것이다. 따라서, 방송 디스크 사이의 액세스율이 방송 디스크 사이의 방송 횟수와 유사할수록 좋은 성능을 보이게 된다.

(그림 4)는 방송 디스크 사이의 액세스율에 따른 응답시간의 변화를 보여준다. 여기에서 방송 디스크 비율은 $bcdisk\_rate_1$ = 60%, $bcdisk\_rate_2$ = 30%, $bcdisk\_rate_3$ = 10%로 고정하였고 bc_rate = 70%로 하였다. 클라이언트의 데이터 요구가 디스크 사이에 분산될수록 전체적으로 응답시간이 증가하고 있다. 이는 방송 디스크 사이의 액세스율 차별화를 통하여 성능을 향상하고자 하는 방송 디스크 기법의 기본 목적과 부합되는 결과로 보여 진다. 특히, BC는 분산될수록 응답시간의 변화폭이 커지는 반면에 BV와 ABV는 그렇지 않음을 볼 수 있으며 이는 BC의 경우 미스가 발생하는 경우 다음 방송 시까지 기다려야하므로 pull을 위한 대역폭을 사용할 수 있는 BV나 ABV보다 대기 시간이 길어진 결과로 해석할 수 있다. 한편, 전반적으로 클라이언트의 요구를 능동적으로 반영할 수 있는 BV와 ABV가 BC보다 좋은 성능을 보이고 있으며 클라이언트의 요구 패턴 변화를 복합적으로 고려하고 있는 ABV가 BV보다 다소 좋은 성능을 보이고 있다. 캐시 히트율도 (그림 5)에서 볼 수 있는 바와 같

이 데이터 요구가 분산될수록 전체적으로 떨어지고 있다.



(그림 4) 방송디스크 액세스율 변화에 따른 응답시간



(그림 5) 방송디스크 액세스율 변화에 따른 캐시히트율

5. 결 론

이동 컴퓨팅 환경에서 다수의 클라이언트에게 효과적으로 데이터를 전송하는 방법으로 데이터 방송이 주목을 받고 있다. 데이터 방송이 효율적이려면 클라이언트의 데이터 요구 사항 변화를 반영하는 것이 바람직하다. 이를 위해 본 논문에서는 최근에 요청이 증가하고 있는 데이터를 효율적으로 방송할 수 있도록 기존의 혼합 데이터 방송 시스템에서 pull을 위한 방송 전략을 수정하였다. 아울러, 방송 알고리즘의 기본적인 특성을 고려한 캐싱 전략을 제안하고 데이터 방송 시스템의 성능을 평가하였다. 제안한 캐싱 전략은 클라이언트의 데이터 액세스 빈도수와 최근성을 고려하였고 방송 알고리즘이 캐시에 있는 데이터의 방송 가능성이나 방송 빈도수를 줄이기 때문에 캐시에 존속한 기간과 방송 횟수도 함께 고려하였다.

성능 평가 결과에 따르면, 클라이언트의 요구 패턴이 변하는 환경에서는 당연히 클라이언트의 요구 변화를 능동적으로 수용할 수 있는 혼합 기반의 시스템이 우수한 성능을 나타내고 있다. 아울러, 전반적으로 클라이언트의 요구 패턴을 반영하여 제안한 전략이 응답시간 측면에서 다소 좋은 성능을 보이고 있다. 그러나 성능의 차이가 크지 않게 나타난 것은 혼합 데이터 방송 시스템에서는 push를 위한 데이터 방송 알고리즘이 시스템의 성능에 더 큰 영향을 미치고 있음을 입증하고 있다.

참 고 문 헌

[1] D. Aksoy, M. Altinel, R. Bose, U. Cetintemel, M. Franklin, J. Wang, and S. Zdonik, "Research in Data Broadcast and Dissemination", AMCP, pp.194-207, 1998.

[2] D. Barbara, "Mobile Computing and Databases-A Survey", IEEE Transactions on Knowledge Engineering, Vol.11, No.1, pp.108-117, January/February, 1999.

[3] S. K. Madria and B. K. Bhargava, "A Transaction Model to Improve Data Availability in Mobile Computing", Distributed and Parallel Databases, 10(2): pp.127-160, 2001.

[4] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast Disks : Data Management for Asymmetric Communication Environments", Proc. of ACM SIGMOD, pp.199-210, 1995.

[5] S. Khanna and V. Liberatore, "On Broadcast Disk Paging", Proc. of the 30th ACM Symp. on the Theory of Computing, pp.634-643, 1998.

[6] D. Aksoy and M. Franklin, "RxW : A Scheduling Approach for Large-Scale On-Demand Data Broadcast", IEEE/ACM Transactions on Networking Vol.7, No.6, pp.846-860, 1999.

[7] S. Acharya, M. Franklin, and S. Zdonik, "Balancing Push and Pull for Data Broadcast", Proc. of ACM SIGMOD, Tuscon, Arizona, pp.183-194, May, 1997.

[8] Q. Hu, D. L. lee, and W. C. Lee, "Dynamic Data Delivery in Wireless Communication Environment", white Paper, GTE Lab. Incorporated. 1999.

[9] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments", Proc. 6th Int. Conf. Mobile Computing and Networking, pp.200-209, 2000.

[10] A. Kahol, S. Khurana, S.K.S. Gupta and P.K. Srimani, "A Strategy to Manage Cache Consistency in a Disconnected Distributed Environment", IEEE Trans. on Parallel and Distributed Systems, 12(7), pp.686-700, July, 2001.

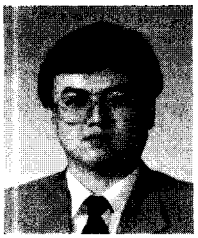
[11] X. Shao and Y. Lu, "Maintain Cache Consistency of Mobile Database Using Dynamical Periodical Broadcast

Strategy", 2003 International Conference on Machine Learning and Cybernetics, pp.2389-2393, Nov., 2003.

[12] S. Galvin and P. B. Galvin, Operation System Concepts, 4th Edition, Addison Wesley, 1994.

[13] E. O'Neil, P. O'Neil, and G. Weikum, "The LRU-K Page Replacement Algorithm for Database Disk Buffering", Proc. of ACM SIGMOD, pp.297-306, May, 1993.

[14] H. Schwetman, CSIM User's Guide for Use with CSIM Revision 16, Microelectronics and Computer Technology Corporation, 1992.



신 동 천

e-mail : dcshin@cau.ac.kr

1985년 서울대학교 컴퓨터공학과(학사)

1987년 한국과학기술원 전산학과(석사)

1991년 한국과학기술원 전산학과(박사)

1991년~1993년 한국전산원 선임연구원

1993년~현재 중앙대학교 정보시스템학과 교수

2004년~2005년 Indiana University Bloomington, School of Informatics 방문 교수

관심분야 : 이동 컴퓨팅, 데이터 웨어하우스, 데이터베이스