멀티 디스크 방송 환경을 위한 인덱스 기술

박기영*, 정성원*
*서강대학교 컴퓨터공학과
e-mail:joy710@sogang.ac.kr

An Indexing Technique for Multi-Disks Broadcast Environments

KiYoung Park*, Sungwon Jung*
*Dept of Computer Science and Engineering, Sogang University

요 약

모바일 환경에서는 상향링크와 하향링크의 대역폭이 비대칭적이며 전력이 한정되어 있기 때문에 효율적인 데이터 전송기술로 브로드캐스팅 방법이 연구되어 왔다. 브로드캐스트에서 인덱스를 사용하면 원하는 데이터가 언제 방송되는지를 알 수 있어 튜닝 시간을 줄이고, 전력의 소비를 줄이는 효과가 있다. 지금까지 연구된 싱글 채널 인덱스 기법들은 모든 데이터 아이템이 동일한 확률로 접근되는 flat 브로드캐스트 방송에 적합한 인덱스 기법들이다. 데이터 아이템에 대한 접근 확률이 편향되는 경우에는 멀티디스크 방송 기법을 사용해야 효과적이지만, 기존의 인덱스 기법들은 인덱스가 한 방송 주기 내에서 반복되어 방송되는 데이터 아이템을 가리킬 수 없기 때문에 멀티디스크 방송 기법에는 효과적이지 않다. 본 논문에서는 싱글 채널 인덱스 기법으로서 멀티디스크 방송에 적용되는 인덱스 기법인 MDEI (Multi-disk Exponential Index) 기법을 제안한다. 제안 하는 MDEI 기법은 각 디스크 별로 인덱스를 구성하기 때문에 데이터에 대한 접근확률이 편향되는 경우에 멀티디스크 방송을 기반으로 이 인덱스 기법을 사용하면 flat 브로드캐스트를 사용하는 다른 인덱스 기법을 사용했을 때보다 평균 접근지연시간 시간을 크게 줄일 수 있다. 실험 결과는 데이터에 대한 접근 확률이 편향된 환경에서 MDEI가 평균 접근지연시간에 있어서 매우 좋은 성능을 갖는 것을 보여준다.

1. 서론

모바일 환경에서는 업링크와 다운링크의 대역폭이 비대칭적이며 전력이 한정되어 있기 때문에 효율적인 데이터 전송기술로 브로드캐스팅 방법이 연구되어 왔다. 이 방법은 서버가 계속하여 데이터들을 브로드캐스트 하고, 클라이언트는 필요로 하는 데이터를 그 중에서 선택적으로 받는 방법이다. 브로드캐스팅의 성능은 클라이언트가 원하는 데이터를 얻기 위해 채널을 탐색하는 시간인 튜닝시간과, 클라이언트가 데이터에 대한 요청을 보낸 후부터 그 데이터를 얻을 때까지 걸린 시간인 접근지연시간으로 평가할 수 있다. 데이터를 얻는데 걸리는 시간을 줄이기 위해서는 접근지연시간을, 소비되는 전력을 줄이기 위해서는 튜닝시간을 줄여야 한다.

브로드캐스팅에서 인덱스를 사용하면 원하는 데이터가 언제 방송되는지를 알 수 있어 튜닝 시간을 줄이고, 전력의 소비를 줄이는 효과가 있다. 이에 따라 브로드캐스팅에서 다양한 인덱스 방법들이 개발되어 왔다. (1,m) 인덱스나 분산 인덱스 방법과 같은 트리기반의 인덱스 방법[4], 해싱을 이용한 방법[5], 시그니처를 이용한 방법[6], 지수 인덱스 방법[1] 등이 그 예이다. 이 기법들은 모든 데이터 아이템이 동일한 확률로 접근되는 flat 브로드캐스트 방송에 적합한 인덱스 기법들이다. 데이터 아이템에 대한 접근 확률이 편향되는 경우에는 멀티디스크 방송 기법을 사용해야 효과적이지만, 기존의 인덱스 기법들은 인덱스가 한 방송 주기 내에서 반복되어 방송되는 데이터 아이템을 가리킬 수 없기 때문에 멀티디스크 방송 기법[2]에는 효과적이지 않다.

MHash 기법[3]은 두 개의 인수를 갖는 해시 함수를 사용하여 non-flat 브로드캐스트 방송에 적합하게 만들어진 인덱스 기법으로, 데이터 아이템에 대한 접근 확률이 편향되는 경우에 기존의 flat 브로드캐스트 방송에 대한 인덱스 기법들보다 좋은 성능을 보인다. 그렇지만 MHash 기법은 두 가지 이유에서 멀티디스크 방송에 적합하지 않다. 첫째, 멀티디스크 방송 프로그램의 모든 데이터 아이템을 정확하게 가리킬 수 있는 해시 함수는 없다. 두 번째 이유는 MHash 기법이 데이터 아이템의 최대 반복 횟수를 일정 숫자 이하로 제한한다는 것이다. 이는 데이터 접근 확률이 편향되는 경우에 멀티디스크 방송 프로그램에서 요구하는 반복 횟수가 MHash 기법의 최대 반복 횟수보다 클 수 있다는 것을 의미한다.

싱글채널 인덱스 기법 중에 멀티디스크 방송 기법에 적합한 기법은 없다. 멀티채널 환경에서는 이와 같은 연구가 진행되어 왔으나[7][8] 이들은 직접적으로 싱글 채널 환경에의 적용이 불가능하다.

본 논문에서는 멀티디스크 방송에 적합한 싱글 채널 인덱스 기법인 MDEI(Multi-disk Exponential Index)를 제안한다. MDEI 기법은 데이터 아이템 접근 확률이 편향될 때 멀티디스크 기법과 함께 사용하면 평균 접근지연시간을 줄일 수 있다.

MDEI는 멀티디스크에서의 동작을 위해 인덱스가 한 방송주기 내에서 여러 번 방송되는 데이터 아이템을 가리킬 수 있도록 지수 인덱스 기법을 확장했다. 기존의 지수인덱스는 하나의 키에 대해 하나의 위치만을 지정할 수 있기 때문에 멀티디스크에서의 동작이 불가능하지만, MDEI는 이를 멀티디스크의 각 디스크마다 별도의 인덱스를 구성하도록 확장했기 때문에 멀티디스크에서의 동작이 가능하다.

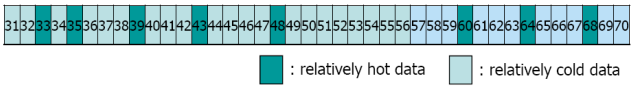
특별히 지수인덱스를 사용한 것은 멀티디스크에의 확장에서 몇 가지 장점을 갖기 때문이다. 본 논문에서처럼 디스크별로 인덱스를 구성할 경우에는 디스크 내의 데이터 아이템 개수가 인덱스의 크기에 영향을 준다. 그런데 멀티디스크에서 각각의 디스크가 포함하는 데이터 아이템의 개수는 데이터에 대한 접근 확률의 편향성 정도에 따라 달라진다. 트리 기반의 인덱스 기법들은 인덱스 트리가 하나의 루트를 갖는 트리로 구성되기 때문에 데이터 아이템의 개수 변화가 인덱스 트리의 크기 변화에 큰 영향을 주지만, 지수인덱스는 서로 다른 루트를 갖는 인덱스 트리들이 방송 전체에 분산되어 나타나기 때문에 데이터 아이템의 개수 변화가 인덱스 테이블의 크기 변화에 미치는 영향이 작아서 멀티디스크에의 확장에 적당하다. 또한 지수인덱스는 인덱스 테이블이 짧은 간격으로 자주 방송되기 때문에 상대적으로 초기 탐색 시간이 짧아서 평균 접근지연시간을 줄일 수 있다는 장점이 있다.

본 논문의 2장에서는 MDEI 브로드캐스트 프로그램의 구성 방법을 보여주고, 3장에서는 클라이언트가 MDEI 브로드캐스트 프로그램에서 원하는 데이터 아이템에 접근하는 프로토콜을 설명할 것이다. 4장에서는 이 기법의 실험환경과 실험 결과를 설명하고, 5장에서 이 논문의 결론을 맺을 것이다.

2. MDEI 프로그램 구성

이 장에서는 MDEI에서 방송하는 브로드캐스트의 프로그램이 어떻게 구성되는지를 설명한다.

(그림 1)은 방송할 데이터 아이템의 집합을 보여준다. 총 40개의 데이터 아이템을 방송하며, 그 중 8개의 데이터가 나머지 32개의 데이터보다 인기 있는 경우이다. 이 때, 인기 있는 데이터들은 나머지 데이터 아이템보다 두 배 더 자주 방송되도록 하면, [2]에 따라 (그림 2)와 같이 멀티디스크를 구성할 수 있다. 즉, D33, D35 등의 인기 있는 데이터 아이템들은 disk 1에 포함되어 한 번의 방송주기 내에서 두 번 방송되며, 나머지 인기 없는 데이터 아이템들은 disk 2에 포함되어 한 번의 방송주기 내에서 한 번 방송된다. 여기서 각각의 디스크는 데이터 아이템의 키 값에 의해 정렬시킨다.



(그림 1) 방송할 데이터 아이템의 집합

프로그램에서 모든 버켓은 컨트롤 정보와 인덱스 테이블, 그리고 데이터 아이템을 가지고 있다. 인덱스 테이블은 형태적으로는 지수 인덱스[1]와 같은 형태를 갖는다. 즉, 테이블은 {distInt, maxKey}의 튜플로 이루어져 있으며, 여기서 distInt는 현재 버켓으로부터의 거리, maxKey는 distInt만큼 떨어진 버켓의 최대 키 값이다 ($1 \le k \le \log N$, N은 방송되는 데이터 아이템의 개수). 하지만 그 내용은 다른 값을 갖게 되는데, MDEI는 멀티디스크에서 동작하기 위하여 인덱스 테이블을 디스크별로 따로 구성한다. 즉, 디스크 1에서 방송되는 인덱스 테이블은 디스크 1의 데이터만을 대상으로 구성된다. 예를 들면, (그림 2)에서 버켓 1의 인덱스 테이블이 나타내는 것은, 한 버켓 뒤에 데이터 아이템 D35가, 세 버켓 뒤에는 D43이, 일곱 버켓 뒤에는 D68이 방송된다는 것을 나타낸다.

하나의 인덱스 테이블은 자신이 포함된 디스크에 대해서만 정보를 가지고 있기 때문에, 인덱스 테이블만 가지고서는 여러 디스크들이 인터리빙되어 방송되는 프로그램 내에서 원하는 데이터의 방송 위치를 찾을 수 없다. 그래서 각 버켓에서는 컨트롤 정보도 함께 방송되는데, 그 내용은 <표 1>과 같다. 데이터의 방송 위치를 찾기 위하여 컨트롤 정보가 어떻게 사용되는지는 데이터 접근 프로토콜과 함께 3장에서 설명하겠다.

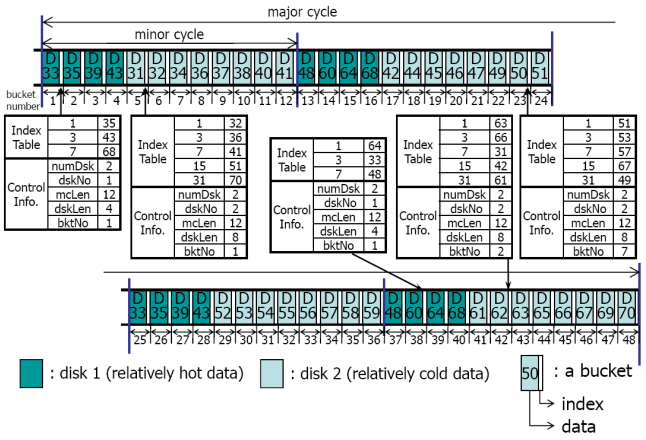
<표 1> 컨트롤 정보

notation	description
numDsk	disk의 개수
dskNo	현재 방송되는 disk의 번호
mcLen	하나의 minor cycle의 길이
dskLen	하나의 minor cycle 내에서 현재 디스크의 길이
bktNo	현재 디스크에서 현재 버켓의 상대적인 위치

3. MDEI의 데이터 접근 프로토콜

3.1 데이터 접근 프로토콜

(그림 2)에서 데이터 아이템 D33 (i.e., 버켓 1)이 방송되기 직전에 클라이언트가 데이터 아이템 D44를 요청했다고 가정하자. 클라이언트는 버켓 1에서 방송되는 인덱스 테이블을 보게 된다. D44은 D43보다 크고 D68보다 작기 때문에 D44는 현재 위치에서 네 버켓에서 일곱 버켓 만큼 떨어진 곳 사이에 존재한다. (단, D44가 디스크 1에 존재할 경우에만.) 그러므로 디스크 1에서는 네 버켓 만큼 뒤에서 튜닝해야 한다는 것을 알 수 있다. 디스크 1만 고려했을 때 네 버켓 만큼 떨어진 곳은 버켓 13이다. 클라이언트는 컨트롤 정보를 이용해 이를 계산해낼 수 있다. (이의 계산 방법은 3.2절에서 설명한다.) 그러므로 디스크 1에서의 다음 튜닝 위치는 버켓 13이라는 것을 저장



(그림 2) MDEI 프로그램 구성

해두고 절전 모드로 디스크 2로 간다.

디스크 2는 버켓 5에서부터 시작되며, 이는 역시 컨트롤 정보를 통해 계산해낼 수 있다. (이의 계산 방법은 3.2절에서 설명한다.) 클라이언트는 버켓 5의 데이터 아이템을 체크한다. 찾고자 하는 데이터가 버켓 5에서 발견되지 않기 때문에 클라이언트는 버켓 5의 인덱스 테이블을 확인한다. D44는 D41보다 크고 D51보다 작으므로 클라이언트는 8번째 버켓까지 절전 모드로 기다린다. 이 때 디스크 2의 8번째 버켓은 버켓 17이므로 다음 튜닝 위치는 버켓 17이며, 이것을 저장해둔다.

디스크 1과 디스크 2의 다음 튜닝 위치가 각각 버켓 13과 버켓 17이므로, 클라이언트는 둘 중 먼저 방송되는 버켓 13까지 절전 모드로 진행한다. 버켓 13에서 D44가 발견되지 않고, 인덱스 테이블을 확인해봤을 때, D44가 존재할 버켓이 없으므로 디스크 1에서는 D44가 방송되지 않음을 알 수 있다. 그러므로 디스크 1에서는 더 이상 찾아볼 필요가 없다. 디스크 2의 다음 튜닝 위치인 버켓 17까지 절전 모드로 진행한다. 버켓 17에서는 D44가 발견되지 않지만, 인덱스 테이블에서 D44를 발견할 수 있다. 따라서 클라이언트는 하나 뒤의 버켓에서 원하는 데이터 D44에 접근할 수 있다.

3.2 튜닝 위치 계산

튜닝 위치를 계산하기 위해서는 현재 버켓에서 인덱스 테이블을 살펴보고 다음 튜닝 위치에 대한 오프셋을 구해야 하며, 이를 실제 거리로 환산해야 한다. 인덱스 테이블에 의해 처음 구해지는 오프셋과 이를 환산한 실제 오프셋을 다음과 같이 정의한다.

정의 1. 현재 위치에서 인덱스 테이블을 살펴보면 다음 튜닝 위치를 결정할 수 있다. 인덱스 테이블은 인터리빙 되어 있는 다른 디스크를 고려하지 않고 현재 디스크만을 고려하는데, 이렇게 현재 디스크만을 고려한 오프셋을 *logical_offset*이라 정의한다.

정의 2. *physical offset*은 *logical_offset*까지의 실제 거리이다. 즉, 현재 위치로부터 다음 튜닝 위치까지 인터리빙 되어 있는 다른 디스크들을 고려한 실제 거리이다.

인덱스 테이블을 탐색하면 *logical_offset*을 알 수 있다. 그렇지만 *logical_offset*은 인터리빙 되어 있는 다른 디스크들은 고려하지 않은 오프셋이기 때문에 *logical_offset*을 *physical_offset*으로 변환해야 한다. *physical_offset*은 다음과 같이 계산될 수 있다.

$$physical\_offset = q*mcLen - bktNo + r + 1 \quad (1)$$

여기서

$$q = \lfloor (logical\_offset + bktNo - 1)/\mathrm{dskLen} \rfloor \quad (2)$$

$$r = \begin{cases} logical\_offset & \text{if } q = 0 \\ \lfloor (logical\_offset + bktNo - 1)\%dskLen \rfloor & \text{if } q > 0 \end{cases} \quad (3)$$

이다.

q는 현재 위치에서 *logical_offset*만큼을 진행하기 위하여 몇 번의 마이너 주기를 지나가야 하는지를 의미한다. r은 q만큼의 마이너 주기를 진행한 후에 몇 개의 버켓을 더 지나가야 하는지를 의미한다. 그러므로 식 (1)의 의미는 q회 만큼의 마이너 주기를 진행하고 r개의 버켓을 더 진행하는 만큼이 *physical_offset*이라는 의미이다. *mcLen, bktNo, dskLen* 등은 모두 컨트롤 정보를 통해 알 수 있는 정보들이고, *q*와 *r*은 인덱스 테이블로부터 계산되는 *logical_offset*을 이용해 계산되므로 클라이언트는 방송에서 *physical_offset*의 계산에 필요한 모든 정보를 얻을 수 있다. (그림 3)은 클라이언트가 인덱스와 컨트롤 정보를 통해 원하는 데이터에 접근하는 알고리즘이다.

```
Input: Key k
Output: location of data item k
Set nt_i =∞ for all 1≤i≤numDsk
Read the first bucket.
while(true) {
    if ( Current bucket contains data item k )
        receive data item k
        return
    }
    else {
        if ( Current disk doesn't have data item k )
            nt_dskNo = -1
        else
            nt_dskNo = q*mcLen-bktNo + r + 1
    }
    if ( nt_(dskNo+1)%numDsk == ∞)
        nt_(dskNo+1)%numDsk = dskLen-bktNo + 1
    if ( nt_i == -1 for all 1≤i≤numDsk) {
        search failure
        return
    }
    Go to doze mode and tune in after
        min_(1≤i≤numDisk) nt_i
}
```

(그림 3) 데이터 접근 알고리즘

4. 실험

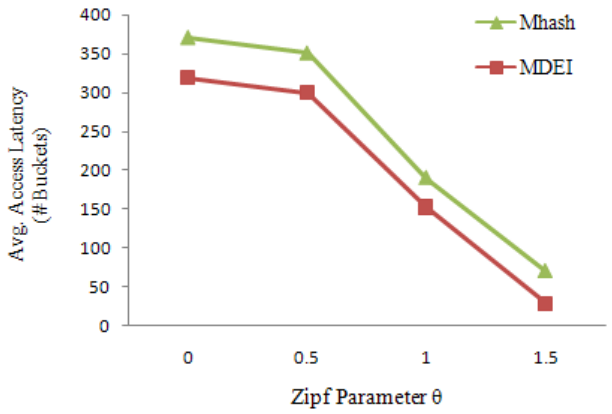
우리는 본 논문의 MDEI 인덱스 방법을 non-flat 인덱

스 방법인 MHash 인덱스 방법과 비교했다. 실험 환경은 <표 2>와 같다.

<표 2> 실험 환경

parameter	description	default value
n	number of data items	5000
bktSize	size of a bucket	1 KB
dataSize	size of a data item	128 bytes
indexSize	size of an index entry	4 bytes

(그림 4)는 데이터에 대한 접근의 편향성 변화에 따른 평균 접근지연시간을 비교한 결과이다. MDEI가 MHash보다 좋은 성능을 내는 것을 볼 수 있는데, 편향성 정도가 더 심해질수록 성능의 차이가 커져서 Zipf 파라미터가 1.5일 때는 MDEI가 MHash의 절반 이하의 평균 접근지연시간을 갖는 것을 볼 수 있다. 이는 편향성 정도가 커질 수록 상대적으로 인기 있는 데이터 아이템의 반복 횟수가 커지게 되는데, MHash에서는 이 반복 횟수가 일정 숫자 이하로 제한되기 때문이라고 분석된다.



(그림 4) 접근 편향성 변화에 따른 평균 접근지연시간

5. 결론

우리는 멀티디스크에서 동작하는 싱글 채널 인덱스 기법인 MDEI (Multi-disk Exponential Index)를 제시했다. MDEI는 데이터에 대한 접근 확률이 편향된 환경에서 접근지연시간을 효과적으로 줄이는 멀티디스크를 기반으로 동작하는 유일한 인덱스 기법이다. 따라서 데이터에 대한 접근 확률이 편향될 때, flat 브로드캐스트를 기반으로 하는 다른 인덱스 기법들이나, 멀티디스크가 아닌 non-flat 브로드캐스트를 기반으로 하는 MHash보다 짧은 접근지연시간을 갖는다. 실험 결과는 MDEI가 접근지연시간을 효과적으로 줄이며, 데이터에 대한 접근 확률의 편향성이 커질수록 더 좋은 성능을 발휘한다는 것을 보여준다.

참고문헌

[1] Jianliang Xu, Wang-Chien Lee, Xueyan Tang, Qing Gao, and Shanping Li, "An Error-Resilient and Tunable Distributed Indexing Scheme for Wireless Data Broadcast", IEEE Trans. Knowledge and Data Eng., vol. 18, no. 3, pp. 392-404, March 2006.

[2] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast Disks: Data Management For Asymmetric Communications Environments," Proc. ACM SIGMOD Conf. Management of Data, pp. 199-210, May 1995.

[3] Yuxia Yao, Xueyan Tang, Ee-Peng Lim, and Aixin Sun, "An Energy-Efficient and Access Latency Optimized Indexing Scheme for Wireless Data Broadcast", IEEE Trans. Knowledge and Data Eng., vol. 18, no. 8, pp. 1111-1124, August 2006

[4] T. Imielinski, S. Viswanathan, and B. Badrinath, "Data on Air: Organization and Access", IEEE Trans. Knowledge and Data Eng., vol. 9, no. 3, pp. 353-372, May 1997.

[5] T. Imielinski, S. Viswanathan, and B.R. Badrinath, "Power Efficient Filtering of Data on Air," Proc. Fourth Int'l Conf. Extending Database Technology, pp. 245-258, Mar. 1994.

[6] W.-C. Lee, and D.L. Lee, "Using Signature Techniques for Information Filtering in Wireless and Mobile Environments," Distributed and Parallel Databases, vol. 4, no. 3, pp. 205-227, July 1996.

[7] Narayanan Shivakumar, and Suresh Venkatasubramanian, "Efficient indexing for broadcast based wireless systems", Mobile Networks and Applications, vol. 1, no. 4, pp. 433-446, December 1996

[8] Sungwon Jung, Byungkyu Lee, and Sakti Pramanik, "A Tree-Structured Index Allocation Method with Replication over Multiple Broadcast Channels in Wireless Environments", IEEE Trans. Knowledge and Data Eng., Vol. 17,no. 3, pp. 311-325, March 2005