멀티채널 환경에서 데이터 브로드 캐스트를 위한 효율적인 인덱싱 방법

이병규 정성원[C]
서강대학교 컴퓨터학과
bkblue@sogang.ac.kr jungsung@ccs.sogang.ac.kr

An Efficient Indexing Technique for Wireless Data Broadcast in Multiple Channels

Byung-Kyu Lee Sungwon Jung[0]
Dept. of Computer Science, Sogang University

요 약

본 논문에서는 높은 대역폭(Bandwidth)을 가지는 싱글채널(Single Channel)로 결합될 수 없는 낮은 대역폭의 멀티채널(Multi Channel) 환경에서의 브로드캐스트(Broadcast)를 위한 효과적인 인덱스 방법을 제안한다. 최근에 들어 멀티채널에서 데이터 브로드캐스트의 인덱스 방법에 대한 많은 연구가 행하여 졌는데, 트리구조의 사용 여부에 따라 두 가지 방법으로 구분해 볼 수 있다. 트리 구조를 이용한 방법은 서버의 부담을 증가시키지만 트리 구조를 이용하지 않은 방법보다 모바일 유저에게 낮은 대기시간을 제공할 수 있다. 이 논문에서는 기존의 방법들이 가지는 채널의 수나 데이터의 크기에 제한을 두는 단점을 해결하고 접근 빈도가 높은 데이터에 대한 인덱스들의 반복 정도를 높여서 모든 인덱스에 대한 대기시간(Latency)의 평균값을 현저하게 줄일 수 있는 효과적인 인덱스 스케줄 생성 방법을 제시한다.

1. 서 론

빠르게 발전하고 있는 컴퓨터 하드웨어와 무선 네트워크 기술은 모바일 컴퓨팅 분야의 발전을 가속화 하고 있으며 이에 따라 모바일 기기의 사용이 급증하고 있다. 모바일 환경에서의 낮은 대역폭과 이동 단말기들이 가지게 되는 휴대성으로 인한 전원문제 때문에 모바일 유저들에게 어떻게 데이터를 효과적으로 전달할 수 있는가의 문제가 대두된다. 이러한 문제들을 해결하기 위한 방법 중에 하나로 브로드캐스트 방법이 연구되어 왔다.

기존의 브로드캐스트에 대한 연구는 싱글 채널에 대한 것이 대부분이였다. 따라서 현재까지 싱글 채널에서의 브로드캐스트에 대한 많은 문제 제기와 해결 방법들이 제시되어 왔다[1]. 하지만 다음의 예와 같이 멀티채널에서도 브로드캐스트 방법이 필요하다[2].

- 한 기지국이 전송능력을 가진 여러 대의 서버를 가지는 경우에 다른 기지국의 브로드 캐스트를 대신하는 할 경우
- 무선 환경에서 낮은 대역폭으로 인해 하나의 채널만으로는 효과적인 브로드 캐스트가 어려울 경우
- 서버에서의 Application이 사용자의 증가를 수용하면서 안정된 서비스를 제공하기 위해 서로 인접하지 않은 주파수를 추가로 필요로 할 때

최근에 들어 멀티채널에서의 데이터 브로드캐스트에 대한 인덱스 방법에 대해서도 많은 연구가 행하여 졌는데, 트리구조의 사용 여부에 따라 두 가지 방법으로 구분해 볼 수 있다. 트리 구조를 이용한 방법은 서버의 오버헤드를 증가시키지만 트리 구조를 이용하지 않은 방법과 비교해 볼때 보다 낮은 대기시간을 제공하는 장점을 가진다[1]. 트리를 이용하지 않은 방법은 서버에서의 오버헤드가 적고, 스케줄 방법자체가 간단한 대신에 트리구조에서 보다 큰 대기시간을 가진다[1][2].

트리 구조를 이용한 방법은 데이터와 인덱스를 함께 브로드캐스트 하는 Topological 트리를 이용한 방법과[3], 데이터와 인덱스를 분리하여 브로트캐스트 하는 Alphabetic 호프만 트리를 이용한 방법으로 나누어 볼 수 있다[4]. Alphabetic 호프만 트리를 이용한 방법은 충분한 수의 채널이 사용 가능하지 않을 경우에 모바일 유저의 접근 분포를 잘 반영하지 못하기 때문에 좋은 성능을 보이지 못하며, Topological 트리를 이용하는 방법은 데이터와 인덱스를 함께 채널에 분배하여 브로드캐스트 하는 방식이기 때문에 인덱스와 데이터의 크기가 같다는 가정을 필요로 하는 방법이므로 대역폭의 활용이 효과적이지 않다는 단점을 가진다.

본 논문에서는 앞에서 언급한 기존의 브로드캐스트 생성 방법들이 가지는 단점인 채널 수와 인덱스 크기에 대한 제한을 두는 것을 해결하면서 접근 빈도가 높은 데이터에 대한 인덱스들의 반복 정도를 높여서 인덱스에 대한 대기시간을 현저하게 줄이는 효과적인 인덱스 스케줄 생성 알고리즘을 제시한다. 또한 브로드캐스트 되는 인덱스들의 보다 효율적인 인덱스 배열 방법을 제시한다.

2장에서는 이 논문에서 제안하는 방법을 설명하기 위한 가정과, 용어의 정의를 기술하고, 접근 빈도를 반영한 인덱스 스케줄 작성 알고리즘을 제시한다. 3장에서는 최적의 index broadcast cycle을 작성하기 위한 인덱스의 재배열 방법에 대해 설명하고, 4장에서는 결론과 앞으로의 연구에 대해 언급한다.

2. 브로드캐스트 인덱스 스케줄링

이 장에서는 본 논문에서 사용되는 인덱스 트리인 Alphabetic 호프만 트리에 대한 설명과 기본적인 용어, 정의에 대해 설명하고 이 논문에서 제안하는 인덱스 스케줄 방법을 설명한다. 그림 1은 Alphabetic 호프만 트리이다.

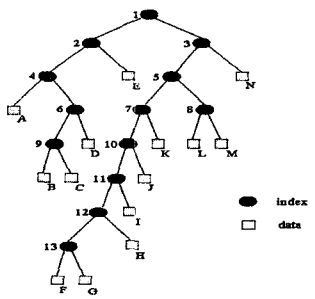

그림 1 Alphabetic Huffman Tree

Alphabetic 호프만 트리는 멀티채널 환경에서의 브로드 캐스트에 사용되는 대표적인 인덱스 방법이다[3][4]. Alphabetic 호프만 트리에서는

데이터를 정렬한 후에 각 접근 빈도에 따라 데이터에 대한 트리의 깊이가 정해진다. 즉, 접근 빈도가 높은 데이터는 트리의 상위 부분에 위치하고, 접근 빈도가 낮은 데이터는 트리의 하위 부분에 위치하게 함으로서 접근 빈도가 높은 데이터는 접근 빈도가 낮은 데이터보다 작은 대기시간을 가지게 된다[4].

우선 인덱스에 대한 접근 빈도를 나타내기 위해, 제한된 수의 데이터 채널에 접근 빈도에 따라 데이터를 분배시키기 위한 방법이 있다고 가정한다. 이 가정은 인덱스에 대한 접근 빈도가 그 인덱스가 포인트하고 있는 데이터에 대한 접근 빈도에 의해 결정되기 때문에 필요하다. 이렇게 함으로서 데이터에 대한 접근 빈도를 이용하여 인덱스의 접근 빈도를 결정할 수 있다. 또한 인덱스는 각각 포인트 하는 다음 인덱스나 데이터가 브로드캐스트 되는 채널의 번호와 각 채널에서 브로드 캐스트 되는 시간을 포함하고, 포인트 하고 있는 블록이 인덱스 또는 데이터인지를 구별하기 위해 타입 정보를 유지한다고 가정한다.

2.1 용어

제안하는 방법을 설명함에 앞서 몇 가지 용어에 대한 정의가 필요한데 여기서는 각 용어에 대해 언급한다. 데이터와 인덱스에 대한 온도는 각각에 대하여 접근되는 빈도를 나타낸다. 즉 데이터와 인덱스에 대한 온도가 높다는 것은 접근 빈도가 높다는 것을 의미하고, 온도가 낮다는 것은 접근 빈도가 낮다는 것을 의미한다. ***index replication set***은 인덱스 트리의 모든 인덱스를 온도에 따라 구분한 인덱스들의 집합이다. 이때 인덱스 구분의 방법은 데이터 채널에서의 데이터 분배 방법을 사용한다. ***index block***은 index replication set에 채널의 제한을 반영한 인덱스들의 집합을 말한다. 따라서 인덱스 트리의 모든 인덱스는 몇 개의 index replication set으로 나누어지게 되고, 여기에 채널의 제한을 반영하여 같은 수의 index block이 생성된다. ***index latency***는 인덱스 트리의 모든 인덱스에 대해서 채널을 듣기 시작한 후부터 각 인덱스 정보를 받을 때까지의 대기시간의 평균값을 말한다. 그리고 브로드 되는 인덱스의 한 주기를 ***index_broadcast_cycle***이라 한다. index block과 index replication set에 대한 온도는 각 인덱스 집합 안의 인덱스들이 포인트 하는 데이터들의 온도의 합을 의미한다. 그림2는 index latency와 index broadcast cycle에 대한 예를 보여주고 있다.

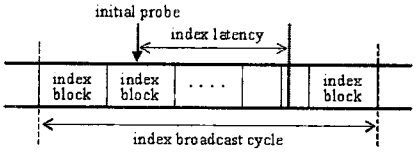

그림 2 index broadcast cycle

그리고 인덱스 스케줄 생성 과정에서 사용하여 만들어진 index block을 각각 $I_1, I_2, \ldots, I_n$이라 할 때 $I_1I_2\ldots I_n$과 같이 모든 index block들이 인덱스 트리에 포함된 인덱스의 순서를 유지하면서 배열되는 인덱스 블록의 순열을 ***full index schedule***이라 하고, $I_1, I_1I_2, \ldots, I_1I_2\ldots I_{(n-1)}$과 같이 I_1으로 시작하여 I_n을 제외한 각 인덱스 블록까지 이어지는 연속된 인덱스 블록의 흐름을 ***sub index schedule***이라 부르기로 한다. 그림 3은 그림1의 인덱스 트리로부터 생성 가능한 full index schedule과 sub index schedule의 예를 보여주고 있다.

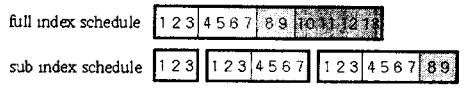

그림 3 full index schedule과 sub index schedule

2.2 인덱스 스케줄 작성 알고리즘

아래의 알고리즘은 접근 빈도가 높은 인덱스의 반복정도를 높이는 방법을 사용하여 인덱스 트리로부터 제한된 인덱스 채널의 수에 맞는 index broadcast cycle을 생성하는 알고리즘이다.

그림 4는 데이터 채널의 분배방법을 이용하여 index replication set이 생성되는 과정을 나타낸 그림이다.

Phase 4에서 각 index replication set은 각각에 포함되어 있는 인덱스의 개수에 따라 반복 정도가 정해지게 되는데, 반복되는 정도의 편차

Input : Alphabetic Huffman tree(Index Tree)

Output : k-channel Index Broadcast Cycle

Phase 1 : index replication set 생성

데이터 채널에서의 분배 방법을 사용하여 index replication set을 구성한다.

Phase 2 : index replication set의 온도 근사

index replication set에 포함된 인덱스의 수를 각각 $t_1, t_2, .., t_n$이라 할 때, 이 값들을 Dx(D는 근사 변수, x는 자연수)와 차이를 최소로 하는 x값을 찾아서 그때의 Dx값을 해당되는 t_i의 값으로 갱신하고, $t_1, t_2, .., t_n$의 공약수로 각각을 나눠준다. 이 과정은 t_n의 값이 인덱스 트리의 깊이(depth) 보다 작아질 때까지 계속 한다.

Phase 3 : index replication set을 이용하여 채널의 제한에 맞는 index block 생성

각 index replication set의 원소에 대해서 개수가 최대 k개인 부분집합을 만들고 인덱스 트리의 상위에 위치한 인덱스를 포함한 집합부터 나열하여 index block을 생성한다.(인덱스 트리에서의 부모와 자식이 같은 시간대에 브로드캐스트 되는 것을 방지하기 위해 인덱스 트리에서 부모 자식 관계에 있는 인덱스는 같은 집합 안에 포함되지 않게 한다.)

Phase 4: full index schedule과 sub index schedule을 생성

index block을 각 반복회수에 따라 복제시킨 후 full index schedule과 sub index schedule을 생성한다.

Phase 5: k-channel index broadcast schedule 생성

full index schedule과 sub index schedule을 생성된 순서대로 배열하여 index broadcast cycle을 생성한다. 이렇게 생성된 index broadcast cycle은 최대 k개의 원소를 가지는 집합들의 배열이 되는데 각 집합에 대해서 원소를 채널의 같은 시간대에 위치시킴으로서 k개의 채널에 맞는 index broadcast cycle을 생성할 수 있다.

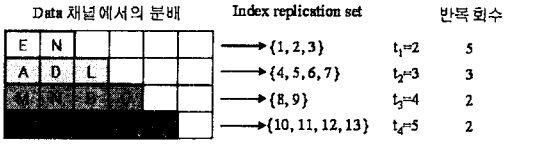

그림 4 index replication set의 생성

가 클 경우에 각 index replication set이 적어도 한번 브로드 캐스트 되는 것을 보장하고, 스케줄 자체가 지나치게 길어지는 것을 방지하기 위해 Phase2의 근사시키는 과정이 필요하다.

Phase 4에서 index block의 반복 회수는 $\left\lfloor \frac{t_i + depth}{t_i} \right\rfloor$에 따라 결정되므로($t_1, t_2, \ldots, t_n$은 각 인덱스 블록에 포함되어 있는 인덱스의 개수) 온도가 높은 index block은 온도가 낮은 index block과 비교해 볼 때 보다 많이 반복됨을 알 수 있다. 이렇게 복제 과정에 따라 생성된 모든 index block들을 이용하여 full index schedule과 sub index schedule이 생성된다. Phase 2에 의해서 가장 온도가 낮은 index block도 적어도 한번 반복되는 것이 보장되므로 적어도 하나의 full index schedule과 sub index schedule이 생성 가능함을 알 수 있다. 이렇게 생성된 full index schedule과 sub index schedule은 Phase 5에 의해 브로드캐스트 채널에 분배되게 된다.

위의 알고리즘을 통해 온도가 높은 인덱스를 자주 반복시켜 index latency를 감소시킬 수 있었다. 또한 Phase 3에서 채널에 대한 제한을 반영함으로서 제한된 인덱스 채널의 수에 맞는 인덱스의 스케줄을 작성할 수 있으며, 데이터와 인덱스 채널이 분리된 상태에서의 브로드캐스트를 가정하므로 인덱스의 크기에 대한 제한을 가정할 필요가 없으므로 대역폭의 효율적인 사용이 가능하다.

그림 5는 위의 알고리즘을 통해 생성된 index broadcast cycle을 보여주고 있다. 알고리즘을 사용하여 온도가 높은 인덱스를 하나의 index broadcast cycle안에 자주 반복시킴으로서index latency를 줄일 수 있다. 하지만 위의 알고리즘에 의해 생성된 index broadcast cycle을 살펴보면 full index schedule들이 위치한 후에 sub index schedule이 연속적으로 놓이게 됨으로서(그림 5의 점선으로 표시된 부분) 이 부분에서 온도가 낮은 인덱스 정보에 대한 대기시간이 길어지게 된다.

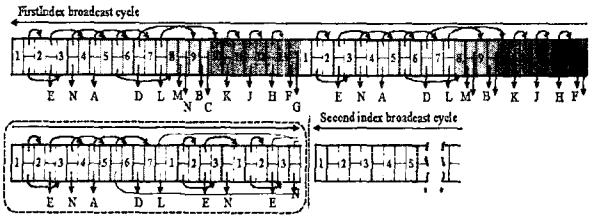

그림 5 알고리즘을 통해 생성된 index broadcast cycle

이러한 온도가 낮은 인덱스에 대한 대기시간의 증가는 데이터와 인덱스의 수가 증가할수록 커지게 된다[1]. 이러한 것을 방지하기 위해 만약 sub index schedule들을 full index schedule사이에 위치시킨다면 온도가 낮은 인덱스에 대한 대기시간의 증가를 완화 시킬 수 있을 것이라 생각할 수 있다. 다음 장에서는 index latency를 보다 줄일 수 있는 주어진 index broadcast cycle에서의 최적의 배열을 찾는 방법에 대해 알아본다.

3. 작성된 인덱스 스케줄의 재배열

2장에서의 알고리즘을 통하여 index broadcast cycle이 생성되는데, index broadcast cycle안에서 full index schedule과 sub index schedule의 배열 순서에 따라 각 index에 대한 index latency의 값이 달라짐을 쉽게 예상할 수 있었다. 이 장에서는 index latency를 최소화 시키는 full index schedule과 sub index schedule의 배열 방법에 대해 살펴본다. 다음은 모든 index에 대해 index latency의 값이 최소인 경우를 나타내는 optimal index latency의 정의이다.

정의 1. 인덱스 트리에 의해 생성될 수 있는 모든 full index schedule과 sub index schedule의 배열 중에서 index latency가 최소일 때의 인덱스들의 ordering을 ***optimal index ordering***이라 정의하고, 이때의 index latency를 주어진 index broadcast cycle에 대한 ***optimal index latency***라고 정의한다.

다음의 정의2는 인덱스 트리 사용하여 2장에 설명된 인덱스 스케줄 알고리즘에 의해 생성된 index broadcast cycle이 optimal index ordering을 가지도록 하기 위해서 만족해야 할 조건에 대한 것이고, 이 내용을 나타내 주고 있는 것이 정리 1이다.

정의 2. index_broadcast_cycle에 n개의 full index schedule이 있고, m개의 sub index schedule이 있을 때, 다음의 규칙을 만족한다면, ***fairly distributed***되어 있다고 정의한다. ($Len(I_{ij})$은 i번째 full index schedule뒤에 위치한 j번째 sub index schedule의 길이를 의미한다.)

1. *sub index schedule은 최대* $\lceil \frac{m}{n} \rceil$ *개 까지 연속적으로 위치할 수 있다.*
2. $\exists k,l \in \{1,\ldots,\frac{n}{n}\}, \forall i \in \{1,\ldots,n\}$ *if* $l > k$, *then* $Len(I_{il}) \geq Len(I_{ik})$
3. $\forall i,j \in \{1,\ldots,n\}, \exists k \in \{1,\ldots,\frac{m}{n}-1\}$, *if* $Len(I_{ik}) > Len(I_{jk})$ *then* $Len(I_{i(k+1)}) \leq Len(I_{j(k+1)})$

정리 1. index_broadcast_cycle에서 여러 개의 full index schedule과 sub index schedule이 있을 경우에 index broadcast cycle이 optimal index latency를 가지려면 fairly distributed되어 있어야 하고 이 경우에만 optimal index latency를 가진다.

정리 1을 통하여 index broadcast cycle이 optimal index latency를 가진다는 사실과 fairly distributed 되어야 한다는 것은 서로 필요충분 조건 관계에 있어야 한다는 것을 의미한다. 따라서 이 정리를 이용하여 optimal index latency를 가지는 index broadcast cycle을 생성할 수 있다. 이 정리에 대한 증명은 [4]를 참조하기 바란다.

그림 6은 인덱스 채널이 하나인 경우에 정리 1을 이용하여 생성된 optimal index latency를 가지는 index broadcast cycle이다. 이것을 그림 5에서 나타내는 재배열되기 전의 index broadcast cycle과 비교해 보면 그림 6에서는 sub index schedule들을 full index schedule사이에 위치시킴으로서(그림 6의 점선으로 표시된 부분) 온도가 낮은 인덱스들이 재배열되기 전보다 낮은 대기시간을 가지도록 할 수 있다. 그림 5에서와 같이 온도가 낮은 인덱스에 대한 대기시간의 증가는 데이터와 인덱스의 수가 증가할수록 커지게 되는데 여기서 제시한 재배열 방법을 사용함으로서 대기 시간의 증가를 완화시킬 수 있다.

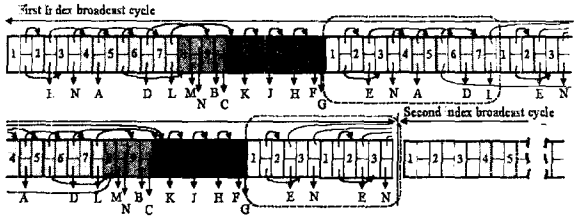

그림 6 재배열 과정을 통해 생성된 optimal index latency를 가지는 index broadcast cycle(index channel이 하나인 경우)

지금 까지 살펴본 방법과 같이 인덱스에 대한 접근 빈도에 따라 반복 정도를 다르게 하여 index broadcast cycle을 생성하고, 재배열 과정을 거침으로서 주어진 index broadcast cycle안에서 최적인 index latency를 얻을 수 있었다. 실험 결과를 통해서도 같은 결과를 확인해 볼 수 있는데 이에 대해서는 [1]를 참조하기 바란다. 그림 7은 같은 인덱스 트리를 사용하여 인덱스 채널이 두개인 경우의 index broadcast cycle이다.

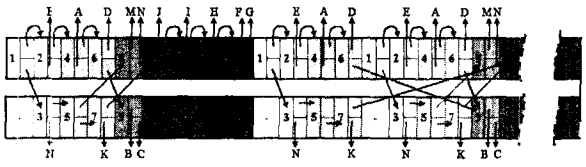

그림 7 재배열 과정을 통해 생성된 optimal index latency를 가지는 index broadcast cycle (index channel이 2개인 경우)

4. 결론

기존의 연구된 멀티채널에서의 인덱스 방법들은 자체적으로 채널수와 데이터의 크기에 대한 제한을 가지고 있었다. 본 논문에서 제시한 알고리즘을 통해 기존의 제시된 방법들의 단점들을 해결하고 접근 빈도가 높은 인덱스를 자주 반복시킴으로서 전체 인덱스에 대한 index latency를 감소시키고 재배열 과정을 통해서 index latency를 최적화 시킬 수 있었다. 하지만 접근 빈도가 높은 인덱스의 반복 정도를 높이는 방법을 사용함으로서 온도가 낮은 인덱스에 대해서는 접근시간이 증가하는 오버헤드가 발생할 수 있다. 그러나 접근 빈도가 낮은 인덱스에 대하서 index latency가 증가하는 것보다 접근 빈도가 높은 인덱스에 대한 index latency의 접근 빈도가 감소하는 폭이 더 크기 때문에 전체적인 성능에 크게 영향을 주지 못함을 알 수 있다.

향후 연구 과제로 제한된 수의 채널이 주어진 상태에서 인덱스와 데이터 채널의 최적의 분배 방법과, 제한된 데이터 채널에서의 데이터 브로드 캐스트를 위한 효율적인 데이터 브로드캐스트 스케줄 방법에 대해 연구 중이다.

참 고 문 헌

[1] S. Jung and B. Lee, Efficient Indexing Technique for Wireless Data Broadcast in Multiple Broadcast Channels, Technical Report, 2002

[2] K. Prabhakara, K. Hua, and J. Oh, Multi-Level Multi-Channel Air Cache Design for Broadcasting in a Mobile Environment Proc. 16th International Conf. on Data Engineering, 2000.

[3] S. Lo and A. Chen, Optimal Index and Data Allocation in Multiple Broadcast Channels, In proc. 16th international conf. on Data Engineering, 2000

[4] Narayanan Shivakumar, Suresh Venkatasubramanian, Energy Efficient indexing for Information Dissemination In Wireless Systems, ACM, Journal of Wireless and Nomadic Application ,1996