효율적인 갱신이 가능한 디스크 기반 역색인 구조

박은주, 이기용
숙명여자대학교 컴퓨터과학과
e-mail: eunjupark@sookmyung.ac.kr
kiyonglee@sookmyung.ac.kr

A update-efficient disk-based inverted index structure

Eun Ju Park, Ki Yong Lee
Dept. of Computer Science, Sookmyung Women's University

요 약

소셜 네트워크 서비스(SNS)로 인해 스트리밍 환경에서 발생되는 데이터들이 급격하게 증가하고 있다. 이러한 많은 데이터 사이에서 특정 키워드를 담고 있는 문서를 찾고자 한다. 문서를 찾는 대표적인 색인인 역색인을 사용한다. 그러나 데이터가 증가하게 되면 데이터를 참조하는 색인의 크기 또한 증가한다. 결국 데이터뿐만 아니라 색인 또한 디스크에 저장되어야 한다. 본 논문에서는 역색인을 지수적으로 증가시키면서 관리하는 방법을 다룬다. 새로운 문서는 가장 작은 역색인에 저장이 되고, 후에 더 큰 역색인으로 옮겨지게 된다. 매번 전체 역색인을 읽지 않고 작은 역색인을 갱신함으로서 갱신 부하를 줄이게 된다.

1. 서론

디지털 경제의 확산으로 많은 정보와 데이터가 생산되면서 빅데이터(Big Data)[1]에 대한 관심이 매우 커지고 있다. 빅데이터란 용량이 매우 크거나, 증가 속도가 매우 빠르거나, 형태가 매우 다양해서 현존 기술로는 효율적으로 처리하기 어려운 데이터를 말한다[2]. 특히 소셜 네트워크 서비스(SNS)로 인해 스트리밍 환경의 데이터들이 급격하게 증가하였다. 대표적으로 트위터의 경우 하루에 수백만 개의 문서를 만들어낸다.[3]

본 논문에서는 주어진 키워드를 포함하는 가장 최근 k개의 문서를 찾는 질의를 처리하려고 한다. 이를 위해 문서 데이터를 찾는 전통적인 색인기법인 역색인을 사용한다. 그러나 데이터의 크기가 끊임없이 커짐에 따라 그 데이터를 참조하는 역색인의 크기도 계속해서 커지게 된다. 그 결과 역색인이 메모리에 모두 저장되기 어려워지게 되고, 디스크에 저장되어야 한다. 하지만 디스크에 저장된 전통적인 역색인은 스트림 데이터를 반영하기 위한 잦은 업데이트에 매우 비효율적이다.

이를 위해 본 논문에서는 디스크 기반의 스트리밍 환경에서 효율적인 갱신이 가능한 역색인 기법에 대해 제안한다. 역색인은 I_0, I_1, I_2, ... ,I_m 형태이며 지수적으로 크기가 증가한다. I_0에만 append-only형태로 데이터를 받고, I_0에 들어간 데이터의 개수가 일정 값이 된 경우 이미 정렬되어 있던 I_1와 병합한다. 이 때, I_1의 크기는 I_0의 두 배이다. 같은 방식으로 I_1의 크기가 일정 값이 되면 I_1은 I_2와 병합된다. 개수가 적은 색인끼리의 병합이 자주 일어나기 때문에 모든 색인을 읽고 갱신하는 것보다 효율적이다.

본 논문의 구성은 다음과 같다. 2장에서 배경 지식과 문제 정의를 기술하고, 3장에서는 연구 동기를 설명한다. 4장에서는 제안 방법을 설명한 후 5장에서 성능 분석을 한다. 마지막으로 6장에서는 추후 연구에 대해 언급한 후 결론을 맺는다.

2. 배경 지식 및 문제 정의

2.1 질의 정의

스트림으로 들어오는 문서는 단어들의 집합으로 구성되어 있고 처리하고자 하는 질의 q는 다음과 같다.

q = (w, k)

즉, 단어 w가 포함된 최근 k개의 문서를 찾고자 하는 질의이다.

2.1 역색인

역색인은 낱말이나 숫자와 같은 내용물로부터의 매핑 정보를 데이터베이스 파일에 저장하는 색인 데이터 구조이다. 역색인의 목적은 문서의 빠른 검색이다. 문서 검색 시스템에 쓰이는 가장 대중적인 데이터구조로서, 검색엔진과 같은 대규모 데이터 검색에 사용된다.[4]

본 논문에서 역색인은 하나의 해시 테이블과 하나의 Posting list세트로 구성되어 있다. 이 때 Posting list는 단어와 해당 단어를 포함하고 있는 문서 정보를 담고 있

다. 따라서 특정 단어가 담긴 문서를 찾을 때는, 먼저 찾고자 하는 단어의 해시 테이블로 접근하여 해당 Posting list를 스캔한 후, 일치하는 문서를 찾는다.

2.2 기타 환경

스트리밍 환경에서 데이터가 메모리(I_0)에 들어오고, 들어온 데이터의 개수가 I_0의 임계값인 τ_0가 될 때마다 디스크에 있던 Posting list과 병합하여 역색인을 갱신한다. 디스크에 있는 Posting list를 갱신할 때, 디스크에 있던 Posting list는 최신 것부터 시간순서대로 정렬이 되어 있다. 따라서 갱신 전 메모리(I_0)에 있는 데이터들을 시간순서대로 정렬 한 후, 디스크의 Posting list와 합치게 된다.

3. 연구동기

기존 연구는 메모리 안에 모든 색인 정보를 가지고 있다고 가정하고 연구를 진행하였다.[5] 그러나 데이터의 크기가 무한히 커진다면 그 정보를 포함하고 있는 색인도 메모리에는 다 올라갈 수 없을 정도로 증가하게 된다. 그에 따라 색인도 디스크에 저장해야 한다.

2.1에서 언급한 디스크에 있는 Posting list를 최신 데이터 순서대로 정렬해 놓을 경우 검색 시 빠른 성능을 보인다. 그러나 스트리밍 환경에서 정렬된 역색인을 계속 유지하려면 매번 대용량 파일을 읽어 갱신해야하기 때문에 갱신 부하가 발생한다.

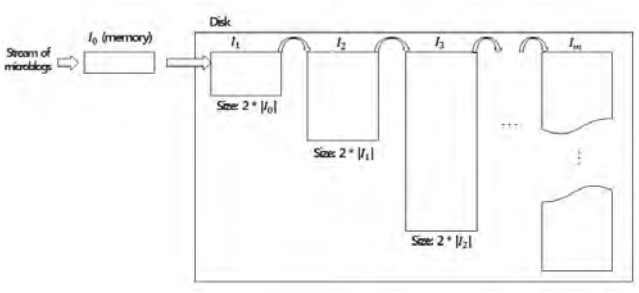
따라서 본 논문에서는 스트리밍 환경에서 효율적인 갱신이 가능한 디스크 기반의 역색인 기법을 제안하려고 한다.

4. 제안 방법

제안 방법은 (그림 1)과 같이 역색인을 I_0, I_1, I_2, ... , I_m로 두고 I_i의 크기는 I_{i-1}의 임계값 τ_{i-1}의 2배로 정한다. 다시 말해서 I_i의 크기는 I_{i-1}가 가질 수 있는 최대 크기의 2배가 된다. 스트리밍 데이터는 메모리에 위치하는 가장 작은 역색인 I_0에 들어오게 된다. I_0가 특정 값(τ_0)이 되면, I_0를 I_1와 병합한 후, I_0는 비워준다. 같은 방식으로 I_1가 특정 값(τ_1)이 되면 I_2와 병합한 후 I_1은 비운다. 이 방식을 이용하여 역색인을 갱신할 수 있고, 전체적인 갱신비용을 줄일 수 있다. 왜냐하면 갱신 시, 주로 개수가 작은 I_0가 갱신을 하며, I_0보다 가끔 I_1을 갱신하고 그 보다 더 가끔 I_2를 갱신하기 때문에 갱신할 때마다 전체를 읽고 새로 갱신하는 것보다 효율적이다.

4.1 갱신 방법

I_0는 메모리에 존재하며 동적배열의 구조를 가지고 있다. I_0에 새로운 데이터가 들어오면 해당 단어가 존재하는



(그림 1) 제안 방법의 역색인 구조

문서의 문서번호가 Posting list에 저장된다. 동적 배열에 담긴 데이터의 개수가 특정 임계값(τ_0)을 만나게 되면 기존 I_0에 있는 데이터는 I_0의 두 배 크기인 I_1에 저장된다. 같은 방식으로 I_i에 저장된 데이터의 개수가 임계값(τ_1)을 만나면 I_{i+1}의 크기는 2×(I_i의 최대크기)가 된다. I_0를 제외한 I_i (i∈$[1,m]$) 는 디스크에 저장된다.

두 개의 Posting list를 병합할 때 해시 테이블로 합칠 수 있는 Posting list를 구별한다. 두 개의 Posting list가 정렬되어 있다면 순서대로 두 개의 Posting list를 읽어가면서 병합한다. 그렇지 않은 경우(하나의 Posting list가 I_0에 존재한다면) 먼저 I_0의 Posting list를 시간 순서대로 정렬 시킨 후, 나머지 Posting list와 병합시킨다.

4.2 검색 방법

검색을 하여 문서 파일을 찾기 위해 후보군 공간을 두게 된다. q = (w, k) 질의가 들어오면 w를 검색하기 위해 해시함수를 이용해 해당 Posting list위치로 이동하게 된다. I_0를 탐색하면서 해당 w와 일치하는 문서번호를 후보군 공간에 넣게 되고, I_0 탐색이 끝나면 후보군 공간에 있는 후보의 개수가 k보다 큰지 작은지 확인한다. k보다 후보들이 많다면 후보들을 시간 순서대로 정렬 한 후 순서대로 k개를 선택해 디스크에서 문서를 찾을 수 있다.

만약 k보다 후보들의 수가 적다면, I_1으로 가서 탐색하고 순서대로 읽으며 후보의 개수가 k의 개수와 같아질 때까지 읽는다. I_1의 탐색이 끝나고 후보의 개수가 k의 개수에 미치지 않는다면 I_2, I_3, ...로 확장해간다.

5. 성능 분석

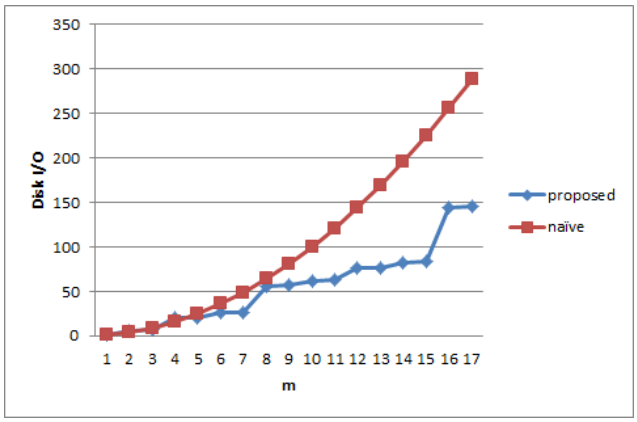
메모리 I_0 에 데이터 n개가 들어왔을 때 I_1와 병합한다고 가정하자. 즉 임계값 t = n이며 데이터가 n개씩 m번 들어와 m번 갱신을 해야 한다고 가정하자.

기존 방법에서 발생하는 디스크 접근 횟수는 m번째 일 때 '(m-1)n+mn'번이다. 그 이유는 기존에 있던 데이터를 모두 읽고 들어온 n개의 데이터를 합쳐서 추가해 주어야 하기 때문이다.

제안 방법은 최소 n번에서 최대 (4m-3)n번 디스크를

접근하게 된다. 그러나 I_i가 I_{i+1}과 병합할 때만 (4m-3)n 번의 디스크를 접근하게 되고, 상대적으로 I_0의 갱신이 자주 일어나므로 병합 시에만 디스크 접근 횟수가 증가하게 된다. 그러나 병합을 하더라도 기존 방식보다 더 적은 디스크 접근 횟수를 갖는다. Chaining update 발생하더라도 누적 디스크 접근 횟수는 기존 방법보다 적다.

그래프로 확인해보면 (그림 2)와 같다. 데이터가 많아질수록 제안 방법이 더 높은 갱신 성능을 보이는 것을 확인할 수 있다. 기존 방법은 디스크 접근 횟수가 기하급수적으로 늘어나는 반면, 제안 방법은 계단식으로 디스크 접근 횟수가 증가하는 것을 확인할 수 있다.



(그림 2) 성능 분석 그래프

6. 결론 및 추후 연구

본 논문은 스트리밍 환경에서 문서 검색을 위한, Disk기반의 효율적인 갱신이 가능한 역색인 방법을 제안하였다. 제안 방법은 기존의 단순 방법에 비해 역색인을 효율적으로 갱신한다.

추후에는 제안 방법의 효율성을 이론적으로 증명하고자 한다. 또한 실험을 통해 성능 비교를 가시적으로 보여주고자 한다.

참고문헌

[1] Big Data, http://en.wikipedia.org/wiki/Big_Data

[2] Mark Beyer, "Gartner Says Solving 'Big Data' Challenge Involves More Than Just Managing Volumes of Data," Gartner, June 27, 2011.

[3] M. Busch, K. Gade, B. Larson, P. Lok, S. Luckenbill, and J. Lin, "Earlybird: Real-time search at twitter," in ICDE, 2012, pp. 1360 - 1369.

[4] Inverted Index, https://en.wikipedia.org/wiki/Inverted_index

[5] Lingkun Wu, "LSII: An Indexing Structure for Exact Real-Time Search on Microblogs", in ICDE, 2013

Acknowledgement

이 논문은 2015년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. 2015R1C1A1A02037071)