SSD에 적합한 동적 색인 저장 구조 : SPM

Efficient Dynamic Index Structure for SSD (SPM)

진두석*, 김진숙*, 류범종*, 정회경**
한국과학기술정보연구원 정보기술연구실*, 배재대학교 컴퓨터공학과**

Du-Seok Jin(dsjin@kisti.re.kr)*, Jinsuk Kim(jinsuk@kisti.re.kr)*,
Beom-Jong You(ybj@kisti.re.kr)*, Hoe-Kyung Jung(hkjung@pcu.ac.kr)**

요약

역파일 인덱스 구조는 대용량 텍스트 데이터의 색인저장 기법을 위한 효율적인 데이터 구조로 널리 활용 되고 있다. 특히, 최근 이슈가 되고 있는 온라인 색인관리 측면에서는 동적 검색 환경에 적합한 In-Place 방식과 Merge-based 색인 방식이 주로 사용 되고 있다. 위 방법들의 핵심은 검색 처리시간을 줄이기 위해서 포스팅 정보의 저장 연속성(Contiguity)을 보장하면서 동시에 색인정보 관리(Index Maintenance) 시간을 최소화 하기위한 색인저장 구조에 중점을 두고 연구가 진행 되었다. 그러나 최근 기존 저장장치(HDD)와 근본적으로 구조가 다른 새로운 저장장치(SSD, SCRAM)가 데이터 저장소로 이용되면서 이러한 장치들의 특성을 효과적으로 활용할 수 있는 새로운 형태의 색인저장 기법 또한 필요하게 되었다. 따라서 본 논문에서는 새로운 저장장치의 빠른 접근 속도(Low access latency) 특성을 최대한 활용할 수 있는 분할(Segmentation) 포스팅 구조를 기반으로 새로운 저장장치에 적합하도록 변형된 In-Place 방식(Pulsing)과 수정된 Merge-based 방식(Merging)을 혼합하여 검색 처리시간 및 색인정보 관리시간을 크게 향상시킬 수 있는 새로운 색인저장 구조(SPM)를 제안한다.

■ 중심어 : | 동적 색인 | 역파일 구조 | 색인 구조 | SSD |

Abstract

Inverted index structures have become the most efficient data structure for high performance indexing of large text collections, especially online index maintenance, In-Place and merge-based index structures are the two main competing strategies for index construction in dynamic search environments. In the above-mentioned two strategies, a contiguity of posting information is the mainstay of design for online index maintenance and query time. Whereas with the emergence of new storage device(SSD, SCRAM), those do not consider a contiguity of posting information in the design of index structures because of its superiority such as low access latency and I/O throughput speeds. However, SSD(*Solid State Drive*) is not well suited for traditional inverted structures due to the poor random write throughput in practical systems. In this paper, we propose the new efficient online index structure(SPM) for SSD that significantly reduces the query time and improves the index maintenance performance.

■ keyword : | Index Update | Inverted File | SSD |

접수번호 : #091007-002
접수일자 : 2009년 10월 07일
심사완료일 : 2009년 12월 07일
교신저자 : 정회경, e-mail : hkjung@pcu.ac.kr

I. 서 론

최근 전통적인 디스크 기반의 저장장치(HDD)보다 수십배 빠른 접근 속도를 보장하는 새로운 저장장치(SSD)의 용량 증가와 가격 하락으로 서버 저장소로서 SSD의 활용 방안에 대한 많은 연구가 진행 되고 있다. 그러나 SSD는 HDD와 근본적인 구조부터 다르고 동작 특성이나 사용 환경에도 차이가 있다. 따라서 이를 변환해주는 FTL(Flash Translation Layer)과 같은 별도의 소프트웨어가 필요할 뿐만 아니라 기존의 HDD환경에서 동작하였던 소프트웨어들이 SSD환경에서 적절한 성능을 발휘하기 위해서는 SSD의 특성을 고려한 형태로 수정이 필요하다. 그 중에서도 특히 본 논문에서 다루고자 하는 색인정보 관리 측면에 중점을 두고 살펴보면 HDD에서 가장 중요시 여겨졌던 포스팅(Posting) 정보의 연속성(Contiguity)이 SSD의 빠른 디스크 접근 속도(Low disk access latency)로 SSD에서는 거의 영향을 주지 않는다. 또한 색인 정보 수정시 HDD방식에서는 포스팅 정보의 연속성 보장을 위해 많은 추가적인 블록(Block)의 정보 수정이 발생한다. 하지만 이러한 점은 임의 쓰기(Random write) 성능이 취약한 SSD에서는 색인정보 관리 성능을 크게 저하시킨다. 최악의 경우에는 HDD를 사용한 경우 보다 못한 성능을 보이기 때문에 기존 HDD를 기반으로 연구된 색인 구조를 SSD에 그대로 적용하기에는 적절하지 못하며 SSD에 적합한 형태의 새로운 색인 저장 구조가 필요하다. 따라서 본 논문에서는 HDD를 대상으로 오랫동안 연구되어온 색인저장 구조를 새로운 저장장치(SSD)에 적합하도록 개선하여 SSD의 우수한 읽기 성능을 최대한 활용하면서 SSD의 단점인 임의 쓰기 문제를 최소화 시킬 수 있는 새로운 형태의 색인 저장 구조를 제안한다.

II. 관련연구

2.1 색인 관리(Index Maintenance)

텍스트 정보검색 분야에서 색인정보의 저장 구조로는 B-tree 기반의 역화일 구조가 가장 보편적이다. 역파일 구조란 문서에서 검색어가 될 만한 것들의 정보를 미리 추출하고 정렬하여 구축해 놓는 방식으로 빠른 검색속도를 제공한다. 특히, 1990년대 중반부터는 추가적인 문서에 대해 즉시 검색 가능한 동적 역파일 구조에 대한 연구의 필요성이 제시되었고 수많은 연구가 진행되고 있다[1-3]. 동적 색인 구조의 대표적인 방식은 Rebuild, In-Place, Merge 방법이 있다[4]. 그 중에서도 In-Place 방식의 개선된 형태나 Merge-based 방식이 가장 우수한 성능으로 평가된다[5]. 위 방식에서 검색 성능을 결정짓는 요인은 디스크 접근 속도(Access latency time)에 영향을 주는 포스팅 정보의 연속성 이다. 물리적으로 연속된 위치에 검색 및 관리에 필요한 정보들(Posting information)이 함께 모여 있으면 디스크 탐색 시간(Seek time)이 줄어들어 검색 시간이 줄어들기 때문이다.

대표적인 동적 색인 구조 방식들의 가장 큰 차이점은 기존 색인 포스팅 정보와 추가되는 포스팅 정보의 병합(Merge) 방법이다. 각각에 적용된 병합 알고리즘에 따라 매우 큰 성능차이를 보이며 각 방식들의 특징을 살펴보면 다음과 같다.

첫째, Rebuild 방식은 매우 간단한 방법으로 추가되는 문서의 색인 정보는 메모리에 저장되었다가, 일정량 이상이 되면 전체를 다시 디스크에 저장하는 방식이다. 동적인 환경에는 매우 비효율적인 구조임에도 불구하고, 데이터 집합의 크기가 비교적 작고 추가되는 문서가 빈번하지 않는 경우에 가장 널리 쓰이고 있는 방식이다.

둘째, Merge-based 방식은 추가 색인(Incremental Indexing)을 지원하기 위한 구조로서 새로 추가된 문서의 색인 정보를 별도의 저장구조에 유지하다가 일정량 이상이 되는 경우 추가 색인정보 들 끼리 병합하거나 또는 주색인(Main Index) 정보에 병합한다. 이때 검색 속도를 빠르게 하기위해서 전체 색인어 리스트를 순차적으로 읽어서 새로운 영역에 정렬된 형태로 디스크 상에 연속적으로 저장한 후 기존 색인정보들은 제거한다. 위 방식은 색인 정보 전체를 병합해야 하는 단점을 가지고 있다.

셋째, In-place 방식은 새로운 문서가 추가 되었을 때

가능하면 기존 색인어 리스트의 포스팅 정보 마지막에 새로운 정보를 추가하는 방식으로 수정되는 색인 정보의 양을 최소화 시키는 방식이다. 그러나 기존 포스팅 정보의 마지막에 새로운 정보를 추가 할 수 없는 경우에는 새로운 연속된 공간을 할당하고 기존 정보를 새로 할당 받은 공간으로 이동한 후 추가된 정보를 병합한다. 이때 발생한 새로운 할당(Relocation)에 의해서 처음에 구성된 색인정보의 순차적인 순서가 파괴된다. 이로 인해서 색인 정보 수정 시 많은 수의 비순차적 디스크 접근(Non-sequential disk access)이 발생한다. 따라서 이러한 비순차적 디스크 접근을 줄이기 위한 방법으로 새로운 공간을 할당할 때 여유 공간을 포함한 포스팅 저장 공간 할당(Over-allocation)을 하기도 한다.

2.2 Tomasic(1994)

Tomasic(1994)[6]은, 연속된 공간에 포스팅 정보를 저장하는 것은 위에서 설명한 In-place 방식과 유사하나, 포스팅 정보가 커짐에 따라 긴 리스트(Longlist) 관리 정책에 따라 "연속적인(Contiguous) 가변길이 블록들", Chunk, 에 저장하거나 비연속적(Discontiguous)인 Chunk 리스트 구조로 포스팅 정보를 저장한다. 즉 Tomasic 방식은 포스팅 리스트를 긴 리스트(Longlist)와 짧은 리스트(Shortlist)로 구분하고 짧은 리스트는 "고정길이(Fixed size) 영역", Bucket, 에 저장하고 긴 리스트들은 분할되어 Chunk 리스트 형태를 유지함으로써 갱신되는 포스팅 정보의 영역을 리스트의 마지막 Chunk 영역으로 한정 할 수 있다. 따라서 새로운 문서가 추가되었을 때 이전 리스트들은 수정하지 않고 마지막 리스트 Chunk만 갱신하거나 추가적인 Chunk를 할당하여 마지막 리스트에 연결하는 방식으로 구성된다. 그러나 이 방법은 추가되는 문서가 많아짐에 따라 단편화(Spread)가 심해져서 검색 시간이 느려지는 단점을 가지고 있다.

2.3 Buttcher(2006)

Buttcher(2006)[7]는, Merge-based방식의 단점인 대다수의 변경되지 않은 색인어 정보 리스트의 수정과 In-Place방식의 단점인 포스팅 정보 수정시 발생하는 많은 비순차적 디스크 접근 문제를 보완하기 위한 새로운 색인저장 구조를 제안하였다. 일반적으로 긴 리스트는 디스크 탐색 연산(Disk seek operation)보다 전체 리스트를 복사하는 시간이 더 많이 걸리며, 짧은 리스트는 리스트를 복사하는 시간보다 디스크 탐색 연산이 시간이 더 걸린다. 이러한 사실에 기반 하여 Buttcher 방식은 긴 리스트와 짧은 리스트에 서로 다른 색인 관리 방식을 적용한 혼합 색인 관리(Hybrid index maintenance) 방식을 제안하였다. 즉 짧은 리스트는 Merge-based 색인 방식을 이용하고 긴 리스트는 In-Place 방식을 이용함으로써 각각을 이용한 방식들에 비해 거의 비슷한 수준의 검색 성능을 유지하면서 우수한 색인 관리 성능을 보였다.

2.4 SSD Storage

SSD(Solid State Disk)는 크기와 전력 소모가 작고, 임의 접근 속도가 빠르며, 충격과 진동에 강하기 때문에 노트북과 같은 모바일 멀티미디어 정보기기의 저장 매체로 사용되고 있다. 최근에는 플래시 SSD의 집적도가 증가하고 단위 용량 당 가격이 감소함에 따라 플래시 메모리를 기반으로 한 서버용 데이터베이스 저장장치로의 활용도 증가하고 있다[8][9]. 그러나 플래시 메모리는 제자리 갱신이 되지 않기 때문에 읽기, 쓰기 이외에 소거 연산이 필요하며 한 번 쓴 영역을 다시 쓰기 위해서는 소거 연산이 선행되어야 한다. 플래시의 내부 구조는 512Byte 또는 2Kbyte의 페이지들이 모여 블록을 구성하고 다수 개의 블록이 모여 플래시 메모리 칩이 구성되는데 읽기와 쓰기 연산은 페이지 단위로 이루어지고 소거 연산은 블록 단위로 이루어진다. 이러한 구조로 인해 읽기와 쓰기 속도에 비해 갱신 속도가 매우 느린 단점을 가지고 있다. 따라서 플래시 메모리를 서버용 데이터베이스 저장장치로 사용하기 위해서는 플래시 메모리의 느린 갱신 속도를 보완할 수 있는 IPL(In-Page Logging) 방식[10]과 같은 새로운 갱신 연산 방식이 필요하다. 또한, 최근에는 SRAM과 동일한 인터페이스를 가지면서 수십 ns의 접근속도로 읽기와 쓰기가 가능하고 제자리 갱신이 가능한 FRAM, MRAM, PRAM과 같은 차세대 NVRAM에 대한 연구

가 활성화 되고는 있지만 이들 메모리는 아직까지는 집적도가 낮아 서버 저장소로서의 상용화에는 많은 시간이 필요하다.

III. SPM 색인 구조(SPM INDEX STRUCTURE)

본 논문에서는 메모리 기반의 저장장치들 중에서 현재 서버용 저장소로 상용화되고 있는 SSD를 대상으로 SSD의 특성을 최대한 활용할 수 있는 새로운 형태의 동적 색인 저장구조를 제안한다. 제안하는 색인 저장구조는 분할(Segmentation) 포스팅 구조에 기반 하여 SSD에 적합한 형태로 변형된 In-Place 방식(Pulsing)과 수정된 Merge-based 방식(Merging)의 혼합 형태이다. 이러한 구조를 채택한 이유는 다음과 같다. Tomasic[6] 방법에서 분석한 결과 텍스트 데이터의 특징은 빈도수 상위 5.0%의 단어가 전체의 93.6%를 차지하는 대략적인 지수분포(exponential distribution : the Zipf curve) 형태이다. 즉 문서가 추가될 때 빈도수가 높은 단어들의 포스팅 정보는 빠르게 증가하며 반대로 빈도수가 낮은 단어들의 포스팅 정보는 매우 느리게 증가하거나 전혀 증가하지 않는다. 따라서 일반적인 색인 저장구조에서는 긴 포스팅 리스트들이 짧은 포스팅 리스트 보다 빈번하게 갱신이 되기 때문에 문서가 추가될수록 점점 갱신 속도가 느려지는 현상이 발생한다. 위의 문제점을 해결하기 위한 가장 효과적인 구조가 분할 구조이다. 또한 SSD를 이용한 경우 낮은 갱신 속도를 보완할 수 있는 형태의 수정된 In-place 방식과 Merge-based 방식의 적용이 필요하다. 따라서 본 논문에서는 위의 문제점을 보완하고 SSD에 적합한 동적 색인 저장구조, SPM을 제안한다. SPM 색인 저장구조는 아래와 같은 3가지 방식을 혼합하여 구성한다.

i. **분할(Segmentation)** - Tomasic 방법의 단편 리스트 아이디어에 기반 하여 정해진 임의의 값에 따라 고빈도 단어와 저빈도 단어로 구분한다. 그리고 고빈도 단어의 포스팅 정보는 고정길이 세그먼트 리스트(Segment list)로 구성하고, 저빈도 단어의 포스팅 정보는 직접 수정하는 In-place 방식과 동일하게 처리한다. Buttcher 방식의 혼합색인관리와 유사하나 저빈도 및 고빈도 단어의 색인관리에서 Buttcher 방식은 각각 Merge-based, In-place 방식을 사용하지만 SPM 색인저장구조에서는 SSD의 특성에 적합한 분할, In-place 방식을 사용한다.

ii. **펄싱(Pulsing)** - 본 연구의 색인 관리 방식은 디스크 기반의 주 색인정보 저장소와 보조 색인정보 저장소를 가진 이중 구조로 되어 있다. 추가되는 문서는 보조 색인 정보 저장구조에 저장되고 저장된 단어의 색인정보 빈도수가 임계치 이상이 되면 고빈도 단어로 변경되고 새로운 세그먼트를 할당하여 주 색인정보 저장소에 저장한다. 그리고 해당 단어의 색인정보는 보조 색인 저장구조에서는 삭제된다. 이러한 과정을 본 연구에서는 펄싱이라 정의한다.

iii. **병합(Merging)** - 일반적으로 새로운 문서가 추가될수록 고빈도 단어들의 색인정보 뿐만 아니라 저빈도 단어들의 색인 정보도 점차 늘어난다. 따라서 보조 색인정보 저장소의 크기가 지속적으로 커지면서 갱신 속도가 점점 느려지는 현상이 발생한다. 이러한 문제점을 해결하기 위해서 본 연구에서는 추가되는 문서 수에 따라 주기적인 병합(Merging)을 수행한다. 이때 병합 속도를 빠르게 하기위해서 기존의 병합 방식처럼 새로운 영역에 전체 색인정보를 재구성 하는 게 아니라 주 색인정보 저장소에 본 논문에서 제안하는 분할 구조를 활용한 펄싱 방식으로 병합한다. 이러한 방식을 사용하는 이유는 SSD의 특성상 연속된 공간에 위치하지 않아도 접근속도에 큰 영향을 받지 않기 때문이다. 이러한 분할구조 기반의 펄싱 및 병합 부분이 기존 방식들과 다른 특징이며 검색속도에 거의 영향을 미치지 않고 색인관리 성능을 크게 향상시킬 수 있는 핵심적인 원인이다.

위에서 정의한 SPM 의 알고리즘을 정의하면 [표 1]과 같다. 현재까지 추가된 문서의 빈도수가 분할 크기와 병합 크기보다 작으면 보조 색인에 포스팅 정보를 삽입(INSERT) 하고, 병합 크기보다 작지만 분할 크기 이상이면 보조 색인과 주 색인을 병합(MERGE)한다. 그리고 분할 크기보다 크고 병합 크기보다 작은 경우에는 보조 색인에 저장되어 있는 해당 세그먼트를 주 색인으로 펄싱(PULSING) 한다.

표 1. SPM 색인 관리 알고리즘

```
IF DF <= SS AND DF <= MS THEN
    INSERT(Pi) insert Pi into aux index
ELSE IF DF <= SS AND DF > MS THEN
    MERGE(aux) aux index merge with main
               index
ELSE IF DF > SS AND DF <= MS THEN
    PULSING(S) S(segment) pulsing into main
               posting space with the S list

SS : Segment Size, MS : Merge Size
Pi : Posting information,
aux : auxiliary index
```

[그림 1]은 SPM 주 인덱스 구조를 나타낸다. 세그먼트 크기는 3인 경우이며 흰색 노드는 초기 데이터를 구축(Build)해 놓은 것이며 검정 노드는 동적으로 추가된 노드들이다. 따라서 초기 구축된 노드들은 세그먼트 리스트로 구성되어있지만 연속된 공간(Contiguous region)에 저장되어있고 추가로 삽입된 노드들은 기존 노드에 여유 공간이 있는 경우 연속적으로 추가되고 여유 공간이 없는 경우 추가적인 세그먼트를 할당 받아서 링크로 연결된다. 포스팅 정보의 숫자들은 문서 식별자(document identifier)를 의미하며 문서가 추가된 순서대로 1부터 순차적으로 부여된다.

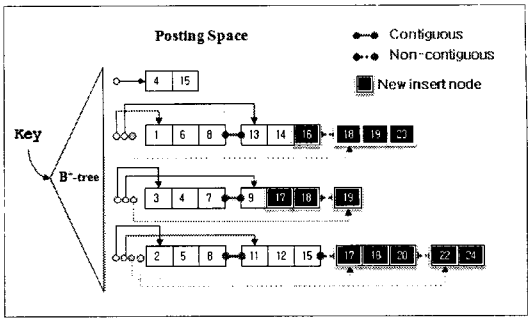


그림 1. 분할 색인 구조(Segmentation)

[그림 2]는 주 색인정보 저장소와 보조 색인정보 저장소 사이에서 색인정보가 펄싱되는 과정을 보여준다. 예를 들면 단어(Term) K의 경우 보조 색인 정보 저장소의 단어 K에 대한 빈도수가 세그먼트 사이즈 이상이 되면 주 색인정보 저장소로 해당 세그먼트 정보가 이동되는 펄싱작업이 수행된다. 이때 주 색인 정보 저장소에 단어 K가 존재하는 경우에는 마지막 링크 리스트에 세그먼트가 추가 되고, 단어 K가 존재하지 않는 경우에는 단어 K와 색인정보 세그먼트가 생성된다. 주 색인정보 저장소로 펄싱한 후 보조 색인정보 저장소에서는 해당 단어의 포스팅 정보를 삭제한다. 즉, 펄싱작업은 보조 색인 정보 저장소에는 항상 단어 당 하나의 세그먼트만 존재하도록 유지시켜 주는 역할을 한다.

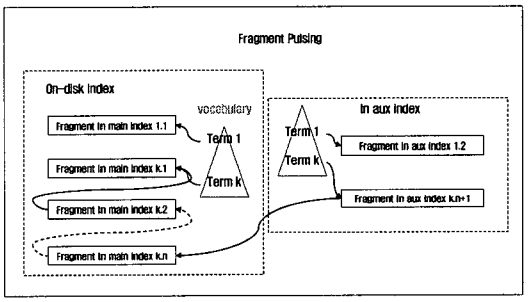


그림 2. 펄싱 색인 구조(Pulsing)

[그림 3]은 SPM 색인 저장구조의 전체적인 구조를 나타낸다. 주 색인정보 포스팅 구조는 고정길이 세그먼트 리스트로 구성되어 있으며 보조 색인정보 포스팅 구조는 가변길이 레코드로 구성되어 있다.

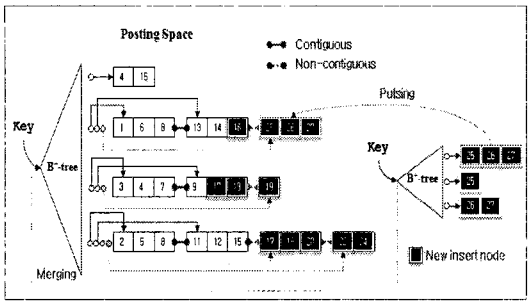


그림 3. 병합 색인 구조(Merging)

보조 색인정보 포스팅 크기가 임계치 이상이면 펄싱 과정을 통하여 주 색인정보 저장소로 병합된다. 또한 보조 색인정보 저장소의 크기가 지속적으로 커져서 색인 관리 속도가 점점 느려지는 문제점을 예방하기 위해서 주 색인정보 저장소와 보조 색인정보 저장소를 병합하는 과정을 주기적으로 수행한다. 즉 보조 색인정보 저장소의 모든 단어들에 대해서 주 색인정보 저장소로 펄싱 작업을 수행한다. 이때 기존 병합 방법과의 차이점은 주 색인정보의 포스팅 정보와 보조 색인정보의 포스팅 정보를 연속된 공간에 병합시키지 않고 링크로 연결한다는 점이다. 이러한 이유는 리스트(List) 방식으로 연결하는 경우 기존 병합 방식에서 요구되는 디스크 읽기/재할당(Reallocation) 연산이 필요 없어 처리 속도가 매우 빠르고 리스트 방식의 단점이었던 단편화로 인한 검색 속도 저하 문제는 SSD를 사용함으로써 보완되기 때문이다. 또한 SPM 색인 저장구조는 변경되는 페이지 또는 블록의 수가 포스팅 정보 전체가 아니라 실제 수정되는 문서 정보가 저장된 세그먼트로 한정된다. 따라서 기존의 동적 색인정보 수정 방법에 비해 SPM 방식은 매우 적은 페이지만 변경되기 때문에 색인정보 수정 속도가 크게 향상된다.

IV. 실험

본 논문에서는 2006년도에 등록된 미국 특허데이터 10만 건에 대한 색인 관리 속도 및 검색 속도를 측정한다. 실험장비는 쿼드 코어 CPU(Intel Xeon 2.83GHz), 메모리 16G, HDD(Dell 7200RPM SATA), RAID-5 SSD(Intel 80G * 4)를 사용한다. [표 2]는 실험데이터의 간략한 스키마 및 색인 단어의 통계 정보를 보여준다. 특허번호, 특허명, 등록일, 발명자, 요약, 청구항 등 총 90개의 필드로 구성 되어있고 평균 문서 크기는 8.92Kbyte이며 문서 당 평균 단어 수는 1,601개 이다.

표 2. 실험데이터 (USA PATENT)

Field	Data Type	Index Type	Total Terms/doc
PatNo	char[20]	index as is	1.0
Title	string	token	14.2
IssueDate	char[8]	index as is	1.0
Inventors	string	by token	10.8
Assignee	string	by token	3.57
Abstract	string	token	67.9
Claims	string	token	626.9
:	:	:	:
Class	string	index as is	1.0
SUM (90) fields			1601

SPM에 대한 동적 문서 관리 성능 테스트를 위해서 IP(In-Place), S(Segmentation), SP(Segmentation + Pulsing), SM(Segmentation + Merging) 그리고 SPM(Segmentation + Pulsing + Merging), 5가지 방식의 속도를 SSD 상에서 측정한다. SM 방식은 SPM방식의 부분으로서 세그먼트 크기와 병합주기가 동일한 경우에 해당한다. 관리 및 검색 속도에 매우 큰 영향을 주는 요소는 세그먼트 크기와 병합 주기이다. 그 중 세그먼트 크기는 수많은 실험을 바탕으로 다양한 서비스에 적용해본 결과 문서 빈도수(DF:Document Frequency)가 1K일 때 가장 우수한 성능을 보였다. 즉, 빈도수가 1K인 경우 세그먼트 크기는 포스팅 정보(12byte * 1K)와 부가정보(1K 미만) 의 크기를 더한 것으로 약16K보다 조금 작은 크기이다. 이는 디폴트 블록 크기(Block Size) 4K의 4배이며, 약간의 여유 공간이 있는 상태로서 In-Place 의 추가 할당과 같은 효과를 얻을 수 있는 영역이다. 따라서 본 실험에서는 세그먼트 크기를 가장 일반적으로 사용하는 시스템 환경을 고려하여 실험상 매우 우수한 성능을 보였던 1K로 고정하였으나, 시스템의 입출력 블록 크기, 버퍼 크기, 입출력 스케줄러의 알고리즘에 따라서 다른 크기가 더 우수할 수도 있다.

또한 병합 주기는 세그먼트 크기인 1K와 두 배인 2K의 경우를 측정하여 병합 주기에 따른 색인 관리 속도를 측정한다. 또한 추가적으로 HDD 상에서도 본 논문에서 제시하는 색인 관리 방식, S, SP, SPM의 성능을 실험하여 HDD에서의 성능 향상 정도를 측정한다.

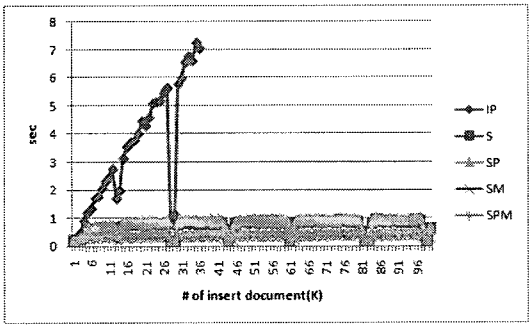


그림 4. 색인 수정 시간(SSD 사용)

[그림 4]는 SSD를 사용한 경우 5가지 방식의 문서 추가 속도를 나타낸다. 첫 번째 IP 방식은 가장 기본적인 구조의 In-Place 방식으로 추가 되는 데이터가 증가할수록 급격하게 관리 속도가 느려진다. 따라서 IP 방식은 데이터의 추가 작업이 빈번한 상황에는 매우 비효율적인 방식이다. 두 번째 S 방식은 보조 색인 저장구조를 사용하지 않고 주 색인저장 구조에 분할 방식만 적용한 경우이며 평균 문서 추가 시간은 0.8초 정도 소요된다. 세 번째 SP 방식은 보조 색인 저장 구조에 추가적인 문서를 저장하면서 펄싱 방법을 이용한 경우이며 약 0.8~1.0초 사이의 속도를 보인다. 펄싱 방식이 분할 방식 보다 문서 추가 시간은 약간 느리지만 HDD를 사용한 경우에는 검색속도가 빠른 장점을 가지고 있다. 단, SSD를 사용하는 경우에는 [그림 6]에서 보이는 것처럼 4가지 방식 모두 검색 속도가 거의 동일하다. 네 번째 SM 방식과 마지막 SPM 방식은 모두 분할 기반의 주 색인정보 저장 구조와 보조 색인정보 저장 구조를 사용하며 병합 방식을 적용한 경우이다. 두 방식의 차이점은 SPM 방식의 경우 보조 색인정보 관리에 펄싱 방법이 적용되어 검색 속도가 조금 빠르지만 색인 관리 속도는 SM 방식에 비해 약간 느리다. 실험 결과를 보면 SM 방식 또는 SPM 방식이 색인 관리 측면에서 S, SP방식에 배해 약 3배 정도 우수한 성능을 보인다.

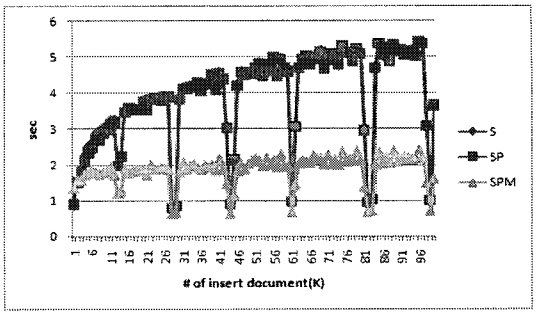


그림 5. 색인 수정 시간(HDD 사용)

[그림 5]는 HDD를 사용한 경우의 문서관리 속도를 나타낸다. 전체적으로 SSD를 사용한 경우에 비해 약 5배 정도 느리지만 각각의 방식에서 보여주는 특징은 SSD를 사용한 경우와 거의 동일하게 나타난다. S 방식과 SP 방식은 문서 추가 시간이 평균 4~5초 정도 소요되며 SPM 방식은 약 2.5배 정도 빠른 2초 정도 소요된다.

검색 성능 테스트를 위해서 위 3가지 방식과 전체를 재구성한 경우에 대하여 SSD를 사용한 경우와 HDD를 사용한 경우 각각의 속도를 측정하였다. [그림 6]에서 SSD를 사용한 경우는 모든 방식에서 약 30ms정도로 일정한 검색 속도를 보이고 있으며 HDD를 사용한 경우에는 검색에 최적화 된 Rebuild 방식에 비해 S 방식이 9.5배, SP 방식은 3.5배 그리고 SPM 방식은 7.5배 정도 느리다. HDD에서 SP 방식이 S방식과 SPM방식에 비해 빠른 이유는 긴 포스팅 색인정보와 짧은 포스팅 색인정보가 주 색인정보 저장소와 보조 색인정보 저장소에 분산 저장되어 있는 효과가 있기 때문이다.

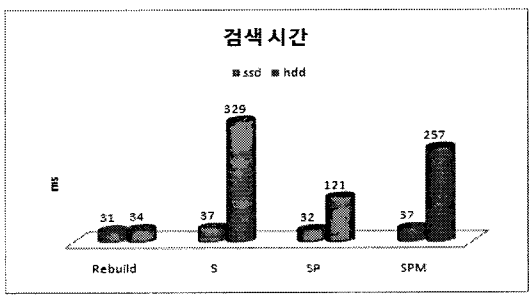


그림 6. 검색 시간

일반적으로 검색 시간과 색인정보 관리 시간은 역 상관관계(Trade-off)를 가지고 있다. 이러한 현상은 본 실험에서도 HDD를 사용한 경우에는 동일하게 나타난다. 그러나 SSD를 사용한 경우 본 논문에서 제안하는 SPM 방식을 적용하면 검색 시간은 거의 변화 없이 색인정보 관리 시간이 크게 향상된 결과를 볼 수 있다. 구체적으로 살펴보면 본 논문에서 제안하는 SPM 방식이 HDD 상에서 검색에 최적화된 구조보다 7.5배 정도 느리지만 색인정보 관리 성능에서는 S 방식에 비해 약 2.5배, SP 방식에 비해 약 2.4배 빠르다. 그리고 SSD 상에서는 검색속도는 거의 동일한 반면 색인 정보 관리 성능은 S 방식, SP 방식에 비해 약 3배 정도 빠르다. 뿐만 아니라 S 방식, SP 방식을 사용하지 않는 가장 기본적인 In-Place 방식의 인덱스 구조에 비해서 색인 정보 관리 속도는 수십~수백 배 이상 빠른 성능을 보인다.

V. 결론

본 논문에서는 온라인 색인 관리 측면에서 가장 널리 활용되고 있는 In-Place 방식과 병합기반(Merge-based) 색인 방식의 장단점을 분석하고 이 방법들의 장점을 활용하고 단점을 보완하여 검색 처리 시간에 최소한의 영향을 주면서 동시에 색인정보 관리(index maintenance) 시간을 최대한 줄일 수 있는 색인 저장 구조를 제안하였다. 또한 최근 기존 저장장치(HDD)와 근본적으로 구조가 다른 새로운 저장장치(SSD, SCRAM)에 적합한 색인정보 저장관리 구조를 제시하였다. 다시 말하면 본 논문에서는 SSD의 매우 낮은 접근시간(Low access latency) 특성을 최대한 활용할 수 있는 분할(Segmentation) 포스팅 구조를 기반으로 SSD에 적합하도록 변형된 In-Place 색인 방식(Pulsing)과 Merge-based 방식(Merging)을 효과적으로 조합하여 검색 처리시간 및 색인정보 관리시간을 크게 향상시킬 수 있는 새로운 색인저장 구조(SPM)를 제안하고 성능을 평가하였다.

참 고 문 헌

[1] D. R. Cutting and J. O. Pedersen, "Optimization for dynamic inverted index maintenance," In Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval, pp.405-411, 1990.

[2] T. Chiueh and L. Huang, "Efficient real-time index updates in text retrieval systems," Technical Report ECSL-TR-66, Computer Science Department, SUNY at Stony Brook, 1999.

[3] L. Lim, M. Wang, S. Padmanabhan, J. S. Vitter, and R. Agrwal, "Dynamic maintenance of web indexes using landmarks," In Proceedings of the 12th international conference on World Wide Web, pp.102-111, 2003.

[4] N. Lester, J. Zobel, and H. Williams, "In-Place versus Re-Build versus Re-Merge: Index Maintenance Strategies for Text Retrieval Systems," In Computer Science, 27th Australasian Computer Science Conference, pp.15-22, 2004.

[5] N. Lester, A. Moffat, and J. Zobel, "Efficient Online Index Construction for Text Database," J. of ACM Trans. Database Systems, Vol.33, No.3, Article 19, 2008.

[6] A. Tomasic, H. Garcia-Molina, and K. A. Shoens. "Incremental updates of inverted lists for text document retrieval," In Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, pp.289-300, 1994.

[7] S. Buttcher, C. L. A. Clarke, and B. Lushman, "Hybrid index maintenance for growing text collections," In Proceedings of the 2004 ACM SIGMOD international conference on

Management of data, pp.1-4, 2004.

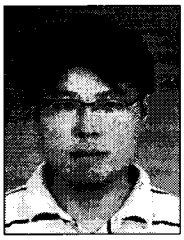
[8] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron, "Migrating enterprise storage to SSDs: analysis of tradeoff," MSR-TR-2008-169, 2008.

[9] John A. Garrison and A. L. Narasimha Reddy, "Umbrella File System: Storage Management across Heterogeneous Devices," ACM Trans. Stor., Vol.5, No.1, Article 3, 2009.

[10] S. W. Lee and M. Bongki, "Design of Flash-Based DBMS: An In-Page Logging Approach," In Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pp.55-66, 2007.

저 자 소 개

진 두 석(Du-Seok Jin) 정회원



- 1999년 2월 : 전북대학교 컴퓨터공학과(공학사)
- 2001년 2월 : 전북대학교 컴퓨터공학과(공학석사)
- 2001년 2월 ~ 현재 : 한국과학기술정보연구원 정보기술연구실

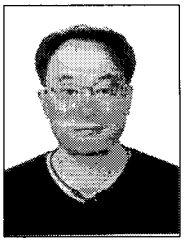
<관심분야> : 정보검색, 저장구조

김 진 숙(Jinsuk Kim) 정회원

- 1995년 2월 : 한국과학기술원 생명과학과(공학석사)
- 2002년 2월 : 한국과학기술원 전산학과(공학석사)
- 2002년 2월 ~ 현재 : 한국과학기술정보연구원 정보기술연구실

<관심분야> : Bio-informatics, 문서분류, 정보검색

류 범 종(Beom-Jung You) 정회원



- 2000년 2월 : 충남대학교 문헌정보학(이학석사)
- 2005년 8월 : 충남대학교 문헌정보학(이학박사)
- 2001년 1월 ~ 현재 : 한국과학기술정보연구원 정보기술연구실장

<관심분야> : 시멘틱웹, 지식베이스, 마이닝

정 회 경(Hoe-Kyung Jung) 정회원

- 1987년 2월 : 광운대학교 컴퓨터공학과(공학석사)
- 1993년 2월 : 광운대학교 컴퓨터공학과(공학박사)
- 1994년 3월 ~ 현재 : 배재대학교 컴퓨터공학과 교수

<관심분야> : MPEG-21, XML, Semantic Web