FAST-INV를 이용한 정보검색 시스템에서의 B-트리의 병렬접근에 대한 연구

김수영, 고지현, 박순철
전북대학교 정보통신공학과
e-mail:avenger@internet.chonbuk.ac.kr

A Parallel Approach on the B-tree for an Information Retrieval System using FAST-INV

Soo-Young Kim, Ji-Hyun Goh, Soon-Chol Park
Dept of Information and Communication, Chonbuk-Buk University

요약

인터넷 상의 문서양이 기하급수적으로 증가하면서 검색엔진의 성능평가가 대두되고 있다. 이를 위해 검색엔진 인덱스 모듈부분의 좋은 성능이 요구되는데 빠른 대용량 역파일 구성을 위한 알고리즘을 사용하게 되면 인덱스 속도를 향상시킬 수 있다. 그러나, 병렬처리가 되지 않는 문헌벡터화일 제작시 트리검색 모듈에서 병목 현상이 발생하게 된다. 본 논문에서는 병목현상이 발생하는 트리를 병렬로 접근함으로써 시스템의 병목현상을 해소하고 인덱스 시스템의 전체적인 성능을 개선할 수 있는 방안을 연구한다.

1. 서론

최근 인터넷상 문서양이 급속도로 증가하면서 검색엔진의 성능이 인터넷 정보검색에서 더욱 중요한 요소가 되었다. 검색엔진을 통해 구성된 인터넷의 자료들은 시간이 지남에 따라 삭제·변경된다. 현재 사용되는 대부분의 검색엔진에서는 현재 존재하는 색인을 업데이트하지 않으며 일정한 시간이 지난 후에 색인을 다시 구성하는 방식을 사용하고 있다. 이런 방식의 경우에 색인주기에 따라 정보의 신용도가 결정된다. 따라서, 색인 시스템의 속도에 따라 검색엔진의 신뢰도와 성능이 좌우된다고 할 수 있다. 지금까지 검색엔진의 성능을 개선하기 위한 색인방법에 대한 많은 연구가 있었다.

본 논문에서는 전통적인 FAST-INV알고리즘을 사용하여 역화일을 구성하며 FAST-INV의 특성상 병렬처리시 병목 현상이 발생하게 되는 색인어 검색부분의 병렬 알고리즘을 제안함으로써 FAST-INV방식의 시스템에서 향상된 성능을 낼 수 있다.

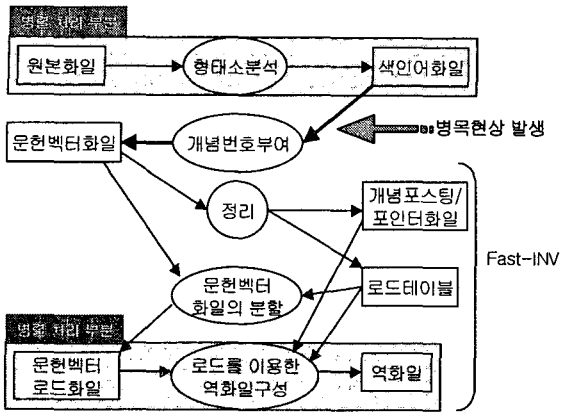
2. FAST-INV 알고리즘을 이용한 정보검색

2.1 FAST-INV 알고리즘의 장점

FAST-INV는 정렬된 배열의 역화일을 만들기 위한 알고리즘으로 대용량의 주기억장치와 입력자료에서부터 얻어지는 순서를 갖는 장점을 갖는다. 입력이 정렬되어 있으므로 병렬처리시 전체 데이터를 적은 양으로 분할하여 각각 처리하게 된다. 디스크의 데이터양이 많다 하더라도 분할되어 주기억장치로 적재되어 처리될 수 있어서 시스템의 비용이 절감되게 된다.

이 알고리즘에서 병렬처리의 이점은 다음의 (그림 1)에서 반전되어 보이는 병렬처리 부분이다. 초기의 형태소 분석은 출력이 다른 프로세서에 영향을 미치지 않으므로 병렬로 처리될 수 있으며, 역화일을 구성하는 부분은 FAST-INV의 핵심 부분으로 각 데

이터를 로드로 분할하고 각각 처리하여 역화일에 저장하게 된다.



(그림 1) Fast-INV알고리즘과 전처리기의 구조도

2.2 색인 시스템에서의 병목현상

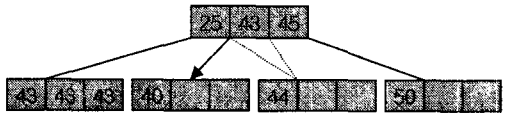
한글 정보검색 시스템에서는 로봇이 수집해온 문서를 태그를 없앤 후 형태소 분석을 거쳐 색인어에 해당하는 단어를 추출해낸다. 일반적으로는 이 색인어만을 가지고 작업을 하게 되지만 FAST-INV에서는 색인어를 개념번호로 치환·정렬하여 문헌벡터화일로 만든다. 색인어를 개념번호로 치환할 때 대용량의 색인어를 B-Tree를 통해 구현하고 이를 이용한다. 이 B-Tree는 삽입과 검색이 빈번하게 발생하게 되므로 여러 프로세서가 하나의 B-Tree를 접근하게 되면 무결성이 손상되며 하나의 프로세서가 배타적으로 트리를 탐색하여 문헌벡터화일을 제작하게 되면 형태소 분석부와 역화일 구성부의 병렬처리의 이점이 문헌벡터화일 제작부에서 손실되게 된다. 따라서, 다중 프로세서가 B-Tree를 병렬로 접근하는 lock메카니즘을 적용하여 병목현상을 해결하면 문헌벡터화일 제작의 속도가 향상되므로 전반적인 색인 모듈부분의 성능을 향상시킬 수 있다.

3. Tree에 대한 병렬 접근

3.1 B-Tree에 대한 병렬 접근의 문제점

B-Tree를 사용하는데 있어 병렬로 접근할 때 다음과 같은 상황이 발생할 수 있다. (그림2)에서 트리에 두 개의 프로세서가 동시에 "41"을 삽입(p_1)하고 "44"를 검색(p_2)하는 경우를 살펴보자.

p_1이 먼저 작동하게 되면, 가운데 노드가 분할된다. 그 순간에 하위 노드로의 포인터가 변경된다. 이 때, p_2가 "44"를 검색하면 "44"가 들어있는 노드의 포인터의 위치가 변동되어 검색 시 오류가 발생한다. 삽입을 위해서 트리가 분할될 때, 검색 프로세서가 작동하게 되면 오동작 가능성이 생기게 된다. 또한, "43"을 검색한다 하더라도 분할의 순간에는 데이터의 위치가 불분명해지게 된다. 따라서, 트리를 병렬로 삽입·검색하게 되면 이와 같은 오류가 발생하게 된다.



(그림 2) Leaf 노드의 분할

3.2 Ellis 알고리즘

AVL트리에 대한 병렬 접근 알고리즘인 Ellis알고리즘에서는 ρ, α, ξ lock을 이용하여, 회전하는 트리에 대한 병렬 접근을 가능하게 했다. root로부터 진행하여 subTree의 높이가 틀린 노드를 critical node로 설정하고 critical node를 기준으로 lock을 사용한다.

- ρ lock : 검색 프로세서가 설정하며, 다중 프로세서가 하나의 ρ lock을 공유할 수 있다.
- α lock : 삽입 프로세서가 다른 프로세서의 삽입을 차단하기 위해서 사용하며, α lock이 설정되어 있을 경우에는 삽입프로세서의 접근이 불가능하다.
- ξ lock : 회전이 일어나는 노드에 검색 프로세서가 들어오는 것을 배제하기 위해서 사용한다.

• ρ와 ξ는, α와 ξ는 서로 배제하며 ρ와 α는 노드를 공유한다.

4. B-Tree에서의 병렬접근 알고리즘

4.1 B-Tree에 대한 기본적인 제한사항

B-Tree에 Ellis알고리즘을 적용하는데 몇 가지 문제가 발생한다. B-Tree는 노드의 수가 홀수인 경우와 짝수일 경우의 알고리즘에 차이가 있다. 노드의 수가 짝수일 경우에는 메모리 공간이 충분히 활용되는 장점이 있는 대신에 역추적을 해야 하므로 Ellis 알고리즘을 적용시키는데 문제를 발생시킨다. 본 논문에서는 노드의 수가 홀수인 경우에 대해서만 방법을 제시한다. 또한, 정보검색 시스템에서는 키를 삭제하는 경우가 거의 발생하지 않으므로 삭제의 경우는 고려하지 않는다. AVL 트리에서는 critical node를 선정하여 기준을 설정하지만 B-Tree에서는 critical node의 선정이 용이하지 않다. 따라서, Ellis

알고리즘에서는 critical node의 아래 노드들에 lock을 설정했지만, B-Tree에 적용할 때 각 노드에 lock을 설정하며 검색•삽입을 처리한다.

4.2 Ellis 알고리즘의 B-Tree 적용

앞에서 설명한 제한된 상황하에서 B-Tree에 Ellis 알고리즘을 적용시킨다. 각 노드단위로 lock을 설정하는 페이지단위의 lock메카니즘을 사용한다. 하나의 Search lock을 여러 프로세서가 공유하기 때문에 "The Readers and Writers Problem"이 발생하게 된다. 이 경우 ***P*** lock을 사용하여 write first algorithm을 적용한다.

B-Tree에 병렬로 키를 삽입•검색하기 위해서 노드에 4개의 lock 필드를 정의하고 동작을 제어한다.

• **R lock**(Root lock) : root노드를 lock시켜 하나의 프로세서만이 트리에 삽입 할 수 있게 한다.

• **I lock**(Insert lock) : 삽입을 위한 프로세서는 탐색 도중에 노드를 분할하고 포인터를 변경하기 때문에 I lock을 통해 다른 프로세서의 접근을 막는다.

• **S lock**(Search lock) : 프로세서가 노드를 검색하고 있을 때, 다른 프로세서가 접근하여 노드 분할을 하는 경우를 방지하기 위해서 사용한다. 다중 접근이 가능하며 접근 프로세서의 수로 정한다.

• **P lock**(Preoccupy lock) : 다중의 프로세서가 S lock을 공유하며 lock이 설정되어 있는 경우 P lock을 설정하여 노드에 대해 선점을 한다. 이후, S lock이 해제되면 I lock을 설정하고 P lock을 해제한다.

• I, S, P Lock의 공통사항 : 탐색 경로중의 모든 노드에 lock이 설정되며 현재 탐색중인 노드에 lock을 설정하고 다음 노드를 찾아 움직일 때 lock을 해제한다.

• 검색 프로세서는 다음 진행노드에 I lock이 설정되어 있으면 현재 상태에서 대기한다. I lock이 해제되면 현재의 노드를 다시 검색한 후 다음으로 진행한다.

[B-Tree 병렬접근 알고리즘]

p_i : i번째 프로세서

• time(p_i) : 프로세서 p_i의 시작시각

$node_{root/pre/now/next}$: 루트/이전/현재/다음 노드

p_1:search A, p_2:insert B, p_3:search C, p_4:insert D

• 조건 : 트리에서 같은 Path를 탐색, 노드에 동시접근

time(p_1) < time(p_2) < time(p_3) < time(p_4)

• 트리 탐색시 하나의 노드에 대해서 발생하는 상황

①	insert ▶ insert
②	insert ▶ search
③	search ▶ insert
④	search ▶ search

i) ($p_1/p_2/p_3$ 작동) - ②, ③의 경우

begin
 p_1 lock $node_{now}$ using S
 p_2 lock $node_{root}$ using R then lock $node_{now}$ using I
 p_3 wait for unlock P
 p_1 unlock $node_{now}$ / S then lock $node_{next}$ using S
 p_2 lock $node_{now}$ using I then unlock $node_{now}$ / P
 p_2 Unlock $node_{now}$ / P then unlock $node_{now}$ / I
 if($node_{next}$ locked by S) p_2 lock $node_{next}$ using P
 else p_2 lock $node_{next}$ using I
 p_3 lock $node_{now}$ / S
end

ii) ($p_2/p_3/p_4$ 작동) - ①, ②, ③의 경우

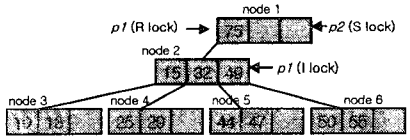
begin
 p_2 lock $node_{root}$ using R then lock $node_{now}$ using I
 p_3 wait for unlock I
 p_4 wait for unlock R
 p_2 unlock $node_{now}$ / I then unlock $node_{root}$ / R
 p_2 **terminate**
 p_3 lock $node_{now}$ using S
 p_4 lock $node_{root}$ using R then lock $node_{now}$ using P
end

iii) (p_1/p_3 작동) - ④의 경우

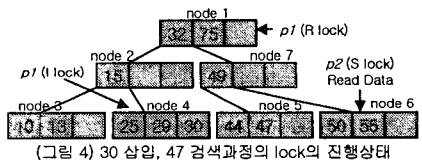
begin
 p_1 lock $node_{now}$ using S
 p_3 share lock S
 p_1 unlock $node_{now}$ / S then lock $node_{next}$ using S
 p_3 unlock $node_{now}$ / S then share lock S
end

앞에서 설명한 바를 아래의 예에서 실제로 적용시켜 보았다. <표1>과 <표2>는 시간에 따른 프로세서와 노드의 상태를 나타낸 것이다.

예1) p_1 : *key30* 삽입, p_2 : *key47* 검색



(그림 3) 30 삽입, 47 검색의 과정에서 lock의 상태



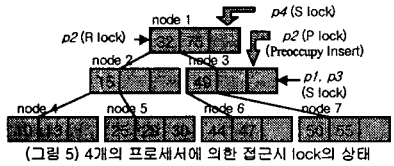
(그림 4) 30 삽입, 47 검색과정의 lock의 진행상태

	p_1(*Insert 30*)	p_2(*Search 47*)
초기상태 (그림 3)	*R lock* (node root) *I lock* (node 1) *I unlock* (node 1) *I lock* (node 2)	**waiting** for *S lock* *S lock* (node 1)
진행상태 (그림 4)	**Split** (node 2) *I unlock* (node 2) *I lock* (node 4) *I unlock* (node 4) *R unlock* (node root) **Terminate**	**waiting** for *S lock* *S unlock* (node 1) *S lock* (node 7) *S unlock* (node 7) *S lock* (node 5) *S unlock* (node 5) **Terminate**

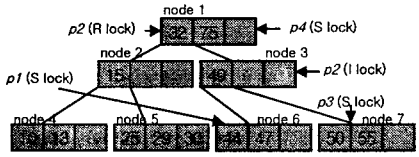
<표 1> 병렬 삽입시 프로세서와 노드의 상태

예2) p_1 : *key44* 검색, p_2 : *key60* 삽입

p_3 : *key50* 검색, p_4 : *key49* 검색



(그림 5) 4개의 프로세서에 의한 접근시 lock의 상태



(그림 6) 다중 접근 진행시 node의 lock 상태변화

	p_1(*search 44*)	p_2(*Insert 60*)	p_3(*Search 50*)	p_4(*Search 49*)
초기상태 (그림 5)	*S lock*(N_3) *S unlock*(N_3)	P lock(N2)	*share S lock* (N_3)	
진행상태 (그림 6)	*S lock*(N_6) *S unlock*(N_6) **Terminate**	*I lock*(N_2) *P unlock*(N_2) *I unlock*(N_2) *I lock*(N_7) **insert** *60* *I unlock*(N_7) **Terminate**	*S lock*(N_7) *S unlock*(N_7) **Terminate**	S lock(N_1) **waiting** for *S lock* *S unlock*(N_1) *S lock*(N_3) *S unlock*(N_3) **Terminate**

<표 2> 다중프로세서의 트리 접근시 노드와 lock 상태

5. 결론 및 앞으로의 연구방향

본 연구는 홀수개의 노드를 가지는 B-Tree에 대한 병렬접근을 lock메카니즘을 통해서 FAST-INV 알고리즘에서의 병목현상을 해결했다. AVL 트리에서의 병렬접근을 가능하게 해주는 Ellis알고리즘의 lock메카니즘을 B-Tree에 적용함으로써 병렬접근을 가능하게 하며 "The Readers and Writers Problem"문제를 해결한다. 또한, B-Tree에 대한 병렬 접근 알고리즘은 대용량의 색인모듈뿐만이 아니라 화일기반의 여타 정보검색 시스템에서도 널리 활용될 수 있다.

앞으로의 이 알고리즘의 개선 방향은 많은 제한점들을 개선하여 사용범위를 더욱 확대하고 일반적인 B-Tree에의 적용과 하나의 삽입 프로세서만이 트리에 접근하는 문제를 해결하도록 연구를 진행해야 할 것이다.

참고문헌

[1] Quinn, Michael J., "Designing Efficient Algorithms f or Parallel Computers", McGraw-Hill, 1987

[2] William B.Frakes and Richard Baeza-Yates., "Inform ation Retrieval.", Prentice Hall, 1992

[3] 이재규, "C로 배우는 알고리즘", 도서출판 세화, 1996

[4] Richard Baeza-Yates. , "Modern Information Retriev al.", Addison Wesley, 1999

[5] Abraham Silberschatz, Henry F. Korth and S. Sudar shan, "Database System Concepts", McGraw-Hill, 19 97

[6] Abraham Silberschatz and Peter Galvin, "Operating System Concepts", John Wiley & Sons Inc., 1997

[7] Robert R. Korfhage., Information Storage and Retrie val.,1997

[8] 한국어정보처리연구소, C로 구현한 인터넷 정보검색 시스템, 1999

김 수 영

1993-2000 전북대 정보통신공학과

2000-현재 전북대 정보통신공학과 인터넷 연구실

관심분야 : 데이터구조, 정보검색

고 지 현

1994-1999 전북대 자연과학대학 수학과

2000-현재 전북대 정보통신공학과 인터넷 연구실

관심분야 : 정보검색, 데이터마이닝

박 순 철

1972-1979 인하대 응용물리학과

1986-1991 미국 루이지애나 주립대 전자계산 박사

1991-1993 한국 전자통신 연구소 데이터베이스 연구실 과제 책임자

1993-현재 전자정보공학부 부교수

관심분야 : 데이터베이스, 네트워크 컴퓨팅, SAN(Storage Area Network),