플래시 메모리 상에서 고차원 데이터의 색인을 위한 효율적인 입출력 방법

유정수 낭종호
서강대학교 컴퓨터공학부
yjs@mlneptune.sogang.ac.kr, jhnang@sogang.ac.kr

Efficient I/O Scheme for Indexing High-dimensional data on Flash Memory

Jeongsoo Yu Jongho Nang
Department of Computer Science and Engineering, The Sogang University of Korea

1. 서 론

최근 들어 플래시 메모리는 이러한 사용자의 요구에 맞추어 가격 하락과 성능 향상을 이루어 대용량화 되는 추세이며, 이에 따라 휴대용 기기에 대량의 사진을 보관하는 일이 많아졌다. 따라서, 휴대용 기기에 적합한 효율적인 검색 방법을 필요로 하며, 내용 기반의 클러스터링을 통한 계층적 브라우징이 가능한 방법 중 하나이다. 내용 기반 검색은 거리 공간 기반의 고차원 색인이 필요하며[1] M-Tree 계열[2][3][4]이 대표적인 색인 방법이다. 대부분의 거리 공간 기반의 색인은 디스크 기반의 인덱싱을 고려하였기 때문에 입출력 성능에 크게 영향을 받는다. 따라서, 휴대용 기기에서 디스크 기반의 인덱싱을 플래시 메모리 상에서 구축할 때 플래시 메모리의 특성을 고려할 필요가 있다[5][6][7]. 플래시 메모리의 가장 큰 특성은 쓰기 연산을 할 때에 소거 연산을 병행함으로 쓰기 비용이 읽기 비용에 비해서 매우 크다는 점이다[8]. <표 1>은 플래시 메모리와 하드 디스크에 대한 특성 비교이다[6].

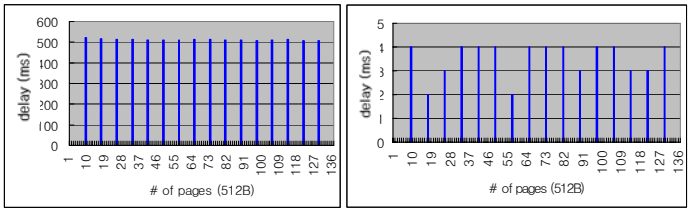
<표 1> 플래시 메모리와 하드 디스크의 성능 비교

	Read	Write	Erase
NAND Flash	36㎲(512B)	226㎲(512B)	2ms(16KB)
Hard Disk	12.4ms(512B)	12.4ms(512B)	-

하지만 위와 같은 특성은 플래시 자체만을 고려한 이론적인 정보이다. 실제 응용에서는 버스, 컨트롤러, FTL(Flash Translation Layer), 시스템의 캐싱 메커니즘, 커널의 인터럽트 관리에 이르기까지 다양한 영향을 받는다. 따라서 응용 레벨에서 <표-1>의 특성만을 기준으로 하여 색인의 효과를 높이는 방법으로는 정확한 성능 향상을 얻기 힘들다. 이러한 응용에서의 실제 체감은 시스템에 따라 다양하겠지만, 본 연구에서는 가장 널리 쓰이는 WinCE PDA를 대상으로 플래시 상에서 고차원 데이터 색인의 성능을 향상시키는 방법을 통하여 향후 다양한 시스템에서의 응용을 위한 해법을 찾을 수 있는 기반을 마련하고자 한다.

2. 본 론

<그림 1-a>는 파일의 끝부분에 크기를 증가시키면서 순차적으로 쓴 결과이며, <그림 1-b>는 파일의 앞부분부터 다시 쓰기를 하는 과정의 결과이며, 쓰기의 경우에 파일 시스템의 페이지 캐시 버퍼의 효과로 인한 4096바이트 간격으로 버스트 현상이 나타나고, 특히 끝부분에 새로 쓰기의 버스트는 그 크기가 약 500ms 정도로 매우 크다. 이는 응용 레벨에서, 새로 쓰기의 경우 심각한 지연 현상이 나타나는 것을 고려해야 하며, 특히 M-Tree와 같은 색인 파일은 일반적으로 노드의 용량이 차면 분할을 통하여 크기가 증가하기 때문에 이러한 특성은 매우 중요함을 알 수있다.

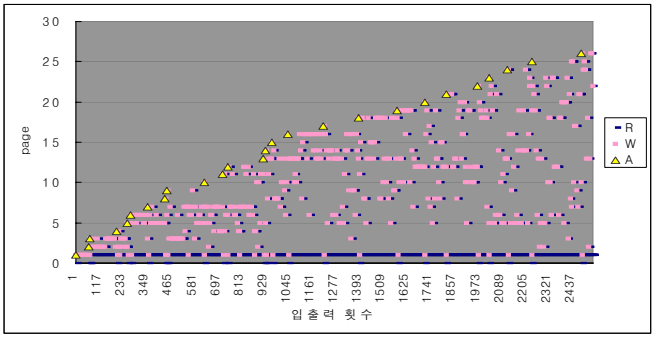


a. 순차적 새로 쓰기 b. 순차적 다시 쓰기

<그림 1> WinCE에서 SD카드의 파일에 대한 입출력 측정

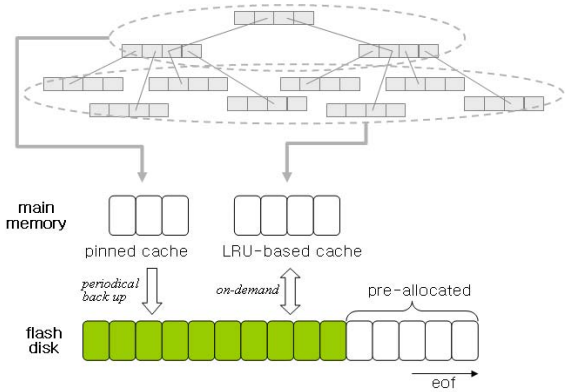
플래시 메모리상에서 고차원 데이터 색인을 사용할 때 입출력 성능을 분석하기 위해서, 본 연구에서는 SD카드 상에서 거리 공간 기반의 대표적인 색인 방법인 M-Tree를 사용하였다. 데이터는 실제 휴대용 기기에 저장된 사진 파일로부터 추출된 MPEG-7 Color Structure Descriptor[9]를 사용했으며, 유사도 거리는 L_2, 차원은 128이고, 데이터 개수는 500개, 페이지 크기 (M-Tree Node 크기)는 32KB이며 입출력은 페이지 단위로 이루어 진다. <그림 2>는 색인 구축 동안의 디스크 접근 패턴으로, R은 읽기, W는 다시 쓰기, A는 새로 쓰기를 나타낸다. 새로 쓰기의 경우 리프 노드의 분할 시 발생하며, 이로 인한 업데이트가 루트 노드인 1번 페이지까지 이어져서 루트 노드에 다시 쓰기 연산이 발생하는 것을 볼 수있다. 또한 M-Tree 는 루트부터 아래 방향으로 데이터가 포함 될 적합한 노드를 찾는 식으로 데이터 삽입이 이루어 지므로, 루트 노드는 데이터 삽입때마다 읽기 연산이 발생하며, 또한 전체적으로 볼때 지역성(locality)이 발견되었다. 이는 데이터 셋의 특성과 삽입 순서에 따른 결과로서 항상 지역성이 발생할 것이라고는 볼 수없다. 하지만, 본 연구가 휴대용 기기에서 찍힌 사진 파일에 대한 검색을 위한 색인을 대상으로 한다는 전제하에, 사용자가 찍는 사진은 이벤트에 따라 클러스터링 되며 각 클러스터는 비슷한 배경 및 조명등의 특성을 가지므로[10][11] 지역성

을 가질 확률이 높다는 특성을 얻을 수 있다.



<그림 2> 플래시 메모리상의 M-Tree 구축 시 디스크 접근 패턴

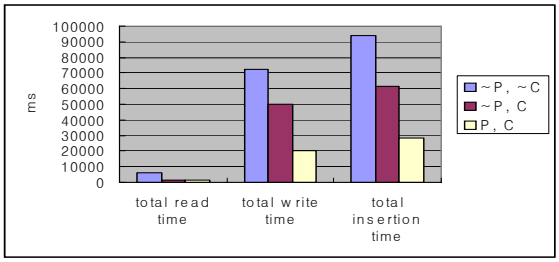
본 연구에서는 상기 내용을 기반으로, 파일 크기 증가시 새로 쓰기를 막기 위한 미리 할당하는 방법과 지역성을 활용하여 페이지 단위의 응용 레벨에서의 버퍼 캐시를 사용하는 기법을 통하여 입출력 성능 향상 방법을 제안한다. <그림 3>은 제안하는 방법을 설명하는 구조도이다. 상위 노드는 리프 노드가 분할될 때마다 지속적으로 업데이트 되며, 데이터 삽입때마다 항상 읽혀지므로 메인 메모리상의 캐시에 고정(pinning)시킨다. 만일 색인 파일의 무결성을 고려한다면 주기적으로 백업하게끔 할 수 있으며, 이를 통하여 매번 디스크에 직접 접근에 비하여 입출력 비용을 절약 할 수가 있다. 하위 노드 및 리프 노드들은 정해진 캐시 페이지에 일반적인 LRU(Least Recently Used) 기법으로 캐싱하므로써 지역적인 접근을 이용하여 디스크 접근 횟수를 줄인다. 또한, 일정량의 페이지를 미리 써놓으므로써 입출력 비용에서 가장 큰 부분을 차지하는 새로 쓰기 비용을 줄일 수 있다. 미리 쓰기 역시 새로 쓰기 비용이 발생하지만, 본 연구에서 대상으로 하는 휴대용 기기의 사진 파일에 대한 색인 구축 문제는 사용자가 사진을 찍었을 때 저장 시 색인 삽입 시간을 줄이는 것이므로, 미리 쓰기를 유휴(idle) 상태 또는 초기화 상태에서 처리 하므로써 삽입 지연 시간을 줄일 수 있는 효과가 있다.



<그림 3> 버퍼 캐시 사용과 미리 할당하는 방법에 대한 구조도

3. 결 론

<그림 4>는 본 연구에서 제안한 방법을 실험 결과로써, ~P는 미리 쓰기를 수행 하지 않는 경우이고 P는 그 반대의 경우이며 ~C는 캐싱을 하지 않은 경우이고 C는 그 반대의 경우이다.
<그림 4>에서 총 삽입 시간을 비교해 보면, 캐싱을 하였을 경우 약 65%로 감소하였으며, 캐싱과 미리 쓰기를 함께 적용하였을 경우 약 30%로 감소하였다. 본 연구는 WinCE를 대상으로 진행하였으며, 다른 시스템에서도 비슷한 접근 방법을 통하여 응용 레벨에서의 입출력 성능을 향상시키는 것이 가능할 것이다. 또한 본 연구에서는 색인 알고리즘으로는 M-Tree를 전제하에 입출력 성능을 고려하였으며, 향후 응용 레벨에서의 플래시 메모리 특성을 고려한 색인 알고리즘에 대한 연구로 확장 시킬 계획이다.



<그림 4> 1000개 데이터 세트에서 삽입 비용의 비교

참고문헌

[1] P. Zezula, G. Amato, V. Dohnal and M. Bako, 2006, *Similarity Search The Metric Space Approach*, Springer

[2] P. Ciaccia, M. Patella, F. Rabitti, and P. Zezula, "Indexing metric spaces with mtree", *in Proceedings of the 23rd International Conference on Very Large Data Bases, Athens, Greece, August 25-29, 1997*, pp. 426-435

[3] X. Zhou, G. Wang, J. X. Yu, and G. Yu, "M+-tree: A new dynamical multidimensional index for metric spaces," in *Proc. Fourteenth Australasian Database Conf. Database Technologies 2003*, Adelaide, Australia, *Feb. 2003*, pp. 161-168.

[4] C. Traina, Jr., A. J. M. Traina, B. Seeger, and C. Faloutsos, "Slim-trees: High performance metric trees minimizing overlap between nodes," in *Proceedings oEDBT 2000, Konstanz, Germany, Mar. 2000*, pp. 51-65.

[5] Chin-Hsien, Li-Pin Chang, and TeiWei Kuo, "An Efficient B-Tree Layer for Flash-Memory Storage System", *in Proceedings RTCSA, Tainan, Taiwan, 2003b*, pp. 409-430.

[6] , "F-Tree : 휴대용 정보기기를 위한 플래시 메모리 기반 색인 기법", *in Proceedings Journal of Information Technology Applications & Management Vol.13, No.4, Dec. 2006*, pp. 257-271

[7] Chin-Hsien Wu, Li-Pin Chang, and Tei-Wei Kuo. "An Efficient R-Tree Implementation over Flash-Memory Storage Systems", *in Proceedings of the 11th ACM international symposium on Advances in geographic information systems, New Orleans, 2003.* pp. 17-24,

[8] Keun Soo Yim, Jihong Kim, and Kern Koh, "A Fast Start-Up Technique for Flash Memory Based Computing Systems", *2005 ACM Symposium on Appkied Computing, Santa Fe, New Mexico, USA, March, 2003*, pp. 843-849.

[9] *ISO/IEC JTC 1/SC 29/WG 11/N3966 15938-5* "Information Technology Multimedia Content Description Interface PartMultimedia Description Schemes".

[10] A. Girgensohn, J. Adcock, , M. Cooper, J. Foote, and L. Wilcox, "Simplifying the management of large photo collections", *in Proceedings of Human-Computer Interaction, IOS Press (2003)*

[11] K. Rodden and K. Wood, "How People Manage Their Digital Photos?", *ACM CHI, USA, 2003 (to appear)*