캐시 및 선반입 정책 결정을 위한 액세스 패턴 분류 방법

석성우, 김재열, 서대화
경북대학교 전자공학과
swsok@palgong.knu.ac.kr, redgauri@palgong.knu.ac.kr, dwseo@ee.knu.ac.kr

A method of access pattern classification for decision of caching and prefetching policies

Song-Woo Sok, Chei-Yol Kim, Dae-Wha Seo
Dept. of Electronic Engineering, Kyungpook National University

요 약

대용량 데이터 베이스, 멀티미디어 서버, 복잡한 과학 기술계산 등의 대용량의 데이터를 처리해야 하는 응용이 증가하고 있다. 이에 따라 파일 서비스의 속도를 높이기 위한 연구가 다방면에서 진행되고 있다. 파일 서비스 속도를 결정하는 중요한 두 부분 중의 하나인 액세스 시간 개선을 위해서 좀더 효율적인 캐시, 선반입을 수행하기 위한 선수작업으로써 액세스 패턴을 실시간으로 분석하여 특징을 추출하고 관리하는 방법을 연구하였다.

1. 서론

디스크 저장 장치의 기계적인 한계로 인해 디스크에서의 파일 입출력 속도가 프로세서의 속도증가를 따라가지 못하고 있다. 거기에 더해 고속통신망의 보급과 멀티미디어 데이터의 증가로 인해 대용량의 파일 입출력을 요구하는 응용이 차지하는 비중이 점점 높아지고 있다. 또한 네트워킹 기술의 발달로 100Mbps의 고속 이더넷의 속도를 뛰어넘어 1Gbps 이상의 전송 대역폭을 가지는 네트워크 기술이 선보이고 있다. 이로 인해 디스크 저장 장치의 입출력 속도와 실제 요구되는 입출력 속도와의 간격이 점점 커지고 있다. 이에 따라 파일 입출력의 속도를 개선하기 위한 많은 연구가 있어 왔다. 파일 입출력 속도의 개선을 위한 연구는 대역폭의 확장과 액세스 시간 단축의 두 분야에서 이루어지고 있다. 대역폭을 확장하기 위한 방법으로 제안된 대표적인 예는 병렬파일시스템이다. 병렬파일시스템은 한 파일을 고속의 네트워크로 연결된 여러 노드의 디스크에 저장해서 병렬로 읽어 들임으로써 디스크의 입출력 대역폭을 확장할 수 있다. 하지만 액세스 시간의 개선 효과는 거의 없다. 액세스 시간을 개선하기 위해 제안되고 있는 것들은 개선된 메시지전달 시스템, 클러스터간 통신을 위한 오버헤드가 적은 프로토콜, 개선된 캐시, 선반입 구조 등이다. 예전에 비해 파일의 평균크기가 커지고 접근 패턴이 변하고 있기 때문에 LRU등의 기존의 캐시, 선반입 정책만을 사용하는 것은 최적의 성능을 보장하지 못하고 있다. 본 논문에서는 캐시, 선반입의 효율을 높이기 위해 각각의 액세스 패턴에 맞는 캐시, 선반입 정책을 결정하는데 선행연구로서 액세스 패턴의 특징을 실시간으로 추출하고 기록하는 방법을 제시한다.

2. 액세스 패턴 분석과 실시간 패턴 검출

다양한 종류의 액세스 패턴이 존재하는 시스템에서 한 종류의 캐시, 선반입 정책으로는 최적의 성능을 구현할 수 없다. 이에 따라 파일 액세스 패턴을 분류, 분석하고 분류된 액세스 패턴에 맞도록 파일 시스템의 정책을 최적화 시키고자 하는 연구가 있어 왔다.

2.1 액세스 패턴 연구

많은 연구자들에 의해 전통적인 UNIX 플랫폼에서의 파일 액세스 패턴이 연구되어 왔다. 이러한 연구에 의해 일반적인 UNIX 파일시스템에서 대부분을 차지하는 것은 순차적인 읽기, 쓰기라는 것이 알려져 있다[1]. 또한 병렬 처리 방법을 사용하고 대용량의 데이터 입출력이 필요한 응용의 대표적인 예인 과학기술계산 분야의 파일 입출력 패턴은 simple-strided 패턴이 많은 것으로 조사되었다[2]. David Kotz는 병렬 프로세서 상황에서 액세스 패턴을 각 프로세스의 액세스 패턴인 로컬 액세스 패턴과 모든 프로세스의 액세스 패턴의 합인 글로벌 액세스 패턴으로 분류하고 글로벌 액세스 패턴이 로컬액세스 패턴에 비해 검출하기 힘들지만 파일 시스템의 성능을 향상시키는데 더 중요하다고 밝히고 있다[3]. Tara Madhyastha는 파일 액세스 패턴을 순차성의 정도, 액세스 크기의 양과 변화율, 읽기/쓰기의 혼합 유무의 세가지 특성에 따라 나누었다[4].

본 논문에서는 현재까지 알려진 대표적인 액세스 패턴을 검출하는데 집중해서 패턴 정보를 간략화 시켜 수집함으로써 액세스 패턴 기록이 필요로 하는 메모리공간과 디스크공간 소모량을 줄였다.

2.2 실시간 액세스 패턴 검출

현재의 파일 액세스 패턴에 따라 파일 시스템의 캐시, 선반입 정책을 변경하고자 하는 연구가 있어 왔다. David Kotz는 간단한 파일 액세스 패턴 검출 알고리즘을 사용해서 액세스 패턴을 검출한 다음 그에 맞춰서 4가지 선반입 정책을 적용하는 방법을 연구하였다[3]. Tara Madhyastha는 그의 논문에서 그가 나눈 액세스 패턴을 Artificial Neural Network로 검출해서 캐시와 선반입 정책을 적용하는 방법에 대해 언급하고 있다[4].

또 다른 방향에서는 운영체제가 응용프로그램의 액세스 패턴을 검출하지 않고 응용 프로그래머가 최적의 캐시, 선반입

정책을 미리 운영체제에 알려줌으로써 액세스 패턴 검출의 오버헤드를 줄이고 검출 오류를 막는 방법에 대한 연구가 있어 왔다. 이런 것에는 응용프로그램이 앞으로 사용하게 될 블록을 알려주는 Informed Prefetching and Caching 알고리즘[5]과 응용프로그램이 운영체제가 제공하는 정책 중 자신에게 최적인 캐시, 선반입 정책을 사용하는 Open Implementation[6] 등이 있다.

본 논문에서 제안하는 실시간 액세스 패턴 검출 방법은 Kotz가 제안한 방법보다는 유용한 정보를 더 많이 수집하면서 인공지능을 사용한 방법들이 가지는 오버헤드를 감소시켰다. 또한 Hint-based 나 Open Implementation이 기존 응용프로그램의 재작성이 필요한데 비해 응용프로그램 재작성이 필요 없고 프로그래머가 응용프로그램의 저수준 액세스 패턴에 대해 알지 못해도 되는 장점이 있다.

3. 액세스 패턴 분류 방법

파일 입출력의 액세스 시간 개선을 위해서는 액세스 패턴을 검출하고 분류해서 최적의 캐시와 선반입 정책을 적용해야 한다. 캐시와 선반입은 그 특성이 상이하므로 캐시 정책을 결정하기 위해 필요한 패턴 정보와 선반입을 위한 패턴 정보는 다르다. 따라서 패턴 정보를 수집하는 방법 또한 다를 수 밖에 없다.

3.1 캐시를 위한 패턴 수집

캐시 정책을 결정하는 경우에는 현재 캐시에 존재하는 블록의 재사용성의 높고 낮음이 주요한 정보이다. 따라서 파일단위가 아니라 블록단위로 패턴 정보를 수집한다. 또한 블록이 캐시에 존재할 때의 액세스 패턴이 관심 사항이므로 캐시에 처음 등록되면 패턴 정보의 수집을 시작하고 캐시에서 제거되면 정보 수집을 종료한다. 수집되어야 할 패턴 정보는 다음과 같다.

- 참조 회수
- 참조간의 평균거리

참조 회수는 재사용성의 유무와 정도를 판단하는데 쓰이고 참조간의 평균거리는 재사용의 빈도를 판단하는데 필요하다.

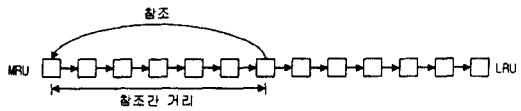

그림 1. LRU 체인에서의 참조간 거리

이렇게 수집된 정보를 정수로 표현한다면 한 블록당 8byte를 차지하게 된다. 파일 저장공간의 크기가 작다면 문제가 되지 않겠지만 TeraByte 단위 이상의 저장 공간이라면 1GB 이상의 공간이 필요하게 된다. 이러한 저장공간의 낭비를 줄이기 위해서 패턴 표현에 필요한 저장공간을 줄일 필요가 있다.

일반적으로 각 블록에 대한 액세스 패턴은 Zipf 분포를 보인다[7]. 즉 자주 액세스 되는 블록은 그 개수가 적고 자주 액세스 되지 않는 블록이 대부분을 차지한다. 또한 자주 액세스 되는 블록의 참조회수는 자주 액세스 되지 않는 블록의 참조회수에 비해 매우 높다. 따라서 블록의 재사용성에 대한 정보를 재사용성이 높다/낮다의 두 가지로 표현해도 자세히 표현할 때에 비해 정보량이 그리 차이 나지 않는다. 본 논문에서는 1bit를 사용해서 재사용성을 0과 1로 표현하고 한 블록 당 2bit를 할당하여 최근 재사용성과 기준 재사용성을 같이 기록하였다.

그림 2.에서는 재사용성 기록포맷의 구조와 변경 방법에 대해 나타내었다.

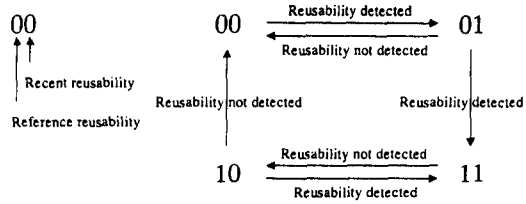

그림 2. 재사용성 기록과 변환 방법

최근 재사용성과 현재 나타난 재사용성이 동일하다면 기준 재사용성을 변경한다. 즉 두 번 연속해서 같은 재사용성이 검출되면 기준 재사용성 bit을 변경한다. 재사용성의 판단 기준은 블록 참조의 회수가 설정된 값 이상이고 블록참조의 평균거리가 설정된 값이 이하일 때 재사용성이 1이고 둘 중에 하나라도 만족하지 않는다면 0으로 판단한다. 캐시 관리자에서 참조하는 것은 기준 재사용성이다. 이러한 방법으로 재사용성을 판단하게 되면 일시적인 재사용성의 비정상적인 변화에 대해 내성을 가지면서 시간에 따라 재사용성이 변화하는 것을 검출해서 그 블록의 재사용성을 변경할 수 있다.

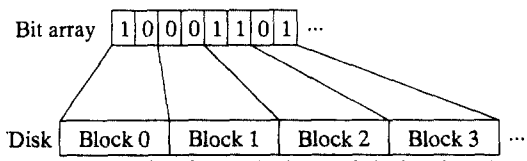

그림 3. 재사용성 비트와 디스크 상의 블록의 상관관계

블록의 재사용성을 저장할 때는 그림 3과 같이 전체 저장공간을 캐시에서의 관리하는 단위로 나누어 그 개수만큼의 비트배열을 할당해서 저장한다. 이 방법을 사용하면 블록번호를 사용해서 해당 재사용성 비트를 찾는데 걸리는 시간을 최소화할 수 있다.

3.2 선반입을 위한 패턴 수집

선반입은 현재 열린 파일에서 다음에 읽을 부분을 예측하여 미리 읽어 둘임으로써 파일 입출력의 유휴 대역폭을 효율적으로 사용하고 입출력 요청이 한 순간에 집중되어서 응답시간이 느려지는 것을 완화할 수 있다. 선반입을 위해 다음에 쓰일 블록을 예측하려면 단일 프로세스 또는 스레드가 하나의 파일을 열었을 때의 액세스 패턴인 로컬 액세스 패턴보다는 전체 프로세스의 파일 액세스 패턴인 글로벌 액세스 패턴을 계산해야만 한다. 하지만 글로벌 액세스 패턴은 똑같이 재생될 확률이 희박하고 글로벌 액세스 패턴에서 특징을 추출하기도 힘들다. 반대로 로컬 액세스 패턴의 경우는 일반적으로 일정하기 때문에 특징을 추출하기 쉽다. 현재의 글로벌 액세스 패턴을 구하려면 현재 열린 파일들의 로컬 액세스 패턴을 조합하면 된다. 따라서 수집해야 할 정보는 로컬 액세스 패턴이다.

선반입을 위한 패턴 수집은 파일단위로 이루어져야 하며 한 번의 파일 열기에서 닫기까지의 액세스 패턴을 기록해야 한다. 일반적으로 하나의 파일은 한 응용프로그램에 의해서만 사용되는 경우가 많고, 여러 응용프로그램에 의해 사용되어도 자주 사용하는 응용프로그램은 정해져 있고 나머지 응용프로그램의 사용빈도는 매우 낮다. 또한 여러 응용프로그램에서 사용한다 하더라도 파일의 종류에 따라 액세스 하는 패턴이 거의 일정하다. 따라서 자주 나타나는 패턴이 다시 나타날 가능성이 매우 높으므로 자주 나타나는 패턴을 기록, 사용하는 것은 모든 액세스 패턴을 수집하는 것에 비해 성능이 그리 떨어지지 않으면서도 사용하는 저장공간과 오버헤드 면에서의 장점은 매우 크다.

이전의 파일 액세스 패턴에 대한 연구결과를 바탕으로 다음

과 같은 대표적인 패턴을 가정하였다.

- Sequential
- Simple strided
- Variably strided
- Random

Sequential 패턴은 일반적인 파일 시스템에서 대부분을 차지한다[1]. 응용프로그램이 파일을 열고 순차적으로 처음부터 읽거나 쓰는 것이다. Simple strided 패턴은 일정한 크기의 데이터 조각을 일정한 간격으로 건너뛰면서 읽어 들이는 패턴으로서 병렬 처리 과학 기술 계산 응용에서 많이 발견되는 액세스 패턴이다[2]. Variable strided는 일정하지 않은 간격을 건너뛰면서 일정하지 않은 크기의 데이터 조각을 읽어 들이는 것으로서 Simple strided보다 적게 발견되지만 과학 기술 계산 응용에서는 많이 발견되는 액세스 패턴이다. Random은 위의 범주에 포함되지 않는 패턴을 모두 포함한다.

이러한 패턴을 검출하기 위해 수집해야 할 패턴 정보를 다음과 같이 결정하였다.

- 읽기/쓰기 여부
- 참조(읽기/쓰기)의 평균 크기, 분산
- 평균 데이터 요구량
- 정방향 seek 회수, 평균거리, 분산, 빈도
- 역방향 seek 회수, 평균거리, 분산, 빈도

수집된 정보를 바탕으로 위의 네 가지 패턴중의 하나로 판단한 후, 수집된 정보와 결정된 네 가지 분류중의 하나를 디스크에 저장한다. 다음에 파일이 열리면 그림 4와 같이 이전에 저장된 패턴 정보를 사용해서 예측 패턴을 만들어 낸다. 그리고 최종적으로 예측 패턴들을 조합해서 글로벌 예측 패턴을 만든다.

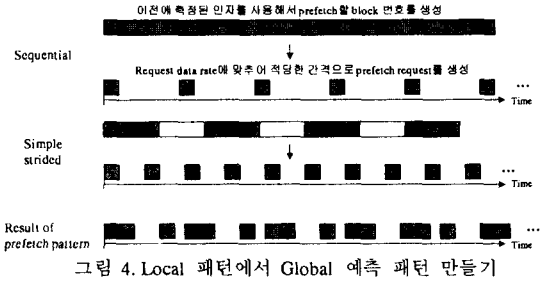

그림 4. Local 패턴에서 Global 예측 패턴 만들기

선반입 정책을 결정하는 시스템에서는 Global 예측 패턴을 바탕으로 선반입의 대상과 양, 시기를 결정할 수 있다.

표 1은 캐시와 선반입을 위한 패턴 수집의 특성을 비교한 것이다.

표 1. 캐시와 선반입 패턴 수집 비교

	캐시	선반입
수집단위	블록	파일
수집기간	캐시에 들어왔을 때부터 캐시에서 제거될 때까지	파일이 열릴 때부터 닫힐 때까지
수집대상	글로벌 액세스 패턴	로컬 액세스 패턴
정보의 종류	블록의 재사용성 여부	알려진 액세스 패턴과의 유사성

4. 캐시, 선반입 정책에의 적용

본 논문에서 제안한 방법으로 어떤 블록을 캐시하는 것이 효율적인지, 어떤 패턴으로 선반입을 하는 것이 적중률이 높은지를 예측할 수 있다. 하지만 실제 시스템의 환경에 따라 위의 방법에서 검출해낸 패턴을 캐시, 선반입 정책에 적용하는 방법은 달라진다. 만약 네트워크 대역폭의 여유가 없다면 선반입을 하지 않는 것이 더 좋은 성능을 보일 것이다. 또 캐시 사이즈에 따라 위에서 구한 블록 패턴을 적용하는 방법이 달라질 것이다. 즉, 본 논문에서 제안한 방법은 실제 캐시, 선반입 정책을 적용할 때 더 효율이 좋은 방법을 결정하기 위한 자료를 수집하는데 목적이 있다. 실제 시스템에 적용될 캐시, 선반입 정책은 그 환경과 목적에 따라 다를 수 있다.

5. 결론 및 향후 과제

본 논문에서는 파일 입출력 시스템에서의 높은 응답 성능을 얻기 위한 방법의 일환으로 효율적인 캐시, 선반입을 구현하기 위해서 기존의 연구에서 알려진 액세스 패턴을 바탕으로 각 액세스 패턴을 분류하고 저장하는 방법을 제안하였다. 캐시의 경우에는 각 블록의 재사용성에 따라 분류함으로써 캐시 정책 결정 시스템이 재사용성이 높은 블록에 우선 순위를 두어 관리할 수 있도록 했고, 선반입의 경우에는 현재 열려 있는 파일들에 대한 과거의 액세스 패턴을 유사하게 재구성하여 조합함으로써 앞으로 요청될 패턴을 예측할 수 있도록 하였다.

대부분의 상용 파일시스템은 일반적인 상황에서의 효율이 높다고 알려진 LRU 등의 정책을 사용하고 있지만, 현재와 같이 멀티미디어 정보의 양이 급격하게 늘어나는 상황에서는 그 효율이 떨어지고 있다. 이에 따라, 새로운 정책들이 제안되고 있지만 그 역시 특정한 액세스 패턴에서만 높은 효율을 보이도록 고안된 것이어서 범용성이 떨어진다. 본 연구에서 제안한 방법은 실제 시스템의 변화하는 액세스 패턴을 수집하여 그에 맞는 정책을 실시간으로 선택할 수 있도록 함으로써 액세스 패턴이 변화하는 상황에서도 동적으로 최적의 캐시, 선반입을 수행할 수 있도록 하는데 기반이 된다.

현재까지는 기존의 연구를 바탕으로 알고리즘이 정의된 상태이며 실제 파일 액세스 패턴을 사용한 시뮬레이션을 통해 효과가 있음을 입증할 것이다. 시뮬레이션이 끝나면 리눅스 파일 시스템을 사용하여 실제 파일시스템에 구현할 계획이다.

6. 참고문헌

[1] John K. Ousterhout, Hervé Da Costa, David Harrison, John A. Kunze, Mike Kupfer, and James G. Thompson, "A Trace-Driven Analysis of the UNIX 4.2 BSD File System", Proceedings of the 10th Symposium on Operating System Principles, Orcas Island, WA, December 1985, 15-24.

[2] Nils Nieuwejaar, David Kotz, Apratim Purakayastha, Carla Schlatter Ellis, and Michael Best. "File-access characteristics of parallel scientific workloads", IEEE Transactions on Parallel and Distributed Systems, 7(10):1075-1089, October 1996.

[3] David Kotz and Carla Schlatter Ellis. "Practical prefetching techniques for multiprocessor file systems", Journal of Distributed and Parallel Databases, 1(1):33-51, January 1993.

[4] Tara Madhyastha, "Automatic Classification of Input/Output Access Patterns", Tech. Rep., University of Illinois at Urbana-Champaign, Department of Computer Science, August 1997.

[5] R.H. Patterson, G.A. Gibson, E. Ginting, D. Stodolsky, and J. Zelenka. "Informed prefetching and caching". In Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles, pages 79-95, December 1995.

[6] Chul-Jin Ahn, Seong-Uk Choi, Myong-Soon Park and Jin-Young Choi, "The Design and Evaluation of Policy-Controllable Buffer Cache", Proc. of International Conference on Parallel and Distributed systems, pp. 764-771, December, 1997

[7] Lee Breslau, Pei Cao, Li Fan, Graham Phillips and Scott Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications", Proceedings of IEEE Infocom '99, pages 126-134, New York, NY, March, 1999.