참조패턴을 이용한 선반입의 개선

이효정° 도인환 노삼혁
홍익대학교 컴퓨터공학과
hjlee@mail.hongik.ac.kr° {ihdoh, samhnoh}@cs.hongik.ac.kr

Improving Prefetching Effects by Exploiting Reference Patterns

Hyojeong Lee° In Hwan Doh Sam H. Noh
Department of Computer Engineering, Hongik University

디스크와 메모리의 속도 차이를 극복하기 위한 캐시에서의 선반입(prefetch)은 교체정책과 함께 중요한 기법으로 연구되고 사용되어 왔다. 지금까지의 선반입 기법은 일반적으로 참조를 관찰하여 비교적 탐지하기 쉬우면서 효과를 거둘 수 있는 참조의 순차성을 탐지하고 그에 기반한 예측을 통해 수행되어 왔다. 본 논문은 참조 패턴을 자동으로 탐지하고 탐지된 참조 패턴의 특성에 따라 참조의 순차성에 기반한 기존의 선반입을 조정하여 개선할 수 있는 기법 IPRP(Improving Prefetching Effects by Exploiting Reference Patterns)를 제안한다.

선반입은 잘못 수행될 경우 성능에 악영향을 가져올 수 있다. 선반입이 부정적인 영향을 가져오게 되는 가장 큰 이유는 잘못된 선반입을 수행할 경우 캐시가 오염될 수 있다는 것이다[3]. 사용되지 않을 블록을 선반입할 경우 그 비용이 작을 수 있다고 해도 추가적인 I/O가 발생한다. 특히 잘못 선반입한 블록 때문에 기존의 유용한 블록이 쫓겨나게 될 경우 쫓겨난 블록을 재 반입 하기 위한 오버헤드가 발생한다. 이와 같이, 사용되지 않을 블록으로 유용한 블록이 쫓겨나게 되는 것을 캐시 오염이라고 부른다. IPRP는 참조 패턴을 탐지하여 참조 패턴의 특성에 따라 기존의 선반입에 간단한 조정을 가해 잘못된 선반입을 최대한 방지함으로써 선반입 성능을 개선하고자 한다.

순차성에 기반한 기존의 선반입은 참조 패턴의 순차성이 클 때 상당한 효과를 거둘 수 있다. 이러한 기법은 오버헤드가 낮고 순차성이 높을 때 효율적이므로 일반적인 커널 선반입에서 널리 사용되고 있다[1][4][5][6]. 참조에 순차성이 적을 경우에는 이러한 기법이 잘못된 선반입을 수행할 위험이 높으므로 순차성의 강도에 따라 선반입의 강도를 조절하는 기법이 널리 사용된다. 그럼에도 불구하고 참조의 순차성이 상당히 낮을 경우에는 잘못된 선반입이 지속적으로 발생하여 수행시간이 최대 두 배까지 증가하는 성능 악화의 위험이 있음이 밝혀진 바 있다[1]. 순차성이 높은 경우에도 잘못된 선반입은 지속적으로 일어나면 선반입의 성능 향상을 감소시킬 수 있다. 모든 순차 참조는 결국 끝나는 시점이 발생하는데, 이 때 순차성을 기반으로 지나친 선반입을 수행할 수 있다. 이것은 비교적 낮은 위험이라 할 수 있지만 만일 이것이 반복적으로 발생한다면 성능에 악영향을 미칠 수 있다.

본 논문에서 제시한 IPRP는 참조 패턴을 일정 개수 이상의 연속된 블록 참조들이 한번만 발생되는 순차 참조, 순차 참조들이 규칙적인 간격으로 반복해서 발생되는 순환 참조, 순차 참조와 순환 참조에 속하지 않는 기타 참조의 세 가지로 분류한다. 이와 같은 분류법은 선반입을 포함하지 않는 캐시 관리기법인 통합버퍼관리기법(UBM)에서 교체정책을 결정하기 위해 사용되어 좋은 성능을 이끌어 낸 바 있으며, IPRP는 UBM에서 제시한 것과 유사한 방식으로 이러한 패턴들을 온라인으로 탐지한다[2].

IPRP의 선반입 제어기는 참조 패턴의 특성에 따라 다음과 같은 제한을 두어 선반입의 성능을 개선하고자 한다.

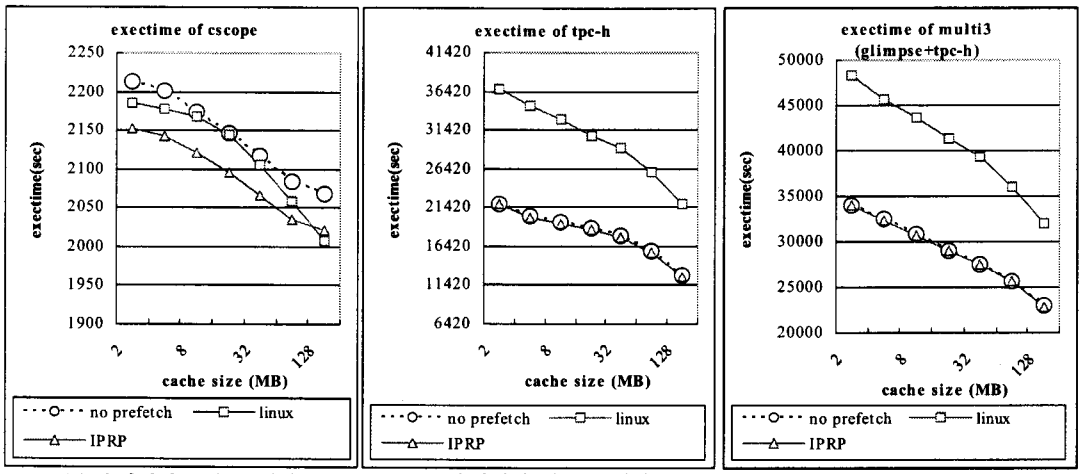
첫째, 순차 참조에 대해서는 순차 참조라는 것이 탐지되면 인접한 블록을 선반입한다.

둘째, 순환 참조가 탐지되면 기존의 선반입을 수행한다. 다만 순환 참조를 구성하는 순차 참조의 마지막 블록 이후의 블록을 선반입 하는 것을 방지한다. 이것은 순차성이 예측보다 일찍 끝나는 일이 발생하여 순차성이 있을 때에도 잘못된 선반입이 일어나는 것을 방지하기 위한 것이다.

셋째, 위의 두 참조가 아닌 기타 참조로 판단되었을 경우에는 선반입을 금지한다.

IPRP는 위와 같은 방법으로 참조 패턴이 선반입에 유리한 형태라는 것이 판단될 때까지 선반입의 수행을 지연하고 순차성이 존재할 때에도 지나치게 적극적인 선반입이 반복되어 사용되지 않을 블록을 반입하는 것을 방지한다.

IPRP의 성능을 평가하기 위한 실험환경으로는 리눅스 2.4 커널의 미리 읽기 선반입과 I/O 클러스터링을 충실히 구현하고 실행시간을 측정할 수 있도록 Butt 등이 제작한 시뮬레이터 Accusim을 사용했다[1]. Accusim에 참조 패턴 탐지기와 참조 패턴 탐지 결과에 따라 선반입 여부 및 선반입 정도를 조정하는 선반입 제어기를 추가하였다. Accusim과 함께 다양한 특성을 갖는 트레이스들이 제공되었으므로 비교의 공정성을 위해 해당 트레이스들을 사용



(a) 순차성이 높은 트레이스 (b) 순차성이 낮은 트레이스 (c) 병렬적인 트레이스

그림 1: 트레이스 특성 별 캐시 크기 변화에 따른 수행시간 비교

했다. 리눅스의 미리 읽기 선반입과 가장 일반적인 교체정책 중 하나인 LRU를 기반으로 IPRP의 성능을 측정했다.

그림 1은 트레이스 특성별로 IPRP의 적용이 선반입을 수행하지 않았을 때나 리눅스 미리 읽기 선반입이 수행되었을 때에 비해 어떤 변화를 가져왔는지를 보여준다. 그림 1(a)에서 볼 수 있듯이, 순차성이 높은 트레이스의 경우 IPRP의 적용은 기존의 선반입을 적용한 것과 유사한 결과를 보여준다. 이를 통해 참조 패턴을 적용한 기법의 선반입 지연이 참조의 순차성이 매우 높을 때에도 선반입 이익 감소를 거의 수반하지 않음을 알 수 있다. 그림 1(b)에 사용된 트레이스 tpc-h는 약 3%정도의 낮은 순차성을 가진다. 기존의 선반입을 그대로 적용했을 경우 선반입을 수행하지 않았을 때에 비해 67%정도의 성능 악화가 발생한다. 그러나 IPRP를 적용할 경우 효과적으로 선반입의 악영향을 방지할 수 있음을 확인할 수 있다. 선반입은 병행하는 트레이스 하에서는 단일 어플리케이션 트레이스에서와 다른 결과를 보일 수 있을 뿐 아니라 실제 상황은 일반적으로 병행하는 트레이스에 가까울 수 있으므로 병행하는 트레이스에 대한 실험은 선반입 연구에서 중요한 문제이다. 그림 1(c)는 참조패턴을 적용한 기법이 병행하는 트레이스에서도 효과적으로 악영향을 방지하고 선반입 성능을 향상시킬 수 있음을 보여준다. 리눅스 미리 읽기가 약 41%의 성능 악화를 유발하는 multi3에 대해서는 성능악화를 방지할 뿐 아니라 약 0.3%의 성능 향상을 가져왔다.

감사의 글

이 논문은 2007년도 정부(과학기술부)의 재원으로 한국과학재단의 국가지정연구실사업으로 수행된 연구임(No. R0A-2007-000-20071-0)

[참고 문헌]

[1] A. R. Butt, C. Gniady, and Y. C. Hu, "The Performance Impact of Kernel Prefetching on Buffer Cache Replacement Algorithms, " In *Proceedings of the ACM International Conference on Measurement & Modeling of Computer Systems (SIGMETRICS)*, pp. 57-168, 2005.

[2] J. M. Kim, J. Choi, J. Kim, S. H. Noh, S. L. Min, Y. Cho, and C. S. Kim, "A Low-Overhead, High-Performance Unified Buffer Management Scheme That Exploits Sequential and Looping References," In *Proceedings of the 4th Symposium on Operating Systems Design and Implementation (OSDI)*, pp.119-134, 2000.

[3] P. Cao, E. W. Felten, A. R. Karlin and K. Li, "A Study of Integrated Prefetching and Caching Strategies," In *Proceedings of the ACM International Conference on Measurement & Modeling of Computer Systems (SIGMETRICS)*, pp.188-197, 1995.

[4] B. S. Gill and D. S. Modha, "SARC: Sequential prefetching in adaptive replacement cache," In *Proceedings of the USENIX Annual Technical Conference*, pp. 293–308, 2005.

[5] B. S. Gill and L. D. Bathen, "AMP: Adaptive Multi-stream Prefetching in a Shared Cache," In *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST 07)*, pp.185-198, 2007,

[6] C. Li, K. Shen and A. Papathanasiou, "Competitive prefetching for concurrent sequential I/O," In *Proceedings of 2nd European Conference on Computer Systems (EuroSys 2007)*, Mar, 2007.