선반입 통합 기법을 이용한 GNU/Linux 파일 시스템의 성능 향상에 관한 연구*

전 홍 석　　노 삼 혁
홍 익 대 학 교　컴퓨터공학과
{hsjeon, noh}@cs.hongik.ac.kr

Improving the Performance of the GNU/Linux FIle System using an Integrated Prefetching Scheme

H. Seok Jeon　Sam H. Noh
Dept. of Computer Engineering, Hong-Ik University

요 약

버퍼 캐쉬의 관리를 위해 많은 교체 정책들과 선반입 정책들이 연구되어져 왔다. 그러나, GNU/Linux를 포함한 많은 실제의 운영체제들은 일반적으로 Least Recently Used (LRU) 교체 정책을 사용한다. 본 논문에서는 끊임없이 적극적인 선반입이 이루어지는 가운데 버퍼 관리와 선반입을 효율적으로 통합하는 SA-W^{2}R 정책을 제안한다. 이 정책은 구현이 단순하여 실제 시스템에서 채택하기 용이하다. 이 정책은 기본적으로 버퍼 교체를 위하여 LRU 정책을 사용한다. 그러나 정책의 모듈성으로 인해 어떤 교체 정책도 이 정책에 적용될 수 있다. SA-W^{2}R 정책에서는 선반입을 위해 오버헤드가 적어 일반적으로 많이 사용되는 LRU-One Block Lookahead (OBL) 정책을 사용한다. GNU/Linux 커널 버전 2.2.14에 구현된 SA-W^{2}R 정책은 응용 프로그램의 실행 시간에 있어 현재 버전의 GNU/Linux 보다 최고 23%의 성능 향상을 보였다.

1. 서론

버퍼 캐쉬의 사용을 최적화하기 위한 연구가 교체뿐만 아니라 선반입의 측면에서도 많이 이루어져왔다 [1,2,3,4,5,6,7,8]. 그러나 많은 선반입 정책들이 현실성이 없거나 정책의 복잡함에 의해 구현하기 어려워 실제의 시스템에서 사용하기에는 실용적이지 못한 문제점을 가지고 있다 [9].

Weighing-Waiting Room (W^{2}R) 정책은 이러한 문제를 해결하는 효율적인 버퍼 관리 정책이다 [9]. W^{2}R 정책은 그림 1처럼, 버퍼 캐쉬를 두 개의 공간, 즉, Weighing Room과 Waiting Room으로 분할한다.

Weighing Room은 참조된 블록들을 위한 공간이며 Waiting Room은 선반입된 블록들을 위한 공간이다. W^{2}R 정책에서의 선반입은 LRU-One Block Lookahead (OBL) 정책과 동일하게 이루어진다. 즉, 현재 참조된 블록의 논리적 다음 블록이 버퍼 캐쉬에 없으면 선반입

본 연구는 한국과학재단 특정기초연구과제(과제번호:98-0102-09-01-3)의 지원을 받았음.

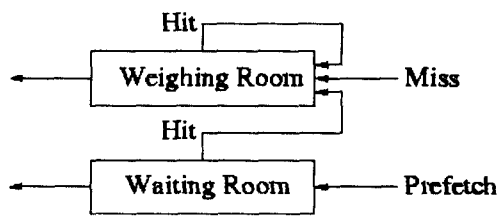

그림 1. W^{2}R 정책의 구조

을 한다. 선반입된 블록들은 버퍼 캐쉬의 이 부분에 머무르며 실제로 참조되었을 때에 Weighing Room으로이동한다.

W^{2}R 정책의 유일한 문제는 고정된 크기의 버퍼 캐쉬를 어떻게 두 개의 영역으로 분할하느냐이다. 본 논문에서는 이를 위해 Weighing Room과 Waiting Room간의 분할 비율을 동적으로 조절하여 버퍼 캐쉬의 효율을 최대화하기 위한 SA-W^{2}R 정책을 제안한다.

본 논문의 나머지 중 2 절에서는 이 논문에서 제안하는 SA-W2R 정책을 소개한다. 구현을 통한 성능 평가가 3절에 기술되며 4절에서 요약과 향후 연구에 대한 방향을 제시하며 결론을 맺는다.

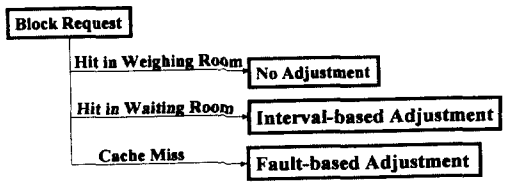

그림 3. SA-W^{2}R 정책.

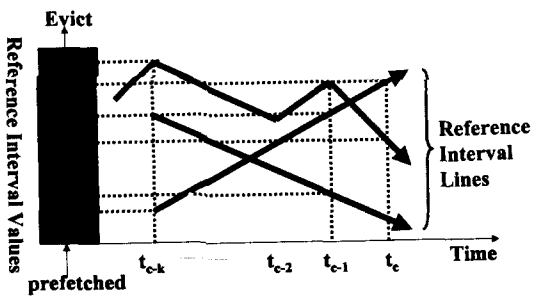

그림 4. Interval-based Adjustment.

2. Self-Adjusting W^{2}R

SA-W^{2}R 정책은 Weighing Room과 Waiting Room 간의 분할 비율을 그림 3과 같이 두 단계의 과정을 통해, 즉, Interval-based와 Fault-based adjustment 단계를 통해 적응한다.

2.1 Interval-based Adjustment

Interval-based Adjustment는 Waiting Room의 크기를, 그림 4에서 보여주듯이, Waiting Room에서 참조된 최근 k개의 블록들의 선반입부터 참조까지의 간격인 reference interval을 유지함에 의해 조절한다. (유지의 비용을 최소화하기 위해 우리는 k를 3으로 고정하였다.) 만일 최근 k개의 블록들의 reference interval 이 증가 추세를 보이면 reference interval의 증가 추세를 보장하기 위하여 Waiting Room의 크기를 증가한다. 유사하게 만일 reference interval 들이 감소추세를 보이면 Waiting Room은 감소된다. 만일 reference interval이 어떤 규칙성도 보이지 않으면 Waiting Room의 크기는 변하지 않는다.

표 1. 블록 i에 대한 미스 발생 시 블록 i, i-1, i+1의 위치에 따른 9가지 상황.

Cases	Weighing Room	Waiting Room	Disk	Adjustment made
case 1	i-1, i+1		i	increase Waiting Room
case 2	i-1	i+1	i	Increase Weighing Room
case 3	i-1		i, i+1	Increase Waiting Room
case 4	i+1	i-1	i	*No Adjustment*
case 5		i-1, i+1	i	Increase Weighing Room
case 6		i-1	i, i+1	No Adjustment
case 7	i+1		i-1, i	No Adjustment
case 8		i+1	i-1, i	Increase Weighing Room
case 9			i-1, i, i+1	No Adjustment

2.2 Fault-based Adjustment

*Fault-based Adjustment*는 각 영역의 크기를 조절하기 위해서 블록 요청에 대한 미스 정보를 활용한다. 구체적으로, 블록 i에 대한 요청이 미스일 경우에는 표 1과 같이 블록 i, i-1, i+1의 위치에 따라 모두 9가지의 가능한 상황이 존재한다.

먼저 블록 i-1이 공통으로 Weighing Room에 있는 case 1과 case 3부터 살펴보자. 블록 i-1이 Weighing Room에 존재한다는 사실은 블록 i가 이전에 교체되기 전에 Waiting Room에 존재했었음을 의미한다. 그런데 블록 i가 Waiting Room의 크기가 너무 작아서 교체되었다고 판단한다. 그러므로 이러한 경우에는 Waiting Room의 크기를 증가해야한다.

이번에는 case 5와 8을 살펴보자. 블록 i+1이 Waiting Room에 있다는 사실은 블록 i가 이전에 Weighing Room에 있었음을 말해준다. 이는 Weighing Room의 크기가 너무 작아서 곧 사용될 블록을 Weighing Room으로부터 교체하였었음을 의미한다. 따라서 이러한 경우에는 Weighing Room의 크기를 증가해야 한다.

Case 4, 6, 7, 그리고 9는 어떤 명확한 관계를 추론해 낼 수 없다. 따라서 이들의 경우에 대해서는 어떠한 적응도 하지 않는다.

표 2. 다양한 버퍼 관리 정책에 대한 응용 프로그램들의 평균 실행 시간 (단위: 초).

Applications	Linux	LRU-OBL	SA-W2R
gcc	244	241	230
cp	59.40	53.31	48.71
tar (create)	61.02	59.87	55.31
tar (extract)	41.70	39.93	36.26
gzip (compress)	72.87	64.73	56.11
gzip (uncompress)	21.45	19.99	17.23
sort	47.32	45.14	42.57
grep	46.92	38.28	37.01

3. 성능 평가

SA-W^{2}R 정책은 128MB의 주기억장치를 가진 Pentium III 430 MHz PC 상의 GNU/Linux 커널 버전 2.2.14에서 구현되었다. 성능 비교를 위하여 본 논문에서는 LRU-OBL 정책 또한 구현하였다.

표 2는 각 응용 프로그램의 실행 시간을 보여준다. SA-W^{2}R 정책은 원래의 GNU/Linux나 LRU-OBL 정책에 비해 가장 좋은 성능을 보여준다. 구체적으로, 제안된 정책의 성능 향상은 원래의 GNU/Linux 정책에 비해 5에서 23% 정도로 나타난다.

4. 결론 및 향후 연구 과제

이 논문에서는 선반입과 교체 정책을 통합하는 단순하고 실용적인 정책인 SA-W^{2}R 정책을 제안하였다. 이 정책은 또한 어떤 교체 정책도 이 기법에 적용될 수 있는 모듈성을 가지고 있다.

SA-W^{2}R 정책을 GNU/Linux에 구현하여 성능을 평가한 결과 SA-W^{2}R 정책이 원래의 GNU/Linux와 LRU-OBL 정책에 비해 가장 좋은 성능을 보여주었다.

현재 SA-W^{2}R 정책에서 Weighing Room의 모듈성 혹은 힌트 기반의 선반입 정책을 통한 이익이 어느 정도의 이익을 가져올 수 있는지에 관해 연구 중이다.

또한, 선반입의 성능은 디스크 시스템의 성능에 의해 매우 강하게 영향을 받는다. 따라서 이에 대한 보다 정확한 연구가 이루어져야 할 것이다.

참고 문헌

[1] Pei Cao and Edward W. Felton. Implementation and Performance of Integrated Application-Controlled File Caching, Prefetching, and Disk Scheduling. *ACM Transactions on Computer Systems*, 14(4):311-343, November 1996.

[2] Pei Cao, Edward W. Felton, Anna R. Karlin, and Kai Li. A study of integrated prefetching and caching strategies. In *Proceedings of 1995 Joint ACM SIGMETRICS and Performance Evaluation Conference*, pages 188-197, 1995

[3] Fay Chang and Garth Gibson. Automatic I/O Hint Generation through Speculative Execution. In *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation*, February 1999.

[4] David kotz and Cara Schlatter Ellis. Practical Prefetching Techniques for Multiprocessor File Systems. *Journal of Distributed and Parallel Databases*, 1(1):33-51, January 1993

[5] R. Hugo Patterson, Garth A. Gibson, Eka Ginting, Daniel Stodolsky, and Jim Zelenka. Informed Prefetching and Caching. In *Proceedings of the 15th ACM SOSP*, pages 79-95, December 1995.

[6] Alan Jay Smith. Sequential program prefetching in memory hierarchies. *IEEE Computer*, 3(3):7-21, December 1978.

[7] Alan Jay Smith. Disk cache-miss ratio analysis and design considerations. *ACM Transactions on Computer Systems*, 3(3):161-203, August 1985.

[8] Andrew Tomkins, R. Hugo Patterson, and Garth A. Gibson. Informed Multi-Process Prefetching and Caching. In *Proceedings fo the 1997 ACM SIGMETRICS Conference*, pages 100-114, June 1997

[9] H. Seok Jeon and Sam H. Noh. A Database Disk Buffer Management Algorithm based on Prefetching. In *Proceedings of the Seventh ACM CIKM Conference*, pages 167-174, Washington, DC November 3-7, 1998