메모리 참조 패턴 및 페이지 교체 기법

*이승훈, **최종무, *조성제, **조유근
*단국대학교 전산통계학과, **서울대학교 컴퓨터 공학과
*{zor4, sjcho}@dankook.ac.kr
**{choijm, cho}@ssrnet.snu.ac.kr

Memory Reference Patterns and Page Replacement Policies

*Seunghoon Lee, **Jongmoo Choi, *Seongje Cho, **Yookun Cho
*Dept. of Computer Science and Statistics, Dankook University
**Dept. of Computer Engineering, Seoul National University

요 약

페이징 기법의 효율성은 어떠한 페이지 교체 기법을 쓰느냐에 따라 결정된다. LRU 기법은 작업 부하의 변화에 잘 적응하여 많은 경우 좋은 성능을 보인다. 그러나 참조의 횟수에 대한 정보를 이용하지 못한다. LFU 기법은 지역성을 가지는 참조 패턴이 발견되면 좋은 성능을 발휘한다. 그러나 작업 부하가 변하는 경우 이에 적응하지 못한다. 여러 응용에 대해 참조 패턴을 분석하여 보면 참조되는 페이지는 최근성과 참조 횟수에 의해 가치가 결정되며, 따라서 LRU나 LFU 기법 한 가지만으로 페이지 교체 정책을 최적화 시킬 수 없다. 본 논문에서는 LRU 기법과 LFU 기법을 결합한 새로운 교체 기법을 제안한다. 이 교체 기법에서는 LRU 리스트와 LFU 리스트를 결합하여 사용함으로써 참조 시간뿐만 아니라 참조 횟수를 이용하여 페이지들을 교체한다. 트레이스 기반 모의 실험에서는 제안 기법이 순수 LRU 기법보다 나은 성능을 보일 때가 있다.

1. 서론

프로세스가 수행되면서 현재 주기억장치에 없는 페이지를 참조하면 페이지 폴트(page fault)가 발생한다. 그러면 트랩이 걸려 운영체제가 제어를 넘겨받게 되고 운영체제는 자유 페이지 프레임(free page frame)을 할당하여 폴트를 유발한 페이지를 반입하게 된다. 이때 자유 페이지 프레임이 없다면 페이지 교체가 발생하게 된다. 이처럼 페이지 폴트를 처리하기 위해 최악의 경우에는 디스크 접근이 두 번 필요하게 되며 이는 시스템 성능을 저하시키는 주요 요인이다. 따라서 페이지 폴트 수를 줄이기 위해 효율적인 페이지 교체 기법이 필요하며, 페이지 교체 기법은 다시 참조될 가능성이 큰 페이지들을 주기억장치에 계속 유지시켜 페이지 적중율을 높이고자 한다. 페이지 교체 기법으로 많이 사용되는 것이 LRU와 LFU 기법이다.

LFU 기법은 참조 횟수가 가장 작은 페이지를 교체한다. 이 기법은 참조패턴이 변하지 않는 안정적인 작업부하(workload)에서는 좋은 성능을 보이지만 작업부하가 변하는 경우 이에 적응하지 못하여 성능이 떨어진다. LRU 기법은 가장 오랫동안 참조되지 않은 페이지를 교체한다. 이 기법은 마지막 참조의 최근성만을 기준으로 페이지를 교체하기 때문에 페이지 참조 횟수를 알 수가 없다.

본 논문에서는 먼저 8개의 응용에 대해 메모리 참조 패턴을 분석해 보았다. 대부분의 응용에서 최근에 참조된 페이지가 계속 참조되는 경향이 있으며, 또한 자주 참조된 페이지도 계속 참조되는 경향이 있다. 이는 페이지 교체 알고리즘이 순수한 LRU나 LFU 기법이 최적의 페이지 교체 기법이 아니라는 것을 의미한다. 본 논문에서는 페이지의 최근성과 참조 횟수를 동시에 고려할 수 있도록 페이징 리스트를 LFU 리스트와 LRU 리스트로 분할하여 관리하는 새로운 페이지 교체 기법도 제안한다.

2. 관련연구

페이지 참조 패턴에 따라 최적의 페이지 교체 기법을 적용하고자 하는 연구들이 수행되어 왔다. SEQ 기법[1]은 정상적으로는 LRU 기법을 사용하다가 연속적인 페이지 폴트가 오랫동안 나타나면 그것을 감지하고, 그 때 MRU 기법을 사용한다. 이러한 연속적인 페이지 폴트가 일어나는 응용들에 대해서는 LRU 기법보다 좋은 성능을 보여주고, 일반적인 응용들에 대해서는 LRU 기법과 비슷한 성능을 보인다. EELRU[2] 기법은 SEQ 기법과 같이 정상적으로는 LRU 기법을 사용하다가 주기억장치보다 큰 반복 패턴을 감지했을 때 미리 페이지들을 제거한다.

LRFU 기법은 참조 횟수와 참조의 최근성 모두를 이용하는 블록 교체 기법이다. 이 기법에서는 블록의 재참조 가능성을 블록의 가치로 표현한다. 이 기법의 가장 기본적인 동작은 블록에 대한 참조가 발생하면 블록의 가치를 증가시키며, 시간이 지남에 따라 이러한 블록의 가치를 감소시키는 것이다.[3]

3. 페이지 참조 패턴

3.1 트레이스의 구성

리눅스 시스템 상에서 트레이스 응용인 VMTrace를 기반으로 제작된 8개 응용들의 트레이스는 표 1과 같다.

표 1 트레이스의 구성

트레이스로 사용된 응용	각 응용에 대한 설명	트레이스 크기	가상 메모리 크기
espresso	circuit simulator	326,938,361	77
gcc-2.7.2	GNU C/C++ compiler	37,524,334	458
gnuplot	the GNU plotting utility	68,458,509	7,718
grobner	grobner calculated Grobner basis functions	77,87,835	67
gs3.33	GhostScript, a software PostScript interpreter	134,371,942	558
lindsay	hypercube simulator	123,690,749	521
p2c	Pascal->C transformer	30,722,431	132
rscheme	implementation of Scheme	151,606,733	2,039

3.2 페이지의 참조패턴

표 1의 8개 응용 트레이스에 대한 참조패턴을 가상 시간 2,000,000으로 해서 분석하였고, 그중 gnuplot과 gs의 트레이스의 패턴은 그림 1과 같다. 트레이스의 일부분만 분석하여 보니, gnuplot와 lindsay를 제외한 나머지 응용들은 최근에 참조된 페이지가 강한 지역성을 보이기도 하지만 자주 참조된 페이지가 계속 참조되는 패턴을 보인다.(gs 패턴 참조) gnuplot 응용의 참조 패턴은 순차적인 참조와 지역적인 참조로 이루어져 있다.

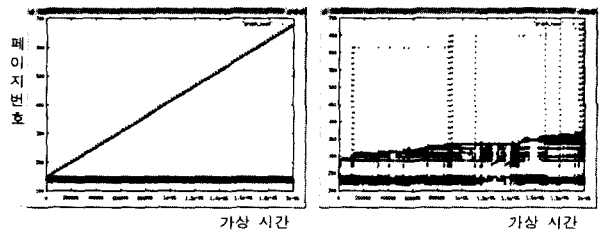

그림 1. gnuplot과 gs의 참조패턴

3.3 연관 참조기간을 고려한 페이지 참조 패턴

최초 트레이스를 정확히 분석하기 어려워, 연관 참조기간(co-relation period)[3]을 2000으로 하여 8가지 응용들에 대한 페이지 참조 패턴을 분석하였다. 연관 참조기간이 2000이라는 것은, 어떤 페이지 참조 시 이 페이지가 최근 가상시간 2000 이내에 참조된 적이 있다면 그 페이지가 처음에 한번 참조된 것으로 본다는 의미이다. 연관 참조기간을 고려한 참조 패턴의 일부가 그림 2와 3에 나타나 있다. 연관 참조기간을 고려한 패턴을 살펴보아도 대부분의 응용에서 페이지의 가치는 최근성 뿐만 아니라 참조 횟수에 의해서도 영향을 받는 다는 것을 보여 준다. gnuplot 응용의 경우 반복 패턴이 보인다. p2c 응용의 경우 지역적인 참조 패턴을 보인다.

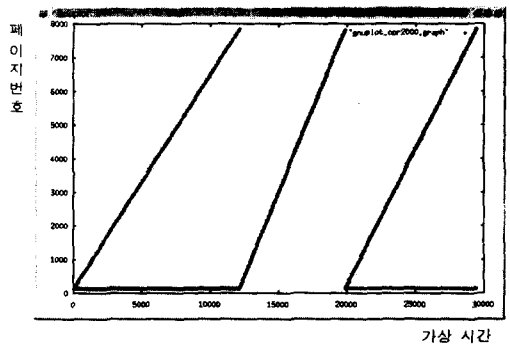

그림 2 gnuplot의 참조패턴

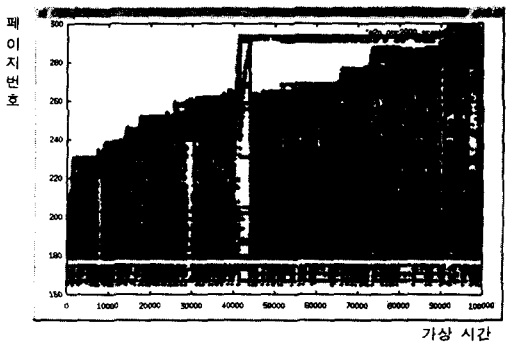

그림 3 p2c의 참조패턴

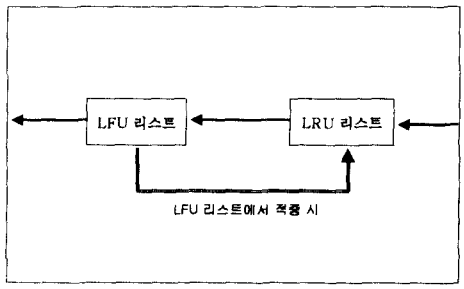

그림 4 제안 기법의 구조

3.4 제안 기법

참조 패턴들을 분석한 결과 페이지의 최근성과 참조 수가 중요하기 때문에, 두 특성 모두를 고려한 새로운 기법을 제안한다.(그림 4) 제안 기법에서는 페이지 리스트의 3/4은 LRU, 1/4은 LFU 리스트로 나누어 관리한다. 페이지가 처음 참조되면 LRU 리스트에 삽입되는데 LRU 리스트가 가득 차게 되면 가장 오랫동안 사용되지 않은 페이지를 LFU 리스트로 보낸다. 참조된 페이지가 LFU 리스트에서 있으면 그 페이지는 LRU 리스트에 보낸다. 두 리스트가 가득 찼을 때 페이지 폴트가 발생되면 먼저 LFU 페이지를 교체하고, LRU 페이지를 LFU 리

스트로 이동시킨 다음 새로 참조된 페이지를 LRU 리스트의 앞에 삽입한다.

4. 모의 실험 및 평가

연관 참조기간을 2000으로 한 트레이스들에 대해서 제안 기법, LRU 기법, FIFO 기법, LFU 기법, 그리고 MRU 기법으로 모의 실험한 결과가 표 2에 나타나 있다. p2c 응용에서는 제안 기법이 모든 프레임 수에 대해 좋은 성능을 보이며 최대 17% 좋은 성능을 보인다. gs 응용에서는 최대 2.7%, espresso 응용에서는 최대 12%, rscheme 응용에서는 최대 26% 좋은 성능을 보인다.

표 2 각 응용별 모의 실험 결과

p2c - 전체 참조 횟수: 99306

교체기법 \ 프레임 수	50	60	70	80	90	100	110	120	132
LRU	31585	22900	16333	10925	6396	3441	1599	644	132
제안 기법									132
FIFO	33906	25554	18124	12363	7570	4220	2167	1045	132
LFU	45546	36029	30318	21947	14514	9362	5374	2081	132
MRU	50083	40250	31372	24096	17548	11976	7590	3752	132

lindsay - 전체 참조 횟수: 1500286

교체기법 \ 프레임 수	150	200	300	400	500	521
LRU	633	633	633	633	524	521
제안 기법	633	633	633	633	524	521
FIFO	633	633	633	633	524	521
LFU	633	633	633	633	524	521
MRU	803178	160559	65688	65688	65688	521

gs - 전체 참조 횟수: 634197

교체기법 \ 프레임 수	100	200	300	400	500	558
LRU			1141	712	590	558
제안 기법	14556	2164				558
FIFO	23359	3449	1866	1140	808	558
LFU	296501	83396	55574	1266	638	558
MRU	470620	329714	202146	93605	24101	558

grobner - 전체 참조 횟수: 20384

교체기법 \ 프레임 수	35	40	45	50	55	60	67
LRU	711						67
제안 기법		300	153	104	80	72	67
FIFO	1319	574	234	149	112	78	67
LFU	1156	760	624	378	241	74	67
MRU	5839	4297	2824	1815	1075	378	67

gcc - 전체 참조 횟수: 173581

교체기법 \ 프레임 수	100	200	300	400	458
LRU					458
제안 기법	31848	8171	1707	700	458
FIFO	35332	8090	2736	1085	458
LFU	119958	110773	37079	3212	458
MRU	139009	98931	58250	19936	458

espresso - 전체 참조 횟수: 328973

교체기법 \ 프레임 수	20	30	40	50	60	70	77
LRU				175	93	88	77
제안 기법	48854	4957	1488				77
FIFO	49588	6800	1500	329	123	119	77
LFU	203173	101968	83407	78689	1703	80	77
MRU	249720	195087	153711	118847	69737	17026	77

gnuplot -전체 참조 횟수: 29394

교체기법 \ 프레임 수	5000	5500	6000	6500	7000	7500	7718
LRU	23141	23141	23141	23141	23141	23141	7718
제안 기법	21623	21419	21213	21007	20801	20596	7718
FIFO	23144	23144	23144	23144	23144	23144	7718
LFU	20502	20502	20502	20502	20502	20502	7718
MRU	15359	13958	12563	11142	9734	8329	7718

rscheme -전체 참조 횟수: 2342871

교체기법 \ 프레임 수	1000	1200	1400	1600	1800	2000	2039
LRU		15059	10930	7386	4442	2242	2039
제안 기법	21700						2039
FIFO	30926	21909	14586	9624	5242	2264	2039
LFU	1132707	782500	167708	99105	30777	2089	2039
MRU	1131836	891360	645159	389408	169232	11202	2039

* 각 항목은 페이지 폴트수

grobner와 gcc에서는 LRU가 좋은 성능을 보인다. 반복 패턴을 보이는 gnuplot 응용에서는 MRU 기법이 가장 좋은 성능을 보인다.

5. 결론

참조 패턴 분석 및 실험 결과를 보면, 순수 LRU 기법이나 순수 LFU 기법만으로는 페이지 교체를 최적화 시킬 수 없다. 어떤 응용들에 대해서는 두 가지를 결합한 기법이 좋은 성능을 보인다. 그리고 반복적인 패턴을 갖는 gnuplot 응용의 경우는 MRU 기법이 좋은 성능을 보인다. 향후, 프로세스의 세그먼트별로 나누어서 참조 패턴을 분석하고, 그 참조 패턴을 반영하는 세부적인 교체 기법을 적용하여 성능을 향상시킬 계획이다.

참고 문헌

[1] G. Glass and P. Cao, "Adaptive Page Replacement Based on Memory Reference Behavior", Proceedings of the 1997 ACM SIGMETRICS Conference, pp115-126, June 1997

[2] Y. Smaragdakis, S. Kaplan, and P. Wilson, "EELRU : Simple and Effective Adaptive Page Replacement", in SIGMETRICS '99

[3] 이동희, "LRFU 블록 교체 기법", 서울대학교, 1998. 2

[4] 최종무, 조성제, 노삼혁, 민상렬, 조유근, "적응력있는 블록 교체 기법을 위한 효율적인 버퍼 할당 정책", 정보과학회 논문지 : 시스템 및 이론 제 27권 제3호, 2000.3

[5] Andrew S. Tanenbaum, "Mordern Operating Systems", Prentice-Hall, Inc. , 1992

[6] A. Silberschatz, P. Galvin, "Operating System Concepts" Addison-Wesley, 1998