선택적 압축방식에 기반한 확장된 플래시 메모리 스와핑 시스템

임근수° 양훈모 차호정
연세대학교 컴퓨터과학과
zmnlks@chollian.net portent@nownuri.net hjcha@cs.yonsei.ac.kr

An Extended Flash Memory Swapping System Based on Selective Compression

Keun-Soo Yim° Hoon-Mo Yang Hojung Cha
Dept. of Computer Science, Yonsei University, Seoul 120-749, Korea

요 약

가상메모리를 사용하는 범용 컴퓨터 시스템, 특히 실시간 시스템에서 상대적으로 속도가 느린 자기디스크로의 스와핑은 해당 시스템의 성능을 저하시키는 핵심적인 요인이다. 본 논문에서는 일반적인 가상메모리 시스템의 성능을 향상시키고 실시간 프로세스에 대해 페이지 스와핑이 발생하여도 수행시간을 보증할 수 있는 방법으로 선택적인 압축방식을 사용한 확장된 플래시 메모리 스와핑 시스템을 제안한다. 그리고 제안하는 시스템을 세부적으로 설계하였으며 해석과 시뮬레이션을 통하여 지연시간과 공간 활용도를 평가하고 시스템의 특성을 상위 수준에서 DRAM 을 확장한 경우와 비교해 분석 및 고찰하였다. 그 결과 제안하는 시스템은 매 페이지 스와핑 시에 일정한 수행시간을 단축하며 선택적 압축방식과 수정된 버디 시스템을 사용하여 물리적인 플래시 메모리의 공간을 논리적으로 확장함을 검증하였다.

1. 서론

컴퓨터 시스템의 프로세서와 메모리 사이의 동작속도 차이는 시간이 흐름에 따라 보다 증가하고 있다. 더욱이 시분할방식 활용, 네트워킹 의존도 증가, 그리고 다양한 멀티미디어 자원의 지원으로 보다 많은 물리적 메모리가 요구되고 있다. 지난 10년간 워크스테이션에서의 일반 응용프로그램의 평균 요구 메모리 양은 매년 50-100% 증가해왔다 [1]. 이러한 메모리의 용량 문제와 접근속도 문제는 전통적인 컴퓨터 시스템 분야의 해결과제 중의 하나로 폭넓은 연구와 개발이 진행되고 있다. 대표적으로 캐시를 사용한 방법, 병렬구조를 사용한 방법, 그리고 압축을 통한 방법이 있다. 이러한 해결방법 중에서 가상메모리 시스템은 메모리의 공간문제와 속도문제를 구조적으로 해결하려는 방식으로 현재까지 그 성능을 인정 받아 널리 사용되고 있다 [2].

하지만 가상메모리 시스템에도 몇 가지 문제점이 있는데 그 중에 하나는 페이지폴트 시에 할당할 수 있는 주 메모리 공간이 부족하여 주 메모리에 적재된 페이지 중 일부를 선택적으로 보조기억장치로 스왑아웃하는 경우와 추후에 스왑아웃된 페이지에 접근을 시도할 경우 이를 다시 스왑인하는 경우에 발생하는 접근시간의 지연이다. 연구 [3] 에 의하면 요구되는 메모리의 70-90% 가 제공될 때 전체 프로세스의 수행시간은 몇 배 상승하며, 50% 정도만 제공될 때는 대부분의 프로세스가 5-20 배 느리게 작동하는데, 이러한 성능저하의 주 요인은 스와핑이다. 특히 이러한 페이지 스와핑은 실시간 프로세스와 같이 우선 순위가 높은 프로세스에게는 치명적인 문제가 될 수 있다.

메모리의 활용도와 접근시간의 문제를 해결하기 위한 노력은 계속되어 오고 있으며, 뛰어난 성능을 발휘하는 아이디어가 몇 가지 제안되어 검증된 바 있다. 관련 연구들은 크게 세가지 기본 아이디어를 응용한 형태로 분류할 수 있는데, 데이터에 대한 접근 시에 자연적으로 발생하는 시간적 지역성과 공간적 지역성을 활용하기 위해 캐시를 사용한 방법, 압축을 통해 전송하는 데이터의 크기를 줄여 속도를 향상시키고 동시에 공간의 활용도를 극대화 하려는 방법으로 이때 압축속도가 데이터의 전송속도보다 빠르다. 그리고 프로세서뿐만 아니라 기억장치도 병렬로 구성하여 동작속도를 향상시키려는 방법이 있다.

이 중에 본 논문에서 제안하는 시스템과 가장 유사한 방식은 압축을 통한 방식이다. 압축은 다양한 메모리 계층구조상에서 시도되었는데, 캐시수준의 연구 중에는 선택적으로 압축하는 프로세서 내장 캐시를 제안해 약 35% 의 캐시 실패율 (miss ratio) 이 감소함을 보인 연구가 있다 [4]. 연구 [5] 는 주 메모리에 CMMU 와 압축기/해제기를 추가한 하드웨어 압축방식과 소프트웨어 압축방식을 제안해 요구되는 메모리 양이 증가함에 따라 수배 가량 평균 접근시간이 단축됨을 보였다. 또한 가장 낮은 층인 자기디스크의 속도문제를 해결하기 위해서 압축방식을 사용하는 것도 옳은 방향임을 제시한 논문이 있다 [6]. 이러한 압축을 통한 방식을 사용할 수 있는 전제인 하드웨어 압축속도가 데이터 전송속도보다 충분히 빠름을 보인 연구가 있는데, 이 중에 X-Match 와 이의 확장인 X-RL (X-Match and Run Length) 알고리즘은 100 MB/sec 의 압축속도를 보인다 [7].

위의 조사결과에 의하면 지금까지 플래시 메모리를 주 메모리와 보조기억장치 사이에 추가하여 스와핑 용도로 사용한 시스템은 제안되거나 개발된 적이 없는 새로운 방식으로 가상메모리 시스템의 스와핑 시의 성능지연을 개선하기 위한 새로운 시도이다. 따라서 본 논문에서는 이러한 가상메모리 시스템이 가지는 메모리 접근시간의 문제를 개선하기 위해 기존 메모리 계층구조에 플래시 메모리를 추가하고 이를 지능적으로 사용할 수 있는 몇 가지

방법을 제안한다. 핵심적인 특성은 메모리에 사상된 파일의 페이지, 우선순위가 높은 프로세스의 갱신되지 않은 페이지 등도 스와핑 대상에 포함하는 점과 스와핑 시에 해당 페이지를 압축률에 따라 다양한 크기의 저장블록을 선택해 내부파편을 최소화하도록 플래시 메모리를 관리하는 것이다. 제안하는 시스템의 보다 세부적인 특성은 2장 시스템 구조에 기술한다. 3장에서는 성능을 측정해 평가하고, 4장에서 결론을 맺는다.

2. 시스템 구조

제안하는 선택적 압축방식을 사용한 확장된 플래시 메모리 스와핑 시스템의 구조와 동작방식은 각각 그림 1a 와 그림 2 와 같다. 기존 컴퓨터시스템의 메모리 계층구조의 주 메모리와 보조기억장치 사이에 플래시 메모리와 압축기/해제기를 추가한 형태로 공간 활용도를 향상시키기 위해 압축기/해제기를 사용하여 선택적으로 압축해 플래시 메모리에 스와핑한다. 이때 그림 1b 와 같은 알고리즘을 운영체제의 페이지폴트 처리기에 추가하여 사상된 파일의 페이지와 우선순위가 높은 프로세스의 페이지까지 스와핑 대상에 포함시킴으로써, 실시간 프로세스와 같은 높은 우선순위를 갖는 프로세스의 수행시간을 상당부분 보증할 수 있으며 동시에 보조기억장치의 쓰기 캐시 역할도 수행한다. 또한 플래시 메모리의 영속성을 사용해 예외상황에는 복구기능을 수행할 수 있는 장점이 있다. 뿐만 아니라 단순하게 원 페이지를 저장하지 않고 압축을 통하여 압축률이 임계값 이하인 경우 이를 선택적으로 저장하도록 하여 플래시 메모리의 공간 활용률을 극대화한다.

제안하는 시스템의 하드웨어 구조는 보조기억장치의 I/O 제어기에 크게 I/O 캐시, 압축기/해제기, 그리고 플래시 메모리를 장착한 형태이다. 이때 I/O 캐시는 주 메모리와 통신을 위한 DMA 버퍼 용도와 플래시 메모리에 저장한 페이지들의 테이블을 관리하는 용도로 사용한다. I/O 버스를 통해 호스트 프로세서로부터 페이지 스왑아웃 제어신호를 받으면, 해당 페이지를 압축하고 압축률과 해당 프로세스의 우선순위를 고려하여 선택적으로 플래시 메모리에 기록한다. 추후에 페이지 스왑인 제어신호를 받으면 압축된 페이지를 실시간에 해제하여 I/O 버스를 통해 전달한다. 그리고 보조기억장치의 사용률에 여유가 있을 때에는 사상된 파일의 페이지들을 보조기억장치에 기록하여 예외상황에 보다 적극적으로 대비한다. 만약 시스템이 예기치 않게 동작하는 경우에는 시스템이 다

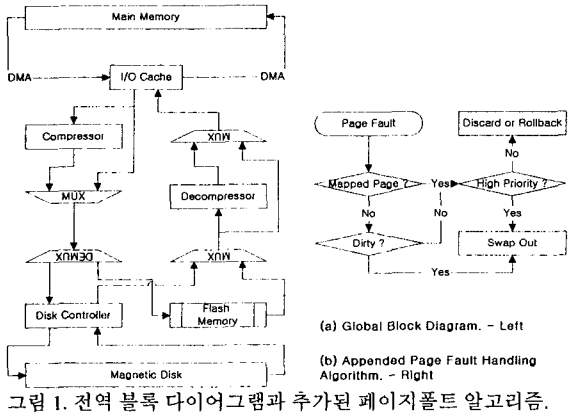

그림 1. 전역 블록 다이어그램과 추가된 페이지폴트 알고리즘.

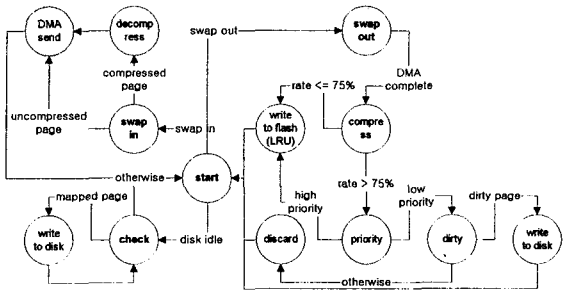

그림 2. 유한 상태기계.

시 시작할 때 플래시 메모리를 검사하여 아직 보조기억장치에 쓰여지지 않은 페이지가 있으면 이를 갱신하는 방식으로 예외상황에 대처한다.

일반적인 운영체제의 페이지 크기인 4KB 의 데이터를 X-RL 알고리즘을 사용해 압축할 경우 약 50% 정도로 압축되며 경우에 따라서 10% 에서 103% 정도 범위의 압축률을 갖는다. 이때 100% 를 초과하는 경우는 원래 데이터보다 압축 후 크기가 더 커진 경우로 데이터팽창 (data expansion) 이라고 하며, 이 경우에는 원 페이지를 사용한다. 그리고 동일한 크기의 블록을 사용하지 않을 경우 이를 관리하기 위한 자료구조가 복잡해지고 갱신할 페이지가 기존 페이지보다 큰 두꺼운 쓰기 (fat write) 현상이 발생할 수 있기 때문에 동일한 크기의 블록을 사용하며 이는 플래시 메모리의 기본 I/O 방식과도 일치한다.

기본적인 플래시 메모리 관리방식은 버디 시스템을 바탕으로 하며 압축률에 따라 표 1과 같은 크기의 블록을 지원한다. 이때 보다 세부적으로 블록을 정의하여 내부파편을 줄이기 위해 두 가지 타입의 버디 시스템을 사용하는데, A 타입은 512B 의 한 블록을 1개부터 시작해서 2의 배수로 4KB 까지 확장한 모델이며 B 타입은 3개부터 시작해서 2의 배수로 4KB 까지 확장한 모델이다. 따라서 제안하는 시스템의 선택적 압축의 임계값은 가장 큰 블록의 임계 압축률인 75% 이다. A 타입과 B 타입의 초기 할당비율은 4장의 결과에 근거해 설정하며, 동작 시에는 시스템의 요청에 맞춰 적응적으로 분할하는 방식을 취한다.

3. 성능 평가

이 장에서는 제안하는 시스템의 성능측정 과정과 분석결과를 기술한다. 접근시간을 살펴보면 IBM™ Deskstar 40GV 자기디스크를 스왑장치로 사용할 경우 4KB 크기 페이지의 평균 전송시간은

$$t_{access} = t_{seek} + t_{rotational} + t_{transfer}$$

$$= 9.5(ms) + \frac{11.1(ms)}{2} + 32(Kbits) \times \frac{1000(ms)}{372(MBits)} \approx 15.1(ms)$$

이다. 그리고 Intel™ StrataFlash 28F128J3A 플래시 메모리를 사용해 3KB 로 75% 압축된 페이지를 읽고 쓰는데 걸리는 시간은

$$t_{read} = (25*10^{-3}) \times 3072 \approx 77(\mu s)$$

$$t_{write} = t_{erase} + t_{program} = (6.8*10^{-3}) \times 3072 \approx 20.9(ms)$$

이다. 이 해석에 사용한 자기디스크와 플래시 메모리는 최근에 출시된 일반 개인용 컴퓨터의 기억장치들이다.

이 결과를 전체 시스템의 성능측면에서 분석해 보면, 일반적으

로 자기디스크 I/O 는 DMA 형태로 이루어지기 때문에 해당 연산을 하는동안 호스트 프로세서는 이를 바쁜 대기 (busy wait) 형태로 기다리지 않는다. 따라서 스와핑 속도가 빠르다고 해서 호스트 프로세서의 불필요한 연산시간이 줄어듬을 의미하지는 않는다. 대신에 응용프로그램 등의 수행시간이 향상됨을 보증할 수 있는데 디스크를 사용한 스왑아웃과 스왑인의 경우 총 30.2 ms 가 소요되는데 반하여 플래시 메모리를 사용한 경우에는 총 21 ms 가 소요되어 매 페이지 스왑아웃마다 최대 8.8 ms 의 수행시간 단축을 보증한다. 그러므로 이러한 제안하는 시스템의 성능 이득은 앞으로 플래시 메모리 기술의 발전으로 더욱 극대화될 수 있을 것이다.

다음으로 공간 활용도를 구하기 위해서 30,000 개 페이지를 포함한 MS-Windows 의 스왑 파일을 입력으로 사용하여 트레이스 기반 (trace-driven) 시뮬레이션을 하였다. 시뮬레이터 구현에는 하드웨어적으로 실시간 구현 가능한 무 손실 압축알고리즘인 X-Match 를 사용하였으며, C++ 언어를 사용해 윈도우용으로 개발하였다. 시뮬레이션 결과 그림 3과 같은 압축률 별 분포를 보였으며, 평균 압축률은 47-51% 이지만 실질적으로 14% 와 103% 부근에 편중됨을 알 수 있다. 표 1은 X-Match 구현에 사용되는 사전의 크기가 증가할수록 압축률이 감소함을 보인다. 사전의 크기를 128 에서 1024 까지 범위로 설정한 경우 평균 공간 활용도는 175-182% 이다. 그리고 초기 A 타입과 B 타입은 57 대 43 비율로 설정하면 된다.

제안하는 시스템과 유사하게 성능을 향상시킬 수 있는 방법인 주 메모리 확장은 여전히 시스템의 성능을 향상시키는 좋은 방법이다. 하지만 많은 프로세스가 동작하며 보다 많은 메모리를 사용할 경우 스와핑은 필연적으로 발생한다. 따라서 제안하는 시스템은 메모리와 보조기억장치의 사용률이 높은 시스템과 실시간 시스템과 같이 수행시간의 보증이 중요한 시스템에 적용될 때 그 효용성이 극대화된다. 그리고 제안하는 시스템에서 보다 빠른 DRAM, 비휘발성 메모리 등의 기억장치를 사용하지 않고 플래시 메모리를 사용한 이유는 사상된 페이지까지 스와핑 대상에 포함시켜 디스크 캐시와 같은 기능을 저비용으로 구현하기 위한 것이다. 즉, DRAM 은 영속성이 없으며 비휘발성 메모리는 비용이 플래시 메모리에 비해 높으므로 사용하지 않았다.

4. 결론

가상메모리 시스템의 성능을 저하시키는 요인 중의 하나인 스와핑 시의 수행속도를 개선하기 위해서 제안된 플래시 메모리를 사용한 선택적 압축방식의 확장된 스와핑 시스템을 해석과 시뮬레이션을 통해 분석한 결과, 기억장치의 종류에 의존적이지만 일

표 1. 사전 크기와 압축률 사이의 관계. (단위: %)

Type	Page Size	Compression Rate	Access Rate depend on Entry Size			
			128	256	512	1024
A	0.5 KB	0.0 ~ 12.5	9.17	9.19	9.21	9.24
B	1.0 KB	12.5 ~ 25.0	21.41	21.45	21.49	21.51
A	1.5 KB	25.0 ~ 37.5	8.28	8.65	8.83	8.97
B	2.0 KB	37.5 ~ 50.0	16.26	17.13	20.06	22.05
A	3.0 KB	50.0 ~ 75.0	21.74	21.27	19.04	17.70
A	4.0 KB	75.0 ~ 120	23.15	22.31	21.37	20.54
Average Compression Rate			57.20	56.58	55.52	54.74
Space Expansion Rate			175	177	180	182

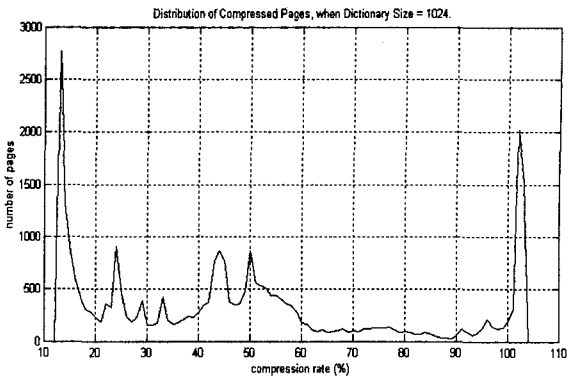

그림 3. 압축률 별 페이지 분포.

반 PC 용 장치를 가정할 경우, 매 스와핑당 최대 8.8 ms 의 수행시간 단축을 보증함을 확인하였다. 그리고 압축과 2가지 타입 6가지 크기의 블록을 지원하는 수정된 버디 시스템을 통해 평균 공간 활용률을 약 175-182% 까지 향상시킬 수 있음을 보였다. 따라서 제안하는 시스템의 수행시간 단축특성을 활용해 일반 가상메모리에 기반한 시스템에 적용하여 전체 시스템의 수행속도를 향상시킬 수 있으며, 실시간 시스템 등의 수행시간이 중요한 시스템에 적용하여 정적인 수행시간 보증이나 예측을 가능하게 할 수 있다. 또한 평균 공간 활용률을 극대화시키는 특성을 활용하여 내장형 시스템과 모바일 시스템 등의 보조기억장치의 크기가 작은 시스템의 파일시스템으로도 응용해 볼 수 있다. 끝으로, 제안하는 시스템을 상용화하기 위해서는 실제로 시스템을 설계/제작하여 성능을 검증해야 하는데, 이는 또한 앞으로 요구되는 중요한 연구과제이기도 하다.

참고 문헌

[1] J. L. Hennessy, D. A. Patterson, David Goldberg, *Computer Architecture: A Quantitative Approach, 3rd Edition*, Morgan Kaufmann Publishers, 2002.

[2] Harvey G. Cragon, *Memory Systems and Pipelined Processors*, Jones and Bartlett Publishers, 1996.

[3] H. Garcia-Molina, A. Park, L. R. Rogers, "Performance through Memory," *Proceedings of the 1987 ACM SIGMETRICS Conference*, pp. 122-131, 1987.

[4] Jang-Soo Lee, Won-Kee Hong, and Shin-Dug Kim, "Design and Evaluation of Selective Compressed Memory System," *International Conference on Computer Design*, Texas, USA, pp. 184-191, October 1999.

[5] M. Kjelso, M. Gooch, and S. Jones, "Performance Evaluation of Computer Architectures with Main Memory Data Compression," *Journal of Systems Architecture*, Vol. 45, pp. 571-590, 1999.

[6] M. Burrows, C. Jerian, B. Lampson, T. Mann, "On-line Data Compression in a Log-structured File System," ACM SIGPLAN Notices, *Proceedings of the fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, Vol. 27, No. 9, September 1992.

[7] S. Bunton, G.. Borriello, "Practical Dictionary Management for Hardware Data Compression," *Communications of the ACM*, Vol. 35, No. 1, January 1992.