모바일 환경에서의 가상 메모리 압축 시스템 설계

정진우[0] 장승주
동의대학교 컴퓨터공학과
jinwoo@dogneui.ac.kr, sjjang@dongeui.ac.kr

Design of Virtual Memory Compression System on Mobile Device

Jin-Woo Jeong[0] Seung-Ju Jang
Dept. of Computer, DongEui University

요 약

일반적으로 메모리 관리에서 가장 큰 문제점은 느린 보조기억 장치의 속도와 빠른 주기억장치의 속도 차이에서 나타나는 성능 저하라고 할 수 있다. 요구 페이징 기법에서 페이지 폴트가 일어나면 페이지 교체 정책에 의해 필요 없는 페이지들을 스왑 디바이스로 이동을 시킨다. 이때 느린 보조기억장치의 접근 속도로 인한 응답시간의 지연은 전체적인 시스템 성능의 저하를 초래한다. 이때 스왑 디바이스로의 접근 횟수와 페이지의 크기를 줄일 수 있다면 페이지 아웃되는 응답시간을 높일 수 있을 것이다. 따라서 본 논문에서는 가상 메모리 압축 시스템을 설계하여 스왑 아웃되는 시간을 줄이며 압축된 페이지들을 사용함으로써 메모리 공간을 절약하여 시스템의 전체적인 성능 향상을 위한 모바일 시스템을 설계한다.

1. 서 론

가상 메모리 시스템에서의 가장 큰 부하는 페이지 폴트가 일어났을 때 느린 스왑 디바이스로 페이지들의 이동에서 발생하는 성능 저하라고 할 수 있다[1]. 가상 메모리 시스템은 페이지 폴트가 일어나면 페이지 교체 정책에 의해 주기억장치 내의 페이지들을 스왑 디바이스로 이동을 한다. 그리고 이후에 이 페이지들이 다시 필요하면 스왑 디바이스에서 주기억장치로 읽어 들이게 된다. 이렇게 페이지 폴트가 일어났을 때 이들 페이지들을 스왑 디바이스로 이동시키지 않고 주기억장치 내의 일정 영역에 압축된 캐시 영역을 할당하여 압축된 페이지를 저장한다면, 스왑 디바이스로 이동하는 시간과 횟수를 감소하여 페이지 폴트 응답시간을 줄이며, 주기억장치에 저장되는 페이지들의 공간 활용도를 높일 수 있다. 또한 모바일 디바이스의 적은 저장 장치의 특성을 고려해서 압축된 스왑 디바이스를 사용하여 보조기억장치의 저장 공간 활용을 높일 수 있다. 본 논문에서는 압축된 캐시 영역을 설계하여 페이지 폴트가 일어났을 때 스왑 아웃되는 페이지를 압축하여 저장하는 가상 메모리 압축 시스템을 설계한다.

2. 가상 메모리 압축 시스템

가상 메모리 압축 시스템은 페이지 폴트 발생시 느린 스왑 디바이스로 페이지를 이동할 때의 응답시간을 줄여 시스템의 성능을 향상시키는 방법이다. 페이지 폴트가 일어나면 가상 메모리 시스템은 주기억장치 내의 비어 있는 공간을 검색하고, 만약 비어 있는 페이지 공간이 존재한다면 보조기억장치에서 프로세스가 필요한 페이지를 주기억장치로 읽어 들인다. 그러나 비어 있는 주기억장치 내에 할당할 페이지가 없을 경우는 페이지 교체 알고리즘에 의해서 주기억장치 내의 사용되지 않는 페이지를 스왑 아웃 시키게 된다. 이런 페이지 교체 알고리즘으로 사용되는 예로는 대표적으로 Least Recently Used(LRU) 알고리즘과 FIFO 알고리즘 등[3]이 있다. 이렇게 페이지 교체 정책에 의해 교체되는 페이지들은 보조기억장치 내의 스왑 디바이스로 이동하게 되는데, 이때 주기억장치의 빠른 속도와 보조기억장치의 느린 처리 속도의 차이에서 낭비되는 시간이 발생한다. 이 문제점을 해결하기 위해 주기억장치에서 스왑 디바이스로 이동하는 페이지들을 압축하여 주기억장치 내의 압축된 캐시 영역에 저장한다. 그리고 이후에 압축된 캐시 영역의 저장 공간이 모자라게 되면 압축된 캐시 영역의 페이지들을 스왑 디바이스로 이동시키게 된다. 이렇게 스왑 아웃되는 응답시간을 향상시켜 전체적인 시스템의 향상을 기하고자 본 논문에서 가상 메모리 압축 알고리즘을 이용한 모바일 환경을 제안한다.

3. 관련 연구

가상 메모리 압축 시스템은 Fred Douglis에 의해서 이미 제한된 바가 있다. Douglis의 연구는 DECstation 5000/200 workstation의 스왑 아웃될 때의 traffic을 줄이기 위해 압축된 캐시를 이용했다[4]. 압축된 캐시는 유동적인 크기의 원형 버퍼의 구조를 이루고 있었으며, 페이지 폴트가 발생하여 스왑 아웃되는 페이지들을 압축하여 원형 버퍼에 저장하며, 이 때 사용되는 압축 알고리즘으로 LZRW1을 사용하였다[10]. Douglis의 연구 결과는 특별한 응용 프로그램의 경우 스왑 아웃될 때의 traffic을 감소시키거나 제거 할 수가 있었으나 대부분

본 논문은 2002년도 동의대학교 자체 학술연구비 지원을 받아 이루어졌음

의 모든 응용 프로그램에서는 적용되지 못했다.

그 원인으로써는 compression, decompression 시간이 페이지를 이동할 때의 시간보다 많이 소모된다는 것이며, 압축된 페이지가 압축되지 않은 페이지 보다 얼마만큼의 크기를 줄일 수가 있느냐에 따라서 성능향상을 기대할 수 있다. 그리고 또 한 가지는 응용 프로그램의 실행 시에 데이터에 접근하는 패턴에 따라서 달라질 수 있다.

본 논문에서는 모바일 환경을 위한 가상 메모리 압축 시스템을 제안한다.

Doug Banks와 Mark Stemm은 포터블 디바이스 내에서의 압축된 메모리 시스템을 제안했는데 Douglis의 연구에서의 문제점으로 Desktop PC나 Workstation에서는 압축된 메모리 시스템에서의 시스템 성능향상은 기대할 수 없으므로 포터블 디바이스 내에서의 압축된 메모리 시스템을 고안하였다. Doug Banks와 Mark Stemm의 연구는 PDA(Personal Digital Assistant)나 Laptop과 같은 포터블 디바이스가 느린 통신 수단으로 연결되어 크기가 큰 응용 프로그램에 접근하는 경우이다. 그래서 그들의 실험은 느린 스왑 디바이스로의 접근하는 양을 줄여 시스템의 성능 향상을 증명하였다[1].

그러나 그들의 연구는 지금으로써는 구형 제품인 PDA나 Laptop을 사용하였다. 현재 출시되는 포터블 디바이스의 경우는 로컬 저장 장치로써 속도가 빠른 RAM이나 하드디스크를 지원하므로 Doug Banks와 Mark Stemm이 제안하는 시스템으로는 시스템의 성능 향상을 기대하기 힘들다. 그래서 본 논문에서는 이들의 연구를 기반으로 포터블 디바이스 내에서의 가상 메모리 시스템의 성능향상을 위해서 좀더 빠른 압축 알고리즘과 압축된 스왑 디바이스를 사용한다.

4. 압축 알고리즘

압축 기술에서 가장 중요하게 고려해야 할 사항은 압축 알고리즘의 압축율과 실행 속도 그리고 메모리의 요구량을 우선적으로 생각해야 한다[5]. 포터블 디바이스의 특성상 페이지의 크기가 일반 PC나 Workstation 시스템에 비해 작다는 것이다. 리눅스의 경우 일반적으로 Alpha AXP 시스템은 8KB 페이지를, Intel x86 시스템은 4KB 페이지를 사용하고 있다. 그에 비해 임베디드 리눅스의 경우는 512B 혹은 1KB 페이지를 사용하고 있으므로 압축 알고리즘은 범용적인 알고리즘 보다는 적은 양의 압축에서 효율이 높고 압축 알고리즘으로 동작 원리가 단순한 알고리즘을 적용하여 페이지의 스왑 아웃되는 속도와 compression, decompression 속도의 차이를 감소시킨다면 시스템의 성능 향상을 기대할 수 있다. 그리고 포터블 디바이스의 경우 한정된 저장 장치를 사용하고 있다. PDA의 경우는 RAM(Random Access Memory)을 보조 기억장치로 사용하는데 RAM에는 일반 응용 프로그램과 데이터, 그리고 스왑 디바이스 영역과 함께 사용이 된다. 따라서 압축된 스왑 디바이스의 사용으로 저장 공간을 늘린다면 적은 저장 공간을 응용 프로그램이나 데이터를 저장하는데 사용하여 시스템의 저장 용량을 향상시킬 수 있다.

압축 알고리즘의 종류에는 크게 무손실 데이터 압축과 손실 데이터 압축으로 나눌 수 있다. 무손실 데이터 압축의 종류에는 RLC(Running Length Coding), VLC(Variable Length Coding), Huffman coding, DBC(Dictionary Based Coding), LZ(Lempel-Ziv)-base와 같은 것들이 있다. 본 논문에서는 논리적이고 수학적인 작업 과정이 간단하여 바르게 동작하면서도 압축율이 좋은 LZ 기반의 알고리즘인 LZW 압축 알고리즘을 사용한다. LZ 기반 알고리즘은 압축할 자료를 코드화 하면서 기억장치 내에 문자열에 대한 색인표(Index Table)를 생성하여, 한 문자열씩 확인하여 그 내용이 같은 문자열을 다시 만나면 그것을 미리 간직해둔 하나의 색인을 갖게 된다. 그렇게 중복되는 모든 부분들은 색인표를 가리키는 하나의 색인을 가지게 돼 그 크기를 줄일 수 있다[9].

5. 가상 메모리 압축 시스템 설계

가상 메모리 압축 시스템을 이해하기 위해서는 페이지의 스왑 아웃과 폐기에 대해서 알아볼 필요가 있다. 시스템에서 물리적 메모리가 부족하게 되면 메모리 관리 시스템은 물리적 메모리를 해제하려 한다. 이 일은 커널 스왑 데몬(kswapd)에게 할당된다. 커널 스왑 데몬의 역할은 메모리 관리 시스템이 효율적으로 동작할 수 있도록 시스템에 충분한 free 페이지가 있도록 하는 것이다. 그림 1에서 보는 것처럼 스왑 데몬은 시스템의 free 페이지의 수가 너무 적지 않은지 확인하여 페이지를 해제해야 할 필요가 있는지 결정한다.

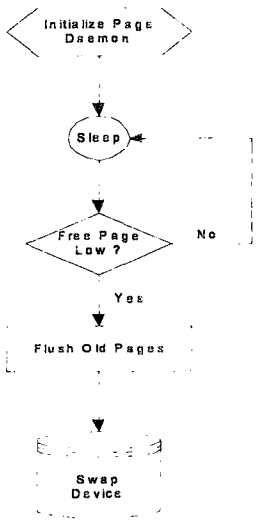

그림 1. Page Out Algorithm

Free 페이지의 수가 많으면 커널 스왑 데몬은 잠들어 있다가 free 페이지의 수가 작아지면 물리적 페이지의 수를 줄이기 위하여 버퍼 캐시와 페이지 캐시의 크기를 줄이고 메모리 페이지를 스왑 아웃하고 폐기한다[7,8]. 이 때 주기억장치와 스왑 디바이스 사이에 처리 속도의

차이로 시스템의 성능저하가 나타나는데 주기억장치와 스왑 디바이스 사이에 캐시 역할을 하는 메모리 영역을 만듦으로써 스왑 디바이스에 접근하는 횟수를 줄여서 시스템의 성능 향상을 기대한다. 그래서 본 논문의 설계는 주기억장치에 스왑 디바이스와의 캐시 역할을 할 수 있는 압축된 캐시 영역을 설정하여 주기억장치에서 스왑 디바이스로 이동하는 페이지들의 수를 감소시키려고 한다.

주기억장치는 그림 2에서와 같이 압축된 메모리와 압축이 되지 않은 메모리로 나누어지며, 서로 간에 페이지들을 이동할 때 compress하고 decompress 하게 된다.

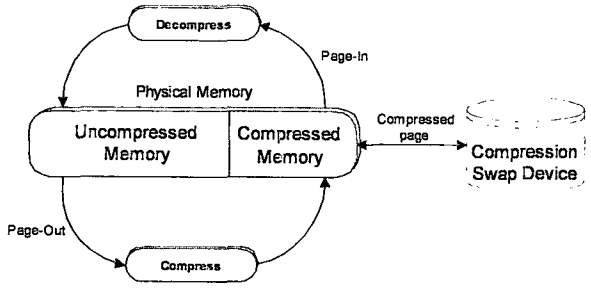

그림 2. Compression Memory Architecture

물리적 메모리 내에 압축된 메모리 공간을 할당하여 페이지 폴트가 일어났을 때 스왑 아웃 되는 페이지를 스왑 디바이스로 이동하지 않고 이들 페이지를 압축하여 압축된 메모리로 이동한다. 그리고 압축된 메모리 내의 여유 공간이 없을 때 실질적인 스왑 영역인 로컬 디스크 내의 스왑 디바이스로 압축된 데이터가 이동한다[2,6].

프로세스는 필요한 페이지를 먼저 압축된 메모리 영역에서 찾고 만약 찾는 페이지가 압축된 메모리 내에 있다면 압축을 풀어서 읽어 들이고, 없을 경우에는 스왑 디바이스에서 찾는다.

이 경우는 보조기억장치의 느린 속도 때문에 압축된 페이지들로 스왑 아웃될 때의 부하를 줄일 수가 있다. 최근의 충분한 물리적 메모리와 보조기억장치를 보유한 시스템에서 로컬 디스크의 느린 속도로 인한 스왑 아웃 응답시간을 줄일 수 있는 방법이라고 할 수 있다. 본 논문에서의 구현은 리눅스 상에서의 Loadable Device Driver를 이용하여 설계한다면, 시스템 개발에서 shutdown이나 reboot이 필요하지 않다는 이점을 가지고 있다. 그리고 본 논문에서의 설계는 PDA에 임베디드 리눅스 시스템을 구성하고 임베디드 리눅스 시스템의 메모리 관리 시스템에 가상 메모리 압축 시스템을 적용하여 PDA의 수행속도를 향상시키기 위한 설계를 하였으며, 압축된 스왑 메모리와 스왑 영역을 사용하여 적은 용량의 PDA를 개선하도록 설계한다.

6. 결 론

본 논문의 제안은 현재 구현 단계에 들어갔으며, 본 논문에서는 설계 부분에 관련된 내용을 소개하였다.

본 논문에서는 스왑 아웃되는 페이지들을 압축하여 주기억장치 내의 압축 캐시 영역에 저장하는 메모리 관리 기법인 가상 메모리 압축 시스템을 설계함으로써 스왑 아웃되는 시간을 줄이고 스왑 영역의 사용량을 줄임으로써 전체적인 시스템의 메모리 공간을 절약하여 시스템 성능 향상을 높일 수 있는 모바일 시스템을 설계하였다.

본 논문의 설계는 주기억장치를 압축되지 않은 영역과 압축된 캐시 영역으로 나누어서 페이지 폴트가 발생했을 때 느린 스왑 디바이스로 이동하는 페이지들을 압축된 메모리 영역에 저장한다. 그리고 이후에 이 페이지들이 프로세스에 의해 필요로 될 때 느린 스왑 디바이스가 아닌 압축된 메모리 영역에서 가지고 옴으로써 보조기억장치로의 접근 횟수를 줄일 수 있도록 설계하였으며, 압축된 스왑 영역을 사용함으로써 보조기억장치의 저장 공간을 늘릴 수 있었다.

그러나 현재 사용되고 있는 포터블 디바이스의 보조기억 장치와 주기억장치는 동일한 RAM을 사용하고 있기 때문에 서로간의 데이터 전송에서 나타나는 처리 속도의 차이를 기대 할 수 없다. 그러나 만약 이들 장치가 느린 무선 통신을 이용한 데이터의 접근이나 응용 프로그램의 실행 시 스왑 아웃되는 시간을 급격히 저하시킬 수 있으며, 앞으로 적용될 대용량의 느린 보조기억장치가 도입될 경우 가상 메모리 압축 시스템은 매우 높은 시스템의 성능 향상을 기대할 수 있다. 향후 연구로는 적은 데이터의 압축에서 보조기억장치로의 전송 속도보다 compress, decompress 수행 속도가 더욱 빠르게 동작하며, 압축율이 높은 압축 알고리즘을 개발할 예정이다.

참 고 문 헌

[1] Dong Banks, Mark Stemm,"Investigating Virtual Memory Compression on Portable Architectures",1995
[2] Caroline Benveniste, "Cache-Memory Interfaces in Compressed Memory System", 2001 IEEE
[3] Peter J Denning, "The working set model for program behavior",communications of the ACM 1968
[4] Fred Douglis. "The compression Cache: Using on-line compression to extend physical memory". USENIX Proceedings, Winter 1993
[5] S. Kwong and Y. F. Ho, "A statitical Lempel-Ziv Compression Algorithm for Personal Digital Assistant(PDA)", 2001 IEEE
[6] Ian McDonald, "The use of a Compressed Cache in an Operating System supporting Self-Paging", September 8, 1999
[7] Sumit Roy, Raj Kumar, Milos Prvulovic, "Improving System Performance with Compressed Memory",2001 IEEE
[8] David A Rusling,"The Linux Kernel",January 1999
[9] M.S. Pinho, W.A Finamore, "Using arithmetic code to improve performance of Lempel-Ziv encoders", Electronic Letters, Aug 31, 2000
[10] Ross N. Williams, "An Extremely Fast Ziv-Lempel Data Compression Algorithm", Data Compression conference, 1991