# 64비트 환경에서 메모리 테스트 영역 확장을 위한 프로그램 재배치 기법*

박한주° 박희권 최종무 이준희¶
단국대학교 컴퓨터 과학, (주)삼성전자¶
{rainbird°, phk0210, choijm}@dankook.ac.kr, joonlee@samsung.com¶

# Program Relocation Schemes for Enhancing Memory Test Coverage on 64-bit Computing Environment

Hanju Park°, Heekwon Park, Jongmoo Choi, Joonhee Lee¶
Computer science Dankook university, Samsung Electronics Co. Ltd¶

요 약

최근 64비트 CPU의 시장 출시가 활발해지고 있으며, 메모리 모듈 또한 대용화가 이루어지고 있다. 이에 대용량 메모리를 64비트 CPU 플랫폼에서 효과적으로 테스트하는 방법을 개발할 필요성이 대두되고 있다. 본 논문에서는 x86-64 기반 리눅스 2.6.11 커널에서 물리 메모리의 테스트 영역을 확장하는 기법을 제안한다. 제안된 기법에는 응용이나 커널에서 물리 메모리에 대한 직접 접근, 프로그램을 사용자가 원하는 물리 메모리로 배치, 프로그램의 동적 재배치 등의 방법을 통해 테스트 영역을 확장 한다. 현재 64비트 CPU를 지원하는 OS는 리눅스와 윈도우즈 64비트 에디션 등이 있다. 기존 리눅스 커널을 그대로 사용하였을 때, 프로그램 등이 이미 사용 중인 물리 메모리에 대해서는 메모리 테스트를 수행 할 수 없었으나, 각 프로그램들을 물리 메모리에서 재배치하여, 원하는 곳의 메모리를 테스트 할 수 있도록 커널 수정을 통하여 구현하였다.

## 1. 서 론

멀티미디어 데이터가 증가하며 사용 환경이 다양해지고 대용량 프로그램이나 효과적인 멀티미디어 데이터의 처리 욕구에 따라 고성능 PC에 대한 필요성이 지속적으로 증가하고 있다. 이에 따라 64비트 CPU들이 시장 주력 상품으로 자리 잡아 가고 있다. 64비트 CPU는 32비트 CPU에 64비트 확장 기능이 있는 제품과 없는 제품이 있다.[1][2] 전자는 AMD와 INTEL에서 각각 AMD64와 EM64T기능이라고 명명된 64비트 확장 기능이 있는 CPU들이며, 후자로는 INTEL과 HP가 공동 개발한 아이테니엄 CPU가 있다. 64비트 확장 기능은 기존 32비트 환경과 하위 호환성을 유지 하면서 64비트로 CPU가 동작하는 기능이다.

64비트 확장 기능을 지원하는 운영체제로는 리눅스와 윈도우즈가 있다. 리눅스는 64비트 확장 기능을 x86-64 아키텍처로 분류하여 AMD64기능 개발 초기부터 지원하여 개발해 왔다. 그리고 배포판별로 정식 지원 플랫폼 제품군으로 판매되고 있다. 윈도우즈는 64비트 확장 기능 제품과 64비트 기능 제품으로 구분하여 출시하고 있다.

에모리 모듈의 고속화, 고용량, 저가격화가 지속적으로 이루어지면서 근래 일반 PC의 기본 물리 메모리 모듈이 512MB나 1GB가 일반화 되고 있다.[4] 개별 메모리 모듈이 증가함에 따라 기존에 사용하던 32비트 기반 메모리 테스트 프로그램을 사용하여 테스트 하는 기법을 64비트 기반으로 변경 하는 것이 필요하게 되었으며, 일반 사용자 환경에서 메모리 모듈의 안정성을 테스트해야 할 필요성에 의해 운영체제가 동작하고 있는 상태에서 메모리를 테스트 할 수 있는 기법에 대한 연구가 필요하게 되었다. 현재 메모리 테스트 프로그램으로 가장 널리 사용되고 있는 것은 memtest86이다. 하지만 memtest86은 32비트 모드로 작동하며, 운영체제가 있는 상태에서 수행되는 것이 아닌 독자 구동(stand alone) 방식이다.

본 논문은 운영체제가 있는 상태에서 효과적으로 메모리 테스트를 수행 할 수 있는 기법을 제안한다. 리눅스 커널 2.6을 기반으로 한 x86-64 아키텍처 운영체제가 수행되는 환경에서 세 가지 단계에 걸쳐 커널을 수정하였다. 첫째, 물리 메모리에 직접 접근 할 수 있는 경로를 확보하였다. 둘째, 커널과 프로그램을 메모리 테스트가 효율적으로 이루어 질 수 있도록 전체 물리 메모리에서 한쪽으로 배치하였다. 셋째, 프로그램에서 실제 사용 중인 물리 메모리의 특정 영역을 다른 영역으로 재배치하여 원하는 물리 메모리 영역을 테스트 할 수 있도록 구현하였다.

본 논문의 구성은 다음과 같다. 2장에서는 64비트 CPU와 물리 메모리 재배치를 위한 세 가지 단계의 설계를 설명한다. 3장에서는 실제 구현 및 실험 결과를 서술하며, 4장에서는 결론 및 향후 연구 내용을 정리한다.

## 2. 설 계

64비트 확장 환경인 x86-64 아키텍처로 만들어진 리눅스 커널 이미지를 통해 부팅을 하게 되면, 64비트 모드로 운영체제가 동작하게 된다. 이 64비트 모드에서 동작할 때에는 레지스터와 메모리의 기본 단위가 모두 64비트가 되며, 32비트 프로그램을 수행할 때엔 CPU레벨에서 32비트 프로그램을 위한 레이어를 통해 수행시간의

* 본 연구는 (주)삼성전자의 테스트 기반기술 연구지원으로 수행된 프로젝트임.

낭비 없이 32비트 CPU에서 수행한 것과 동일한 수행시간을 보장하게 된다. 이를 효율적으로 지원하기 위해 x86-64 아키텍처로 제작된 배포판을 설치해 보면, lib와 lib64 두 개의 런타임 라이브러리들의 저장소가 있는 것을 확인 할 수 있다.

64비트 환경으로 운영체제가 동작할 때에는 64비트, 32비트 프로그램을 둘 다 수행할 수 있다. 물론, 운영체제가 64비트이기 때문에 하드웨어 드라이버 등은 모두 64비트로 제작된 것이라야 한다. 리눅스에서 32비트 운영체제 환경에서 사용할 수 있는 가상 주소는 총 0xFFFFFFFF이다. 하지만 64비트 OS환경에서는 0xFFFFFFFFFFFFFFFF 로 증가하게 되었기 때문에 가상 주소 공간이 32비트로 동작 할 때와는 다르게 관리한다.

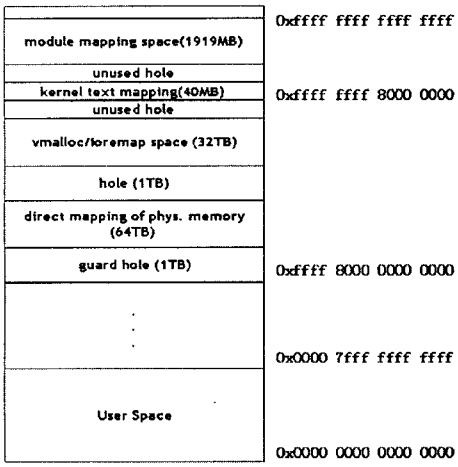


그림 1 x86-64 아키텍처의 가상메모리 구조

그림 1은 x86-64 리눅스 아키텍처의 가상 메모리 구조를 나타내고 있다. 리눅스 커널이 i386아키텍처에서는 0xc0000000 이었으나, x86-64에서는 0xFFFFFFFF80000000에 위치한다. 일반 프로그램이 할당되어 사용하게 되는 유저 공간은 0x0000000000000000부터 0x00007FFFFFFFFFFF까지 할당되어 있다.

물리 메모리를 효율적으로 사용하며, 대용량 프로그램을 수행 할 수 있게 하는 페이징에서 페이지 프레임 사이즈는 4KB로 동일하며, PML4(page map level 4)가 추가되어 사용된다. 그림 2는 페이징을 사용하는 x86-64에서 가상주소를 물리 메모리로 변환하는 4단계 주소 변환이다.

일반적으로 수행되고 있는 운영체제에서 물리 메모리를 테스트 하려면 물리 메모리에 직접 접근할 수 있어야 한다. 리눅스 커널은 가상 메모리 공간에서 수행되며 물리 메모리에 직접 접근을 하기 위해 가상 메모리 공간에 물리 메모리 공간을 사상시켜 주거나, 물리 메모리 공간 전체를 직접 접근할 수 있는 장치를 열어 접근하는 방법

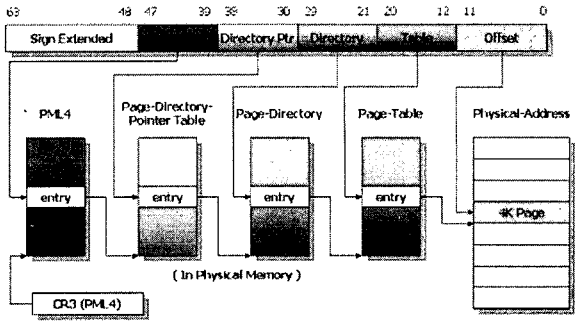


그림 2 x86-64의 주소변환 단계

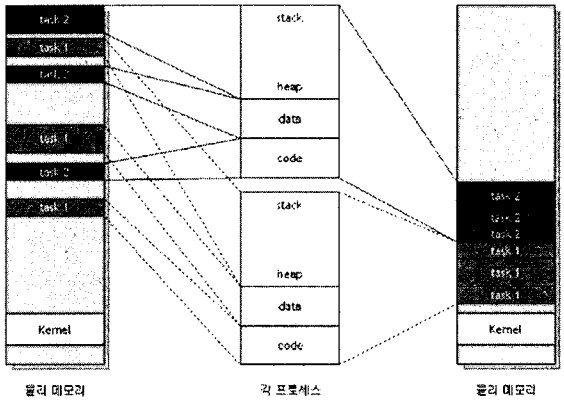


그림 3 프로세스를 물리메모리 특정 영역으로 할당

이 필요하다. 메모리를 사상 시켜주는 방법이 가상 메모리 공간에서 손쉽게 접근 할 수 있고, 기존 메모리 테스트 방법을 그대로 적용시키기도 편리하다. 따라서 본 논문에서는 페이지 테이블을 각각 수정 및 관리하여 가상 메모리를 통해 원하는 물리 메모리를 직접 접근할 수 있도록 하였다.

한편, 운영체제가 정상적으로 동작하고 있을 때, 실제 사용하고 있는 메모리들을 살펴보면, 물리 메모리 곳곳의 페이지 프레임에 산재되어 있는 것을 볼 수 있다. 이것은 메모리 할당 정책과 프로세스들이 수행, 종료를 반복하면서 발생한 일종의 메모리 단편화이다. 하지만 페이지 프레임의 산재는 메모리 테스트에서 문제가 된다. 즉, 그림 3의 왼쪽처럼 되어있는 페이지 프레임 할당을 오른쪽처럼 사용자가 원하는 특정 위치에 모여 있도록 수정하여야 한다. 본 논문에서는 버디 시스템을 수정하여 동작중인 프로그램들이 특정 위치에 자신의 페이지 프레임을 할당할 수 있도록 수정하였다.

이제부터 프로그램 재배치에 대해 설명하겠다. 프로그램들이 수행되고 있을 때, 메모리 테스트 프로그램이 특정 물리 메모리 공간을 테스트 하려고 한다면, 그 물리 메모리 위치에 존재하던 프로그램을 다른 비어있는 영역으로 옮겨 주어야 한다. 실제 수행중인 프로그램을 물리

메모리 상에서 다른 페이지 프레임으로 옮기는 작업은 다음과 같이 구현된다.

① 해당 프로그램 수행을 중단시키고 스케줄링을 멈춘다.
② 프로그램이 사용 중인 페이지 테이블들을 확인한다.
③ 사용 중인 물리 메모리를 확인한다.
④ 비어있는 물리메모리를 확보한다.
⑤ 페이지 프레임 단위로 비어있는 곳으로 옮긴다.
⑥ 페이지 테이블에 지정되어 있는 물리 메모리 주소를 옮긴 곳의 물리 메모리 주소로 바꾼다.
⑦ 프로그램 수행과 스케줄링을 재개 시킨다.

## 3. 구현 및 실험

본 실험은 x86-64기능이 있는 Intel Xeon 3.2GHz CPU(EM64T지원)를 사용하였으며, Gentoo 2005.0 배포판 기반에서 수행되었다. 그리고 리눅스 커널은 2.6.11을 사용하였다. 물리 메모리 크기는 6GB이다.

그림 4 사상 및 테스트 결과 확인

먼저 물리 메모리 직접 접근 경로를 확보하는 것은, 커널 가상 메모리 영역에 물리 메모리를 직접 사상시키는 방법을 사용하였다. 가상 메모리 중 vmalloc/ioremap 섹션으로 분류되어 있는 32TB의 공간에 사상 시켰기 때문에 그림 4를 보면 0xFFFFC20000080000에 사상된 것을 볼 수 있다.

버디 시스템을 수정하여 부팅 과정에서 커널이 차지하고 있는 영역과 데몬, 프로그램이 차지하고 있는 영역을 전체 물리 메모리 중 한쪽으로 제한하였다. 그 결과 커널과 프로그램을 100MB 이내로 제한할 수 있었다.

그림 5는 프로그램 재배치 실험 결과를 보여준다. 그림에서 위의 캡처 화면은 재배치 전의 정보이며 아래의 캡처 화면은 재배치 후의 정보이다. 재배치 전후의 가상 메모리 위치는 동일함을 알 수 있다(그림에서 mm정보). 하지만 물리 페이지 프레임은 0x13275000에서 0x37E02000으로 재배치가 되었음을 알 수 있다.

프로그램을 수행 중에 옮기기 위해서 가용 물리 메모리 공간을 관리하는 버디 시스템의 관리 공간을 2개로 분할하였다. 프로그램을 수행시키면, 첫 번째 공간의 페이지 프레임을 할당 받는다. 그리고 재배치가 요청되면, 두 번째 공간을 관리하도록 버디 시스템을 변경한다. 옮기고자 하는 프로그램의 페이지 테이블 엔트리와 해당 파일과 사상되어 있는 부분을 비워준다. 페이지 테이블이 가리키고 있던 페이지의 내용을 두 번째 공간에 할당

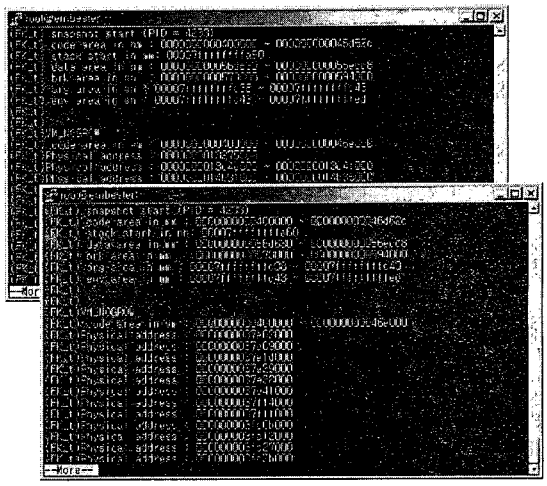

그림 5 프로그램 재배치

받은 페이지에 메모리 카피를 통해 복사하고, 이와 관련된 각종 권한들을 조정한다. 각 프로그램은 프로그램 별로 개별적으로 존재하는 가상 주소 공간에서 수행되기 때문에, 물리 주소 어느 곳에 위치하고 있어도 프로그램 수행에는 영향이 없다.

## 4. 결 론

본 연구에서는 x86-64 아키텍처의 리눅스 커널 2.6이 수행될 때, 물리 메모리의 효과적인 테스트를 위하여, 수행중인 프로그램을 어떻게 제어 하여 물리 메모리 테스트를 수행하기 좋은 환경으로 만들어 줄 것인가를 설계하였다. 대용량 물리 메모리를 테스트 할 때 커널에서 물리 메모리를 일종의 디스크로 보고 경로를 확보하여 테스트 할 수도 있으며, 본 실험에서와 같이 커널을 부분적으로 수정하여, 가상 메모리 영역에 사상시키는 방식을 사용 할 수도 있다. 후자를 사용한 것은 기존 테스트 프로그램의 테스트 기법을 적용하기 쉬웠으며, 일반적으로 프로그램들이 리눅스에서 사용하는 방법을 그대로 사용하기 때문에 문제해결이 쉬우며, 확장성이 좋다.

추후, 전체 메모리 영역에 대한 테스트를 위해 일반 프로그램이 아닌 커널 자체를 옮길 수 있게 설계, 구현하고, 캐시 기능을 효율적으로 활용하여 테스트 성능을 향상시키는 것에 관한 연구를 진행함을 목표로 한다.

## 참고문헌

[1] AMD64 homepage, http://www.x86-64.org
[2] Intel, "Extended Memory 64 Technology Software Developer's Guide", 2004
[3] AMD, "AMD64 Architecture Programmer's Manual Vol1-Vol5", Sep. 2003
[4] Samsung Semiconductor DRAM, http://www.samsung.com/Products/Semiconductor/DRAM/index.htm
[5] memtest86, http://www.memtest86.com/