Journal of Korea Multimedia Society Vol. 7, No. 8, August 2004(pp. 1058-1065)

내장형 리눅스 커널에서 멀티미디어 서비스를 위한 메모리 복사 감소 기법의 구현

김 정 원†

요 약

단순 모니터에서 CPU, 메모리, 그리고 하드디스크를 갖춘 셋탑박스에 이르기까지 내장형 시스템은 다양한 응용에 사용되고 있다. 특히 휴대용 또는 소형기기에 멀티미디어를 서비스하는 경우가 증가하고 있고 이들 시스템에는 내장형 운영체제가 탑재되고 있다. 본 논문에서는 멀티미디어 서비스를 위한 내장형 리눅스가 탑재된 임베디드 시스템에서 응용 프로그램과 운영체제 커널사이의 메모리 복사 요구를 감소시키는 Null copy기법을 제안한다. 저성능의 컴퓨팅 파워 및 저용량의 메모리를 가진 내장형 시스템에서 연속 미디어를 네트워크를 통해 실시간으로 전송하고자 할 때 Null copy 기법은 시스템의 QoS를 만족시킬 수 있다. 웹 카메라를 내장형 리눅스 개발 보드에 장착하여 영상 전송 환경을 구축한 결과 Null copy 기법은 CPU 이용률 및 마감시간 실패측면에서 기존 커널에 비해 우수한 성능을 나타내었다.

Implementation of Memory Copy Reduction Scheme for Multimedia Service in Embedded Linux Kernel

Kim Jeong Won†

ABSTRACT

Embedded system is widely used in various applications from simple monitor to a set-top box with CPU, memory and hard disk drives. Specially, embedded OS is ported in moveable or small machinery since it ordinarily transmits multimedia data. In this paper, we propose Null copy scheme on the embedded linux system for multimedia service, which can reduce memory copy overhead from user address space to kernel one, and vice versa. Since embedded system for networked multimedia service has low level computing power as well as memory, the Null copy scheme can provide more improved QoS. Our image transmission experiment results on embedded linux target board(CPU utilization an Deadline miss rates) installed a web camera have shown that the proposed scheme can increase fast response and lower CPU overhead.

Key words: Linux, Embedded System(내장형 시스템)

1. 서 론

내장형 리눅스는 포스트 PC 시대에서 최적의 운영체제가 될 것으로 기대되면서 다수의 회사 및 연구기관에 의해 개발 및 구현되고 있다[1]. 서버 시장에서 리눅스는 비교적 성공적인 시장 진입을 하고 있지만 전자 수첩, 무선 전화기 등 단순 단말기와 PDA, 스마트 폰 등의 정보 단말기의 중요한 구분 요인이 되는 소프트웨어의 유연성, 즉 새로운 응용 프로그램을 수행할 수 있는 능력을 가지고 있지만 작은 이동형 단말기에 필요한 모든 기능을 수용할 수 있는 운영체제인가 라

※ 교신저자(Corresponding Author) : 김정원, 주소 : 부산시 사상구 괘법동(617-736), 전화 : 051)309-5749, FAX: 051)309-5657, E-mail : jwkim@silla.ac.kr
접수일 : 2003년 4월 9일, 완료일 : 2003년 12월 8일
† 신라대학교 컴퓨터정보공학부 교수
※ 이 연구는 2002년도 신라대학교 연구비로 이루어졌음.

는 질문에 아직은 부족한 점이 많은 것이 사실이다[2]. 즉, 리눅스는 포스트 PC 단말기와 같은 내장형 시스템에 적합한 운영 체제로 인식되고 있지만 이동형 단말기의 커널로서 가져야 할 특성을 완벽하게 만족시키지는 못하고 있다. 이러한 요구사항에는 먼저 커널의 측면에서 전력 관리 기능, 메모리 관리 기능, 플래쉬 파일 시스템, XIP(Execute In Place), 각종 장치 드라이버 지원 등이 있다[2]. 미들웨어 측면에서는 GUI, 보안 관련 솔루션, 원격 데이터베이스 참조 솔루션, 싱크 관련 솔루션 등이 있고 응용 프로그래밍 측면에서는 기존의 마이크로소프트웨어사 문서와의 호환성을 가지는 응용 프로그램들, 다양한 언어 지원, 멀티미디어 플레이어, 각종 플러그 인을 지원하는 브라우저, 개인 정보 관리, 게임들이 있다.

본 논문에서는 위의 요구사항들 중에 네트워크에 연결되어 멀티미디어를 서비스하는 내장형 리눅스 시스템에서 응용프로그램의 데이터를 네트워크로 효율적으로 전송하는 기법을 제안하고자 한다. 내장형 시스템은 일반적으로 저성능의 컴퓨팅 파워를 가지고 있는 반면 네트워크형 멀티미디어 서비스는 지속적으로 데이터를 생성하여 네트워크로 전송해야 한다. 일반적인 데스크톱환경에서도 디스크에서 커널 주소 공간으로의 복사는 DMA(Direct Memory Access) 등에 의해 CPU의 부하가 감소될 수 있지만, 커널 주소 공간에서 사용자 버퍼의 주소 공간으로의 복사는 CPU가 관여한다. 소규모의 비정기적인 요구를 가진 비연속형 데이터의 경우는 큰 문제가 되지 않지만, 대용량의 연속미디어의 경우 시스템 처리율을 저하시키고 QoS를 보장할 수 없다. 즉, 기존의 리눅스 커널은 사용자와 커널 메모리사이의 데이터 복사를 피할 수 없다. 이 메모리 복사는 CPU에 오버헤드를 초래하여 응용 프로그램의 실시간성을 저해한다. 데스크탑에서의 기존 연구에서는 하드디스크에서 커널 메모리 및 사용자 메모리간의 데이터 복사 오버헤드를 한 번으로 감소시켜 실시간성을 증가시킨 바 있다[3]. 본 논문에서는 이것을 내장형 시스템에 확장시킨 Null Copy 기법을 제안하고 성능을 평가하고자 한다.

논문의 구성은 다음과 같다. 2장에서는 관련 연구를 기술하고, 3장에서는 리눅스 주소 변환 기법과 제안하는 Null copy 기법을 설명하고, 4장에서는 커널 구현 환경과 성능 측정 결과를 설명한다. 마지막으로 본 논문의 결론과 향후 연구 방향에 대해 논한다.

2. 관련연구

커널상에서 메모리 복사 오버헤드 감소를 위한 관련 연구에는 IRIX의 direct I/O, Container Shipping, FBuf, 전통적인 유닉스의 raw I/O, 내장형 시스템을 위한 XIP 등이 있다.

SGI IRIX 운영체제의 XFS는 파일 시스템의 데이터를 사용자 버퍼에 직접 전송하는 direct I/O를 제공한다[4]. IRIX의 버퍼 캐시 모듈은 XFS와 사용자 메모리의 버퍼 블록을 상호 연결하여 일반 I/O를 수행하듯이 direct I/O를 수행한다. 이 direct I/O는 대규모 데이터 베이스가 파일시스템에 저장되어 있을 때, 혹은 대규모 파일이 고속으로 전송될 경우 효율적임이 증명되었다[13,14].

Container Shipping[5], FBuf[6]는 가상 메모리 재매핑 기술을 이용하여 운영체제 차원에서 물리적인 데이터 움직임을 최소화 시키는 기법들이다. Container shipping은 응용프로그램이 명시적으로 시스템 버퍼를 할당 및 해제할 수 있고, I/O의 수단으로 네이밍(naming)하여 사용자 버퍼와 시스템 버퍼사이에 데이터를 이동할 수 있다. FBuf는 cross-domain 사이의 데이터 복사 없이 전달만으로 I/O 성능을 향상시킨다.

내장형 시스템에서는 실행 이미지가 이미 CPU의 주소 공간에 존재하는 메모리(플래쉬 메모리, 마스크 롬, 또는 DRAM)위에 존재하기 때문에 파일을 DRAM에 다시 로드하지 않고 그대로 실행할 수 있으며 이러한 기술을 XIP라고 한다. XIP를 사용함으로써 파일 시스템에서 DRAM으로의 실행 이미지 로드가 제거되므로 메모리 요구량 감소, 프로그램 로딩 시간의 개선, 페이지 캐싱이 없어짐으로 인한 문맥 교환 속도 향상, 전력 소비 감소 효과 등이 있다.

Milind[7] 는 데이터 복사 오버헤드를 감소시키기 위한 zero copy 기법을 4.4 BSD Unix에서 설계 및 구현하였다. 이 기법은 mmbuf 라는 새로운 버퍼 관리 시스템을 통하여 디스크에서 커널 버퍼로 전송된 데이터가 사용자 버퍼로 복사되지 않고 네트워크 소켓으로 직접 전송된다.

3. 내장형 리눅스 커널에서 Null Copy의 설계 및 구현

리눅스 파일시스템의 읽기 및 쓰기 연산은 항상

버퍼 캐쉬 시스템을 통과한다. 캐쉬의 적중률이 높은 경우에는 디스크 접근 회수를 줄여 응답시간의 향상을 가져온다. 그러나, MOD(Multimedia on Demand)와 같은 멀티미디어 데이터의 경우는 다음의 결과들을 초래할 수 있다. 1) 연속성의 특성을 가지고, 데이터의 재사용성이 낮아 높은 캐쉬 적중률을 기대하기 어렵다. 2) 사용 가능한 메모리가 부족할 경우에는 잦은 페이지 스왑핑이 발생하여 기대하는 버퍼 캐쉬의 효과를 얻기 힘들다. 3) 대량의 데이터를 장시간 전송해야 하므로 CPU의 메모리 복사 오버헤드가 상당히 증가할 것이다. 따라서 사용자가 할당하는 메모리와 커널이 할당하는 메모리 사이의 복사 오버헤드를 감소시키면 네트워크상에서 멀티미디어 데이터를 서비스하는 저 성능의 컴퓨팅 파워를 지닌 내장형 시스템의 실시간성이 증가될 것으로 기대된다. 이 기능을 구현하기 위해서는 사용자가 할당한 메모리 버퍼를 커널의 프로토콜 스택이 인식할 수 있어야 한다. 따라서, 응용 프로그램은 커널에게 해당 소켓이 Null Copy 임을 플래그로 세팅하여 통보하고, 사용자 버퍼의 시작번지 및 크기를 커널에게 파라메터로 알려야 한다. 커널 내부적으로는 사용자 버퍼의 가상주소를 커널의 물리주소로 변경하기 위한 함수를 구현해야 하며 소켓 버퍼의 포인터를 변경한 커널 물리주소로 매핑하는 과정이 필요하다.

3.1 Null Copy의 I/O 시나리오

Null Copy는 스트림을 목적지로 전송할 때 사용자 주소 공간에서 커널 주소 공간으로 복사가 가상적으로 이루어짐을 의미한다. 따라서, 커널의 시스템 버퍼로는 데이터가 복사되지 않고 사용자 주소 영역에서 네트워크로 직접 전송되게 된다. 현존하는 운영체제에서 DMA는 물리적 디바이스와 주 메모리 사이의 데이터 전송 시 CPU의 오버헤드는 획기적으로 제거할 수 있지만 커널 버퍼와 사용자 버퍼 사이의 메모리 복사 오버헤드는 해결할 수 없다. 장시간의 데이터 전송을 요구하는 멀티미디어 응용에 있어서 이 오버헤드는 시스템 처리율을 저하시킬 수 있다. Null Copy 의 핵심은 기존 버퍼 캐시 시스템을 그대로 사용하면서 사용자 버퍼 영역과 소켓을 연결하는 것이다.

다음은 Null-Copy의 입출력에 대한 시나리오를 설명한다.

◎ 단계 1 : 메모리 할당 및 소켓 생성

정적 할당 또는 동적인 할당 방법으로 메모리를 할당한다. 본 논문에서는 malloc()을 사용하여 메모리를 할당한다. 또한 응용 프로그램이 사용할 소켓을 생성한다.

◎ 단계 2 : 멀티미디어 데이터 생성

단계 1에서 할당한 메모리 주소에 멀티미디어 데이터를 생성한다. 본 논문의 실험에서는 카메라에서 캡쳐한 영상 데이터를 사용자 버퍼에 저장한다.

◎ 단계 3 : ioctl을 통해 Null Copy임을 통지

fcntl(socket ID, O_NULLCOPY, Buffer_Size) : 소켓 ioctl호출을 통해 해당 소켓 ID가 Null Copy 임을 커널에 명시한다. 커널은 커널내의 소켓 구조체에 본 논문에서 정의한 O_NULLCOPY 플래그를 셋팅한다.

◎ 단계 4 : send

send(socket ID, buffer, size) : 이 단계에서는 사용자 버퍼 영역을 커널이 사용할 수 있는 구조로 변경하며 사용자 버퍼의 시작주소를 커널의 주소영역으로 변환한다. 그리고, 소켓 버퍼의 포인터를 버퍼 헤드의 데이터 포인터로 리다이렉션하여 패킷을 전송한다. 다음은 단계 4의 상세 내용으로서 커널이 내부적으로 소켓이 Null_Copy 를 처리하는 과정을 설명한다.

◎ 단계 4-1 : 플래그 검사

커널 내부 함수인 sys_send() 시스템 콜 인터페이스에서 커널의 소켓 구조체에 O_NULLCOPY 플래그가 설정되어 있는지 검사한다. 성공하면 단계 2로 분기하고 그렇지 않으면 정상적인 소켓 전송 함수를 호출한다.

◎ 단계 4-2 : 커널 페이지 구조체 작성 (*kiobuf()*)

사용자 버퍼의 가상주소공간에 대해 입출력을 수행하기 위한 커널 페이지 구조체를 만든다. 현재 리눅스 커널은 4 킬로바이트의 페이지 크기를 지원하므로 사용자 버퍼공간을 4 킬로 바이트 단위로 분할하여 구조체를 생성하는 것이다. 그림 1은 사용자 주소공간을 커널 주소공간으로 매핑하기 위한 커널 입출력 구조체(kiobuf)이다. 이 구조체의 각 인덱스는 사용자 버퍼에 대한 물리적 주소, 페이지 번호 등에 대한 정보를 유지하여 패킷 전송시 정보를 제공한다.

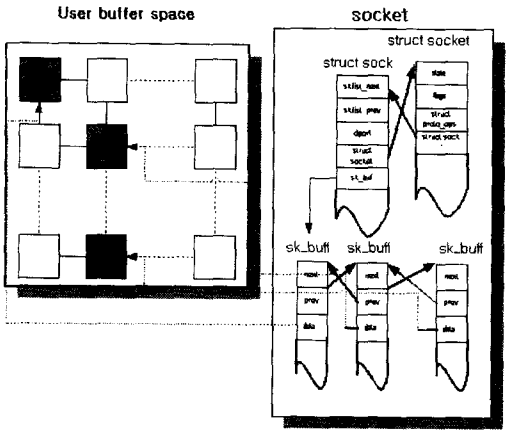

그림 1. 사용자 버퍼와 연결된 소켓 구조체

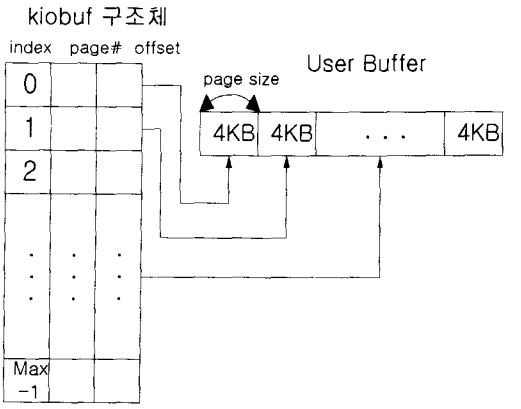

그림 2. 사용자 버퍼를 위한 커널 버퍼 구조체 매핑

◎ 단계 4-3 : 주소 변환(*map_userSpace_kernel-Space()*)

사용자가 커널에 전송한 버퍼의 시작주소는 가상주소로서 커널시 사용하는 물리적 주소로 변경해야 야 한다. 이 단계에서는 kiobuf에 저장된 각 페이지에 대해 가상주소를 물리주소로 변환한다. 주소 변환 방법은 3.3에서 설명한다.

◎ 단계 4-4 : 페이지 잠금 (*page_lock()*)

주소 변환된 각 페이지에 대해 페이지를 잠근다(locking). 이는 커널이 해당 페이지에 대한 입출력 수행 도중 스웹핑이 발생하는 것을 막기 위해서 이다.

3.2 변경된 커널 구조

이 절에서는 커널의 주요 변경 사항을 설명한다. 그림 2는 사용자 버퍼와 연결된 소켓 구조체의 구조를 나타낸다. 그림 2에서 변경된 소켓의 struct socket 구조체는 현재 프로세스가 넘겨준 사용자 주소공간의 버퍼 포인터를 유지한다. 이 포인터는 소켓의 send() 함수에 의해 파라메터로 커널 함수로 넘겨진다. 그리고, 소켓이 Null Copy 임을 커널에 통지하기 위해 본 논문에서는 *fcntl()* 함수에 O_NullCopy라는 플래그를 세팅한다.

소켓에 대한 send() 연산이 수행되면 소켓은 Null Copy 인지를 먼저 검사한다. 소켓은 그림 2에서 보듯이 sk_buff의 data 포인터를 사용자 버퍼 포인터로 사상시킨다. 프로토콜 스택은 이 포인터의 데이터를 네트워크로 전송하게 된다. 소켓 구조체의 sk_buff는 한 번의 전송 시 사용되는 데이터 량을 가리키는 포인터인데 4K 바이트로 세그먼화된 사용자 버퍼를 다시 한번의 전송단위를 가리키는 sk_buff가 다시 세그멘테이션한다.

그림 3은 소켓에 대한 ioctl에서 파라메터로 들어온 소켓이 Null Copy임을 커널에 통지하기 위해서 변경된 sock_no_fcntl() 함수 이다. 본 논문에서는 이 함수의 case 문에 O_NULLCOPY 플래그를 추가하여 Null_Copy 소켓과 normal 소켓을 구분하고 있다.

```
int sock_no_fcntl(struct socket *, int cmd, unsigned
long arg)
begin
    switch(cmd)
    {
        case F_SETDOWN:
                :
        case F_GETOWN:
                :
        case O_NULLCOPY:
                sock->nullcopy = TRUE;
        default:
                return(-EINVAL)
    }
end
```

그림 3. 변경된 시스템 호출 소켓 fcntl()

3.3 가상주소에서 물리주소로 변환

i386 계열의 CPU는 페이지화된 세그멘테이션 기법으로 주소를 변환한다[8]. 메모리에 대한 사용자관점과 물리적인 관점이 동일한 세그멘테이션 기법의 장점과 외부 단편화를 제거할 수 있는 페이징 기법의

장점을 결합한 것이다. 리눅스 커널은 가상 주소에서 동작하므로 라이브러리 호출 read(fd, buffer, size)에 의해 제공된 가상주소는 시스템 호출 인터페이스 sys_read()에서 선형주소(linear address)로 변환된다[9]. 이 선형 주소는 페이지 테이블을 참조하여 물리주소(physical address)로 변환된다. 본 연구에서는 사용자 버퍼의 물리주소를 얻기 위해 page_map() 함수를 제공한다. 그림 4는 주소 변환의 과정을 보여준다.

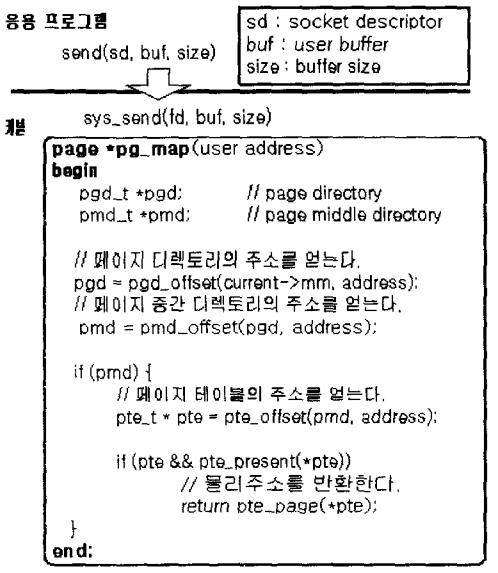

그림 4. 주소 변환 과정

리눅스 커널은 3단계 페이징을 제공한다. 즉, page directory, page middle directory, page table이다. i386 계열의 프로세서에 탑재되는 리눅스 커널은 2단계 주소 변환을 하므로 page middle directory는 하나만 존재한다[10]. 사용자가 제공한 버퍼의 가상주소는 페이지 디렉토리와 페이지 테이블을 참조하여 페이지 프레임의 시작주소를 찾아낸다. 이 시작주소에 선형주소의 오프셋 값을 더하면 실제 물리주소로 변환된다[9]. 다음은 본 논문에서 구현한 그림 4의 주소 변환 과정 주요 부분을 설명한다. *read(), write()* 라이브러리 호출 시 *sys_read(), sys_write()* 시스템 호출이 각각 발생하는데 이때 파라메터로 넘겨진 사용자 버퍼의 주소는 선형주소이다. 먼저 시스템 콜 인터페이스로 넘겨진 선형주소와 시스템 호출을 수행한 프로세스의 가상메모리 구조체를 가리키는 포인터(current → mm)를 *pgd_offset()*함수에 입력하면 페이지 디렉토리의 시작 주소를 얻을 수 있다. 이 반환된 값(pgd)과 선형주소를 다시 *pmd_offset()* 함수에 입력하면 페이지 중간 디렉토리의 주소를 얻을 수 있다. 마지막으로 *pte_offset()* 함수를 호출하면 원하는 물리주소에 해당하는 페이지의 시작주소를 얻을 수 있다.

4. 실험 결과

4장에서는 내장형 리눅스 커널에서 사용자 메모리와 커널 메모리사이의 복사 오버헤드를 감소시키는 Null copy 기법의 성능 측정 결과를 소개한다. 실험을 위해 상용 내장형 리눅스 개발 보드인 (주)휴인스의 PXA255Pro에 카메라를 장착하여 캡쳐된 이미지를 실시간으로 서버에 전송하는 환경을 구축하였다. 그림 5는 실험 환경을 나타내고 있다.

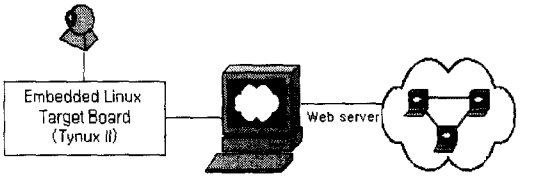

그림 5. 실험 환경

그림 5에서 보듯이 내장형 리눅스 타겟보드에는 실시간 캡쳐가 가능한 PC 카메라가 장착되어 있고 캡쳐된 이미지는 웹서버로 전송되어 인터넷을 통해 클라이언트로 전송되는 구조이다. 타겟보드의 컴퓨팅 파워가 낮기 때문에 카메라가 캡쳐한 이미지를 타겟 보드에서 동작하는 응용 프로그램이 네트워크를 통해 웹서버로 전송한다. 클라이언트는 웹서버에 접속하여 카메라가 보내주는 영상을 실시간으로 볼 수 있다.

본 논문의 관심은 카메라가 주기적으로 보내주는 영상 데이터를 지터(Jitter)를 최소화 하면서 웹 서버로 전송하는 것이다. 따라서, 실험의 파라메터로는 타겟보드의 CPU 오버헤드와 네트워크 전송시 서버 측에서 마감시간 실패 회수로 정할 수 있다. CPU 오버헤드의 경우에는 본 연구에서 작성한 데몬 프로세스에서 시스템 전체의 CPU 부하를 측정하는데 이 사용자 데몬 프로세스는 리눅스 표준 함수인 sysinfo (struct sysinfo *)를 호출하여 현재 시스템의 상태를

모니터링한다. struct info의 경우 unsigned long loads[3]라는 필드가 있는데 각각 1, 5, 15분 주기의 CPU 부하의 평균을 저장하고 있다. 본 논문에서는 5분 주기의 부하 평균값을 측정하고 있다. 표 1은 (주)휴인스의 (주)휴인스의 PXA255Pro[11] 의 사양이고 그림 6은 타겟 보드의 그림이다.

표 1. 타겟 보드의 사양

항목	설명
타겟보드	Intel XScale 128MB Main memory, 32MB Flash 640×480 16bit Color TFT LCD
소프트웨어	Latest Embedded Linux kernel 2.4.x Qt/Embedded 2.3.0 gcc
응용	영상 감시

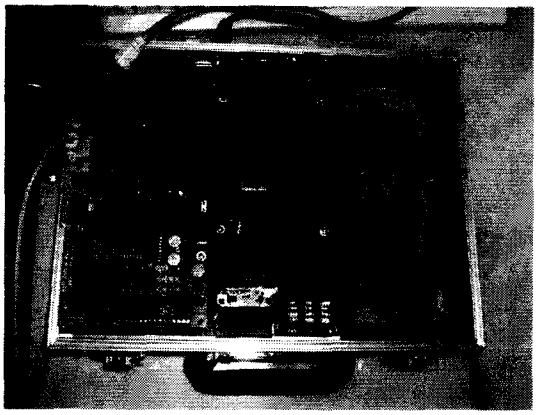

그림 6. 타겟 보드

4.1 CPU overhead 측정

실험은 스트레스 측정을 위해 24 시간 동안의 영상을 웹 서버로 전송할 때 타겟 보드의 CPU 이용율을 주기적으로 측정하여 Null copy의 성능을 측정하는 것이다. 실험을 위해 소켓(socket)을 오픈할 때 옵션 Null copy가 TRUE 일 때는 Null copy 방법으로 네트워크로 이미지를 전송하고, FALSE 일 경우는 기존 방식으로 데이터를 전송한다. 그림 7은 실험의 결과를 보여 주고 있다. normal path 의 경우 91%의 사용도를 보인 반면 Null copy는 72%의 CPU 사용율을 보여 주고 있다. 이는 normal 의 경우 카메라로부터 전송받은 이미지 데이터가 사용자 주소 영역에서 소켓 버퍼가 사용하는 커널 주소 영역으로 복사할 때 CPU가 모든 바이트를 복사하는데 관여하기 때문에 CPU 오버헤드를 초래하기 때문이다. 반면 Null copy의 경우 사용자 주소영역을 바로 커널 주소 공간으로 변환시키고 소켓이 이 영역을 사용하여 데이터를 네트워크 디바이스 버퍼로 보내기 때문에 CPU의 오버헤드가 상당히 감소시키는 효과를 얻을 수 있는 것이다.

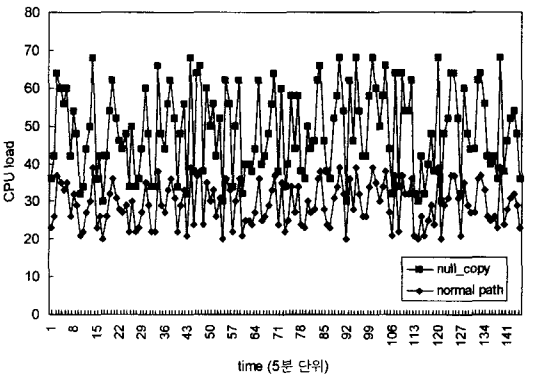

그림 7. 데이터 경로에 따른 CPU 부하

4.2 deadline miss 측정

타겟보드가 이미지 데이터를 네트워크로 전송하여 웹 서버에 도착하는 시간은 실시간성이 보장되어야 한다. 따라서, 전송 지연시간에 때한 변이(Variation), 지터(Jitter)를 측정해야하는 데 본 논문에서는 분석은 간단성을 위해 기준시간, 즉 한 프레임의 데이터 전송시 타겟보드와 서버와의 평균 지연 시간을 정하고 이 시간을 벗어나서 도착하는 패킷은 마감시간을 어긴 것으로 정하고 있다. 그러나, 마감시간을 지키지 못한 것이 서비스의 실패를 의미하지는 않고, 단지 타겟 보드의 실시간성의 정도를 판단하는 기준이 될 수 있다. 그림 8은 두 가지 기법의 마감시간 실패 회수를 나타내고 있다. 스트레스 테스트를 위해 24시간 측정하였는데 normal path 의 경우 360회의 실패를 나타내었고 null_copy의 경우 80회 정도의 마감시간 실패를 보였다.

카메라는 초당 10 프레임의 이미지 데이터를 타겟보드로 전송하고 1 프레임의 평균 크기는 10K바이트이다. 실시간으로 전송되고 캐싱의 효과가 없는 응용이기 때문에 사용자가 할당한 메모리 영역의 데이터

는 지속적으로 서버로 전송되어야 한다. normal path의 경우 Null copy path에 비해 상당한 마감시간 실패 회수를 나타내고 있다. 타겟 보드는 카메라 버퍼에 있는 데이터를 사용자 메모리로 복사해야 하고 이것을 다시 소켓이 사용하는 커널 버퍼로 다시 복사해야 하기 때문에 두 가지의 작업을 시 분할하여 처리하기 때문이다.

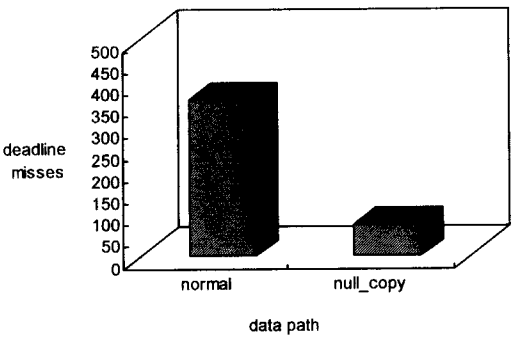

그림 8. 데이터 경로에 따른 마감시간 실패 회수

5. 결 론

본 논문에서는 네트워크 상에서 멀티미디어 데이터를 서비스하는 내장형 리눅스 커널을 위한 메모리 복사 감소 기법인 Null copy 기법을 제안하였다. 제안된 기법은 사용자 메모리영역을 네트워크 전송시 커널 메모리 영역으로 복사하지 않고 직접 커널 소켓 구조체와 연결한다. 따라서, 장시간의 멀티미디어 데이터를 전송하고자 하는 경우 CPU가 사용자와 커널간의 메모리 복사 오버헤드를 제거할 수 있어 응용 프로그램의 응답성과 실시간성을 증가시킬 수 있다. 물론 버퍼 캐시의 캐싱 기능을 이용하고자 하는 경우 Null copy 기법이 지역성을 이용할 수 없는 것은 사실이지만 일반적으로 멀티미디어 데이터는 재사용성이 거의 없다는 점을 고려한다면 응용가능성은 있을 것으로 판단된다.

실험의 결과 Null copy 기법은 CPU의 메모리 복사 오버헤드가 상당히 감소되었고 마감 시간 실패율도 현저히 낮아짐을 확인할 수 있다. 향후 연구로는 단순한 이미지 뿐만 아니라 동영상 데이터 전송을 위한 Null copy를 확장하는 것이다.

참 고 문 헌

[1] "Presentation: The State of Embedded Linux," LinuxDevice.com Homepage, http://www.linuxdevices.com/articles/AT2113794413.html, July, 2000.

[2] 이민석, 모바일 기기를 위한 임베디드 리눅스, pp.112-119, 정보처리학회지, 제9권1호, 2002.

[3] 김정원, 리눅스 커널에서 네트워크 멀티미디어 서비스를 위한 메모리 복사 감소 기법 구현, 한국통신학회논문지, pp.129~137, Vol.28, Feb, 2003.

[4] SGI워크스테이션의 매뉴얼 http://techpubs.sgi.com/library/manpages/open.html

[5] Pasquale, Joseph, Anderson, Eric, and Muller, P. Keith, "Container Shipping: operating system support for i/o intensive applications," IEEE Computer Magazine, 27 (3):84-93, March 1994

[6] N.T. Moti, A.K Yousef, "An Efficient Zero-Copy I/O Framework for UNIX," Tech Report, Sun Microsystems Laboratories. 1995.

[7] M.M.Buddihikot, X,J.Chen, D.Wu, and G.M. Parulkar, "Enhancements to 4.4 BSD unix for efficient networked multimedia in project MARS," IEEE ICMCS, pp.326-337, 1998.

[8] Silberschatz, galvin, Operating systems concepts, fifth edition, pp.304, 1998.

[9] M. Beck, H. Bohme, M. Dziadzka, U. Kunitz, R. Magnus, D. Vervorner, Linux Kernel Internals, pp.148-151, Addison Wesley.

[10] Remy card, eric dumans, and frank mevel, The Linux kernel book, Wiley, pp.286~298, 1999.

[11] 휴인스 홈페이지, www.huins.com.

[12] Prabhat k. andleigh, Kiran thakrar, Multimedia Systems Design, pp.112, Prentice Hall PTR, 1996.

[13] Steen R. Siltis, Thomas M.Ruwart, Matthew T.O'keefe, "The global file system", Proc.of the fifth NASA goodard space center conference on mass storage systems and technologies, sept 17-19, 1996.

[14] Jim Mostek, William earl, and dan Koren, Porting the SGI XFS File System to Linux, white paper, http://oss.sgi.com/projects/xfs/

김 정 원

1995년 부산대학교 전자계산학과 (학사)
1997년 부산대학교 대학원 전자계산학과(석사)
2000년 부산대학교 대학원 전자계산학과(박사)
2000년~2001년 기술신용보증기금 기술평가역(차장)
2002년~현재 신라대학교 컴퓨터정보공학부 교수
관심분야 : 내장형시스템, 멀티미디어, 운영체제