동적 IDE 장치 검사 기법을 이용한 리눅스 커널의 빠른 부팅

Fast Booting of Linux Kernel using Dynamic IDE Device Probing Scheme

김영주
신라대학교 컴퓨터정보공학부

Young-Ju Kim(yjkim@silla.ac.kr)

요약

기존의 임베디드 시스템은 Intel x86 계열의 CPU를 장착한 PC(Personal Computer) 플랫폼에 기반하여 구현되었는데, 이는 PC 환경에서 개발된 많은 솔루션들을 이용할 경우 전체적인 제품 개발 기간을 단축할 수 있을 뿐만 아니라 제품의 신뢰성을 높일 수 있기 때문이다. 또한 PC 플랫폼 기반의 임베디드 시스템은 데이터 저장을 목적으로 하는 경우가 많아 주로 대용량 하드디스크를 데이터 저장장치로 사용한다. 최근 하드디스크의 용량은 빠른 속도로 증가하고 있는 반면에 하드디스크의 구동 준비 시간이 30초 이상으로 길어지는 문제점이 발생하고 있다. 길어진 하드디스크의 구동 준비 시간은 시스템의 부팅 시간에 영향을 미쳐 전체 시스템의 부팅 시간이 길어지고 사용자의 불편함을 가중시키게 된다. 본 논문에서는 대용량 저장장치를 지원하는 PC 플랫폼 기반의 임베디드 시스템에서 리눅스 커널이 빠른 부팅을 수행할 수 있도록 개선된 시스템 구조와 동적 디스크 장치 검사 기법을 제안하고, 성능을 평가하여 제시한다.

■ 중심어 : | PC 플랫폼 | 리눅스 커널 | 빠른 부팅 | 하드디스크 구동 준비 시간 | 동적 IDE 장치 검사 |

Abstract

Most of embedded systems have been developed practically based on the PC platform equipped with Intel x86 CPU since it is able to reduce the total time for product development and improve the reliability of product by making use of a variety of solutions developed for a long time in the PC environment. Also, embedded systems based on PC-platform mainly use a high-capacity hard disk as data storage device for applications intending to store multimedia data. Recently, while the capacity of hard disk is increasing rapidly, the start-up ready time of hard disk is growing longer more than 30 seconds. The lengthened start-up ready time may delay the booting time of embedded system seriously, weighting users down with inconvenience. This paper proposes the refined system architecture and the dynamic IDE device probing method for fast booting of linux kernel in the embedded system based on PC platform with high-capacity hard disks, and the performance of proposed methods is evaluated and presented.

■ keyword : | PC Platform | Linux Kernel | Fast Booting | Start-Up Ready Time | Dynamic IDE Device Probing |

I. 서 론

DVR, 디지털 TV, 휴대용 단말기, 인터넷 셋톱박스 등 고성능 임베디드 시스템을 탑재한 정보가전제품이 늘어나는 추세이다. 현재 이러한 임베디드 시스템은 Intel x86 계열의 CPU를 장착한 PC(Personal Computer) 플랫

접수번호 : #061113-006
접수일자 : 2006년 11월 13일
심사완료일 : 2006년 12월 04일
교신저자 : 김영주, e-mail : yjkim@silla.ac.kr

폼에 기반하여 구현하는 경우가 지배적인데, PC 환경에서 개발된 많은 하드웨어 및 소프트웨어 솔루션들을 이용할 경우 전체적인 제품 개발 기간을 단축할 수 있을 뿐만 아니라 제품의 신뢰성을 높일 수 있기 때문이다[1]. 또한 PC 플랫폼 기반의 임베디드 시스템은 데이터 저장을 목적으로 하는 경우가 많아 주요 저장장치로서 대용량 하드디스크를 적용하는 경우가 대부분이다. 대용량 저장장치를 지원하는 PC 플랫폼 기반의 임베디드 시스템은 다음의 요인들로 인한 동작 지연으로 시스템 부팅 시간이 길어져 사용자의 불편함을 가중시키며, 제품의 경쟁력을 약화시키게 된다.

첫째, 긴 BIOS 동작 시간[2][3]
둘째, 최적화되지 않는 운영체제 커널[4][5]
셋째, 대용량 하드디스크의 길어진 구동 준비 시간 (Start-up Ready Time)[5][6]

본 논문은 대용량 저장장치를 장착하는 PC 플랫폼 기반의 DVR(Digital Video Recorder) 시스템을 대상으로 위의 세 번째 요인으로 유발되는 리눅스 커널의 부팅 지연을 최소화하기 위해 개선된 시스템 구조와 동적 IDE 장치 검사 기법을 제안하고 구현한다. 그리고 제안된 기법으로 개선된 시스템에 대해 리눅스 커널의 부팅 시간을 측정하여 성능을 평가한다.

본 논문은 2장에서 관련 연구를 살펴보고, 3장과 4장에서 본 논문에서 제안하는 기법을 제시하며, 5장에서 실험을 통한 성능 평가를 제시한다. 마지막으로 6장에서 결론으로 마무리 짓는다.

II. 관련 연구

관련 연구에서는 임베디드 시스템에서 리눅스 커널의 부팅 지연 요소와 그에 대한 해결책을 제시한 연구 결과에 대해 살펴본다.

1. BIOS 동작 시간

x86 기반의 시스템은 시스템의 기본적인 초기화 및 부팅을 지원하는 BIOS(Basic Input Output System)가 전원인가와 동시에 동작하는 구조를 갖는다. 기존의 BIOS는 MS-DOS와의 호환성을 유지하기 위해 복잡하고 불필요한 기능을 많이 가지고 있을 뿐만 아니라 리눅스와 같은 범용 운영체제가 이러한 기능들은 동일하게 수행하도록 구성됨으로써 중복된 기능 수행으로 인한 부팅 시간의 증가를 초래한다[2]. BIOS에 의한 부팅 지연에 대한 해결책으로는 긴 부팅 시간을 획기적으로 단축할 수 있도록 최적화된 BIOS을 개발하여 적용하는 것으로 라이센스 비용이 필요 없는 공개 소프트웨어인 LinuxBIOS[3] 등이 대안이 될 수 있으나, 전통적인 BIOS 기능을 제공하지 않아 LILO와 같은 부트로더를 사용할 수 없을 뿐만 아니라 지원하는 chipset의 종류에 제약이 있어 추가적인 개발 시간을 요구한다는 문제점을 안고 있다.

2. 리눅스 커널 최적화

현재의 리눅스 커널은 PC에서 지원되는 다양한 하드웨어 장치를 수용하기 위해 폭넓게 확장되어 있을 뿐만 아니라 새로운 H/W 추가 여부를 자동으로 체크하고 추가된 H/W가 부팅과 동시에 사용자가 원활하게 활용할 수 있도록 모든 H/W를 안정화시킨 이후에 시스템을 구동시킨다. 그러나 이러한 부팅 메커니즘은 긴 부팅 시간을 요구하는 단점을 지닌다[4][5]. 범용 컴퓨터는 임의적으로 새로운 H/W를 추가하거나 다양한 응용프로그램을 수행하는 반면에 임베디드 시스템은 정해진 응용프로그램을 수행하며, 새로운 H/W 추가와 같은 예외적인 상황이 거의 발생하지 않는 특성을 가지고 있다. 따라서 임베디드 시스템의 경우 커널 설정 최적화 및 커널 소스 최적화를 통해 부팅 시간을 단축할 수 있다[4]. 그러나 커널 소스 최적화 작업은 타켓 시스템의 구조에 맞게 커널의 부팅 과정과 관련된 모든 부분을 수정하여야 하는 부담이 크다.

3. XIP(eXecute-In-Place) 기법

임베디드 시스템에서 커널의 빠른 로딩 및 부팅을 구현하기 위해 XIP(eXecute-In-Place) 기법에 관한 연구가 다수 진행되었다[7]. XIP 기법은 실행 코드를 RAM으

로 복사하지 않고 플래시 메모리 영역에서 바로 실행하는 기법으로서, 플래시 메모리를 보조 저장 장치로 채택하고 있는 임베디드 시스템 환경에서 구현이 가능하여 많은 관심을 받고 있다. 그러나 [표 1]에서 살펴보듯이 플래시 메모리는 충분히 빠른 읽기 접근 시간을 제공하고 있지만 SDRAM에 비해 10배 정도 느린 접근 속도를 가지고 있다. 이는 실행 빈도가 낮은 소프트웨어에 적용할 경우에 시스템 전체 성능에 많은 영향을 미치지는 않지만, 운영체제 모듈과 같이 실행 빈도가 높은 코드를 플래시 메모리에 위치시켜 실행하는 운영체제 XIP 기법은 시스템 전체 성능을 저하시키는 문제점을 발생시킬 수 있다. 그러므로 최근에는 XIP 기법을 통해 부팅 시간 감소보다는 주기억장치(RAM) 사용량을 줄이기 위한 기법으로 연구되고 있다[8].

표 1. 플래시 메모리 vs. DRAM의 읽기접근시간 비교

특징	플래시 (Intel 28F640J3A)	SDRAM	DRAM		
			FPM	EDO	BEDO
읽기 접근 시간	100~150ns	10~15ns	25~35ns	20~30ns	15~20ns

출처 : Intel, EDN

4. 하드디스크의 구동 준비 시간 (Start-up Ready Time)

최근 하드디스크의 용량은 빠른 속도로 증가하고 있는 반면에 하드디스크의 기계적인 구동 메카니즘으로 인하여 저장된 데이터에 대한 안정된 접근을 위한 구동 준비 시간(Start-up Ready Time)이 30초 이상으로 길어지는 문제점이 초래되고 있다[6]. 길어진 하드디스크의 구동 준비 시간은 하드디스크를 장착한 PC 플랫폼 기반의 임베디드 시스템에서 커널 부팅을 크게 지연시키게 된다. 그리고 하드디스크 장치를 부팅 장치로 사용하는 경우, 하드디스크의 초기화 시간은 피할 수 없는 제약 조건이 된다.

특히, 대용량 비디오 데이터를 저장하기 위해 대용량 하드디스크를 장착하는 DVR(Digital Video Recorder) 시스템의 경우 하드디스크 장치 검사(Device Probing)와 파일시스템 마운팅(Mounting)에 소요되는 시간이 부팅 시간에 미치는 영향이 매우 크다. 그러나 최근까지 이러한 문제를 직접 다룬 연구가 없는 실정이다.

본 논문에서는 리눅스 운영체제와 대용량 저장 장치를 채택하고 있는 DVR 시스템에서 하드디스크 구동 준비 시간에 의해 지연되는 부팅 시간을 최적화를 위하여 개선된 시스템 구조와 동적 IDE 장치 검사 기법을 제안하고 구현한다.

III. 제안된 시스템 구조

x86 기반의 PC 플랫폼에서 적용하는 BIOS 기반의 부팅 구조에서는 디스크 장치를 부팅 장치로 사용하여야 하며, 그에 따른 부팅 지연은 BIOS 동작 방식을 변경하지 않는 한 피할 수 없는 요소이다[2][3]. 이에 본 논문은 x86 기반 시스템에서 하드디스크를 부팅 장치로 사용할 때에 발생하는 부팅 지연을 최소화하기 위해 [그림 1]과 같은 시스템 구조를 제안하고 적용하였다.

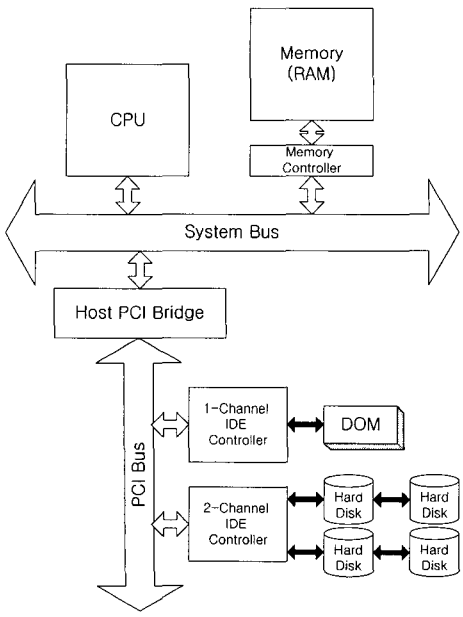


그림 1. 제안된 시스템 구조

[그림 1]에서는 기본적인 PC 플랫폼에서 추가적으로 변경된 부분만을 나타내었으며, 크게 부팅 장치와 대용

량 저장장치에 대한 인터페이스 장치(IDE H/W Controller)의 분리 그리고 부팅 장치로서 DOM(Disk On Module) 플래시 메모리 드라이브를 적용한 점을 특징으로 제시할 수 있다.

2개의 디스크 인터페이스 장치를 지원함으로써 더 많은 하드디스크를 장착할 수 있을 뿐만 아니라 부팅 장치에 대한 인터페이스 지연 시간을 단축하고 대용량 하드디스크에 대해 부팅 시간 장치 검사가 아니라 원하는 시점에 원하는 장치만을 검사할 수 있는 동적 IDE 장치 검사(dynamic IDE device probing)를 가능하게 한다. 또한, DOM 플래시 메모리 드라이브를 부팅 장치로 적용한 것은 하드디스크와 달리 구동 준비 시간이 필요 없어 부팅 지연을 줄일 수 있을 뿐만 아니라 부팅 장치로서만 동작하기 때문에 적은 비용으로 적은 저장 공간이 요구되는 필요성에 부합되고, 작은 크기와 높은 신뢰성이 임베디드 시스템의 요구 조건에 적합하기 때문이다.

IV. 동적 IDE 장치 검사 기법

1. 하드디스크의 구동 준비 시간에 의한 부팅 지연

본 논문의 연구 대상이 되는 DVR 시스템에서는 대용량의 비디오 데이터를 저장하기 위해 다수의 대용량 하드디스크를 사용하는 것이 필수적이다. 반면, 최근 하드디스크가 대용량화되면서 구동 준비 시간이 길어지고 이로 인해 점차 초기화 지연 시간이 증가하고 있는데, 이러한 하드디스크의 초기화 지연은 DVR 시스템에 탑재되는 리눅스 커널의 부팅 과정에 영향을 미친다.

[그림 1]에서 제안된 시스템 구조에서는 부팅 장치로 DOM 플래시 메모리 드라이브를 사용함으로써 부트로더 및 리눅스 커널의 로딩과 실행에서 불필요한 지연이 발생하지 않으나, 리눅스 커널 부팅 과정에서 하드디스크 초기화가 완료되기 전에 IDE 장치 검사를 수행함으로써 장치 인식 오류가 발생하고 하드디스크 장치가 장착되지 않는 것으로 인식하여 시스템이 정상적으로 동작하지 못하게 된다.

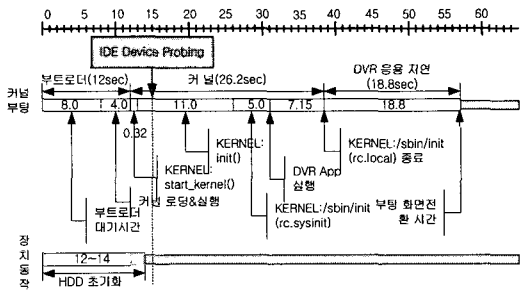


그림 2. 커널 부팅 지연 및 IDE 장치 검사 시점

현재 이러한 문제에 대한 해결책으로는 [그림 2]와 같이 부트로더에서 일정 시간(약 8초 정도)을 무의미하게 대기한 후에 부팅을 시작하는 방법이 적용되고 있으며, 이는 전체 부팅 지연에 큰 영향을 미칠 뿐만 아니라, 대용량 IDE 하드디스크의 유형 및 용량 등 시스템 환경 조건이 바뀔 때마다 지연 시간을 조정하여야 하는 문제가 발생한다.

본 논문에서는 제안된 시스템 구조에서 무의미하게 지연되는 리눅스 커널의 부팅 시간을 단축하기 위해 동적 IDE 장치 검사 기법을 제안하고 구현하였다. 리눅스 커널의 하드웨어 초기화 과정에서 수행하는 IDE 장치 검사 기능을 원하는 시점, 즉 커널 부팅 종반부나 커널 부팅 후의 응용프로그램 수행 시점 등에서 원하는 IDE 장치만을 동적으로 검사할 수 있는 기능을 지원함으로써 하드디스크의 초기화에 병행하여 커널 부팅을 정상적으로 지속할 수 있도록 IDE 장치 검사를 적절하게 지연시켜 부트로더의 지연 시간을 없애고, 전체 커널 부팅 지연 시간을 줄이도록 하였다.

2. 동적 IDE 장치 검사 기법

2.1 리눅스 커널의 IDE 장치 검사 과정 분석

리눅스 커널은 IDE 장치 드라이버를 로딩할 때에 [그림 3]과 같은 과정으로 IDE 장치 검사를 수행하는데, 리눅스 커널은 검출된 PCI 장치 목록을 검색하여 PCI IDE controller 장치를 찾고 IDE H/W interface 정보를 수집하여 관련 데이터 구조를 초기화한 다음, IDE H/W interface 별로 IDE drive를 검사한다.

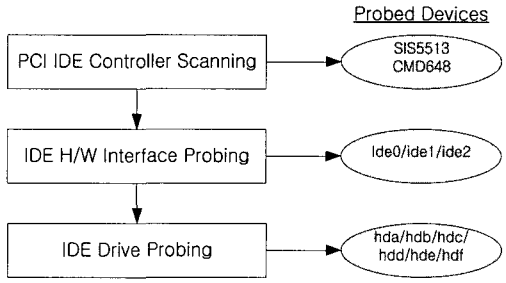


그림 3. 리눅스 커널에서의 IDE 장치 검사 과정

현재 리눅스 커널에서는 IDE 장치 검사에 대해 IDE H/W Interface 및 IDE drive 별로 장치 검사 여부를 지정할 할 수 있는 커널 옵션 설정이 가능하며, IDE 장치 드라이버의 프로그램 구조가 IDE H/W interface 중심으로 구성되어 있다. 따라서 본 논문은 장치 드라이버 프로그램의 확장성과 제안된 시스템 구조에서의 IDE H/W interface 분리 등을 감안하여 IDE H/W interface 별로 동적 IDE 장치 검사를 수행할 수 있도록 제안된 기법을 설계함을 전제하였다.

2.2 동적 IDE 장치 검사의 동작 모델

우선, 동적 IDE 장치 검사 기법에 대한 요구사항을 분석하면 다음과 같다.

첫째, 원하는 IDE 장치를 원하는 시점에 검사할 수 있어야 한다.

둘째, 커널 실행과 비동기적으로 수행되어야 한다.

셋째, 파일시스템 마운트 프로그램과 동기화 기능이 제공되어야 한다.

본 논문에서는 첫 번째 요구사항을 지원하기 위하여 [그림 4]과 같은 동적 IDE 장치 검사 기법의 동작 모델을 설정하였다. 동작 모델은 IDE 장치 검사를 두 단계, 즉 top-half probing 및 bottom-half probing으로 나누어 장치 검사를 수행하도록 하며, top-half probing은 커널 부팅의 하드웨어 초기화 과정에서 수행하고 부팅과 관련된 IDE 장치, 즉 루트 파일시스템 저장 장치 등을 검사한다. Bottom-half probing은 동적으로 원하는 시점에 수행하도록 하며, 주로 부팅과 관계없는 데이터 저장 목적의 IDE 장치를 검사한다.

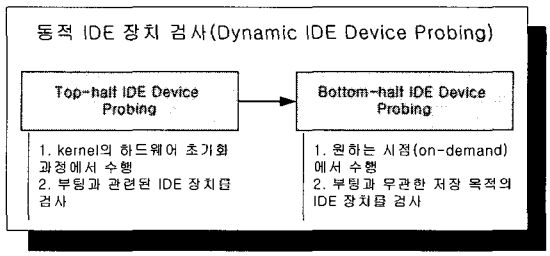


그림 4. 동적 IDE 장치 검사의 동작 모델

2.3 동적 IDE 장치 검사의 구현

본 논문은 [그림 4]의 동작 모델에 근간하여 위에서 분석된 요구사항을 만족하도록 동적 IDE 장치 검사 기법을 구현하고 리눅스 커널에 추가하였다. 동적 IDE 장치 검사 기법은 이미 구현되어 있는 IDE 장치 드라이버의 기능을 거의 동일하게 사용하기 때문에 별도의 모듈로 구현하기 보다는 IDE 장치 드라이버를 확장하는 개념으로 구현하였으며[9][10], 구현 방법의 특징을 요약하면 다음과 같다.

첫째, 하드웨어 장치 검사는 커널 문맥을 이용하여 수행되어야 하며[9], 커널 동작과 비동기적으로 실행되어야 하므로 별도의 커널 쓰레드를 이용하여 동적 IDE 장치 검사 기능을 구현하였다. Bottom-half probing 기능이 호출되면 커널의 scheduler queue에 bottom-half probing task를 생성하여 추가하고, keventd 커널 쓰레드에 의해 비동기적으로 실행된다.

둘째, 커널 부팅 과정에서는 커널 함수 호출을 통해 동적 IDE 장치 검사를 실행할 수 있지만 응용프로그램 실행 시점에서는 커널의 장치 검사 기능을 요청하여 실행하여야 함으로 별도의 시스템 콜을 추가하였다. 추가된 시스템 콜이 호출되면 앞에서 언급한 것과 같이 별도의 커널 쓰레드가 생성되어 IDE 장치 검사를 수행하고, 시스템 콜을 호출한 사용자 프로세스와는 비동기적으로 IDE 장치 검사가 이루어진다.

셋째, IDE 장치 검사를 커널 부팅 과정의 하드웨어 초기화 시점보다 지연시켜 수행하는 경우에 PCI 버스의 IRQ 공유로 인해 IDE 장치가 사용하는 IRQ를 자동으로 검출하지 못하며, 이로 인해 정상적인 IDE 장치 검사를 수행하지 못한다. 이러한 IRQ 미검출 문제는 IDE 장치

드라이버가 장치의 호환성 등을 고려하여 ISA 버스 방식의 자동 IRQ 검사 기능을 사용함으로써 PCI 버스의 IRQ 공유 기능을 고려하지 못하여 발생한다. 본 논문은 IDE 장치 드라이버의 개발 취지를 살리기 위해 자동 IRQ 검사 기능에 다음과 같은 IRQ 재초기화 및 복구(IRQ reinitialization & recovery) 기법을 적용함으로써 IRQ 미검출 문제를 해결하였다.

① 장치 검사 과정의 IRQ 검사 시점에 IDE I/F가 사용하는 IRQ에 대해 IRQ 할당 여부 및 IRQ 핸들러 정보를 보관한 후, IRQ 할당 정보를 재초기화한다.
② IDE I/F에 대해 IRQ 검사를 수행하여 IRQ를 검출하여 할당한다.
③ 이전에 저장한 정보를 이용하여 검출된 IRQ에 대한 할당 정보를 복구하면서 업데이트한다.

넷째, 동적 IDE 장치 검사가 별도의 커널 쓰레드에 의해 실행되는 동안에 비동기적으로 다른 쓰레드가 파일시스템 마운트를 시도하는 경우 IDE 장치 검사가 완료되지 않아 정상적으로 마운트할 수 없는 경우가 발생한다. 본 논문은 동적 IDE 장치 검사를 실행한 후에 파일시스템을 마운트할 때에는 동적 IDE 장치 검사가 완료되었는지를 검사하고 일정 시간 지연하도록 함으로써 파일시스템을 정상적으로 마운트할 수 있도록 하였다. 또한, proc 파일시스템에 동적 IDE 장치 검사의 완료 여부를 알려주는 읽기 전용(read-only) entry를 생성하여 접근함으로써 마운트 프로그램과 동기화를 수행하도록 하였으며, 이에 대해서는 다음 절에 자세히 제시한다.

3. 지연된 파일시스템 마운팅 기법

동적 IDE 장치 검사가 별도의 커널 쓰레드에 의해 실행되는 동안에 비동기적으로 사용자 쓰레드가 파일시스템 마운트를 시도하는 경우 IDE 장치 검사가 완료되지 않아 정상적으로 파일시스템을 마운트할 수 없는 경우가 발생한다. 이에 본 논문은 동적 IDE 장치 검사 기능과 동기화하여 IDE 장치 검사가 완료되었는지를 먼저 검사하고 그렇지 않으면 일정 시간 동안 지연하도록 하여 파일시스템을 정상적으로 마운팅하는 지연된 파일시스템 마운팅(delayed file-system mounting) 프로그램을 구현하여 적용하였으며, 다음과 같은 특징을 갖는다.

첫째, 기존의 파일시스템 마운트 프로그램과의 호환성을 지원하여 모든 기능을 동일하게 사용할 수 있다.

둘째, 동적 IDE 장치 검사 기능과 실행 동기화를 수행하여 동적 IDE 장치 검사의 완료 여부를 확인하고 마운팅 실행 시점을 결정한다. 이러한 동기화 동작은 [그림 5]와 같이 proc 파일시스템의 읽기 전용 entry('/proc/ide/bhprobe')을 이용하여 동적 IDE 장치 검사 쓰레드와 파일시스템 마운팅 쓰레드 사이에 동기화를 수행한다.

셋째, 동적 IDE 장치 검사가 오동작하거나 일부 장치에 대해 검사를 실패하였을 경우 동기화에 실패하여 무한 대기 상태에 빠질 수 있는데, 이를 해결하기 위해 일정 시간 동안만을 지연하였다가 파일시스템 마운팅을 시도하도록 마운트 동작 지연 기능을 지원한다.

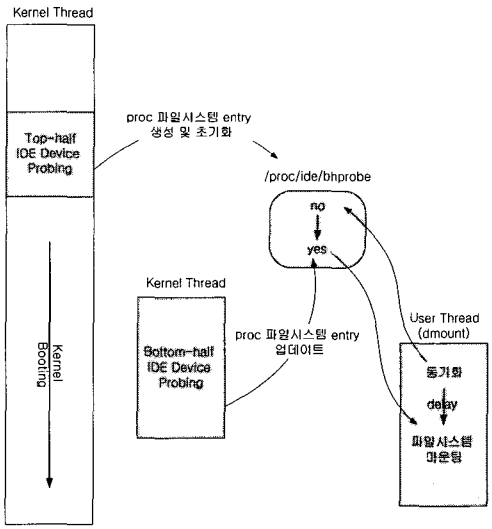


그림 5. 동적 IDE 장치 검사 기능과 파일시스템 마운팅 기능의 동기화 메커니즘

V. 성능 평가

본 논문은 개선된 시스템 구조에서 제안된 동적 IDE 장치 검사 기법의 성능을 평가하고 결과를 제시하였다. 성능 평가는 개선된 시스템 구조에서 원래 버전의 리눅스 커널과 제안된 기법을 사용한 개선된 리눅스 커널을

각각 적용하고 부팅 시간을 직접적으로 측정하여 비교하였다. 성능 평가 실험에 적용된 DVR 시스템의 하드웨어 환경은 [표 2]와 같다. 실험에 사용된 시스템의 사양이 비교적 낮은데, 이는 비디오 및 오디오 처리는 별도의 DSP 프로세서가 담당하고, CPU 장치는 입출력 및 전체적인 동작을 제어하는 프로그램만을 실행하기 때문이다.

실험을 통해 원래의 리눅스 커널과 제안된 기법이 적용된 개선된 커널 간에 커널 부팅 단계별 경과 시간을 측정하여 직접 비교한 결과, [표 3]과 같으며, 동적 IDE 장치 검사 기능을 적용한 개선된 커널이 원래의 커널보다 부팅 시간을 전체적으로 9~10초 정도 단축하였음을 알 수 있다. 이는 부트로더에서의 초기 대기 시간을 제거하였을 뿐만 아니라 커널 부팅 단계에서 동적 IDE 장치 검사 기능과 파일시스템 마운팅 기능의 적절한 동기화를 통해 저장장치에 대한 인식 시간을 단축하였기 때문이다.

표 2. 성능 평가를 위한 하드웨어 환경

시스템 항목	규 격
CPU	200MHz x86 호환 CPU
Memory	256 MB RAM
IDE 장치	DOM, CD-ROM, 400GB HDD 2A
SCSI 장치	장치 없음
출력 장치	VGA 모니터
리눅스 커널 버전	linux-kernel-2.4.26.ksr

표 3. 부팅 단계별 부팅 시간 비교(단위:sec)

		original kernel		refined kernel		비 고
		단계별 시간	누적 시간	단계별 시간	누적 시간	
부트로더 지연	wait delay	8.0	8.0	0	0	
	kernel loading & exec	4.0	12.0	4.0	4.0	부팅화면으로 전환
커널 지연	start_kernel()	0.32	12.32	0.32	4.32	
	init()	13.7	26.00	11.3	15.6	부팅화면에서 스크롤 시작
	/sbin/init (rc.sysinit - IDE device probing & mounting)	5.05	31.05	7.2	22.8	
	/sbin/init (rc.local)	7.1	38.15	7.1	29.9	커널부팅 종료
응용프로그램 지연	부팅화면 전환	19.4	57.5	18.0	48	응용화면으로 전환

VI. 결 론

본 논문에서는 대용량 저장장치를 지원하는 PC 플랫폼 기반의 임베디드 시스템에서 길어진 하드디스크의 구동 준비 시간으로 야기된 리눅스 커널의 부팅 지연을 최소화하여 보다 빠른 부팅을 수행할 수 있도록 개선된 시스템 구조와 동적 IDE 장치 검사 기법을 제안하고 구현하였다. 그리고 개선된 시스템에 대해 리눅스 커널의 부팅 시간을 측정하여 성능을 평가하였으며, 기존 시스템에서 무의미하게 대기하던 시간을 제거하고 저장장치에 대한 인식 시간을 단축하여 빠른 부팅을 수행함을 확인하였다.

향후에는 PC 플랫폼 기반의 시스템에서 BIOS 실행으로 야기되는 부팅 지연을 효과적으로 제거하기 위해 개선된 BIOS을 연구, 개발하여 적용할 예정이다.

참 고 문 헌

[1] http://www.ednkorea.com/article.asp?id=2785

[2] 윤희철, 마진석, 김선자, "LinuxBios를 이용한 X86 기반 임베디드 시스템의 빠른 부팅 기법", 한국정보과학회 2003년 춘계학술발표대회 논문집, pp.160-162, 2003.

[3] http://www.linuxbios.org

[4] T. R. Bird, "Methods to Improve Bootup Time in Linux," Linux Symposium, 2004.

[5] http://tree.celinuxforum.org/CelfPubWiki/BootupTimeReductionHowto

[6] http://www.storagereview.com/guide/guide_index.html

[7] C. I. Park, J. Y. Seo, S. H. Bae, H. J. Kim, S. H. Kim, and B. S. Kim, "A low-cost memory architecture with NAND XIP for mobile embedded systems," Proc. of the 1st IEEE/ACM/IFIP international conference on Hardware/Software codesign and system, pp. 138-143, 2003.

[8] J. S. Lee, J. Y. Park, and S. S. Hong, "Memory

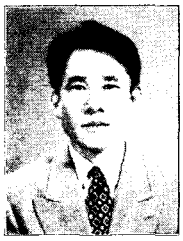
Footprint Reduction with Quasi-Static Shared Libraries in MMU-less Embedded Systems," IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), pp.24-33, 2006.

[9] D. P. Bovet and M. Cesati, Understanding The Linux Kernel, O'Reilly, 2003.

[10] J. Corbet, A. Rubini, and C. K. Hartman, Linux Device Drivers, 3rd-Ed., O'Reilly, 2001.

저 자 소 개

김 영 주(Young-Ju Kim) 정회원



- 1990년 2월 : 부산대학교 계산통계학과 (이학석사)
- 1990년 1월 ~ 1995년 8월 : 큐닉스컴퓨터 응용시스템연구소 선임연구원
- 1999년 8월 : 부산대학교 전자계산학과 (이학박사)
- 2000년 3월 ~ 현재 : 신라대학교 컴퓨터정보공학부 교수

<관심분야> : 멀티미디어통신, 임베디드시스템, 운영체제