임베디드 시스템에서 리눅스의 빠른 부팅

신광무*O 박성호** 정기동*
부산대학교 컴퓨터공학과*, 부산대학교 정보전산원**
{omega*O, kdchung* }@melon.cs.pusan.ac.kr shpark**@pusan.ac.kr

Fast Booting Implementation of the Linux in the Embedded System

Kwangmu Shin*O , Seongho Park** , Kidong Chung*
Dept. of Computer Engineering, Pusan National University*
Information Technology Center, Pusan National University**

요 약

오늘날 생활환경에서 디지털 TV, 휴대용 단말기, 인터넷 셋톱박스 등 임베디드 시스템이 탑재된 정보가전제품을 쉽게 찾아 볼 수 있다. 정보가전제품은 가전제품의 본래 기능뿐만 아니라 다른 정보가전제품과 상호 작용을 통한 인텔리전트한 기능의 수행이 요구된다. 이와 같은 인텔리전트한 기능을 수행하기 위해서는 단순한 기능만을 수행하는 펌웨어 수준의 임베디드 시스템이 아니라 다양한 기능을 수행하는 스마트 임베디드 시스템이 요구된다. 스마트 임베디드 시스템은 인텔리전트한 기능을 제공하기 위해서 네트워킹, 멀티프로세싱 등의 기능이 제공되는 범용 운영체제 수준의 성능을 가진 운영체제의 탑재가 요구된다. 그러나 이러한 범용 운영체제는 수십 초의 긴 부팅 시간을 요구함으로 이전의 파워온 (Power-On)과 동시에 사용할 수 있는 전통적인 가전제품이나 산업기계의 사용자에게는 매우 큰 불편을 초래할 수 있다. 특히, 복잡한 공정을 수행하는 공장 산업기계의 임베디드 시스템은 shutdown 후 정상가동 까지 걸리는 시간이 제품 생산량 및 품질에 큰 영향을 미친다. 이와 같이 다양한 분야에서 적용된 스마트 임베디드 시스템의 부팅시간은 스마트 임베디드 시스템의 성능을 평가하는 중요한 요소가 된다. 본 논문은 임베디드 환경 하에 범용 운영체제인 리눅스를 활용하여 빠른 부팅을 구현하였다. 부팅 단계에서 영향을 미치는 부트로더, 커널 그리고 루트 파일시스템의 각 구성요소를 최적화하는 연구를 수행하였으며, 그 결과 HBE-EMPOS II 기준으로 부팅시간이 11초로 감소되는 성과를 얻었다.

1. 서 론

임베디드 시스템이란 미리 정해진 특정 기능을 수행하기 위해 컴퓨터의 하드웨어와 소프트웨어가 조합된 전자 제어 시스템을 말한다. 임베디드 시스템 중에서도 아주 간단한 제어 기능을 수행하는 8비트 MCU 와 같은 프로세서로 개발되는 임베디드 시스템에는 운영체제를 사용하지 않는다. 간단한 기능을 제공하기 위해 운영체제를 사용한다면 운영체제 자체 비용 외에 운영체제 저장 공간, 운영체제 사용 메모리 등 하드웨어의 추가적인 비용이 커진다. 그러나 근래 들어 휴대 전화, 인터넷 셋톱박스, 네트워크 장비처럼 복잡한 기능을 수행하는 기기들이 늘어나고 자동차나 가전제품 등에도 사용자들의 요구가 많아지고 복잡해지고 있다. 이러한 임베디드 시스템에서는 간단한 제어 기능만을 수행하는 펌웨어 수준의 임베디드 시스템이 아니라 보다 복잡하고 다양한 기능을 수행하는 스마트 임베디드 시스템이 요구된다. 스마트 임베디드 시스템은 인텔리전트한 기능을 제공하기 위해서 네트워킹, 멀티프로세싱 등의 기능이 제공되는 범용 운영체제 수준의 성능을 가진 운영체제의 탑재가 요구된다.

현재 사용되는 다양한 범용 운영체제 중에서도 오픈 소스, 오픈 아키텍처, 안정성 등의 이유로 임베디드 운영체제로서 사용이 증가하고 있다. 본 논문에서는 임베디드 리눅스 환경 하에서 빠른 부팅을 구현하기 위해 부팅 단계에서 영향을 미치는 부트로더, 커널, 루트 파일시스템의 각 구성요소를 최적화하였다. 이 최적화 메커니즘을 통해서 부팅 단계에서 소요 시간을 최소화하도록 설계, 구현하였다.

"이 논문은 교육인적자원부 지방연구중심대학육성사업 (차세대물류IT기술연구사업단) 의 지원에 의하여 연구되었음."

본 논문에서의 구성은 2장에서 관련 연구, 3장에서 빠른 부팅 설계, 4장에서 실험 결과, 그리고 마지막으로 5장에서 결론 및 향후 과제를 기술한다.

2. 관련 연구

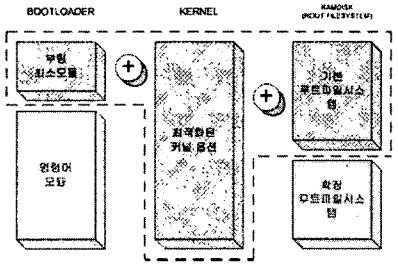
기존에는 리눅스 시스템에서 빠른 부팅을 구현하기 위해 XIP(eXecute-In-Place), Calibration 지연 감소, IDE probing 제거, "quiet" 옵션 사용 등의 방법을 이용하였다[2,3,7]. 그러나 이들 연구에서 제시하고 있는 방법들은 하드디스크를 사용하지 않는 임베디드 시스템에 적용할 수 없거나 현실적인 부팅 시간 감소 효과가 미약하였다. 코드를 램으로 복사하지 않고 플래시 영역에서 바로 실행하는 방법인 XIP (eXecute-In-Place), "loops_per_jiffy" 값을 하드코드하여 Calibration 지연을 감소시키는 방법, 커널 명령어 라인에 "quiet" 옵션을 주어 시리얼 라인을 통해 타겟 보드에 출력되는 부팅 메시지를 제거하는 방법은 부팅 시간 감소 효과가 미약하였다. 특히 XIP 경우에 플래시 메모리를 지우거나 쓰는 순간에는 코드가 실행될 수 없는 문제점이 있다[1]. 최근에는 XIP 를 부팅 시간 감소의 방법으로 제시하기 보다는 주 메모리 사용량을 줄이기 위해서 활용하고 있다[4,5,6]. 코드가 플래시 영역에서 램 복사 없이 직접 수행된다면 램의 사용량이 그만큼 확보되기 때문이다[6].

이 외에 본 논문은 커널 최적화 과정을 수행하기 위해 HBE-EMPOS II 환경의 커널 2.4.19 기준으로 커널 configuration 의 옵션을 파악하였다[8]. 이 방법은 커널 사이즈를 줄임으로써 부팅 시간을 줄이는 기존에 자주 활용된 방법이다[2].

3. 빠른 부팅 설계

3.1 최적화 메커니즘

최적화 메커니즘의 핵심은 부팅 단계에서 램으로 복사되는 모듈을 최소화 하는 것이다. 즉, 부트로더의 부팅 최소모듈, 커널 configuration 최적화로 만들어진 커널 이미지, 기본 루트 파일시스템이 램으로 복사된다. 그 결과 램 복사 및 압축해제 소요시간이 줄어든다. 부팅시간을 줄이기 위해서 이 방법 외에 부트로더 단계에서 불필요한 초기화 과정 제거, 마찬가지로 루트 파일시스템 영역의 부트 스크립트에서 불필요한 초기화 과정 제거하여 부팅시간을 단축시켰다.



<그림 1> 최적화 메커니즘

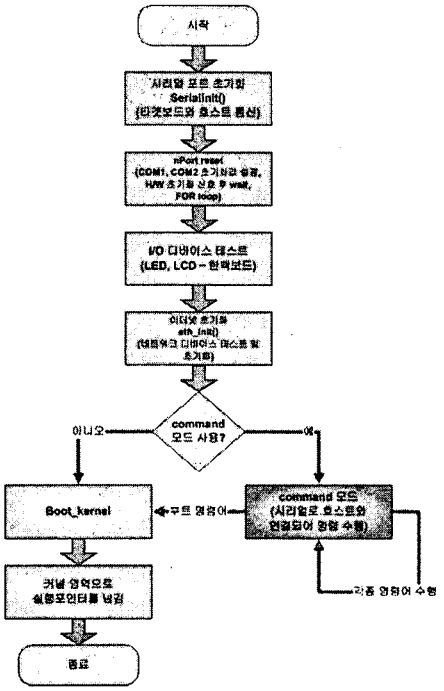
HBE-EMPOS II 기준 부팅 순서는 <표 1>과 같다

<표 1> HBE-EMPOS II 부팅 순서

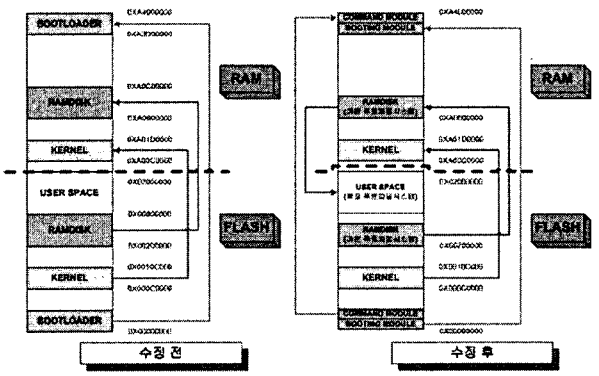
순서	수행 과정
①	시스템 이상 여부 테스트
②	커널 및 램디스크 램으로 복사
③	커널 압축 해제 및 재배치
④	장착된 하드웨어 검사 및 장치 드라이버 설정
⑤	파일 시스템 검사 및 마운트
⑥	파일 시스템 마운트
⑦	/etc/inittab 에서 init 실행을 위한 설정 내용 확인
⑧	/sbin/init 실행 (PID가 1 이 됨)
⑨	/etc/rc.d/rc.sysinit 실행 (호스트 이름, 시스템 점검, 모듈 로딩)
⑩	/etc/rc.d/rc 실행 (runlevel 에 따른 스크립트 실행)
⑪	/etc/rc.d/rc.local (매번 실행할 내용 입력)
⑫	로그인

3.2. 부트로더 (Bootloader)

<그림 2>에서 보는 바와 같이 각종 디바이스 초기화 및 테스트를 거친 후 오토 부팅 여부를 결정하고 그 선택에 따라 커널 영역으로 실행 포인터를 넘기거나 명령어 모드를 실행한다. 명령어 모드는 부트로더, 커널, 루트 파일시스템 등을 바꾸는 경우처럼 특별한 경우가 아니면 특별히 수행을 요구하지 않는다. 본 논문은 명령어 모드에서 수행하는 부분을 하나의 독립된 모듈로 재구성한 한 후 명령어 모드를 수행하지 않는 일반적인 부팅일 경우에는 명령어 모드 모듈을 제외한 부팅 최소 모듈만을 램으로 복사하여 실행하도록 한다. 물론 명령어 모드 사용을 선택하게 되면 명령어 모드 모듈이 설정된 램의 특정 주소로 복사한 후 수행된다. <그림 3>에서 부트로더 구조 변화 및 램 복사 과정을 보여준다.



<그림 2> HBE-EMPOS II 부트로더



<그림 3> 메모리 맵 구조 변화

3.3. 커널 (Kernel)

커널 영역에서 부팅시간을 줄이는 방법은 커널 사이즈를 최소화 하는 것이다. 커널 사이즈를 줄임으로써 램 복사 시간 및 압축 해제 시간이 단축된다. 그리고 불필요하게 선택된 모듈이 로딩되는 시간까지 절약될 수 있다. 그래서 커널 다운 사이징의 한 방법으로 채택한 것이 커널 cofiguration 최적화이다. 네크워크, MTD, 파일시스템 부분을 제외하고 부팅 단계에서 필요한 최소한의 옵션만을 남긴다. 물론 각 옵션 간의 의존성을 해치지 않는 범위 내에서 configuration 최적화가 이루어졌다. <표 2>는 커널 configuration 최적화 과정에서 옵션 선택 최적화의 예이다. <표 2>는 HBE-EMPOS II 보드에서 제공하는 기본 커널 configuration 에서 필요하지 않은 옵션을 제거하는 방식을 취했다.

<표 2> 커널 configuration 최적화 예

상위 메뉴	하위 메뉴	최하위 메뉴	on / off
Kernel Hacking	Verbose user fault messages		off
	Kernel debugging		
	Include debugging information in kernel binary		
Sound	Sound support		
File Systems	DOS FAT fs support		
	Journalling Flash File System v2 (JFFS2)		
	/dev/pts file system for Unix98 PTYs		
	Network File Systems -> NFS file system support		
Input core support	Input core support		
ATA/ATAPI/MFM/RLL support	ATA/ATAPI/MFM/RLL support		
Plug and Play configuration	Plug and Play support		
General setup	Timer and CPU usage LEDs		
	Power Management support (experimental)		
	MMC devices drivers	Multi Media Card support	
	PCMCIA/CardBus support	PCMCIA/CardBus support	
	Support for hot-pluggable devices		
Code maturity level options	Prompt for development and/or incomplete code/drivers		

3.4. 루트 파일시스템 (Root Filesystem)

<그림 3>에서 최적화 전에는 루트 파일시스템이 압축된 램디스크에 포함되어서 부팅 후에 압축 해제 및 램으로 복사되는 과정을 거친다. 본 논문에서는 루트 파일시스템을 이분화 하는 방법을 제안한다. 즉, 부팅 과정에서 필수적인 기본 루트파일시스템과 나머지 데이터를 담은 확장 루트파일시스템으로 나눈다. 기본 루트파일시스템은 압축된 램디스크에 포함시켜 플래시 메모리의 기존 주소에 올린다. 이 과정에서 램디스크 사이즈도 함께 줄였다. 확장 루트파일시스템은 플래시 메모리의 유저 영역에 올렸다. 부팅 과정에서 기본 루트파일시스템은 램으로 복사되지만 확장 루트파일시스템은 플래시 메모리에 그대로 남게 된다. 부팅 후에 기본 루트파일시스템 영역과 확장 루트파일시스템 영역의 연결은 심볼릭 링크를 이용하여 유지하였다. 부팅 과정에서 압축 해제 및 램으로 복사되는 압축 루트파일시스템 사이즈를 최소화함으로써 부팅시간 감소 효과를 얻고자 했다.

4. 실험 결과

실험에서 사용된 임베디드 보드는 한백전자의 HBE-EMPOS II 이다. <표 2>에서 보는 바와 같이 각 구성요소 별로 최적화를 거친 후 부팅시간이 17초 감소되었다. 기준으로 삼은 28초의 부팅시간은 HBE-EMPOS II 에서 제공하는 부트로더, 커널, 루트 파일시스템에 어떤 최적화 과정도 거치지 않고 측정한 시간이다. 부팅시간이 17초 감소되는 성과가 있었지만 아직 안정화 과정을 거치지 않았다. 부트로더에서 디바이스 초기화 과정을 제거한 점, 루트 파일시스템의 램디스크 사이즈를 6MB 로 제한한 점 등이 향후 안정화 과정에서 고려해야 할 사항이다.

<표 3> 실험 환경

	타겟 보드	호스트 PC
명칭	HBE-EMPOS II	-
프로세서	PXA-255	P4-1.8MHz
SDRAM	128MB	256MB
FLASH	32MB	-
부트로더	EMPOS-BOOT v1.0	-
커널버전	2.4.19	

<표 4> 실험 결과

	부트로더	커널	루트파일시스템
부팅시간 감소효과	4초 감소	5초 감소	8초 감소
기타	부팅최소모듈 (30 ⇒ 3)	압축커널 (948 ⇒ 696)	압축램디스크 (3172 ⇒ 2180) 루트파일시스템 (8386 ⇒ 4934)
부팅시간 변화	28초 (최적화 전) ▶ 11초 (최적화 후)		

(사이즈 단위, KB)

5. 결론 및 향후과제

본 연구에서는 리눅스를 활용한 임베디드 시스템에서 간단한 메커니즘을 이용하여 부팅시간을 줄이고자 하였다. 핵심 메커니즘에 대해 재차 언급하면 부팅에 필요한 최소의 모듈만 램으로 올리도록 해서 램 복사 시간을 줄이는 방식이다. 물론 압축 해제 시간도 단축된다. 실험 결과의 성과는 만족스러운 편이었지만 앞서 언급한 부트로더 및 루트 파일시스템 부분의 안정화 과정이 더 필요하다. 그리고 보다 정밀한 커널 configuration 조정으로 커널 사이즈 최적화 및 Source Level Kernel down sizeing 에 관한 연구, 루트 파일 시스템에서 동적 라이브러리의 효율적인 관리에 관한 연구가 남아 있다.

[참고 문헌]

[1] Bill Roman, "Tips and Tricks for Implementing Software Execute-In-Place with Windows CE.NET", Datalight, 2003

[2] David Selvakumar & Chester Rebeiro, "RTLinux on Memory Constraint Systems", Real Time Linux Workshop, 2004

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

[4] 박지용 외 4명, "Shared Library and Execute-In-Place Support in MMU-less Embedded Systems"

[5] 박찬익, "A low-cost memory architecture with NAND XIP for mobile embedded systems", ACM, 2003

[6] 윤진혁, "NAND 플래시 메모리에서 XIP 기능의 지원", 2001

[7] http://tree.celinuxforum.org/

[8] http://kldp.org/KoreanDoc/html/Kernel-KLDP/index.html