부팅 시 생성한 스냅샷 이미지를 이용한 임베디드 리눅스의 부팅 속도 향상 방안

정회원 조 인 휘*, 준회원 이 상 철*

Improving Bootup Time of Embedded Linux using Snapshot Image Created on Boot Time

Inwhee Joe* ***Regular Member***, Sang Cheol Lee* ***Associate Member***

요 약

임베디드 리눅스 기반의 시스템에서 부팅 속도를 향상시키기 위한 방안 중 하나인 Snapshot boot기법을 개선하여 그 방법의 효용성을 높이고자 한다. Snapshot boot란 현재 수행중인 작업들의 대한 정보를 모두 저장하고 복원하는 방법인 Suspend/Resume을 사용하는 기법으로 리눅스의 일반 부팅과정을 거치지 않고, 부트로더에서 이전 작업의 복원을 처리하는 것으로 부팅속도의 향상을 가져온 기법이다. Snapshot boot의 문제점으로는 Suspend 과정에서 이미지를 생성해 저장장치에 저장을 하게 되는데 모든 페이지를 대상으로 하기 때문에 이 시간이 오래 걸리는 단점이 있다. 또한 이미지 생성 중 예외 상황이 발생 되어 전원이 OFF되면 이미지를 생성하지 못하게 되어 Snapshot boot의 Resume 과정을 수행 할 수 없게 된다. 본 논문에서 제안한 기법은 Snapshot image를 단 한번만 생성을 하고 이것을 활용하도록 하여 부팅 시 속도 및 안정성의 향상의 효과를 볼 수 있다.

Key Words : Snapshot, Bootup Time, Embedded Linux, Suspend, Resume

ABSTRACT

This paper attempts to propose a method to improve the snapshot boot method, one of the methods to enhance the booting speed of the embedded Linux based system and to increase its efficiency. Snapshot boot is a method of using suspend/resume that is a method of saving and restoring the entire information of the current tasks, and it enhances the booting speed by processing a restoration of previous tasks from the bootloader instead of processing the Linux booting. The facing problem of snapshot boot is that it takes a long time to create images and save them to the storage device during the suspend process since it targets every pages. Additionally, if the switch is turned off while creating an image by some extraordinary circumstances, then the image is not created and thus, the resume process of snapshot boot cannot be executed. The suggest method in this paper creates the snapshot image for once only and utilize the image to enhance the speed and stability on booting.

Ⅰ. 서 론

모바일 폰, 스마트 폰, MP3 플레이어, 디지털 카메라 등과 같은 개인 용 휴대 장비가 보편화 되면서, 각 기기의 부팅속도는 제품의 경쟁력을 위해 항상 고려되어 왔다. 애플에서 발표한 아이폰의 경우 기존 시장 제품과의 차별화 된 기능 중 하나로 빠른 부팅 속도를 들 수 있다. 이 빠른 부팅 속도는

* 한양대학교 컴퓨터공학부 이동네트워크 연구실 (iwjoe@hanyang.ac.kr)
논문번호 : KICS2010-08-423, 접수일자 : 2010년 8월 30일, 최종논문접수일자 : 2011년 3월 21일

사용자들로 하여금 만족감을 불러일으키는 요인 중에 하나로 자리 잡았다.

부팅속도를 향상시키기 위해 다양한 방법들이 논의 되고 있다. 부팅 속도를 줄이기 위한 간단한 방법으로는 커널의 부팅 시 옵션으로 quiet를 주어 printk() 메시지를 생략하거나, 불필요한 초기화 함수들을 삭제하는 방법들이 있다. 그 중 대표적인 방법 몇 가지를 나열해 보면 아래와 같다.

1.1 Kernel XIP (eXcute In Place)

Uclinux에서 사용하는 기법으로 커널을 실행 가능한 롬이나 플래시 같은 비 휘발성 메모리 영역에 위치시키는 것이다. 일반 커널의 경우 커널 이미지를 램 영역에 복사 하도록 하는데, kernel XIP 기법의 경우 커널이 부팅 할 때 커널 이미지의 대부분을 차지하는 텍스트 영역을 복사 하지 않음으로써 커널 이미지의 복사로 발생 되는 오버헤드를 줄여 부팅 시간을 단축하는 기법이다. 하지만 이러한 개선으로 얻을 수 있는 효과가 미비하고, 또한 플래시 메모리를 지우거나 쓰는 순간에는 코드가 실행할 수 없는 단점이 있다 [1].

1.2 지연된 장치 탐색 기법(Delayed Devices Probing)

운영체제가 부팅 될 때, 시스템은 인터페이스 채널을 초기화하고, 인터페이스 채널에 연결된 디바이스들이 있다면 드라이버를 등록해 초기화를 수행한다. 이러한 과정이 전체 부팅 시간의 약 80%정도를 차지하고 있다. 따라서 이 방법은 부팅 시 불필요한 인터페이스 채널 초기화와 드라이버의 등록을 부팅 이후로 지연시킴으로써 부팅 속도를 향상 시키는 방법이다. 하지만 이러한 개선으로 부팅 속도를 단축시키는 이점을 얻을 수 있지만, 부팅이 완료된 후 해당 디바이스를 사용하지 않는다는 전제가 있어야 한다 [2].

본 논문에서는 기존에 제안된 Snapshot Boot기법 [4]을 소개하고, 이 방식의 단점을 개선한 방법을 제시한다. Snapshot Boot기법을 간략히 설명을 하면 현재 수행 중인 프로세스들을 중지 시키고, 프로세스들이 사용하던 메모리 정보와 각종 메모리 자료들 및 레지스터 값들을 이용해 snapshot image를 생성하고, 이를 보조기억 장치에 저장을 한다. 그리고 다음 부팅 시 보조 저장 장치에 있는 snapshot image를 불러들여 복원하는 방식이다. 이렇게 함으로써 커널의 별다른 초기화 없이 빠른 부팅을 수행할 수 있게 된다. 이 기법은 현재 사용 중인 모든 메모리를 대상으로 snapshot image를 생성하기 때문에 프로세스의 수가 늘어나고, 메모리의 사용량이 많아질수록 snapshot image를 생성하는데 오랜 시간이 걸려 시스템의 종료 시간 또한 늘어나는 단점이 있다. 또한 snapshot image를 만드는 도중 전원의 off등과 같은 예외 상황이 발생 되었을 경우 snapshot image를 생성하지 못하는 불완전 종료가 발생 될 수 있다. 이럴 경우 보조 기억 장치에 있는 snapshot image가 불완전하므로 부팅 시 이상동작을 하거나, snapshot boot를 수행하지 못하고, 원래의 부팅 방식으로 처리 될 수 밖에 없다.

본 논문에서는 이러한 문제점을 개선하기 위해 제시한 방법으로 snapshot image를 시스템의 종료 될 때 마다 생성을 하지 않고, 시스템의 부팅이 완료된 후의 기본 대기 상태에서 한번만 snapshot image를 생성하여 보조 기억장치에 저장을 한 후, 다음 부팅 시 보조 기억 장치에 저장된 snapshot image를 이용해 부팅을 하는 방법을 제시한다. 비록 이전에 사용자가 사용하던 환경 그대로의 복원은 될 수 없는 단점이 있지만, 시스템의 종료 시간을 단축시키고, 부팅속도의 향상과 안정성 개선의 측면에서 보다 큰 장점을 가질 수 있게 된다.

Ⅱ. 스냅샷 기반의 제안 방식

2.1 기존 방식의 문제점

기존의 Software Suspend/Resume Boot, Snapshot Boot 등등은 2가지의 단점이 있다. 첫 번째로는 종료 시 마다 항상 snapshot image를 생성하고 이를 보조기억 장치에 저장을 해야 되는 문제가 있다. snapshot image는 시스템에서 사용하고 있는 모든 페이지들을 대상으로 하기 때문에 그 크기는 클 수 밖에 없다. 본 논문에서 실험 시 생성 된 snapshot image의 크기도 60Mbyte정도의 크기이다. 종료 시 마다 이 정도의 크기의 snapshot image를 생성하고, 보조기억장치에 저장하는 일을 수행을 하게 된다면, 부팅 속도는 올릴 수 있다고는 하지만 종료 시간이 늘어나는 기현상이 발생 할 수 있다. 그리고 두 번째 문제점으로는 안정성의 문제가 있다. snapshot image를 생성하는 동안 예외상황이 발생 하게 되어 전원이 OFF되는 상황이 발생 할 수 있다. 이렇게 되었을 경우 snapshot image를 생성하지 못하거나, 생성된 snapshot image를 보조 기억 장치에 저장하지 못한 상태로 종료를 하게 된다. 종료 시 snapshot image를 보조 기억 장치에 저장을 하지

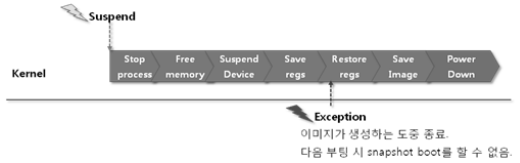

그림 1. Snapshot image 생성 중 예외 상황 발생

못하였으므로 다음 부팅 시에는 Snapshot Boot를 수행 할 수 없어 일반 부팅 모드로 부팅을 할 수 밖에 없는 상황이 발생 하게 된다. 두 번째 문제에 대한 상황을 그림으로 표현 하면 그림 1과 같다.

그림 1에서의 상황의 경우, snapshot image를 생성한 후 프로세서의 상태를 저장하는 도중 OFF되는 예외 상황을 보여준다. 이렇게 될 경우 snapshot image를 생성은 하였으나 보조 기억 장치에 저장을 하지 못했기 때문에 다음 부팅 시 보조기억장치에 저장된 snapshot image가 없으므로 Snapshot Boot를 수행 할 수 없게 된다.

2.2 제안 방식 내용

Snapshot boot의 문제점을 개선하기 위해 본 논문에서는 매번 종료 할 때 마다 snapshot image를 생성하지 않고, 처음 부팅 시 단 한번만 snapshot image를 생성하도록 한다. 그리고 이렇게 생성된 snapshot image를 매 부팅 시 마다 이용 하도록 하여 snapshot boot의 종료 시간을 줄이고, snapshot image 생성 중 발생하는 예외 상황에 대한 안정성을 높이는데 중점을 둔다.

그림 2는 앞에서 설명한 내용을 그림으로 도식화 한 것으로, 제안한 방식의 전체 동작을 나타낸다. 어플리케이션이 구동 된 후에 기본 대기 상태에서 snapshot image를 생성하고, 이를 부팅 시 마다 이용하여 이전 대기상태로 복원을 하는 과정을 보여준다.

본 논문에서 제안한 방식을 자세히 살펴보면 아래와 같다.

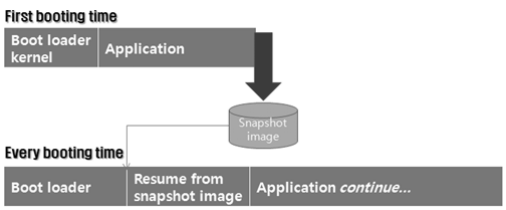

그림 2. 제안 방식의 동작

1) 처음 부팅 시 일반적인 부팅 순서로 부팅을 진행하고, 어플리케이션을 수행한다.
2) 어플리케이션이 구동이 되어 기본 대기 상태로 진입 하게 될 때, snapshot image를 생성하고, 이를 보조 기억 장치 내의 스왑 공간에 저장한다.
3) 시스템을 다시 재부팅 하고, 스왑 공간에 저장된 snapshot image를 보조 기억 장치 내에 snapshot image를 저장하기 위해 따로 할당한 공간에 복사 한다.
4) 앞으로의 모든 부팅 때 마다 저장된 snapshot image를 읽어 부팅을 진행 한다.
5) 부팅이 완료 된 후에 SD Card나 CF Memory와 같은 사용자가 별도로 데이터를 저장하기 위해 사용하는 저장 장치를 마운트 하여 사용한다.

안정적으로 부팅이 완료된 상태에서 오직 한번만 snapshot image를 생성하기 때문에 매 종료 시 마다 부팅 이미지를 생성하는 기존 방식 보다 이미지 생성 중 강제적 power off와 같은 예외 상황이 일어날 확률이 줄어들 것이다. 안정적인 상태에서 생성한 snapshot image를 매 부팅 시마다 이용할 수 있으므로 빠른 부팅을 할 수 있고, 매 종료 시 마다 snapshot image를 생성하지 않으므로 종료 시간의 단축과 snapshot image를 생성하는 중 예외로 인해 image가 생성이 되지 않아 빠른 부팅을 할 수 없는 문제를 해결 할 수 있다. 본 논문에서 제안한 방식을 사용할 경우 사용자의 데이터 저장 용도로 사용되는 저장 장치의 마운트는 부팅이 완료 된 후에 이루어져야 한다. 그 이유에 대해서는 다음 절에서 자세히 설명 하도록 한다.

2.3 제안 방식 구현

본 논문에서 제안한 부팅 방법은 Snapshot Boot 방식을 기반으로 한다. Software Suspend/Resume Boot 방식에도 본 논문에서 제안한 방식을 적용 할 수 있다. 처음 부팅 시 생성한 snapshot image를 이용 하는 것이므로 Snapshot Boot 방식에서 resume만을 수행하고, snapshot image 이미지를 생성하는 suspend과정은 처음 부팅 시 오직 한번 만 수행하는 것이다. 기존 방식에 본 논문에서 제안 한 방법을 적용하기 위해서는 다음 두 가지 사항에 주의를 해야 한다.

첫째는 보조기억 장치에 처음 부팅 시 생성한

snapshot image를 보존 할 수 있는 별도의 공간이 존재해야 한다. 2장에서 설명 한 Software Suspend/ Resume방식과 Snapshot Boot 방식에서는 보조 기억 장치 내의 스왑 공간에 snapshot image를 저장하므로 이 방법을 통해 부팅을 할 경우 스왑 공간을 사용하기 위해 스왑 공간에 있는 snapshot image를 지우는 과정을 수행 한다. 그러므로 처음 부팅 시 생성한 snapshot image를 매 부팅 시 마다 이용하기 위해서는 이 이미지를 보존 할 수 있는 별도의 공간이 필요하다. 본 논문에서는 보조 기억 장치로 256Mbyte의 크기의 Nand 플래시 메모리를 사용하였다. 본 논문에서 이용한 플래시 메모리의 파티션 영역은 표 1과 같다.

처음 부팅 시 생성한 snapshot image를 저장하기 위해 파티션 6번을 사용한다. 그리고 크기는 swap 공간의 크기와 같아야 하므로 78Mbyte를 사용한다. 파티션 중 4번째 파티션의 용도를 android data로 표시 하였는데, 이는 본 논문의 방식을 실험하기 위한 기반 시스템으로 안드로이드를 사용하였기 때문이다. 안드로이드는 플랫폼에서 처음 구동이 될 때 오랜 시간이 소요 되는데, 이는 처음 부팅 이후에 다음 부팅 시 부터는 어플리케이션의 구동 속도를 빠르게 하기 위해 몇몇의 필요한 데이터를 생성하여 저장을 한다. 이런 데이터를 저장하기 위한 공간이 필요하여 Nand 플래시 메모리 내에 저장 공간을 별도로 할당하여 사용하는 것이다. 실제 이 영역은 일반 부팅 모드에서 사용하는 것으로 본 논문에서 제안한 방식과 일반 부팅 모드의 시간을 측정하기 위한 용도로 사용한 것이다. 본 논문에서 제안한 방식의 경우 Nand 플래시 메모리 내에 이런 공간을 할당하여 사용 할 수 없다. 그 이유는 아래에서 설명하도록 한다. 본 논문에서는 안드로이드에 제안한 방식을 적용 후 얻을 수 있는 개선 효과를 측정하는 용도로 이용한 것이어서 안드로이드에 대한 별도의 추가적인 설명을 생략하도록 한다.

표 1. Flash 메모리의 파티션 구성

파티션	용도	크기
1	Boot loader	256 Kbyte
2	Kernel	3.75 Mbyte
3	Root file system	48 Mbyte
4	Android data	48 Mbyte
5	Swap area	78 Mbyte
6	Snapshot image area	78 Mbyte

둘째로는 처음 부팅을 하여 snapshot image를 만드는 시점에 저장을 위한 용도로 사용되는 저장장치가 시스템에 마운트 되어 있지 않아야 한다. 어플리케이션에서 주로 사진이나 동영상 같은 데이터를 저장하는 용도로 사용되는 SD Card나 CF Memory 같은 저장장치가 마운트 된 상태에서 이미지를 만들 경우 전체 시스템에 오류를 일으킬 수 있다. 또한 어플리케이션의 설정 데이터나 생성물을 저장하기 위한 용도로 사용하기 위해 플래시 메모리의 일부 공간을 사용하기도 하는데 이런 공간 역시 마운트 되어 있지 않아야 한다. 이런 저장 장치들의 대부분은 각각의 파일 시스템을 가지고 있다. 만약 처음 부팅 후 snapshot image를 만드는 시점에 이 장치들이 마운트 되어 있고, 이 상태에서 snapshot image를 만들어서 사용 했을 경우를 생각해보자. 시스템에 마운트 된 저장 장치들의 내용을 처음 부팅 시의 내용 그대로의 상태를 유지 하지 않고, 뭔가 다른 데이터를 추가하거나 삭제되었을 경우 이런 데이터들이 적용이 되지 않은 상태로 부팅이 될 수 있다. 또한 각각의 저장장치 내에 파일 시스템을 유지하기 위해 저장 된 메타 데이터와 시스템에서 저장장치를 마운트 한 후에 유지하고 있는 저장 장치에 대한 데이터가 서로 맞지 않아 파일을 접근할 수 없거나 저장 장치에 대한 접근에 오류가 발생 하여 전체 시스템의 심각한 오류를 발생시킬 수 있다. 이런 이유 때문에 본 논문에서 제안한 방식에서는 안드로이드의 데이터를 저장하는 공간을 플래시 메모리로 사용하지 않고, 램 디스크를 이용 한다. 램 디스크의 경우 램을 이용하는 것이므로 처음 부팅 시 상태 그대로 복원을 해도 시스템이 이상 없이 동작을 할 수 있다.

Ⅲ. 실험 결과

3.1 실험환경

본 논문에서는 제안한 방식을 구현 하기위해 사용된 플랫폼으로는 SMDK6410 board가 이용되었다. 이 보드는 ARM1176JZF 프로세서를 장착하고 있고, 128Mbyte의 DDR 메모리, 256Mbyte의 NAND Flash 메모리, 그리고 1Mbyte의 NOR Flash 메모리를 장착하고 있다. 이 외에 100Mbps Ethernet, camera interface, LCD, touch screen등등 다양한 하드웨어가 장착 되어 있다. 사용된 리눅스 커널의 버전은 2.6.29가 사용되었고, 여기에 Software Suspend/Resume Boot와 Snapshot Boot

표 2. 실험 환경

항목	설명
프로세서	S3C6410 (ARM1176JZF, 800 MHz)
타겟 보드	SMDK6410
부트로더	U-boot-1.1.6
운영체제	Linux-2.6.29
어플리케이션	Android
메모리	DDR Memory 128Mbyte
플래시 메모리	K9F2G08U0A (NAND Flash 256Mbyte) AM29LV800 (NOR Flash 1Mbyte)

를 하기 위해 커널 소스를 수정 하였다. 각 실험은 안드로이드 시스템을 기반으로 하여 진행하였다. 표 2는 본 논문에서 실험한 환경을 간략하게 보여준다.

3.2 실험결과

기존의 제안된 부팅 방식과 본 논문에서 제안한 방식의 성능을 비교하기 위하여 Normal Boot, Software Suspend/Resume Boot, Snapshot Boot와 본 논문에서 제안한 방식 등 4가지 방식의 부팅 시간과 종료 시간을 측정하여 비교하였다.

표 3은 Normal Boot, Software Suspend/Resume Boot, Snapshot Boot와 제안 방식의 시스템 부팅 시간과 종료 시간을 보여준다. 먼저 부팅시간의 결과를 보면 Normal Boot의 부팅 시간이 32초로 가장 긴 것을 볼 수 있다. Software Suspend/Resume Boot 방식의 경우 일반 부팅 모드 보다 부팅 시간이 4초가량 단축된다. Snapshot Boot 방식과 제안 방식의 경우 Software Suspend/Resume Boot 방식 보다 3초, 일반 부팅 모드 보다 7초가량(약 22%)이 단축되는 것을 볼 수 있다. Snapshot Boot와 제안 방식의 부팅 속도는 차이가 없다. 이는 snapshot image를 만드는 시점을 모두 안드로이드가 대기 상태로 진입한 상태에서 생성을 했기 때문에 부팅 시간이 동일하게 측정 되었다. 만약 보다 많은 어플리케이션을 수행한 상태에서 종료 후 다시 재부팅을 한다면 Snapshot Boot와 제안 방식의 부팅시간에도 차이가 발생할 것이다.

표 3. 시스템 부팅 및 종료 시간

	Normal Boot	Suspend /Resume Boot	Snapshot Boot	제안 방식
부팅 시간	32 sec	28 sec	25 sec	25 sec
종료 시간	12.3 sec	52.1 sec	52.1 sec	12.3 sec

종료시간의 경우 Software Suspend/Resume Boot와 Snapshot Boot의 종료시간은 52.1초 정도가 소요되고, 일반 부팅 모드와 제안 방식의 경우 종료시간이 12.3초 정도 소요 되는 것을 볼 수 있다. Software Suspend/Resume Boot와 Snapshot Boot 방식의 경우 종료 시간이 일반 부팅 모드와 제안 방식과 비교하여 상대적으로 훨씬 더 많은 시간이 걸리는 것을 볼 수 있다. 이는 Software Suspend/Resume Boot와 Snapshot Boot의 경우 Suspend 과정을 거쳐 종료를 수행하는데, Suspend 과정을 수행 시 현재 시스템에 대한 정보를 저장하기 위해 Snapshot image를 생성하게 된다. 이 이미지를 생성하는 시간이 시스템의 종료시간을 오래 걸리도록 하는 원인이 된다. 이는 본 논문에서 지적했던 Software Suspend/Resume Boot방법과 Snapshot Boot방법의 단점 중 하나이다. 제안한 방식의 경우 시스템의 종료 시간이 일반 부팅 모드와 비슷한데, 이는 제안한 방식의 경우 snapshot image의 생성을 오직 처음 부팅 시 한번만을 수행하게 되므로 종료 시 Suspend 과정을 거치지 않고, 일반 부팅 모드와 같은 방식을 사용하게 된다. 이 때문에 기존 Software Suspend/Resume Boot 방법과 Snapshot Boot 방법의 종료시간 보다 상당히 개선 된 효과를 얻을 수 있었다. 또한 시스템이 종료 할 때 마다 Snapshot image를 생성하지 않으므로 예외 상황으로 인해 시스템이 종료가 되어도 이미 snapshot image를 생성한 것을 가지고 있으므로 언제나 일반 부팅 모드보다 빠른 시간 내에 부팅을 할 수 있게 된다.

그림 3은 각각의 부팅 모드 별 부팅 시간과 종료 시간을 차트로 보여준다. 세로축이 부팅 모드를 나타내고, 가로축이 시간을 나타낸다.

실험 결과를 보면 본 논문에서 제안한 방식은 기존 Snapshot Boot 방식의 단축효과를 얻으면서도 종료 시간은 일반 부팅 모드와 유사하게 단축 할 수 있는 효과를 얻을 수 있는 것을 볼 수 있다. 부팅시간의 경우 일반 부팅 모드의 시간보다 7초가량 단축 된 것을 볼 수 있고, 종료시간의 경우 기존의 제안 된 방식보다 40여 초 가량 단축하여 일반 부팅 모드와 비슷한 종료시간이 되는 것을 볼 수 있다. 또한 매 종료 시 마다 snapshot image를 만들지 않으므로 안정성도 한층 높아졌다 할 수 있다. 오직 처음 부팅 시 단 한번만 snapshot image를 만들어 저장하여 사용 하므로 예외 상황의 발생하여 갑작스러운 종료 시에도 다음 부팅 시 빠른 부팅을

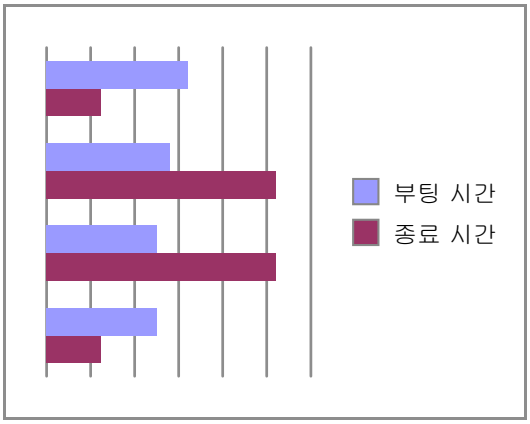

그림 3. 부팅모드별 부팅시간과 종료시간

수행 하는데 지장이 없다. Snapshot Boot와 Software Suspend/Resume Boot의 경우 snapshot image를 매 종료 시 마다 생성 하여 저장 하므로, 이미지를 생성 중 예외 상황으로 인해 갑작스러운 종료 되었을 경우 다음 부팅 시 빠른 부팅을 할 수 없다.

Ⅳ. 결 론

본 논문은 기존에 제안된 Software Suspend/Resume 방식과 Snapshot Boot의 경우 부팅 시간의 단축은 이루었지만, 반대로 snapshot image를 매 종료 시 마다 생성을 해야 하므로 종료 시간의 증가와 이미지 생성 중 예외 상황에 대한 대처가 이루어 지지 못했다. 본 논문에서 제안한 방식의 경우 처음 부팅 시 오직 한번 만 snapshot image를 생성하여 별도의 저장 공간에 저장을 하고, 이를 매 부팅 시 마다 저장된 snapshot image를 이용하여 부팅 시간을 줄이면서도 종료 시간 또한 일반 부팅 모드에서와 같은 종료 시간이 걸리도록 하였다. 그리고 매 종료 시 마다 snapshot image를 생성하지 않으므로 종료 중 예외 상황이 발생 하여도 항상 단축 된 부팅 시간을 얻을 수 있도록 기존 방식을 개선하였다.

참 고 문 헌

[1] CE Linux Forum (CELF) Kernel XIP "http://tree.celinuxforum.org/CelfPubWik/KernelXIP"

[2] 박우람, 나윤주, 박찬익 "지연된 장치 탐색을 이용한 부팅시간 향상 기법" 정보과학회 2006 추계 학술대회

[3] Hiroki Kaminaga, "Imporving Linux Startup Time Using Software Resume," 2006 Linux Symposium

[4] 박세진, 송재환 "개선된 스냅샷 부트를 이용한 임베디드 리눅스의 빠른 부팅 기법." 정보과학학회지 : 컴퓨팅의 실제 및 레터 제 14권 6호 2008.8

[5] http://android.git.kernel.org

[6] http://git.kernel.org/?p=linux/kernel/git/kki_ap/linux-2.6-samsung.git;a=summary

[7] Bovet and Cesati, "Understanding Linux Kernel 3rd Edition" O'Reilly

조 인 휘 (Inwhee Joe) 정회원

1983년 2월 한양대학교 전자공학과 학사
1994년 12월 미국 University of Arizona, Electrical and Computer Engineering, M.S.
1998년 9월 미국 Georgia Tech, Electrical and Computer Engineering, Ph.D.
1992년 12월 (주) 데이콤 종합연구소 선임연구원
2000년 6월 미국 Oak Ridge 국립연구소 연구원
2002년 8월 미국 Bellcore Lab (Telcordia) 연구원
2002년 9월~현재 한양대학교 컴퓨터공학부 부교수
<관심분야> Sensor Networks, Cellular System and PCS, Network Security, Mobility Management

이 상 철 (Sang Cheol Lee) 준회원

2008년 2월 학점은행제 전자공학과 학사
2010년 8월 한양대학교 컴퓨터공학과 석사
2006년 6월 (주)아젝스메디테크 사원
2010년 6월 (주)디지털포캐스트 과장
2010년 9월~현재 KT Innontz 연구소 연구원
<관심분야> 시스템 가상화, 클라우드 컴퓨팅, 멀티미디어 스트리밍, 멀티 코어 플랫폼