한국컴퓨터정보학회 동계학술대회 논문집 제25권 제1호 (2017. 1)

리눅스 기반 로봇 시스템의 부트 시간 단축을 위한 외부 컨텍스트 기반 선별적 자원 사용률 조정 기법

이은성[O], 김정호[**], 양범준[*], 홍성수[*,**]
[O*]서울대학교 전기정보공학부
[**]서울대학교 융합과학기술대학원 융합과학부
e-mail:{eslee, jhkim, bjyang, sshong}@redwood.snu.ac.kr

External Context-Based Selective Resource Utilization Control Technique for Reducing Boot Time of Linux-Based Robot System

Eunseong Lee[O], Jungho Kim[**], Beomjoon Yang[*], Seongsoo Hong[*,**]
[O*]Department of Electrical and Computer Engineering, SNU
[**]Department of Transdisciplinary Studies, GSCST, SNU

● 요 약 ●

지능형 로봇의 사용자 품질을 결정하는 주요 요소들 중 하나는 짧은 부트 시간이다. 로봇 시스템에서는 부팅 과정 중에 침입자 인지, 자택 순찰, 개인 비서, 엔터테인먼트와 같은 다수의 응용들이 동시에 초기화되는데, 고품질의 사용자 경험을 제공하기 위해서는 사용자 응답성이 중요한 응용들이 우선적으로 초기화되어야한다. 이를 위해 리눅스 기반 로봇 시스템에서 부트 시간을 단축하기 위한 다양한 연구들이 진행되어 왔다. 하지만 이들은 단일 응용 각각에 대한 초기화 시간을 단축하는 연구들이며, 응용들 간에 CPU, 메모리, I/O와 같은 자원 경쟁에 의한 지연 요소를 고려하지 않고 있다. 본 논문에서는 응용들 간의 각종 자원 경쟁들을 고려하여 사용자 응답성이 중요한 응용을 우선적으로 초기화하기 위한 외부 컨텍스트 기반 선별적 자원 사용률 조정 기법을 제안한다. 이를 리눅스 기반 시스템 상에 구현하여 검증한 결과 응용의 부트 시간이 기존 대비 33.02% 단축됨을 확인했다.

키워드: 부트 시간(Boot Time), 외부 컨택스트(External Context), 태스크 체인(Task Chain), 교차 계층 최적화(Cross-Layer Optimization)

I. 서론

산업, 의료, 우주탐사, 가정 등 다양한 분야에서 전방위적으로 활용되고 있는 로봇이 급속도로 지능화되고 있다. 이러한 지능형 로봇의 사용자 품질을 결정하는 주요 요소들 중 하나는 짧은 부트 시간이다. 가령, 가정용 휴머노이드 로봇은 높은 소비전력을 요구하기 때문에 사용자가 부재중인 경우에는 로봇의 전원을 꺼놓고 필요에 따라 전원을 켠다. 사용자가 집에 돌아왔을 때 서비스를 빠르게 제공받기 위해서는 짧은 부트 시간을 갖는 것은 중요하다.

이러한 요구사항을 충족하기 위해서 부트 시간을 단축하기 위한 다양한 연구들이 진행되어 왔다. 이들은 일반적으로 범용적인 활용성을 위해 리눅스 기반 시스템 상에서 수행되었다. 게다가 로봇 시스템에서 요구되는 응용이 많아짐에 따라 대다수의 로봇들에 GPOS(General Purpose Operating System)인 리눅스가 탑재되었다[1][2]. 한 예로, 미래형 가정용 로봇으로 전망되는 휴머노이드의 경우, 표 1과 같이 전부 리눅스 기반으로 구동된다.

표 1. 휴머노이드 로봇의 리눅스 사용 여부

국가	한국	일본1	일본2	미국	프랑스	중국
로봇명	HUBO	Kawada	Asimo	Aido	Nao	Pepper
리눅스 사용 여부	O	O	O	O	O	O

리눅스 기반 시스템에서 수행된 연구들은 부트 과정의 단계 별로 구분된다. 부트 과정은 바이오스, 부트로더, 커널 레벨, 사용자 레벨 초기화 순으로 구성된다. 이들 중 바이오스, 부트로더, 커널 레벨 초기화에 관련된 최적화 연구들은 이미 포화상태에 이르러 수 초 이내에 작업이 완료된다[3]. 이는 수십 초 소요되는 전체 부트 시간에서 차지하는 비중이 작다. 반면 사용자 레벨 초기화에 관련된 연구들은 단일 응용 또는 단일 커널 서브시스템에 대하여 초기화 시간을 단축한 연구들이다[5][6]. 따라서 다수의 응용들을 동시에 초기화하는 자능형 로봇 시스템에서 활용되기에는 한계가 있다. 이는 기존 연구들이 전체 부트 시간에서 큰 비중을 차지하는 응용들 간의 자원 경쟁에 의한 지연 시간을 고려하지 않기 때문이다.

이러한 한계를 극복하기 위해 본 논문은 외부 컨텍스트 기반 선별적 자원 사용률 조정 기법을 제안한다. 외부 컨텍스트를 통해 우선적으로 초기화되어야 하는 응용을 파악한다. 그리고 초기화되어야할 응용들의 우선순위에 따라 자원 사용률을 상향 조정하여 초기화 순서를 조정한다. 이를 통해 사용자 응답성이 중요한 응용의 부트 시간을 단축한다. 본 연구진은 제안된 기법을 리눅스 기반 시스템 상에 구현하여 검증한 결과 기존 대비 사용자 레벨 부트 시간이 33.02% 단축됨을 확인했다.

II. 배경지식

이 장에서는 본 논문을 이해하기 위해 필요한 리눅스 기반 시스템의 부트 과정과 리눅스의 자원 사용률 조정 기법인 cgroups를 설명한다.

1. 리눅스 기반 시스템의 부트 과정

본 논문에서는 전원이 켜진 시점부터 특정 응용이 초기화 작업을 마치고 GUI를 화면에 출력하기까지의 시간을 부트 시간(boot time)으로 정의한다. 그림 1은 리눅스 기반 시스템에서의 전체 부트 과정(boot process)을 나타낸다. 부트 과정은 바이오스 초기화, 부트로더 초기화, 커널 레벨 초기화 그리고 사용자 레벨 초기화로 순으로 수행된다. 바이오스 초기화 과정에서는 시스템에 연결된 모든 센서 및 주변 기기와 같은 하드웨어의 정상 동작 여부 확인하고 ROM에서 부트로더 이미지를 메모리로 적재한 후 부트로더를 실행한다. 실행된 부트로더는 스토리지에 있는 커널 이미지를 메모리에 적재하고 커널의 첫 인스트럭션을 수행한다. 커널 레벨 초기화에서는 CPU 스케줄러, 메모리 등의 커널 자원을 초기화한다. 그리고 커널 스레드와 디바이스 장치들을 초기화한다. 마지막으로 사용자 레벨 초기화에서는 프레임워크 계층의 자원을 독점적적으로 관리하는 시스템 서버와 일반 응용을 순차적으로 실행한다.

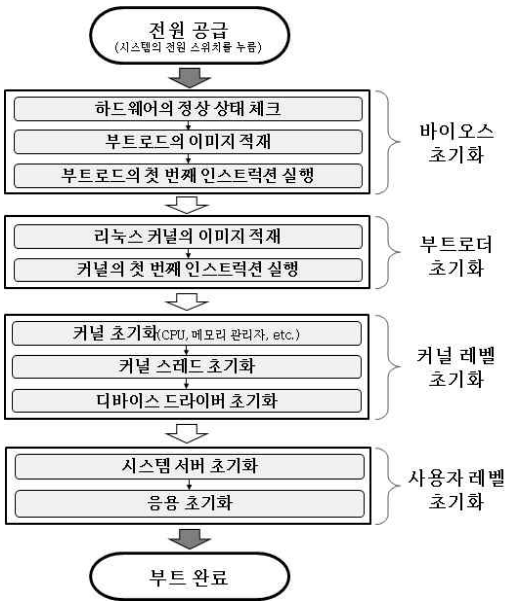

그림 1. 리눅스 기반 시스템의 부트 과정

일반적으로 리눅스 기반 시스템의 부트 과정 중 사용자 레벨 초기화 단계에서 가장 많은 지연이 발생한다[3]. 일본의 대표 휴머노이드 로봇 카와다 로봇과 유사한 스펙 상에서의 사용자 레벨 초기화 지연은 30초 내외로 전체 지연에 약 88%를 차지한다[7]. 구체적인 스펙은 아래와 같다. CPU는 Snapdragon APQ8060 Dual Core 1.5GHz, 스토리지는 Samsung eMMC 4.41이며 읽기 대역폭은 ~30MB/s이다.

2. 리눅스의 자원 사용률 조정 기법: cgroups

cgroups는 리눅스 시스템 상에서 동작중인 태스크들을 임의의 서브시스템 단위로 그룹지어 제어하는 프레임워크이다. 서브시스템은 cgroups에 의해 만들어진 그룹에 대하여 시스템 자원 할당, 우선순위 지정, 모니터링 등의 제어가 가능한 모듈이다. 서브시스템의 예로는 CPU, 메모리, 네트워크, block I/O등이 있다. cgroups으로 특정 서브시스템의 사용률을 조정하기 위해 가중치(weight), 한계치(limit) 등을 사용한다.

cgroups는 다음과 같은 특성을 갖는다. 우선 cgroups는 계층적 구조를 갖는다. cgroups 모델은 트리 형태로 구성된다. cgroup은 트리를 구성하는 노드를 의미하며, 이는 자식 cgroup 또는 태스크로 구성된다. 또 다른 특성은 자식 cgroup은 부모 cgroup의 속성을 상속받는다는 것이다.

III. 외부 컨텍스트 기반 선별적 자원 사용률 조정 기법

제안된 기법은 우선순위에 따른 응용 초기화의 순서 조정으로 사용자 레벨 초기화 시간의 단축을 목적으로 한다. 초기화되는 응용들의 우선순위는 시스템 부팅시의 상황인 외부 컨텍스트에 따라 달라진다. 응용 초기화의 순서 조정은 태스크 체인의 자원 사용률의 순차적

상향 조정을 통해 가능하다. 태스크 체인이란 어떤 응용의 초기화를 위해 연속적으로 수행되어야하는 태스크 리스트를 의미한다. 이러한 태스크 체인은 각 응용마다 한 개 존재한다.

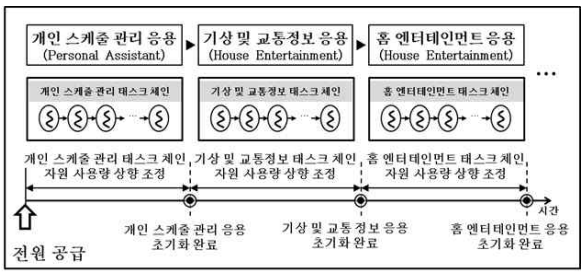

그림 2. 제안된 기법의 동작 예시

그림 2는 사용자가 출근을 준비하는 외부 컨텍스트에서 제안 기법이 적용된 예시이다. 이러한 외부 컨텍스트에서는 개인 스케줄 관리, 기상 및 교통정보, 홈 엔터테인먼트 순으로 응용이 초기화되어야 한다. 이를 위해 가장 먼저 개인 스케줄 관리 응용에 대한 태스크 체인의 자원 사용률을 상향 조정한다. 해당 응용의 초기화가 완료되면 기상 및 교통정보 응용에 대한 태스크 체인의 자원 사용률을 상향 조정한다. 이와 같이 특정 외부 컨텍스트에는 우선시되어야하는 태스크 체인이 존재한다.

그림 3은 프레임워크화된 제안 기법의 구성요소들을 일반적인 리눅스 시스템의 계층적 소프트웨어 스택 상에서 나타낸다. 해당 구성요소는 크게 정보 전달 인터페이스와 리눅스 커널 내부에 구현된 메커니즘으로 나뉜다. 정보 전달 인터페이스는 리눅스 커널과 라이브러리 계층 사이에 존재한다. 이는 애플리케이션, 프레임워크, 라이브러리 계층으로부터 응용별 태스크 체인, 외부 컨텍스트와 부팅 순서 테이블 정보를 리눅스 커널 계층에 전달한다. 태스크 체인은 태스크 ID의 리스트로, 외부 컨텍스트은 각각이 구별 가능한 정수형 변수로 구현된다. 부팅 순서 테이블은 각각의 외부 컨텍스트 별로 등록된 태스크 체인의 부팅 순서로 부팅 순서 테이블은 런타임 이전에 미리 등록된다.

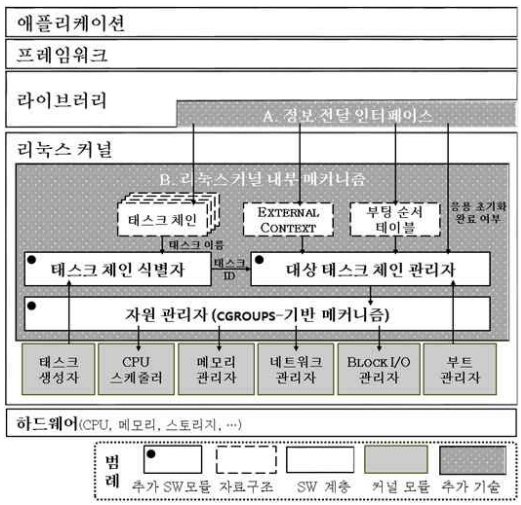

그림 3. 계층적 소프트웨어 스택 상에서 제안된 기법

리눅스 커널 내부에 추가적으로 구현된 SW 모듈은 (1)태스크 체인 식별자, (2)대상 태스크 체인 관리자, (3)자원 관리자이다. 태스크 체인 식별자는 태스크 체인에 속한 태스크의 ID를 식별하는 작업을 수행한다. 이를 위해 태스크 체인 식별자는 리눅스 커널내의 태스크 생성자로부터 태스크에 대한 정보를 전달받고 라이브러리 계층으로부터 태스크 체인 정보를 전달받는다. 해당 정보를 사용하여 어떤 태스크가 생성될 때마다 그 태스크가 특정 태스크 체인에 속하는지를 확인한다. 마지막으로 태스크 체인 정보와 그에 속한 태스크의 ID 정보를 대상 태스크 체인 관리자에게 전달된다.

대상 태스크 체인 관리자는 최초 부팅 시점 또는 태스크 체인의 초기화가 완료될 때 마다 자원 사용률을 상향 조정할 태스크 체인을 결정한다. 해당 작업을 위해 입력으로 외부 컨텍스트, 부팅 순서 테이블, 태스크 체인별 태스크 ID 그리고 태스크 체인 초기화 완료 여부를 받는다. 그리고 전달받은 외부 컨텍스트와 부팅 순서 테이블을 토대로 초기화할 태스크 체인의 순서를 확인한다. 각 태스크 체인이 완료될 때마다 다음에 초기화 되어야할 태스크 체인의 모든 태스크들의 ID를 자원 관리자에게 전달한다.

자원 관리자는 초기화되어야할 특정 태스크 체인의 자원 사용률을 상향 조정한다. 해당 작업은 대상 태스크 체인 관리자로부터 대상 태스크 ID들을 전달받을 때 마다 수행된다. 자원 관리자는 리눅스에 구현되어있는 cgroups기법을 사용하여 자원 사용률을 조정한다. 구체적으로는 태스크 체인안의 태스크들을 하나의 cgroup으로 묶어 CPU와 메모리, 네트워크, block I/O 자원 사용률을 조정한다. 대상 cgroup의 CPU와 block I/O 자원에 대한 사용률은 가중치 값을 통해 결정된다. 또한 대상 태스크 체인의 네트워크 자원의 사용률은 다른 태스크 대비 트래픽 처리 비율의 설정으로 상향 조정가능하다. 한편 메모리 자원은 한계치값의 설정으로 사용량 조정이 가능하다. 모든 자원에 대해 cgroups로 조정 가능한 수치로 대상 태스크 체인의 자원 사용률을 상향 조정하였다.

IV. 실험 환경 및 검증 결과

본 연구진은 제안된 기법을 리눅스 시스템에 구현하였다. 실제 로봇 시스템에 사용되는 응용과 같이 CPU와 메모리 작업량이 많은 벤치마크를 사용해 사용자 레벨 초기화 시간의 단축을 검증했다. 표 2는 실험 환경으로 일본의 대표 휴머노이드 로봇 카와다 로봇에서 사용한 스펙과 유사하게 구성하였다.

표 2. 실험 환경

Hardware	AP	S5P6818
	CPU	Cortex-A53 Octa Core CPU 1.5GHz
	Main Memory	2GB
	Storage	Flash memory 23GB
Software	Kernel	Linux kernel v3.4.39

제안 기법의 검증을 위해 로봇 시스템이 부팅될 때 사용자가 사용할 응용들의 초기화 시간을 측정한다. 미래형 가정용 로봇 시스템의 대표적인 응용인 침입자 인지, 자택 순찰 응용은 종합적인 판단을

위해 많은 계산량이 요구된다. 또한 이 알고리즘은 많은 메모리 I/O 작업을 요구하는 영상, 음성 데이터 등을 입력으로 받는다. 이러한 특징을 가지는 응용을 모델링하기 위해 SPEC CPU2006 벤치마크 태스크들 중 CPU 집약적이고 메모리 집약적인 태스크를 사용하여 태스크 체인을 구성하였다. 실험에는 3개의 태스크 체인을 사용하였으며 각 태스크 체인 안에는 태스크 하나가 존재한다. 검증을 위해 제안된 기법의 적용 전후로 각각 50회씩 반복 실험하였다.

그림4는 제안된 기법 적용 전후에 대한 실험 결과를 나타낸다. 가로축은 사용자 레벨 초기화 시간이고 세로축은 그 횟수이다. 제안된 기술이 적용된 이후 3개의 태스크에 대하여 평균 초기화 시간이 약 33.02% 단축되었음을 확인할 수 있다.

V. 결론

본 논문은 로봇 시스템에서 사용자 경험을 증진하기 위한 부트 시간 단축 기법을 제안하였다. 이를 위해 미래형 가정용 로봇 시스템에서 가장 많이 쓰이는 운영체제인 리눅스 시스템의 부트 과정을 분석하였다. 그리고 해당 시스템에 본 연구진의 제안 기법을 적용한 후, 실험을 통해 부트 시간이 단축되었음을 확인하였다.

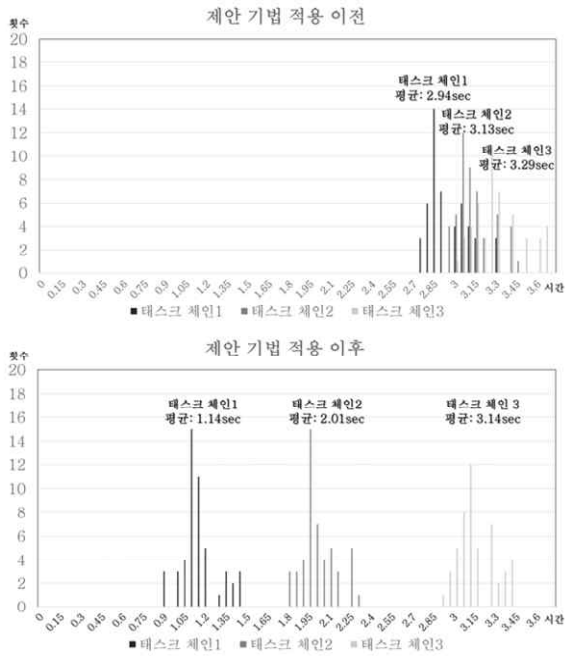

그림 4. 제안 기법 적용 이후 부트 시간의 변화

References

[1] Khafizuddin Azazi et al., "Design of Mobile Robot with Navigation Based on Embedded Linux", International Conference on Computer Science and Computational Intelligenc(ICCSCI), 2015.

[2] Ling Xu et al., "Wireless Control of Mobile Robot Technology Research Environment under Linux", Applied Mechanics and Materials, 2014.

[3] Abhishek Palod and Rishabh Gupta, "Analysis of Boot Time Reduction techniques in Linux", M.S. thesis, Mälardalen Univ., Västerås, 2009.

[4] Sam H. Noh, “Fast Booting Android based on Snapshot Booting”, Linux Forum, 2012.

[5] Mikael Åsberg et al., "Fast Linux bootup using non-intrusive methods for predictable industrial embedded systems", Emerging Technologies & Factory Automation (ETFA), 2013.

[6] Antonio Bovenzi et al., "Towards fast OS rejuvenation: An experimental evaluation of fast OS reboot techniques", International Symposium on Software Reliability Engineering (ISSRE), 2013.

[7] Kawada Specification, http://nextage.kawada.jp/en/specification/