ISSN: 1226-7244 (Print)
ISSN: 2288-243X (Online)
논문번호 13-04-25
172

Journal of IKEEE.Vol.17,No.4,564~569,December 2013
http://dx.doi.org/10.7471/ikeee.2013.17.4.564

멀티코어 기반의 임베디드 시스템에서 안드로이드 부팅 속도 향상 방법

An Improving Method of Android Boot Speed in Multi-core based Embedded System

최 진 용*, 이 재 흥*★

Jin-Yong Choi*, Jae-Heung Lee*★

Abstract

The current embedded devices are growing rapidly in the multi-core, and these demand fast boot time. But method of previous boot uses core only one. The method includes parallel techniques and modification of CPU Frequency policy. Parallel methods, after analyzing the Android boot process with analysis tool, applied to location where a lot of CPU operation. CPU Frequency policy is modified for high performance of core. The proposed method was applied to S5PV310 dual core and Exynos4412 quad core embedded system. As a result of the experiment, we found that the proposed method makes boot time fast about 20.71% and 31.34% in dual core and quad core environment as compared with the previous method.

요 약

현재 임베디드 기기는 멀티코어로 급성장하고 있으며 빠른 부팅 속도를 요구하고 있다. 하지만 기존의 부팅 기술은 하나의 코어만을 사용하고 있다. 따라서 본 논문에서는 분석 도구를 통해 안드로이드 부트 프로세스를 분석 후, CPU연산이 많은 곳에 병렬 기법을 적용하는 방법과 멀티 코어의 성능을 최대로 끌어내기 위해 CPU주파수 정책을 변경함으로써 멀티코어 기반에서 안드로이드 부팅 속도 향상 방법에 대해 제안한다. 본 논문의 제안 방법을 듀얼 코어 S5PV310과 쿼드 코어 Exynos4412에 각각 적용시킨 뒤 부팅 완료 시간을 측정하였으며 기존의 방법과 제안 방법의 시간을 비교한 결과 듀얼코어와 쿼드코어에서 각각 약 20.71%, 약 31.34%의 속도 성능향상을 가져왔다.

Key words : multi-core, embedded system, android boot, Bootchart, CPUfreq governor

Ⅰ. 서론

스마트 폰과 스마트 TV, 디지털 카메라, 내비게이션, 차량용 블랙박스 등과 같은 다양한 임베디드기기는 우리의 실생활에서 흔히 접할 수 있을 정도로 범용화 되어있고 생활하는데 없어서는 안될 만큼 현대인의 필수품으로 자리 잡고 있다. 이렇게 매년 임베디드기기 시장이 증가 될수록 기기 안에 탑재되는 소프트웨어는 요구사항이 많아짐에 따라 규모가 점점 커지고 있으며 이에 따라 프로세서가 처리할 내용도 많아지고 있다. 이 때문에 처음 시스템을 초기화 하는 과정인 부팅작업 또한 늘어나고 있는 실정이다. 이러한 임베디드기기의 부팅 속도의 지연현상은 항상 이슈로 부각되어 왔고 부팅지연 문제를 해결하기 위한 최적화 기술은 제품의 경쟁력을 강화하는 중요한 요소로 자리매김하고 있다.[1] 이러한 최적화 기술은

* Dept. of Computer Engineering, Hanbat University (010-5343-7692 , heaveninj@naver.com)
★ Corresponding author (010-3404-6954 , jhlee@hanbat.ac.kr)
※ Acknowledgment
This research was financially suppoerted by the Ministry of Education (MOE) and National Research Foundation of Korea(NRF) through the Human Resource Training Project for Regional Innovation (No. 2012H1B8A2026119).
Manuscript received Dec. 12, 2013; revised Dec. 27, 2013; accepted Dec 27. 2013

해를 거듭 할수록 발전하고 있지만 대부분 리눅스 환경을 기준으로 적용된 기술이 주를 이루고 있으며, 최근 몇 년 전부터 사용량이 급격하게 증가한 안드로이드 환경에서 적용된 사항은 아직 미흡한 형태이다. 또한, 현재 출시되고 있는 임베디드기기에서는 대부분 멀티 코어 아키텍처를 채택하였지만 순차적인 부팅 흐름의 한계에 부딪혀 싱글 코어와 동일한 수준으로 부팅을 수행하고 있다.[2-4]

본 논문에서는 기존 기술에서 고려하지 않았던 멀티 코어환경에서 안드로이드의 부팅 속도 향상 방법에 대해 제안한다. 제안방법은 기존 부트 프로세스를 분석한 뒤 속도 저하 요인을 찾고 해당 프로세스에 병렬 기법을 적용하는 방법과 기존의 CPU주파수 정책을 변경하는 방법으로 구분된다.

II. 안드로이드 부팅 및 CPU주파수 정책 분석

1. 안드로이드 부팅 분석

가. 부트 프로세스 분석 방안

멀티코어 환경에서의 성능 향상 방법은 어떠한 작업단위에 병렬 기법을 적용하는 것이 대표적으로 쓰레드, fork()함수를 통한 프로세스 복사 등이 있다. 병렬기법은 프로세스 내의 연산 영역을 독립적으로 분할하여 각 코어에서 동시연산을 수행하는 것으로, 반복된 연산 영역이 많은 프로그램에 적용할 경우에 효율성이 높다. 만약 부팅 과정 중 특정 프로세스에 연산이 많아지게 되면 전체 부팅 시간에 영향을 주게 된다. 이러한 프로세스에 병렬 기법의 적용을 통해 전체 부팅시간의 감소 효과를 볼 수 있다. 따라서 병렬 처리가 가능한 구간이나 프로세스를 먼저 선정하는 것이 필요하다.

부트 프로세스의 분석은 그림1과 같이 먼저 멀티코어기반의 임베디드 보드에 부트 프로세스 분석 도구를 이식 한 후 부팅을 시작한다. 부팅 중에는 분석 도구에 의하여 활성중인 프로세스들의 정보들을 모으고 있으며, 부팅이 완료 된 후엔 모은 데이터들을 raw데이터로 저장한다. 이 raw데이터는 PC환경으로 옮겨진 후 분석 도구의 시각화 기능을 사용하여 이미지 파일로 만드는 과정을 거친다. 그 후 이미지파일을 통해 병렬 처리가 가능한 구간이나 프로세스를 선정하는 과정을 거친다.

이 분석 방안에서는 Bootchart라는 도구를 사용한다. Bootchart는 GNU/리눅스 등의 OS의 부트 프로세스의 성능 분석 및 시각화를 위한 도구로서 부팅과정 중 소요된 자원 활용 및 프로세스 정보를 수집하는

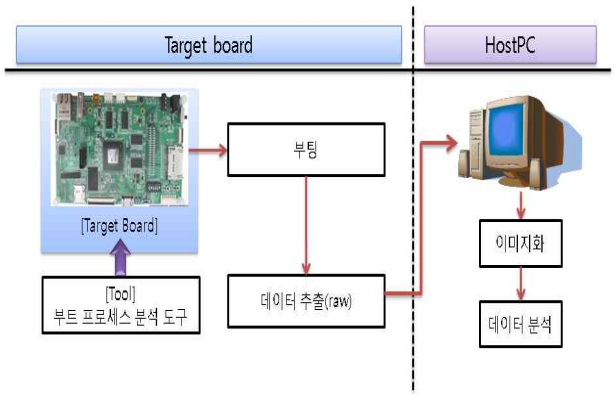

Fig. 1. A method of boot process analysis
그림 1. 부트 프로세스 분석 방안

데이터 로거 유틸리티와 시스템 자원 활용에 대한 정보를 PNG, SVG등과 같은 그림 파일형태로 제공해주는 시각화 유틸리티를 포함하고 있다. Bootchart의 데이터 로거는 init프로세스 대신 실행되며 백그라운드에서 동작하여 proc 파일 시스템에 있는 CPU 통계, CPU I/O와 Idle 시간, 디스크 활용, 모든 활성 프로세스 관련 정보 등을 주기적인 간격으로 샘플링 하는 프로파일러이다. 샘플링을 거쳐 얻게 된 미 가공 데이터는 Bootchart에 포함된 시각화 유틸리티를 사용하여 부팅 로그 정보 이미지 파일로 형성하는데 아래 그림2와 같은 차트 및 그래프 형태의 3가지 구성요소로 표현된다.

그림2의 ①은 해당 부팅 구간의 CPU 상태로 CPU 사용량(푸른색)와 I/O(붉은색), Idle을 나타내고 있고, ②는 해당 부팅 구간에서의 디스크 사용량에 대한 정보를 나타낸다. 마지막 ③은 부팅 시 모든 활성 프로세스의 관련 정보와 해당 프로세스의 CPU 사용량을 나타낸다.

나. 안드로이드 부트 프로세스 분석

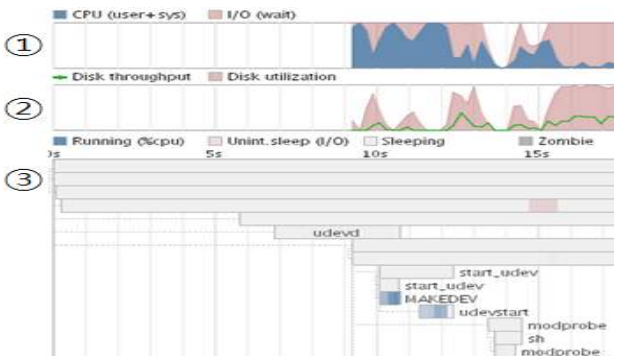

Fig. 2. Component of Bootchart
그림 2. Bootchart 구성요소

병렬 처리 구간이나 프로세스를 선정하기 위해 듀얼 코어 S5PV310과 쿼드 코어 Exynos4412에 Bootchart를 적용하여 그림3와 같은 결과를 얻었다.

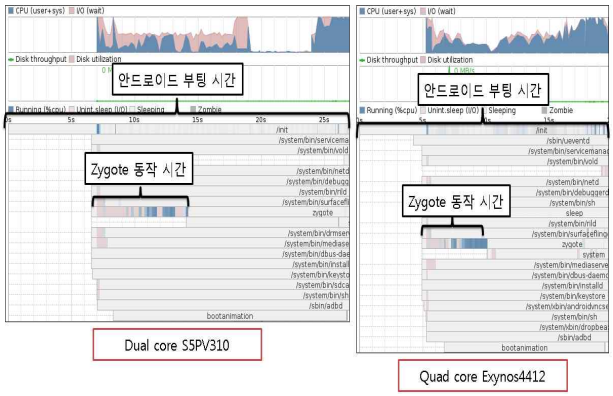

Fig. 3. Android boot for each board
그림 3. 보드별 안드로이드 부트

Bootchart의 결과를 보면 두 보드의 부팅 흐름이 비슷한 양상을 보이고 있고, 공통적으로 Zygote라는 프로세스에서 CPU의 사용량이 많다는 것을 알 수 있었다. Zygote 프로세스는 전체 부팅 시간 중 평균 25~26% 정도를 차지하고 있다.

Zygote프로세스는 안드로이드 어플리케이션을 빠르게 구동하기 위해서 미리 fork되어있는 프로세스로 시스템에서 exec() 호출을 통해 특정 어플리케이션을 실행하고자 하기 전까지는 중립적인 상태를 유지한다. Zygote프로세스는 그림4와 같이 네 가지의 기능들로 구성되어 있으며 각각의 기능에 따른 함수들이 구현되어 있다. Socket Binding은 새로운 어플리케이션의 실행 요청을 수신하는데 쓸 소켓을 바인딩 한다. Preload는 어플리케이션 프레임워크에 포함된 클래스와 자원을 메모리에 적재시킨다. System Server는 Package Manager Service, Activity Manager Service, Location Manager Service등과 같은 자바 시스템 서비스들을 실행한다. 마지막으로 Loop는 새로운 안드로이드 어플리케이션의 생성 요청을 받으면 이를 처리한다.[5-6]

이 내부 함수들의 시간 측정 결과 preload()에서 77.85%~85.70%로 가장 높은 비율을 차지한다는 것을 알 수 있었다. Preload()는 세부적으로 frameworks/base/preload-classes에 담겨있는 약 2300여 가지의 클래스들을 적재하는 preloadClasses()와 문자열, 레이아웃, 색, 이미지, 사운드 등의 리소스들을 적재하는 preloadResource()를 차례대로 호출한다. 따라서 클래스들과 리소스를 적재하는 내용이 많을수록 시간이 비례하게 증가한다.

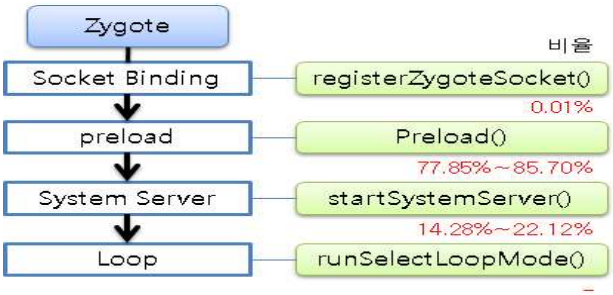

Fig. 4. Zygote 프로세스
그림 4. Zygote process

2. CPU주파수 정책 분석

가. 임베디드 기기의 성능과 저 전력 기법

임베디드 기기나 데스크 탑 프로세서의 클럭 주파수는 시스템의 성능에 매우 많은 영향을 차지하고 있다. 주파수가 높을수록 성능이 증가하지만 전력소모 또한 증가하기 때문에 임베디드 기기에서는 항상 높은 성능만을 내기엔 어려움이 있다. 따라서 이러한 문제를 해결하기 위한 대표적인 저 전력 기법으로 DVFS(Dynamic Voltage and Frequency Scaling)가 있다. 이 기법은 프로세서의 클럭 주파수와 전압을 조정함으로써 열과 에너지 사용량을 줄이는 방식이다.[7]

리눅스에서는 DVFS가 적용된 모듈인 CPUfreq governor가 쓰이고 있다. 이 모듈은 리눅스 2.6버전 이후부터 내장되어 사용되고 있으며 리눅스 커널 내에 CPU의 부하에 따라 주파수를 동적으로 할당하여 전력 효율을 올린다. CPUfreq governor는 아래 그림5와 같이 크게 3부분으로 나눌 수 있다.

CPU-specific driver는 CPU의 주파수를 직접 제어하고, CPUfreq module은 CPU-specific driver와 CPUfreq governor사이를 연결하며 독립적으로 동작해주도록 지원한다.[8] CPUfreq governor는 CPUfreq module을 통해 CPU 주파수를 현재 상황에서 얼마큼

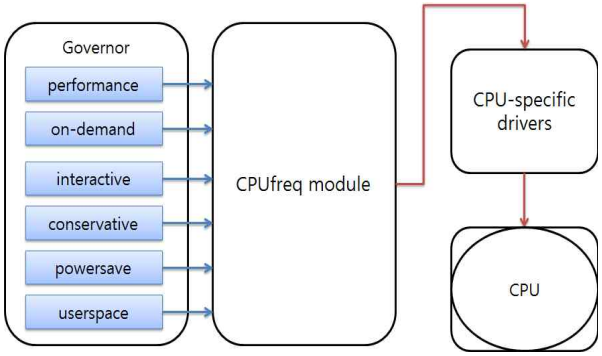

Fig. 5. Structure of CPUfreq governor
그림 5. CPUfreq governor 구조

주파수를 변경할지에 대한 정책을 가지고 있다. 리눅스에서는 다양한 정책을 구현하여 사용자의 요구에 따라 사용할 수 있도록 지원한다. 일반적으로 6가지의 정책을 사용하며 그 내용은 다음 표1과 같다.

Table 1. Type of CPUfreq governor policy
표 1. CPUfreq governor 정책의 종류

정책	특징
performance	최대 주파수로 동작
on-demand (default)	부하에 따라 주파수 변경 - 임계치 ↑ : 최대 - 임계치 ↓ : 단계별 하향
interactive	on-demand 개선 정책 고정 시간에 따라 부하 판단
conservative	on-demand 변형 정책 - 임계치 ↑ : 단계별 상향 - 임계치 ↓ : 단계별 하향
powersave	최소 주파수로 동작
userspace	user영역의 프로그램에 의해 사용자가 임의로 조정 가능

나. 안드로이드 부팅 시 CPU 주파수 변화

안드로이드는 리눅스의 CPUfreq governor가 기본적으로 on-demand정책을 사용하고 있기 때문에 안드로이드 또한 이 방식으로 동작한다. 부팅 중 on-demand정책을 사용한 CPU주파수 수치 흐름을 측정해본 결과 아래 그림6과 같은 그래프로 표현되었다.

CPU부하가 많을 경우 주파수가 높은 반면 CPU부하가 적을 경우 주파수가 낮아지게 되는 지그재그 형태를 보였으나 평균값을 비교하였을 땐 부팅 시작부터 종료 시점까지 점차 주파수가 내려가는 것을 확인할 수 있었다. 이러한 현상의 이유는 그림4의

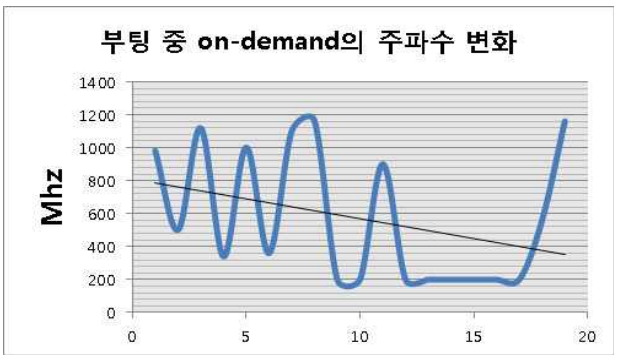

Fig. 6. CPU frequency variation of on-demand on boot
그림 6. 부팅 중 on-demand의 주파수 변화

Bootchart에서 CPU의 I/O시간이나 idle시간이 많아짐에 따라 CPU부하가 없기 때문에 on-demand 정책 특성상 주파수가 내려간다고 볼 수 있다. 따라서 CPU부하 측정치가 상향 임계점을 넘어가기 전까지 낮은 주파수로 동작하기 때문에 부팅속도에 영향을 준다.

III. 본 논문의 병렬 기법 적용 및 CPU주파수 정책

1. 병렬 기법 적용

Preload함수의 앞서 설명한 대로 preloadClasses() 함수와 preloadResource() 함수를 차례대로 호출하는 구조를 가지고 있다. 이 두 가지의 함수 흐름에 병렬 기법을 적용하는 방법으로 fork()와 thread가 있다. Zygote는 자바 언어로 만들어져 있기 때문에 사용하기엔 간편하나 자바 특성상 가상 머신에서 동작을 하기 때문에 실행 환경이 가볍지 않다는 단점이 있다. 따라서 오버헤드를 줄이기 위해 JNI를 사용하여 fork()함수를 호출하는 방법을 구현하고 이를 Zygote 프로세스에서 사용하도록 하였다.

그림7과 같이 preload의 preloadResource()는 부모 프로세스의 pid를 가지고 있어 기본적인 preload의 호출 구조와 동일하다. preloadClasses()의 경우 자식 프로세스에서 동작하기 위해 fork()를 호출해야 한다. 하지만 fork()는 따로 C라이브러리에만 존재하고 따로 선언이 되어 있지 않기 때문에 안드로이드 빌드 전 JNI NativeMethods 구조체에 호출하려는 callfork와 C라이브러리의 fork()를 연결시켜야 한다. 연결이 등록되면 callfork와 연결된 fork()가 호출되어 자식 프로세스를 생성하고 이 프로세스에서 preloadClasses()를 처리하게 된다.

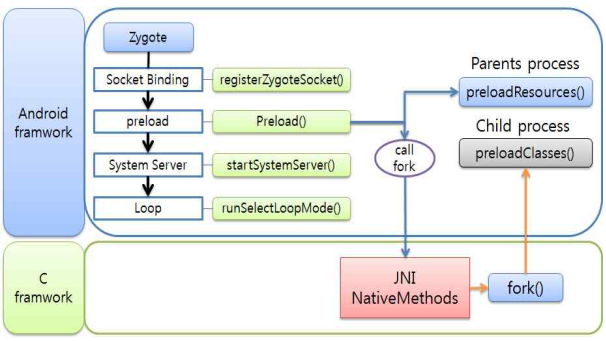

Fig. 7. Structure in the process to call a fork()
그림 7. Preload 프로세스의 fork() 호출 구조

2. CPU주파수 정책

기존 정책의 속도 저하 현상을 해결하기 위하여 기존 on-demand방식을 최대 성능의 정책인 performance와 on-demand를 혼합한 형태로 변경하였다. 변경된 정책의 흐름은 그림8과 같다.

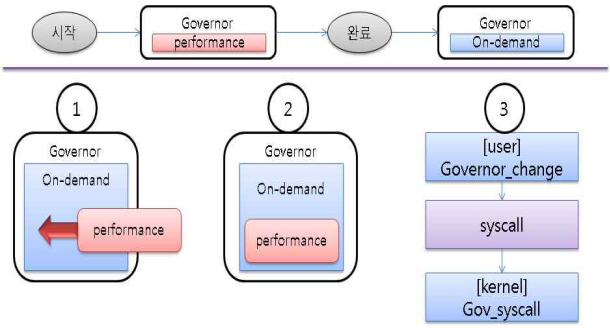

Fig. 8. Modification of CPUfreq governor policy
그림 8. CPUfreq governor 정책의 변경

보드가 부팅을 시작한 뒤 CPUfreq governor의 정책은 맨 처음 performance로 설정된다. 그 후 부팅완료 후에는 사용자영역에서 CPU주파수 정책을 기존의 on-demand로 변경하게 된다. CPUfreq governor의 정책은 커널 빌드 시에 고정되고 커널영역에서 관여하기 때문에 사용자영역에서 접근하기 위해 system call을 사용해야 한다. 따라서 그림8의 1~2번과 같이 CPUfreq governor의 on-demand정책에 performance의 부분을 추가 하고 정책의 변경을 위해 사용자영역의 governor_change()와 커널영역의 gov_syscall()을 생성해주었다.

Ⅳ. 실험 및 고찰

1. 실험 환경

실험 환경은 (주)삼성전자 듀얼 코어 S5PV310과 (주)삼성전자 쿼드 코어 Exynos4412를 탑재한 보드를 사용하였다.

Table 2. Experimental environment
표 2. 실험 환경

Dual core S5PV310		
H/W	SOC	S5PV310 1G/1.3GHz (Cortex-A9 dual core)
	RAM	DDR3 SDRAM 800Mhz 2GB
S/W	kernel	Linux kernel 3.0.15
	android	Ice Cream Sandwich 4.0.4

Quad core Exynos4412		
H/W	SOC	Exynos4412 Prime 1.7Ghz (Cortex-A9 quad core)
	RAM	LP-DDR2 880Mhz 2GB
S/W	kernel	Linux kernel 3.0.51
	android	Jelly Bean 4.1.2

2. 부팅 시간 측정

실험에 사용한 두 보드에 총 4가지 방법(기본 부팅, Zygote프로세스만 병렬처리 했을 경우, CPU주파수 정책만 변경했을 경우, 두 가지 방법을 모두 사용했을 경우)을 5번에 걸쳐 부팅 시간을 측정하였다.

Table 3. Measured boot time(sec)
표 3. 부팅 시간 측정(sec)

Board	적용 방법	횟수 1	2	3	4	5	평균
Dual core	original	26.23	25.69	25.00	23.83	25.00	25.15
	Zygote	20.59	20.79	20.71	20.84	20.62	20.71
	CPUfreq	21.27	22.24	24.39	23.94	21.23	22.61
	Zygote+CPU	20.45	20.26	19.92	20.47	18.61	19.94

Board	적용 방법	횟수 1	2	3	4	5	평균
Quad core	original	15.37	15.28	15.52	15.75	15.77	15.54
	Zygote	13.91	13.76	14.08	13.95	13.82	13.90
	CPU freq	12.21	12.16	12.07	12.08	11.99	12.10
	Zygote+CPU	10.72	10.62	10.63	10.62	10.75	10.67

듀얼 코어의 경우 Zygote 병렬 처리만 적용하였을 경우 약 17.65%, CPU 주파수 정책을 변경했을 경우 10.08%, 둘 다 적용했을 경우 20.71%의 성능 향상을 보였다. 쿼드 코어는 Zygote 병렬 처리만 적용하였을 경우 약 10.52%, CPU 주파수 정책을 변경했을 경우 22.11%, 둘 다 적용했을 경우 31.34%의 성능 향상을 보였다. Zygote의 병렬처리의 경우 두 보드의 안드로이드 버전이 다르기 때문에 성능 향상이 다르게 나타나고 있고 CPU 주파수 정책 부분은 각 코어의

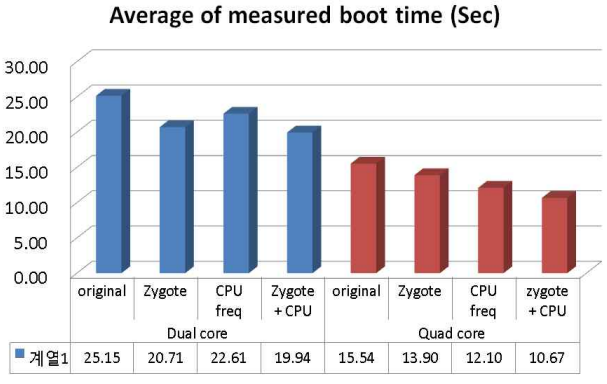

Fig. 9. Average of measured boot time
그림 9. 측정된 부팅 시간의 평균

CPU주파수 수치를 최대로 증가시키기 때문에 듀얼 코어보다 쿼드 코어에서 약 2배 이상의 성능향상 결과를 얻었다.

Ⅴ. 결론

본 논문에서는 멀티코어 기반의 임베디드 시스템에서 안드로이드 부팅 속도 향상방법을 제안하였다. 듀얼 코어와 쿼드 코어 보드에 제안 방법을 적용한 결과 각각 20.71%, 31.34%의 성능향상을 보였다. 측정된 부팅 시간을 보면 제안 방법을 독립적으로 적용한 부팅 시간의 합이 두 가지 방법을 모두 적용했을 때의 부팅 시간과 비슷하다는 것을 알 수 있다. 이 결과는 하나의 적용방식이 다른 적용방식에 영향을 주지 않고 독립적으로 적용되었다는 것을 입증한다.

제안하는 부팅 속도 향상 방법은 안드로이드 버전에 상관없이 적용이 가능한 기술이며 적용범위에 비해 높은 성능향상을 가져오고 있다. 나아가 이 방법은 시스템 전체를 기준으로 본다면 극히 일부분에 불과하다. 따라서 안드로이드 시스템 측면뿐만 아니라 기존 기술에 적용되어 왔던 최적화가 함께 이루어진다면 더 빠른 속도 향상을 가져올 것이다.

References

[1] Se-Jin Pack, Song-Jae Hwan, Chan-ik Park “A Fast Booting Technique using Improved Snapshot Boot in Embedded Linux”. Journal of KIISE : Computing Practices and Letters. Vol.14. No.6. pp.594-598, 2008.8.

[2] Kyung-Ho Chung, Myung-Sil Choi, Kwang-Seon Ahn. "A study on the packaging for fast boot-up time in the embedded linux." Embedded and Real-Time Computing Systems and Applications, 2007. RTCSA 2007. 13th IEEE International Conference on. (pp.89-94) IEEE, 2007.

[3] Kwang-Mu Shin, Seong-Ho Park, Ki-Dong Chung “Fast Booting Implementation of the Linux in the Embedded System”. Proc. of the 32th KIISE Fall Conference. Vol.32. No.2. pp.853-855. 2005

[4] In-Whee Joe, Sang-Cheol Lee “Improving Bootup Time of Embedded Linux using Snapshot Image Created on Boot Time”. Journal of KICS. Vol.36. No.3B. pp.254-259. 2011.

[5] Singh, Gaurav, Kumar Bipin, and Rohit Dhawan. "Optimizing the boot time of Android on embedded system." Consumer Electronics (ISCE), 2011 IEEE 15th International Symposium on. (pp. 503-508). IEEE, 2011.

[6] Xia Yang, Nan Sang, Jim Alves-Foss. “Improving the Boot Time of the Android OS”IEEE Computer Society. Aug. 2013

[7] Tam, David, Winnie Tsang, and Catalin Drula. " Dynamic voltage scaling in mobile devices." Csc2228 project final report, University of Toront (Dec. 2003) (2003).

[8] Miyakawa, Daisuke, and Yutaka Ishikawa. "Process oriented power management." Industrial Embedded Systems, 2007. SIES'07. International Symposium on. (pp. 1-8). IEEE, 2007.

BIOGRAPHY

Choi Jin-Yong (Student Member)

2012 : BS degree in Computer Engineering, Hanbat National University.

2012 ~ Present : MS degree in Computer Engineering, Hanbat National University.

Lee Jae-Heung (Member)

1983 : BS degree in Electronic Engineering, Hanyang University.

1985 : MS degree in Electronic Engineering, Hanyang University.

1994 : PhD degree in Electronic Engineering, Hanyang University.

1989 ~ Present : Professor in Dept. of Computer Engineering, Hanbat University