Linux Kernel Profiling 기법을 통한 안드로이드 부팅 시퀀스 I/O 패턴 분석

유준영*, 이성우** 임승호*
*한국외국어대학교 전자정보공학부
**이에프텍
e-mail : jonnyyu86@gmail.com

Android Boot Sequence I/O Pattern Analysis through Linux Kernel Profiling Techniques

*Jun-Young Yu, Sung-Woo Lee**, and Seung-Ho Lim*
School of Electronics and Information Engineering
Hankuk University of Foreign Studies
**Elixir Flash Technology Co., Ltd.

요 약

최근 안드로이드 플랫폼은 성능 개선의 많은 이슈를 가지고 있다. 그 중 안드로이드 부팅 시퀀스부분이 중요한 부분으로 차지하고 있다. 안드로이드 플랫폼 부팅 과정의 속도 저하가 발생 되는 부분은 블록 I/O 시스템이다. 본 논문에서는 리눅스 블록 레이어의 I/O를 tracing 해주는 blktrace에 대해 소개를 하고 그 기법을 통해 안드로이드 부팅 시퀀스의 I/O 패턴을 분석하고, 개선 방안을 고찰한다.

1. 서론

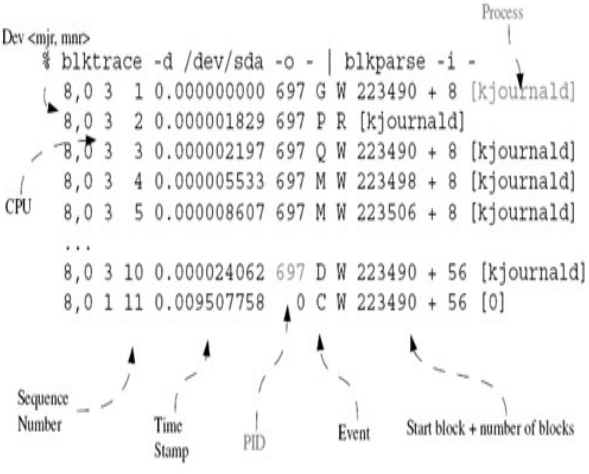
최근 스마트폰은 사용자의 욕구에 따라 빠른 추세로 발전하고 있다. 그에 발맞추어 안드로이드 플랫폼은 버전 업그레이드를 통해 성능이 점점 개선되고 있는 중이다. 안드로이드 플랫폼의 일부분을 차지하고 있는 리눅스 커널 부분은 변함없이 2.6.x 버전 대를 사용하고 있고 지금도 다양한 분야(부팅, 실시간, 전력관리, 신뢰성)별로 기능이 추가되고 있다. 안드로이드 플랫폼내부에 있는 리눅스 커널은 안드로이드에 맞게 수정된 커널이고, HAL과 인터페이스 역할을 하며, 하드웨어를 구동하기 위한 디바이스 드라이버와 메모리 관리, 프로세스 관리, 네트워크 관리 기능을 가지고 있다.

프로파일링은 어떤 상황에서 안드로이드 플랫폼이 발생시키는 I/O 부분을 추출하는 것이다. 프로파일링은 tracing 같은 말로서 block 레벨의 I/O tracing 할 수 있는 도구로서 blktrace가 있다. 이 도구를 이용하여 안드로이드 부팅 시퀀스 I/O 패턴을 추출 할 것이다. 그리고 하드웨어는 Galaxy Tab, huins의 ACHRO-HD라는 개발 보드를 사용할 것이다.

본 논문에서는 안드로이드 기반 블록 디바이스의 I/O 패턴(pattern)을 리눅스 프로파일링 기법을 통해 추출과 분석하고, 그 분석을 바탕으로 부팅 속도에 효율을 보고자 한다. 안드로이드 부팅 시퀀스는 안드로이드 플랫폼에서 안드로이드 커널 속의 init.rc라는 파일이 가장 먼저 실행되는데 그 파일 안에 프로파일링 하는 툴의 명령을 실행하는 셀 스크립트 파일을 넣으면 그 실행 파일이 실행되면서 부팅 시퀀스에서 발생하는 I/O를 추출 할 수 있다.

2. Linux Kernel Profiling 기법 소개 – blktrace

blktrace는 block layer의 I/O를 추적해주는 기능을 제공한다.[1] kernel에서 수행되며 user application에서 요청하는 실제 disk I/O 패턴을 추출 하는데 사용된다. 그림 1 에서는 실제 blktrace와 blkparse command 예제 및 결과 그림인데 Dev, CPU, Sequence Number, Time, Event, Process 등 많은 정보를 알 수 있다.



(그림 1) blktrace 와 blkparse command 예제와 결과

blktrace tool은 세 가지로 구성 요소가 있다. 첫 번째는 kernel patch 부분이다. kernel에서 block 장치 이벤트를 남겨주는 인터페이스로서 trace에서 이 정보를 이용하여 I/O 패턴을 추출하게 된다. 2.6.17 이상 버전의 커널이

라면 패치 할 필요 없이 설정 변경만으로 사용 가능하다. 두 번째는 blktrace 유틸리티이다. blktrace 유틸리티는 커널에서 이벤트를 추출하여 저장장치로 정보를 저장하거나 표준 출력 방식을 제공한다. 세 번째는 blkparse 유틸리티이다. 파일로 저장된 이벤트 정보들을 원하는 포맷으로 정리해주는 기능을 수행하는 유틸리티이다.

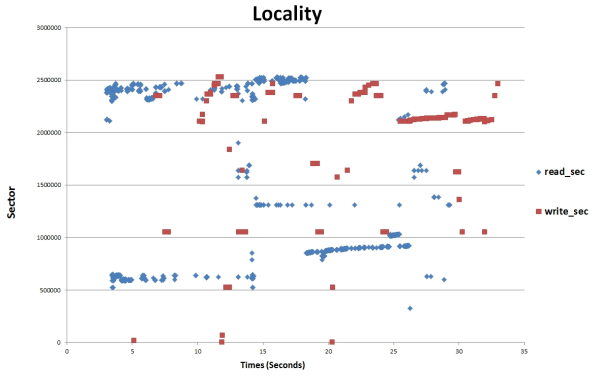
3.1 실험 대상과 부팅 시퀀스 I/O 패턴 분석

실험 대상으로 사용한 안드로이드 장치는 huins사의 ACHRO-HD라는 안드로이드 개발용 보드와 Galay-Tab을 사용하였으며, ACHRO-HD는 안드로이드 2.1버전(이클레어), Galay-Tab은 안드로이드 2.2버전(프로요)이다. 두 장치에 대한 자세한 사항은 표 1에 기록 되어 있다.

	ACHRO-HD Android 2.1 - Eclair				Galaxy-Tab Android 2.2 - Proyo			
	Capacity	**Mount**	**File System**	**Size**	**Capacity**	**Mount**	**File System**	**Size**
RAM	512M				512M			
SD	iNAND 2G	Code			OneNAND 4G In CPU-Pack	Code		
		/system	EXT3	271M		**/system**	RFS	371M
		/data	EXT3	1.1G		**/data**	RFS	540M

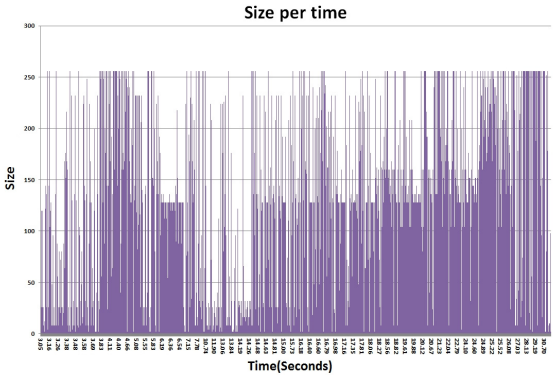
(표 1) 안드로이드 장치의 세부 정보

blktrace는 git 저장장소에서 받을 수 있다. 받은 소스를 가지고 실험 대상에서 동작 할 수 있도록 툴체인(ARM-EABI-2009q3)을 사용 하여 컴파일 하였다.[2] 그리고 최종적으로 추출한 booting.blktrace.o 파일을 btt 유틸리티로 한 번 더 우리가 필요한 자료 분류를 한다.[3] 우리는 먼저 추출한 자료를 바탕으로 두 가지 그래프를 그렸다. 첫 번째는 시간대별 얼마나의 사이즈를 read, write 했는지 알 수 있는 그래프이다.



(그림 2) 시간대별 발생한 I/O의 섹터 위치

그림 2의 그래프는 안드로이드 플랫폼이 올라간 개발보드에서 안드로이드 부팅이 되는 시간동안 read, write의 움직임을 본 그래프인데 처음 15동안은 500,000와 2,500,000번지에서 계속 read를 하다가 15동 이후로는 1,300,000와 900,000번지에서 read를 하는 모습을 볼 수 있고, write는 처음 25동안은 조금씩 움직임이 있다가 25이후로 2,200,000번지에서 write를 하는 모습을 볼 수 있다.

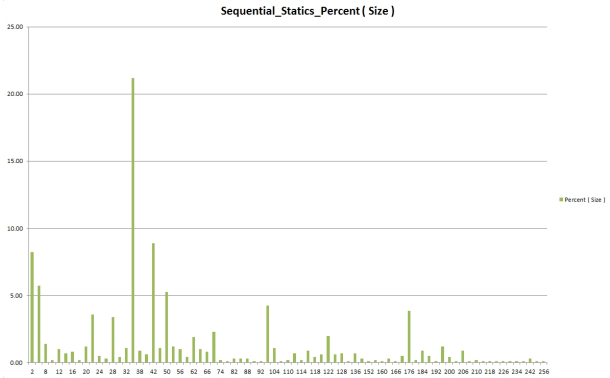


(그림 3) 시간대별 size 크기

그림 3의 그래프는 시간별로 얼마큼의 size를 read하였는지 보여주는 그래프이다. 그림 1의 그래프와 비교하면서 보면 read의 움직임 많은 시간 때(4초~5초, 19초~25초) 그림2의 그래프에서도 size가 높게 나타난다.

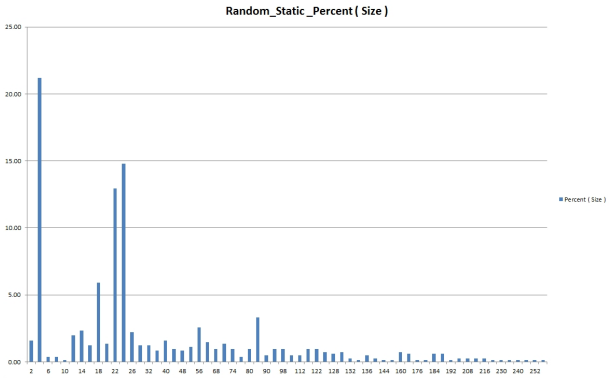
3.2 분석 결과

그림2과 그림3의 그래프를 가지고 더 자세한 자료를 얻기 위해 우리는 비연속적일 때와 연속적일 때 얼마큼의 size를 read를 하면 그 횟수를 얼마나 되는지 분석하였다.



(그림 4) 연속적인 섹터의 사이즈 크기와 비중

그 결과 그림 4의 그래프에서는 size 1의 크기는 512byte이다. size 2는 1Kbyte size 4는 2Kbyte이며, 그래프에서 보듯이 가장 많은 read를 한 size는 21.17%으로 17Kbyte였다.



(그림 5) 비연속적인 섹터의 사이즈 크기와 비중

그림 5의 그래프에서는 비연속일 때의 그래프인데 가장 많이 read 된 사이즈는 21.18%로 2Kbyte였고, 다음으로는 14.78%로 12Kbyte, 그 다음으로는 12.93%로 11Kbyte였다.

4. 결론

안드로이드 플랫폼은 지금 스마트폰 업계에서 애플의 IOS 경쟁하는 또 하나의 OS이다. 그 만큼 플랫폼 내의 리눅스 커널을 바탕으로 구현된 안드로이드 커널이 얼마나 효율적으로 성능을 내는지는 많은 사람들의 관심을 갖는다.

본 논문에서 제안한 안드로이드 부팅 시퀀스 I/O 패턴 분석은 안드로이드 플랫폼이 올라간 개발보드와 Galaxy Tab에 있는 안드로이드 커널에서 크로스 컴파일 환경을 구축하여 자료를 추출한 것이고, 1차적으로 추출한 자료를 바탕으로 시간대별 발생한 I/O 패턴과 size의 크기를 시각화하였고, 2차적으로 간단한 c프로그램을 이용하여 연속일 때와 불연속일 때를 구분하여 각각 얼마큼의 size가 read, write 되는지 또 그 비중을 얼마나 되는지 분석하였다. 그리고 더 나아가 분석한 I/O 패턴을 가지고 데이터의 움직임을 미리 읽고 그 패턴에 맞게 OS를 수정한다면 안드로이드 커널의 속도 향상에 효과가 있을 것이다.

참고문헌

[1] Jens Axboe, Alan D. Brunelle “Blktrace User Guide”

[2] https://opensource.samsung.com/

[3] Jens Axboe, Alan D. Brunelle “Btt User Guide ”