ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 44, No. 3, pp. 232-238, 2017. 3
https://doi.org/10.5626/JOK.2017.44.3.232

AIOPro: 안드로이드 스마트폰을 위한 통합된 스토리지 I/O 분석도구

(AIOPro: A Fully-Integrated Storage I/O Profiler for Android Smartphones)

한 상 욱 † (Sangwook Shane Hahn)　이 인 혁 †† (Inhyuk Yee)　류 동 욱 ††† (Donguk Ryu)　김 지 홍 †††† (Jihong Kim)

요 약 사용자 경험이 중요한 스마트폰에서는 사용자의 입력에 대한 응용 프로그램의 반응 시간에 대한 분석이 필요하며 특히 사용자 반응 시간에 큰 영향을 주는 스토리지 I/O 성능의 분석은 사용자 경험 최적화를 위한 중요한 요건이다. 사용자가 경험하는 반응 시간의 분석을 위해서는 입력을 받아들이는 최상위 계층에서 최하위 저장장치 계층을 수직적으로 아우르는 통합된 분석도구가 필요한데 기존의 도구들은 제한된 범위의 계층들에 맞추어 있어서 효과적인 사용자 경험 분석이 어려운 상황이다. 본 논문에서는 안드로이드 스마트폰을 대상으로 최상위 안드로이드 플랫폼, 리눅스 커널을 포함하는 전체 스토리지 I/O 계층의 I/O 동작을 측정하고, 이를 바탕으로 계층 통합적 분석을 통하여 각 계층간 I/O 동작을 연결하여 사용자 입력에 따른 스토리지 I/O가 미치는 영향의 분석이 가능한 도구인 AIOPro를 소개한다. 검증 실험을 통하여 AIOPro가 0.1% 미만의 동작부하로 정확히 분석할 수 있음을 확인하였다.

키워드: 데이터 입출력 분석도구, 안드로이드 플랫폼 분석도구, 리눅스 커널 분석도구

Abstract Application response time is critical to end-user response time in Android smartphones. Due to the plentiful resources of recent smartphones, storage I/O response time becomes a major key factor in application response time. However, existing storage I/O trace tools for Android and Linux give limited information only for a specific I/O layer which makes it difficult to combine I/O information from different I/O layers, because not helpful for application developer and researchers. In this paper, we propose a novel storage I/O trace tool for Android, called AIOPro (Android I/O profiler). It traces storage I/O from application - Android platform - system call - virtual file system - native file system - page cache - block layer - SCSI layer and device driver. It then combines the storage I/O information from I/O layers by linking them with file information and physical address. Our evaluations of real smartphone usage scenarios and benchmarks show that AIOPro can track storage I/O information from all I/O layers without any data loss under 0.1% system overheads.

Keywords: I/O trace tool, Android platform I/O tracer, kernel I/O tracer, I/O visualizer

· 이 연구를 위해 연구장비를 지원하고 공간을 제공한 서울대학교 컴퓨터연구소에 감사드립니다. 이 논문은 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2015M3C4A7065645)
· 이 논문은 2016 한국컴퓨터종합학술대회에서 'AIOPro: 안드로이드 스마트폰을 위한 통합된 스토리지 I/O 분석도구'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 서울대학교 컴퓨터공학부
shanehahn@davinci.snu.ac.kr
†† 비 회 원 : 서울대학교 컴퓨터공학부
ihyee@davinci.snu.ac.kr
††† 비 회 원 : 삼성전자 소프트웨어연구소 책임
du.ryu@samsung.com
†††† 종신회원 : 서울대학교 컴퓨터공학부 교수(Seoul Nat'l Univ.)
jihong@davinci.snu.ac.kr
(Corresponding author임)

논문접수 : 2016년 8월 18일
(Received 18 August 2016)
논문수정 : 2016년 11월 30일
(Revised 30 November 2016)
심사완료 : 2016년 12월 2일
(Accepted 2 December 2016)

정보과학회논문지 제44권 제3호(2017. 3)

1. 서 론

안드로이드 스마트폰에서는 사용자가 응용 프로그램을 사용하면서 느끼는 사용자 입력에 대한 응용 프로그램의 반응 시간이 사용자 경험에 큰 영향을 끼친다. 즉, 사용자 경험이 안드로이드 스마트폰의 성능에 가장 큰 영향을 주는 요소이기 때문에, 사용자 중심의 응용 프로그램 반응 시간을 결정하는 요소들에 대한 분석이 중요하다[1].

응용 프로그램 반응 시간은 일반적으로 크게 응용 프로그램 수행 시간(유저 레벨과 커널 레벨), 네트워크 I/O 수행 시간 그리고 스토리지 I/O 수행 시간으로 구성된다. 안드로이드 스마트폰의 지속된 성능향상으로 인해 응용 프로그램 수행 시간이 줄어들면서 스토리지 I/O수행 시간의 비중이 크게 증가하고 있다[2,3]. 그림 1은 안드로이드 스마트폰 중 하나인 Nexus 6에서 Gmail, Twitter, Facebook, Youtube, GTA 의 시작 시간 중 응용 프로그램 수행시간, 네트워크 I/O 수행 시간, 스토리지 I/O 수행 시간을 보여준다. 그림 1에서 알 수 있듯이, 스토리지 I/O 수행 시간이 응용 프로그램 런칭 시간 중 많은 시간을 차지하는 것을 확인할 수 있다.

사용자 경험에 영향을 주는 응용 프로그램 반응 시간을 분석하기 위해서는 사용자가 상호작용을 하는 응용 프로그램 수준의 최상위 계층에서부터 데이터를 저장하는 저장장치를 관리하는 최하위 저장장치 계층까지 스토리지 I/O에 관여하는 모든 스토리지 I/O 계층을 수직적으로 아우르는 통합된 분석 도구가 필수적이다. 하지만 기존의 스토리지 I/O 분석 도구들은 커널의 일부 계층들에 집중되어 있기 때문에[4,5], 안드로이드 프레임워크와 커널을 거쳐가는 스토리지 I/O에 대한 통합된 분석에는 한계가 있다.

본 논문에서는 사용자 중심 반응 시간 분석을 위해 사용자와 응용 프로그램의 상호작용에서 시작되는 스토리지 I/O를 최상위 안드로이드 프레임워크부터 리눅스 커널의 저장장치 드라이버까지 스토리지 I/O에 관여하는 모든 스토리지 I/O 계층에서 실시간으로 스토리지 I/O 동작을 관찰 및 분석하는 통합된 스토리지 I/O 분석 도구인 AIOPro를 소개한다. AIOPro는 기존 안드로이드 프레임워크와 커널의 동작을 수정하지 않고 스토리지 I/O 각 계층에서 수집된 정보만을 바탕으로 계층 간 연결고리를 이용하여 각 계층의 스토리지 I/O를 이어서 계층 통합 분석을 가능하게 한다. 또한 AIOPro는 사용자 중심 반응 시간에 영향을 주지 않도록 고안되어 다양한 실험 환경에서 0.1%미만의 무시할만한 동작 부하가 발생하는 것을 확인할 수 있었다.

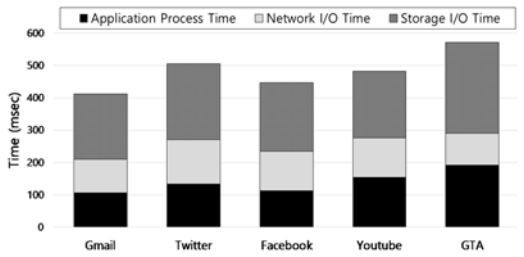

그림 1 응용 프로그램 시작 시 스토리지 I/O 시간 비중
Fig. 1 Storage I/O time during application launching

본 논문의 구성은 다음과 같다. 2장에서는 안드로이드 스마트폰의 스토리지 I/O에 대해 프레임워크와 커널의 스토리지 I/O 계층과 해당 계층에서 분석 가능한 I/O 정보를 소개한다. 3장에서는 제안한 통합된 스토리지 I/O 분석도구인 AIOPro의 구조와 동작 원리에 대하여 설명한다. 4장에서 다른 I/O 분석도구를 사용하여 AIOPro의 동작을 검증한다. 5장에서는 제안한 도구를 사용하여 다양한 시나리오에서 각 스토리지 I/O 계층의 부하를 측정하고, 도구 자체의 동작 부하를 평가한다. 6장에서 기존에 존재하는 스토리지 I/O 분석도구들을 소개하고 7장에서 결론을 맺는다.

2. 안드로이드 스마트폰의 스토리지 I/O

안드로이드 스마트폰의 스토리지 I/O의 부하를 분석하기 위해서는 프레임워크부터 저장장치까지 스토리지 I/O 동작에 관여하는 계층을 분석해야 한다. 안드로이드 플랫폼에서의 스토리지 I/O 계층은 프레임워크, Android Runtime (ART)와 여러 종류의 Library들로 구성된다. Library들 중에서도 SQLite DBMS가 스토리지 I/O에 직접 관여하기 때문에, 스토리지 I/O 계층에 포함되어있다.

리눅스 커널의 스토리지 I/O 계층은 Virtual File System과 파일 시스템들, Block Layer, SCSI Layer, 저장장치 드라이버로 구성되어 있다. 저장장치 드라이버는 저장장치 종류에 따라 MMC, UFS 드라이버 드라이버들로 구성된다.

표 1은 응용 프로그램을 포함한 안드로이드 플랫폼 계층, Virtual File system을 포함한 파일 시스템 계층 그리고 SCSI Layer와 저장장치 드라이버를 포함한 Block Layer 계층에서 분석 가능한 I/O 정보를 나타낸다. 응용 프로그램과 안드로이드 플랫폼 계층에서는 Read/Write 시스템 콜을 호출하여 스토리지 I/O를 수행하는데, 이 때 특정 파일에 접근하기 위해서 FD (file descriptor)를 사용한다. 시스템 콜로 커널 영역으로 진입한 스토리지 I/O는 FD로부터 해당 파일과 관련 메타데이터인 inode, dentry 정보를 통해 해당 파일이 위치한 물리 주소(sector address)를 찾아내 Read/Write를 수행하게 된다.

표 1 I/O 계층 별 분석 가능한 스토리지 I/O 정보
Table 1 Analyzable Storage I/O Information for each storage I/O layers

I/O Information	Application	File System	Block Layer
I/O Type	○	○	○
I/O Size	○	○	△
File Descriptor	○	○	X
Process ID	○	○	△
Sector Address	X	○	○

파일 시스템의 하위 계층인 Block Layer부터는 파일에 대한 정보를 더 이상 유지하지 않는다. 따라서, 시스템 콜로 요청된 스토리지 I/O를 저장장치 드라이버 계층까지 연결 지어 분석하기 위해서는 모든 계층에서 공유할 수 있는 고유한 값의 I/O 정보가 필요하다. 하지만, I/O type을 제외하고는 모든 계층에서 공유하는 고유한 값의 I/O 정보는 존재하지 않는다. I/O size의 경우, 응용 프로그램부터 파일 시스템까지는 byte 단위로 관리되지만, Block Layer부터는 sector 단위(512 byte)로 관리되기 때문에 모든 계층에서 공유하는 고유한 값으로 사용할 수 없다. Process ID 역시 Buffered Write처럼 Page Cache에서 전달된 스토리지 I/O의 경우엔 Process ID가 0으로 초기화되기 때문에, 모든 계층에서 공유하는 고유한 값으로 사용할 수 없다.

3. AIOPro 설계 및 구현

AIOPro는 응용 프로그램 계층부터 파일 시스템 계층까지 FD를 연결고리로 삼아서 스토리지 I/O 정보를 연결하고, 파일 시스템 계층부터 저장장치 드라이버 계층까지 물리 주소로 스토리지 I/O 정보를 연결한다.

최종적으로 모든 스토리지 I/O 계층에서 따로따로 관찰된 스토리지 I/O들을 두 개의 연결고리를 사용하여 전부 연결하는 것이 가능해지기 때문에, 모든 계층에서 얻은 I/O 정보들 중 응용 프로그램에서 유발된 스토리지 I/O를 특정 지어 분석하는 것이 가능해진다.

AIOPro는 특정 I/O에 특정 변수를 추가하여 관련된 모든 함수 및 구조체를 수정하는 방식의 tagging과는 달리, 각 계층간의 연결고리를 이용하여 모든 스토리지 I/O 계층에서 수집된 정보들을 통합 및 연결하여 특정 I/O 정보를 분석하는 점에서 기존의 다른 분석도구들과 차별화된다.

3.1 AIOPro 설계

그림 2는 본 논문에서 제안한 안드로이드 스마트폰을 위한 통합된 스토리지 I/O 분석 도구인 AIOPro의 전체 구조를 나타낸 그림이다. 안드로이드 플랫폼과 리눅스 커널에 구현된 I/O Profiler가 각 계층의 스토리지 I/O 동작에 대한 정보를 수집한 뒤, 이를 메인메모리에 순환

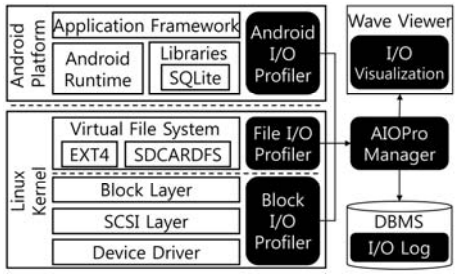

그림 2 AIOPro의 전체 구조
Fig. 2 Overall architecture of AIOPro

표 2 AIOPro가 관찰하는 I/O 계층 별 함수 목록
Table 2 AIOPro's function lists of each I/O layers

I/O Layer	Read	Write
Application	SYSCALL_DEFINE3	SYSCALL_DEFINE3
Virtual F/S	generic_file_aio_read	generic_file_aio_write
Block Layer	submit_bio / blk_done_softirq	
SCSI Layer	scsi_request_fn / scsi_softirq_done	
Device Driver	ufshcd_queuecommand / ufshcd_intr	

버퍼에 저장하여 AIOPro Manager에게 전달한다.

AIOPro Manager는 각 계층의 스토리지 I/O 정보를 연결하여 사용자와 응용 프로그램간의 상호작용으로 발생한 스토리지 I/O를 계층 통합적 분석이 가능하게 한다. 또한, AIOPro Manager는 사용자가 스토리지 I/O에 대한 정보를 다각도에서 분석할 수 있게끔 DB에 입력이 가능한 CSV 파일형식이나 wave viewer를 사용 가능한 waveform 파일형식으로 출력해주는 기능도 제공한다.

I/O Profiler는 모든 스토리지 I/O 계층에 구현되었는데, 그림 2에서는 표 1과 같이 크게 3가지 계층으로 분류하였고, 이는 각 계층에서 얻을 수 있는 스토리지 I/O 정보의 범위에 따라 결정되었다.

Android I/O Profiler 에서는 스토리지 I/O를 유발한 응용 프로그램의 Unique identifier, Process identifier, Thread identifier 정보와 파일 이름, 크기, 오프셋 정보를 저장한다. File I/O Profiler에서는 파일 이름, 크기, 오프셋 외에 파일 시스템의 정보를 활용하여 파일의 물리 주소인 sector address를 저장한다. Block I/O Profiler에서는 파일이 저장되는 블록 저장장치에 관련된 정보와 파일의 물리 주소를 저장하게 된다. 또한, 각 계층에서 스토리지 I/O 동작에 소요되는 시간도 저장한다.

표 2는 AIOPro가 각 스토리지 I/O 계층에서 관찰하는 함수들을 나타낸다. 응용 프로그램 계층부터 파일 시스템 계층까지 Read/Write는 각기 다른 함수가 담당하여 스토리지 I/O를 처리하지만, Block Layer부터는 같은 함수에서 Read/Write 모두가 처리한다. 또한, Block Layer부터는 저장장치에 스토리지 I/O를 전달하는 함수

와 저장장치로부터 스토리지 I/O 처리 종료를 알리는 인터럽트가 전달받아 수행하는 인터럽트 핸들러 함수를 모두 관찰하여 스토리지 I/O 정보를 수집한다.

3.2 AIOPro 구현

AIOPro는 스토리지 I/O 동작에 관여하는 함수들의 파라미터와 파일과 블록 I/O 구조체로부터 스토리지 I/O 정보를 얻는 방식으로 구현되었기 때문에, AIOPro의 동작을 위해서 기존 함수들이나 구조체들의 수정이 필요하지 않다.

AIOPro Manager는 AIOPro의 동작이 안드로이드 스마트폰 성능에 영향을 주지 않도록 각 Profiler로부터 전달받은 각 계층의 스토리지 I/O 정보를 모두 메인메모리에 순환 버퍼 형태로 저장하였다가, 사용자에 의해 AIOPro 동작이 종료되면 사용자가 지정한 파일형식으로 분석된 스토리지 I/O 정보를 저장하게 된다. 멀티코어 CPU 환경에서는 하나 이상의 쓰레드가 동시에 순환 버퍼에 접근하여 I/O 정보를 기록하는 경우가 발생하는데, 순환 버퍼에 Lock을 사용하면 안드로이드 스마트폰 성능 저하가 발생하기에, AIOPro Manager는 CPU 코어 수만큼 별도로 생성해 둔 예비 순환 버퍼에 I/O 정보를 저장하도록 쓰레드들을 유도하여 성능저하를 최소화하였다. 순환 버퍼가 메인메모리에서 사용하는 공간은 최대 50 MB로 다른 응용 프로그램의 동작에 지장을 주지 않을 정도 크기로 할당하고 있다.

AIOPro Manager는 가공된 스토리지 I/O 정보를 DB 형식 혹은 Waveform 형식의 파일 출력을 지원함으로써 사용자의 분석의 용이성을 높이도록 구현되었다. 그림 3은 AIOPro를 이용하여 분석한 스토리지 I/O 정보를 Waveform 파일로 출력하여 GTKWave[6]라는 프로그램을 통해 Visualization한 결과를 나타낸 화면이다. 리눅스 커널의 최상위 계층인 Virtual File System부터 최하위 계층인 저장장치 드라이버까지 시간에 따라 스토리지 I/O가 어떤 계층의 어떤 함수에서 얼마나 동작 부하가 발생하였는지 한눈에 파악이 가능하게 해준다. 이는 스토리지 I/O 분석을 통한 스토리지 I/O 성능 최적화 연구에 AIOPro가 유용하게 사용될 것으로 기대한다.

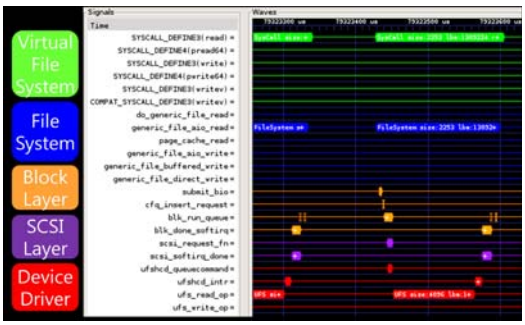

그림 3 AIOPro와 GTKWave를 이용한 I/O 동작 Visualization

Fig. 3 Storage I/O visualization using AIOPro and GTKwave

4. AIOPro 검증

AIOPro의 동작을 검증하기 위해, 기존에 존재하고 널리 쓰이는 2개의 I/O tracer 프로그램인 strace와 blktrace를 사용하였다. strace는 명령어로 설정한 특정 PID(Process ID)에서 발생되는 시스템 콜을 수집한다. blktrace는 Block layer에서의 스토리지 I/O 처리과정을 수집한다. blktrace는 Linux kernel가 지원하는 trace 도구이다. strace는 Linux kernel에는 포함되어있지는 않지만, strace와 blktrace 모두 kernel의 configuration을 수정하면 간단히 실행할 수 있도록 Linux에서 지원하고 있다.

그림 4, 그림 5와 그림 6은 안드로이드 스마트폰으로 사진을 찍으면서 각각 AIOPro, strace와 blktrace로 수집한 로그를 나타낸 화면이다. 그림 5에서 나타나듯이, FD가 55인 파일을 8192 byte 크기만큼 읽으라는 시스템 콜이 호출된 것을 확인할 수 있다. 이는 표 3을 보면, AIOPro는 read 시스템 콜을 400번 함수로 로그를 남기고 있기 때문에, 그림 4에서도 확인할 수 있다. 마찬가지로 blktrace의 결과 화면인 그림 6을 보면 물리주소가 25607336이고 크기가 8 KB(1 sector가 512 byte이기 때문에, 16 sector는 8 KB이다.)인 Read Block I/O가 11.0000045723초에 Queue(Q)된 것을 확인할 수 있다. 또한 해당 주소의 Block I/O가 11.0001181681초에 Complete(C)된 것을 확인할 수 있는데, 이는 표 4에서 따르면, AIOPro는 Block I/O queue가 700번으로 Block I/O complete은 710번으로 로그를 남기고 있고, 그림 4에서 700번과 710번이 같은 시간 차를 두고 발생하는 것을 확인할 수 있다.

```
Start Time    ExecTime UID   PID  TID  Type Func FD Size Address  File Name
133.420376771 2928083  10042 4032 3976 Read 400  55 8192 25607336 20160809_185945.jpg
133.423857979 2125     10042 4032 3976 Read 501  55 8192 25607336 20160809_185945.jpg
133.423356104 66583    10042 4032 3976 Read 700  55 8192 25607336 20160809_185945.jpg
133.424492062 49500    10042 4032 3976 Read 710  55 8192 25607336 20160809_185945.jpg
```

그림 4 AIOPro의 log 예시

Fig. 4 Snapshot of AIOPro's log

```
Start Time       Type FD  Data                    Size    Size
14:47:19.381761 read(55, "\0\0\0\0\0\0"...,8192) = 8192
```

그림 5 strace의 log 예시

Fig. 5 Snapshot of strace's log

```
DEV CPU SEQ Start Time    PID  Event Type Address     Size
8,0 5     1 11.0000042250 4032 A     R    25607336 + 16
8,0 5     2 11.0000045723 4032 Q     R    25607336 + 16
8,0 5    14 11.0001181681 4032 C     R    25607336 + 16
```

그림 6 blktrace의 log 예시

Fig. 6 Snapshot of blktrace's log

표 3 특정 I/O event에 대한 strace와 AIOPro의 표기
Table 3 Predefined symbols of AIOPro and strace for read/write system call

I/O event	strace	AIOPro
Read System Call	read	400
Write System Call	write	410

표 4 특정 I/O event에 대한 blktrace와 AIOPro의 표기
Table 4 Predefined symbols of AIOPro and blktrace for read/write system call

I/O event	blktrace	AIOPro
Block I/O Queue	Q	700
Block I/O Complete	C	710

하지만 그림 4, 그림 5 그리고 그림 6을 보면 로그에 표시된 시작 시간이 다 다른 것을 확인할 수 있는데 이는 3가지 다 수집할 때 판단하는 현재 시각에 대한 기준이 다르기 때문이다. AIOPro는 System이 booting된 순간부터 흘러간 시간을 현재 시각으로 측정하고 있고, strace는 현지 시각을 현재 시각으로 측정하고 있고, blktrace는 blktrace를 수행하기 시작한 순간부터 흘러간 시간을 현재 시각으로 측정하고 있기 때문이다. 따라서 AIOPro를 strace와 blktrace로 검증할 때 시간은 각 스토리지 I/O 간의 상대 시간을 기준으로 검증하였다.

6개의 안드로이드 스마트폰 사용 시나리오를 대상으로 blktrace를 사용하여 AIOPro를 검증을 수행하였다. (각 시나리오의 자세한 정보는 표 5에 나타나있다.) strace는 strace 수행 조건 때문에 파일 내부 복사 시나리오만 대상으로 검증을 진행하였다. 정확한 검증을 위해 AIOPro를 먼저 시작하고, strace와 blktrace를 시작한 뒤, 특정 안드로이드 스마트폰 사용 시나리오를 수행한 뒤, strace와 blktrace를 종료시키고, AIOPro를 가장 나중에 종료시켰다. 6개의 시나리오를 위의 방법으로 수행한 뒤, strace와 blktrace에서 나타난 로그가 AIOPro에 존재하는지 확인하는 간단한 스크립트를 작성하여 분석한 결과, 표 6과 표 7에서 나타나듯이 strace의 모든 read/write 시스템 콜과 blktrace의 모든 block I/O queue/complete가 AIOPro에 하나도 빠짐없이 나타나는 것을 확인할 수 있었다.

표 5 AIOPro 검증을 위한 안드로이드 사용 시나리오
Table 5 Scenarios used for AIOPro verification

Scenario Name	Scenario Explanation
Application Launch	Launching 3D game application
Web Browser	Loading multiple web pages
Camera	Camera burst shot (30 photos)
Download	Downloading 1 GB file
Internal File Copy	1 GB file internal copy
USB File Copy	1 GB file USB copy

표 6 strace와 AIOPro 로그 비교
Table 6 A comparison between strace and AIOPro

Scenario	strace read	AIOPro read	strace write	AIOPro write
Internal File Copy	16048	16048	16032	16032

표 7 blktrace와 AIOPro 로그 비교
Table 7 A comparison between blktrace and AIOPro

Scenario	blktrace queue	AIOPro queue	blktrace complete	AIOPro complete
Application Launch	225166	225166	219658	219658
Web Browser	5108	5108	4795	4795
Camera	19374	19374	18759	18759
Download	15774	15774	10124	10124
Internal File Copy	6567	6567	6434	6434
USB File Copy	4083	4083	3597	3597

5. 실험 결과

제안한 분석 도구의 정확성을 판단하기 위해 본 논문에서는 안드로이드 프레임워크 버전 5.0과 리눅스 커널 버전 3.10.61에 AIOPro를 구현하여 해당 버전의 프레임워크와 커널을 지원하는 안드로이드 스마트폰 에뮬레이션 보드에서 AIOPro의 계층별 스토리지 I/O 동작 부하를 측정하고 AIOPro 자체 동작 부하를 검증하는 실험을 진행하였다.

안드로이드 스마트폰의 I/O 계층별 부하를 측정하기 위해 어플리케이션을 실행하는 시나리오, 카메라 연속촬영 시나리오, 외부에서 USB를 통해 스마트폰 저장장치로 파일을 복사하는 시나리오와 스마트폰 저장장치의 파일을 다른 디렉토리에 복사하는 시나리오를 수행하면서 AIOPro를 통해 계층별 스토리지 I/O 동작 부하를

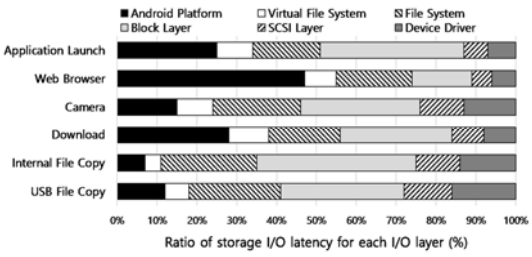

그림 7 AIOPro로 측정한 각 시나리오 별 계층별 스토리지 I/O 동작 부하
Fig. 7 Measured operation latencies for each storage I/O layer using AIOPro

측정하였다. 그림 7은 계층별 스토리지 I/O 동작 부하를 측정한 결과이며, 대부분의 경우 파일 시스템과 Block Layer에서의 부하가 평균 50% 이상을 차지하는 것을 확인할 수 있었다. 또한, 어플리케이션 실행 시나리오에서는 어플리케이션이 최대 27%까지 동작 부하를 차지하는 것을 확인할 수 있었다.

AIOPro 자체의 동작 부하를 검증하기 위해 Andro-bench, Antutu, Storebench 벤치마크 프로그램들에 대해서 AIOPro를 사용하는 경우와 사용하지 않는 경우에 수행 시간을 측정하여 비교하는 실험을 진행하였다. 실험 결과, AIOPro를 사용하는 경우와 사용하지 않는 경우의 수행 시간 차이가 평균 0.1% 미만으로 매우 적은 동작 부하를 발생하는 것을 확인하였다.

6. 관련 연구

리눅스 커널의 스토리지 I/O를 분석하는 도구는 Virtual File System으로 인가되는 시스템 콜을 분석하는 strace[4], 그리고 Block Layer 내부의 스토리지 I/O를 분석하는 blktrace[5]가 존재한다. 그러나 이 두 도구를 모두 사용한다고 해도 사용자와 응용 프로그램의 상호작용에서 시작된 스토리지 I/O가 무엇인지 커널 수준의 정보로는 파악이 불가능한 한계가 존재한다. 또한, 두 도구의 정보를 바탕으로 스토리지 I/O를 연결하는 작업이 불가능한데, 시스템 콜의 정보에는 파일에 대한 정보가 있지만, Block Layer에는 파일에 대한 정보가 존재하지 않기 때문이다.

안드로이드 스마트폰 환경에서 이러한 한계를 극복하기 위해 Androtrace[7]가 기존에 제안되었다. Androtrace는 파일의 이름을 바탕으로 커널 수준에서 SQLite의 스토리지 I/O를 파악하여 Block Layer까지 분석하는 도구이다. 이는 기존 strace와 blktrace의 한계를 넘어서는 발전된 스토리지 I/O 분석도구이지만, 여전히 사용자와 직접 상호작용하는 응용 프로그램 수준의 스토리지 I/O 정보를 파악하기 위해서는 안드로이드 프레임워크의 분석이 추가로 필요하다. 또한, 데이터의 입출력을 수행하는 저장장치까지 스토리지 I/O가 거치는 계층 중 SCSI Layer와 저장장치 드라이버를 분석하고 있지 않기 때문에, 해당 계층의 스토리지 I/O 분석은 불가능하다는 한계가 존재한다. 반면, AIOPro는 최상위 계층인 프레임워크부터 최하위 계층인 저장장치 드라이버까지 스토리지 I/O 정보를 분석함으로써 기존 분석도구 대비 차별화되는 장점을 보인다.

7. 결 론

본 논문에서는 안드로이드 스마트폰의 사용자 경험에 큰 영향을 주는 응용 프로그램의 반응 시간 중 스토리지 I/O의 부하를 정밀 분석하는 통합된 스토리지 I/O 분석 도구인 AIOPro를 제안하였다. 개발된 도구는 응용 프로그램으로부터 발생된 스토리지 I/O를 프레임워크부터 저장장치 드라이버까지 모든 소프트웨어 계층 별로 스토리지 I/O 정보를 수집 및 연결하여 각 계층별 부하에 대한 계층 통합적 분석을 가능하게 해준다. 본 도구를 활용하면 안드로이드 스마트폰에서의 스토리지 I/O 성능 최적화 연구에 많은 기여가 있을 것으로 예상된다.

References

[1] W. Song et al., "Reducing Energy Consumption of Smartphones Using User-Perceived Response Time Analysis," *Proc. International Workshop on Mobile Computing Systems and Applications*, No. 20, 2014.

[2] H. Kim et al., "Revisiting Storage for Smartphones," *Proc. USENIX Conference on File and Storage Technologies*, No. 14, 2012.

[3] D.T. Nguyen et al., "Improving Smartphone Responsiveness through I/O Optimizations," *Proc. ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 337-342, 2014.

[4] strace. [Online]. Available: http://linux.die.net/man/1/strace.

[5] blktrace. [Online]. Available: http://linux.die.net/man/8/blktrace.

[6] GTKWave. [Online]. Available: http://gtkwave.sourceforge.net.

[7] E. Lim et al., "Androtrace: framework for tracing and analyzing IOs on Android," *Proc. Workshop on Interactions of NVM/FLASH with Operating Systems and Workloads*, No. 3, 2015.

한 상 욱

2011년 KAIST 전산학 학사. 2013년 서울대학교 컴퓨터공학부 석사. 2013년~현재 서울대학교 컴퓨터공학부 박사과정 관심분야는 플래시 저장장치, 임베디드 소프트웨어, 모바일 시스템, 운영체제

이 인 혁

2015년 한양대학교 컴퓨터공학부 학사
2017년 서울대학교 컴퓨터공학부 석사
관심분야는 임베디드 소프트웨어, 모바일 시스템, 운영체제

류 동 욱

2012년 성균관대학교 DMC공학과 석사 2012년~현재 삼성전자 소프트웨어연구소 책임 연구원. 관심분야는 임베디드 소프트웨어, 모바일 시스템, 저전력 시스템

김 지 홍

1986년 서울대학교 계산통계학과 학사 1988년 University of Washington 컴퓨터과학과 석사. 1995년 University of Washington 컴퓨터과학 및 공학과 박사. 1995년~1997년 미국 Texas Instruments 선임연구원. 1997년~현재 서울대학교 컴퓨터공학과 교수. 관심분야는 플래시 저장장치, 저전력 시스템, 임베디드 소프트웨어, 컴퓨터 구조