MOST Viewer: MOST(MObile STorage analyzer) 기반의 트레이스 시각화 도구

최경열○ 원유집
한양대학교 컴퓨터·소프트웨어학과
chl4651@hanyang.ac.kr youjip.won@gmail.com

MOST Viewer: Visualization Tool for the Trace based on MOST(MObile STorage analyzer)

Gyeongyeol Choi○ Youjip Won
Department of Computer and Software, Hanyang University

요 약

MObile STorage analyzer(이하 MOST)는 안드로이드 OS 스마트폰을 대상으로 하는 입출력 분석 소프트웨어이다. MOST는 어플리케이션에서 발생한 IO 워크로드를 블록 레이어 수준에서 추출하여 분석하며 blktrace에서 제공하는 IO trace를 기반으로 분석 과정을 통해 얻어낸 추가적인 특성들을 추가하여 결과 파일로 출력한다. 하지만 해당 결과 파일은 발생한 IO들을 개별적으로 나열할 뿐이므로 IO 워크로드의 전체적인 특성을 파악하기에 적절한 형태가 아니다. 본 논문에서는 MOST툴의 기반의 트레이스를 분석하여 그래프로 작성해주는 MOST Viewer를 개발함으로써 MOST를 통해 얻어낸 IO 트레이스를 시각화해서 파악할 수 있도록 하였다. 또한 MOST Viewer는 IO 특성을 기반으로 하는 다양한 종류의 그래프들을 옵션에 따라 선택할 수 있게 설계되었다.

1. 서 론

안드로이드 기반의 스마트폰에서 동작하는 모든 어플리케이션들은 각각의 사용목적에 따른 개별적인 IO 워크로드를 발생시킨다. 어플리케이션에서 발생한 IO 워크로드는 실제 블록 장치로 접근하기 위해 파일 시스템에서 블록 레이어를 경유하게 되는데 이 과정에서 IO는 블록 연산에 필요한 정보를 포함하는 Block IO 구조체로 변환된다. MOST (MObile STorage Analyzer)[1]는 IO 워크로드를 블록 레이어 수준에서 추출한 IO trace를 분석하기 위해 개발되었다. MOST는 blktrace 유틸리티를 통해 출력된 block trace의 LBA (Logical Block Address) 정보를 기반으로 리버스-매핑을 수행함으로써 해당 IO의 블록 타입, 파일 타입 등 유용한 특성들을 추가하여 Output Trace를 생성하게 된다. 그러나 MOST의 Output Trace는 추출한 IO를 개별적으로 나열한 미가공 데이터이기 때문에 발생한 IO 워크로드의 특성을 파악하기에 적합한 형태가 아니다.

본 논문에서는 MOST의 Output Trace를 시각화하기 위한 도구로써 MOST Viewer[1]를 개발하였다. MOST Viewer는 Output Trace를 가공하여 IO 워크로드의 특성을 판단할 수 있는 지표 데이터로 변환한다. 또한 MOST Viewer는 이 지표 데이터를 이용하여 MOST에 제공하는 IO 특성에 대한 그래프를 자동으로 작성해준다.

2. 구 현

2-1. 요구 사항

MOST Viewer가 Output Trace를 처리하여 그래프를 작성하는 작업을 수행하기 위해서 두 종류의 프로그래밍 언어가 사용된다.

1) Python[2]

Python은 MOST의 Output Trace를 읽어 들여 R에서 그래프를 작성하는데 필요한 입력 데이터를 생성하는 역할을 수행한다. 또한 MOST Viewer의 인터페이스를 담당한다.

2) R (Programming Language)[3]

R은 그래픽 및 통계 처리를 목적으로 R 재단에서 개발한 프로그래밍 언어다. R은 Python을 통해 수정된 데이터를 읽어 들여 이를 그래프로 출력해주는 과정을 수행한다. 또한 MOST Viewer에서 R은 단순히 가공된 데이터를 그래프로 작성할 뿐 아니라 MOST의 Output Trace를 직접 가공하여 그래프를 작성해주는 역할을 수행한다.

2-2. 동작 과정

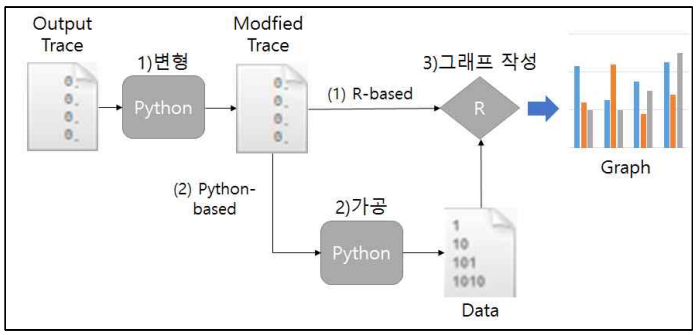


그림 1 MOST Viewer의 Process Flow

1) Link: https://github.com/ESOS-Lab/MOST

MOST Viewer가 Output Trace를 그래프의 형태로 처리하는 과정의 동작 프로세스는 크게 3 부분으로 분류할 수 있다.

1) 변형 (Modification)

MOST의 Output Trace는 IO를 Block IO의 처리 단위로 표시하는데 IO의 Block Type에 따라 표시되는 IO Trace의 형태에 차이가 있다. 변형 과정에서는 각 Block Type에 따른 IO Trace의 형태 차이를 일치시킴으로써 Trace를 R에서 읽어 들일 수 있는 형태로 변경한다.

2) 가공 (Parsing)

MOST Viewer의 그래프는 R에서 인자값으로 받아들이는 입력 데이터의 형태에 따라 R-based, Python-based로 분류된다. 이 중 Python-based형 그래프는 추가적으로 트레이스를 '가공'하여 R에서 필요로 하는 데이터를 생성한다.

3) 그래프 작성 (Drawing)

R-based형 그래프는 수정된 Output Trace를, Python-based형 그래프는 추가적으로 트레이스를 가공하여 생성한 데이터를 가지고 그래프를 작성하게 된다.

2-3. 세부 처리

MOST의 Output Trace에서 Raw값은 각각의 IO를 나타내고 Column값은 해당 IO의 특성을 나타낸다. MOST가 나타내고 있는 IO 특성은 총 9가지로 다음과 같다.

Col no.	IO 특성	Col no.	IO 특성	Col no.	IO 특성
1	Time	4	Sector Number	7	Process Name
2	Action	5	Sector Size	8	Block Type
3	RWBS	6	Process ID	9	File Name

표 1 MOST Output Trace의 IO 특성

MOST Viewer는 위 양식의 Output Trace 파일을 input 파일로 받아들여 그래프 작성을 위한 분석 작업을 수행한다. 표현 가능한 그래프 양식은 MOST Viewer 내 R을 기준으로 해당 그래프가 MOST의 Output Trace 기반으로 작성되었는지, 혹은 추가적인 가공 과정을 통해 생성한 데이터 파일을 통해 작성되었는지에 따라 R-based / Python-based 형으로 1차적으로 분류되며 각 그래프 종류에서 표현 가능한 그래프 양식은 다음과 같다.

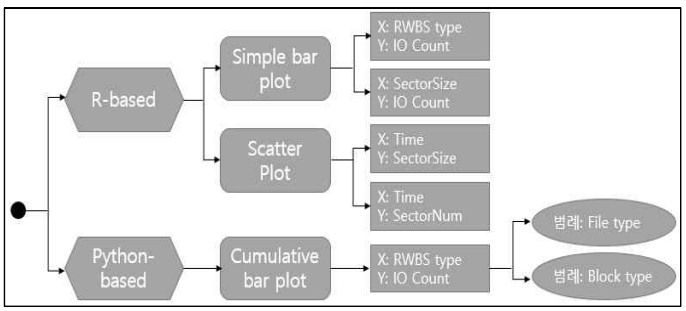


그림 2 MOST Viewer 그래프 분류

1) R-based Graph

R-based형 그래프는 수정된 Output Trace를 R에서 테이블 형태로 읽어 들인다.

- 단순 막대그래프 (Simple Bar Plot): 특정 IO 특성을 지정된 Column 번호를 통해 지시한 다음 이를 각 그래프의 분류된 X축 값과 비교한다. 이를 통해 해당 조건과 일치하는 줄(Row)의 개수를 카운트하고 변수값으로 저장한 다음 그래프를 작성한다.
- 분산형 그래프 (Scatter Plot): Output Trace 내 Time 속성과 <Sector Number 혹은 Sector Size> 속성을 추출하여 (X, Y) 좌표 내 점 형식으로 표시하게 된다.

2) Python-based Graph

Python-based형 그래프는 Trace를 가공하여 얻은 데이터를 이용하여 작성된 누적 막대그래프 (Cumulative Bar Plot)이다. 그래프의 형태는 크게 범례 (Block Type / File Type)에 따라 구분된다.

(a) Block Type

MOST Output Trace에서 Block type은 D (Data), M (Metadata), J (Journal)로 분류된다. MOST Viewer는 Trace에서 각 Block type을 가지는 IO를 Count하여 데이터 파일로 저장한다. 이후 R에서 해당 데이터 파일을 통해 그래프를 작성한다.

(b) File Type

File type도 앞서 설명한 Block type과 동일한 방식으로 트레이스를 가공한다. File type은 해당 IO가 수행된 파일 정보를 특성에 따라 분류한 것으로 MOST Output Trace의 9번째 Column - File name값을 읽어 들여 파일 이름 내 확장자를 통해 분류 작업을 수행한다.

MOST Viewer에서는 확장자를 통한 File Type 분류 과정을 유동성 있게 처리할 수 있게 하였다. MOST Viewer 내 'file_info.txt'에 사용자가 직접 접근하여 정의하고자하는 파일 타입과 확장자 정보를 입력하면 MOST Viewer는 'file_info.txt' 파일을 읽어들여 추가한 파일 타입도 포함하여 분류 작업을 수행한다.

3. 실 험

MOST를 이용해 출력한 Output Trace를 대상으로 실제 작성 가능한 그래프를 리스트별로 출력하였다.

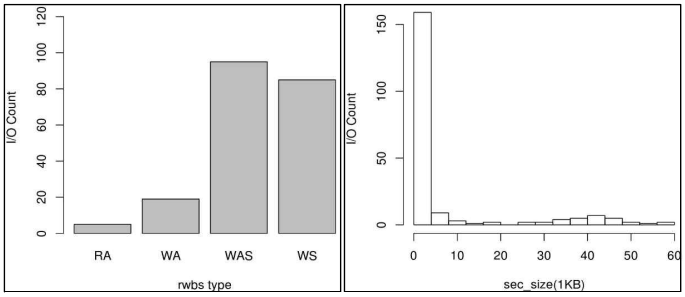
3-1. 실험 환경

MOST Viewer에서 사용할 Output Trace 생성을 위해 안드로이드 스마트폰 NEXUS5 (Android v.4.4, Linux 3.4.0-gadb2201)를 대상으로 MOST 툴을 수행하였다. 실험 워크로드는 7초간 기기 내에서 Google Chrome 어플리케이션을 수행시켰다.

해당 Output Trace를 이용하여 MOST Viewer를 수행한 HOST OS는 Linux 3.13.0이며 MOST Viewer 수행 과정에서 사용된 Programming Language 버전은 Python: 2.7.3, R: 3.2.0이다.

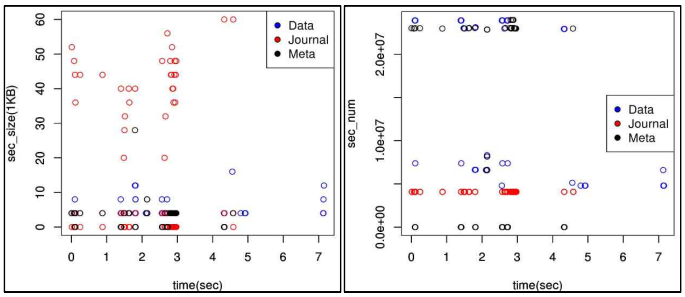
3-2. 실험 결과

1) R-based형 그래프



(a) RWBS type (b) Sector Size

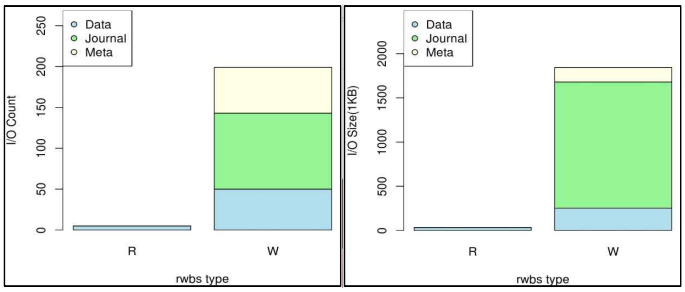
그림 3 단순 막대그래프



(a) Sector Size (b) Sector Number

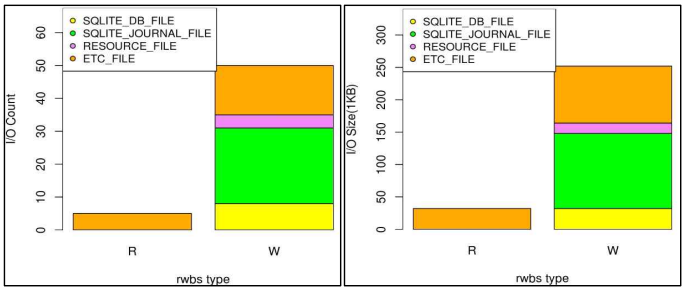
그림 4 분산형 그래프

2) Python-based형 그래프



(a) IO Count (b) Sector Size

그림 5 누적 막대그래프 - Block Type



(a) IO Count (b) Sector Size

그림 6 누적 막대그래프 - File Type

4. 결 론

본 논문에서는 MOST의 Output Trace에 대한 IO 워크로드 분석을 수행하고 이를 시각화하기 위한 도구로 MOST Viewer를 개발하였고 이를 통해 MOST의 Output Trace를 그래프로 변환하여 검증하였다. MOST Viewer의 구성 요소 및 동작 프로세스, 세부 처리 과정을 기술함으로써 MOST Output Trace가 시각화된 그래프 데이터로 처리되는 과정을 기술하였다. 또한 해당 툴이 MOST Output Trace를 통해서 나타낼 수 있는 다양한 IO 특성들을 제시함으로써 I/O 및 파일 시스템 연구에 대한 MOST Viewer의 유용성을 나타냈다.

5. 사 사

본 연구는 지식경제부 및 한국산업기술평가관리원의 산업원천 기술개발사업(정보통신)의 일환으로 수행하였음. [No.10041608, 차세대 메모리 기반의 스마트 디바이스용 임베디드 시스템 소프트웨어]

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 대학ICT연구센터 육성지원사업의 연구결과로 수행되었음 (IITP-2016- H8501-16-1006)

참고문헌

[1] Sooman Jeong, Kisung Lee, Jungwoo Hwang, Seongjin Lee, Youjip Won, “AndroStep: Android Storage Performance Analysis Tool”, Lecture Notes in Informatics, Vol. 215, Stefan Kowalewski and Bernhard pe(Hrsg.) (1st European Workshop on Mobile Engineering), Feb. 26-Mar 1, 2013, Aachen, Germany)

[2] Lutz, Mark, “Learning python”, O’Reilly Media, Inc., 2013.

[3] Venables, William N., David M. Smith, and R Development Core Team. “An introduction to R.” 2004.