멀티파일 워크로드의 생성

김 선 두 ○ 원 유 집

한양대학교

sksioi12@hanyang.ac.kr, yjwon@hanyang.ac.kr

A workload Generation Method for Multiple files

Sundoo Kim ○ Youjip Won

Hanyang University

요 약

SQLite는 Android, iPhone 그리고 Tizen에 탑재된 default DBMS이다. 우리는 실제 mobile 환경에서 발생하는 IO trace을 수집하고 분석했다. Real IO trace를 분석해 본 결과 다수의 multiple file transaction을 수행하는 application들이 존재하는 것을 식별했다. 하지만 현재 상용화된 마이크로 벤치마크 도구에서는 multiple file transaction을 모사할 수 없는 문제가 존재한다. 우리는 마이크로 벤치마크 도구에 multi file transaction을 지원하는 기능을 추가해 실제 IO trace를 유사하게 모사하도록 변경했다.

1. 서 론

SQLite는 Android, iPhone 그리고 Tizen에 탑재된 default DBMS이다. 최근에, 사물인터넷을 비롯하여 자율주행 자동차까지 다양한 영역에서 SQLite가 사용된다. SQLite 같은 경우 database를 file 단위로 관리한다. 많은 데이터가 발생하게 될 경우 큰 IO traffic이 발생된다. 따라서, 대부분의 mobile device vender들은 개발 후 SQLite의 성능 측정을 위해 실제상황을 모사하고 있는 마이크로 벤치마크 도구들을 이용한다. 실제로 발생하는 동작을 모사해 많은 SQLite IO가 유발되는 경우 다양한 configuration을 이용해 SQLite의 성능을 조절한다. 마이크로 벤치마크 도구들은 실제상황 에서 발생하는 IO pattern들을 정확하게 모사하는 것이 중요하다. 하지만 실제 IO trace를 분석해 본 결과 현존하는 마이크로 벤치마크 도구들은 multiple file transaction 동작을 모사하지 못하는 문제가 존재한다. 우리는 실제 IO trace를 분석해 본 결과 multiple file transaction이 다양한 App에서 발행되며 하루에 최대 575번까지 발생하는 것을 식별했다. 우리는 분석된 결과를 바탕으로 실제상황과 좀 더 유사하게 동작하기 위해 multiple file transaction을 지원하도록 마이크로 벤치마크를 변경해 성능을 측정을 실험했다.

2. 배 경

2.1 SQLite

SQLite는 오늘날 널리 사용되고 있는 DBMS 이다. SQLite는 스마트 폰, 분산처리 파일 시스템, 웨어러블 디바이스 그리고 자율주행차 등 거의 모든 컴퓨팅 플랫폼에서 사용된다. 최근에 모바일 어플리케이션 이나 PC 어플리케이션의 기능이 다양해지고 복잡도가 증가하면 서 가볍고, 빠르며, 이식성이 좋은 임베디드 DBMS인 SQLite의 사용률이 크게 증가하였다. 특히, 오늘날의 스마트폰 대명사인 Android, iPhone 그리고 Tizen 에 default DBMS로 적용되어 널리 쓰이고있다. 스마트폰의 운영체제에서는 데이터를 저장하고 관리하기 위해 SQLite를 빈번하게 사용하고 있으며, IO traffic에 큰 부분을 차지한다[1].

SQLite는 Rollback journal mode (Delete, Truncate, Persist) 3개, Rollforward journal mode, Memory mode, 그리고 Non journal mode 총 6개를 가지고 있다. Rollback journal mode 같은 경우 database file과는 별개로 journal file을 두고 변경되지 않은 데이터를 journal file 에 적은 후 실제 데이터를 database file에 작성하는 것을 말한다. 만약 crash가 발생하면 journal file을 참조해 이전에 저장해 둔 데이터를 crash가 발생한 database file로 복원해 일관성을 유지한다. Journal file을 관리하는 방법에서 여러 개의 모드로 나뉘게 된다. Delete mode는 트랜잭션이 완료 후 파일을 지운다. Truncate mode는 완료 후 file size를 0으로 절삭한다. Persist 모드는 완료 후 file을 지우지 않고 Journal header만 0으로 초기화 하여 file을 유지한다. File을 overwriting 한다면 추가적인 file system I/O가 유발되지 않아 3가지 journal mode 중 가장 효율적이다. Rollforward journal mode 같은 경우 database file과는 별개로 WAL file을 두고 변경된 페이지들을 먼저 WAL file에 기록한다. 일정 threshold (default 1,000 pages)[1]가 넘어가게 되면 checkpoint 동작을 이용해 실제 database file에 작성한다. 만약 crash가 발생하게 되면 WAL file을 순회하며 crash가 발생한 지점을 찾은 후 그 지점부터 overwrite하여 다시 트랜잭션을 수행한다. WAL journal mode 같은 경우 최근 android platform 에서 default mode로 설정됐다.

2.2 Multiple file transaction of SQLite

SQLite 는 DB file 의 저장 및 이동 공간의 용의성을 위해서 DETACH / ATTACH 기능을 지원한다[2]. DETACH function을 통해 SQLite 는 connect 된 database file 의 일부 table을 별도의 database file로 분리해서 추출 할 수 있다. 또한, ATTACH function을 통해 SQLite는 다수의 database file들을 하나의 datafile file로 묶어 다수의 file들에 대해 접근하여 데이터를 조작할 수 있다. 이와 같이, SQLite에서 ATTACH function을 통해 다수의 트랜잭션을 접근하는 동작을 multiple file transaction이라고 정의한다[2]. SQLite에서는 Rollback journal mode를 제외한 나머지 mode들에 대해서는 multiple file transaction을 지원하지 않는다. Rollback journal mode에서 SQLite는 multiple file transaction을 지원하기 위해 master journal file을 사용한다. Multiple file transaction이 발생했을 때, transaction atomicity를 보장하기 위해 master journal file에 각 database file 마다 생성된 journal file들의 경로를 string 형태로 저장한다. 이후 journal header 에 master journal file의 full path 명을 저장한다. 만약 crash가 발생하게 된다면, SQLite는 master journal file을 열고 journal file의 경로를 통해 해당하는 journal file들을 읽는다. 이후 journal file 에 해당하는 database file들을 journal file들 내 존재하는 페이지를 복원시켜 일관성을 유지한다. Database file에 발생된 transaction이 모든 database file에 적용되지 않는 이상, 변경사항은 하나도 적용되지 않는다. 따라서, multiple file transaction은 atomicity를 보장 할 수 있다.

3. 분 석

3.1 트레이스 수집방법

	Collection Period (Month)
User 1	2015.07.10. ~ 2015.10.14.
User 2	2015.07.10. ~ 2016.05.06.
User 3	2015.10.05. ~ 2016.11.07.
User 4	2015.10.22. ~ 2017.04.07.

표 1. User 4명에 대한 IO trace 수집 기간

우리는 IO pattern 분석하기 위해, 4명의 User로부터 약 2년간 IO trace를 수집했다. 표 1은 우리가 수집한 IO trace의 기간을 설명한다. 우리는 IO trace를 모으기 위해 Androtrace [3]를 이용한다. Androtrace 는 Android 기반의 mobile device에서 IO trace를 추출해 수집하는 도구이다. Androtrace는 수집을 위한 client 부분과 저장을 위한 back end 부분 두 가지로 구성된다. 구체적으로, Client 부분은 수정된 linux kernel 과 추출된 데이터를 서버에 전송하는 전송 모듈로 나뉜다. 수정된 linux kernel 같은 경우 linux에서 사용되는 모든 trace를 수집해 disk 동기화한다. 전송 모듈은 disk에 동기화된 데이터를 back end server로 전송한다. Androtrace 는 user의 개입 없이 IO trace를 수집할 수 있다는 수집할 수 있는 장점이 있다.

3.2 실제 트레이스 결과

우리는 Androtrace로 추출한 IO trace를 기반으로 실제 mobile device에서 multiple file transaction이 얼마나 발생하는지 조사했다. Multiple file transaction 은 crash consistency를 위해 master journal file을 사용한다. Master journal file은 우리가 접근하는 database file의 이름에 *-mj 형식을 추가한 형태로 지정된다. 따라서, 우리는 추출한 IO trace에서 master journal file 이름을 구별해 실제 기기에서 발생한 multiple file transaction의 빈도 수를 분석했다.

표 2. User 별 하루에 발생하는 master journal file의 호출 횟수를 Quantile statistics을 나타낸 것이다. User별로 크게 차이가 난다. User1같은 경우 4명의 User 중에서 평균적으로 multiple file transaction을 가장 많이 호출한다. 하지만 User1의 수집기간이 짧은 영향도 있다. 수집기간이 가장 긴 User는 User4이다. User4 같은 경우 약 2년간 수집 되었으며, 하루 최대 575번, 평균 44번의 multiple file transaction이 수행되었다. 우리는 실제 발생한 IO trace를 통해 수집된 master journal file 이름들을 기반으로 multiple file transaction을 수행하는 어플리케이션을 조사했다. 수집된 master journal file의 이름들은 contact2.db-mj, EmailProvider.db-mj, History.db-mj, KakaoTalck.db-mj 등이 존재한다. Master journal file을 기반으로 각 database file의 이름을 조사한 후 multiple file transaction을 사용하는 어플리케이션을 역으로 조사했다. 웹 브라우저, 메일 클라이언트, 전화번호부, SNS 등의 어플리케이션이 multiple file transaction을 사용하는 것을 식별했다.

	Avg,	50%	75%	MAX
User 1	118.8	117.5	148.3	296.0
User 2	59.3	51.0	70.0	353.0
User 3	27.0	25.0	37.0	271.0
User 4	43.9	23.0	62.3	575.0

표 2. User 별 하루 동안 multiple file transaction 발생 횟수

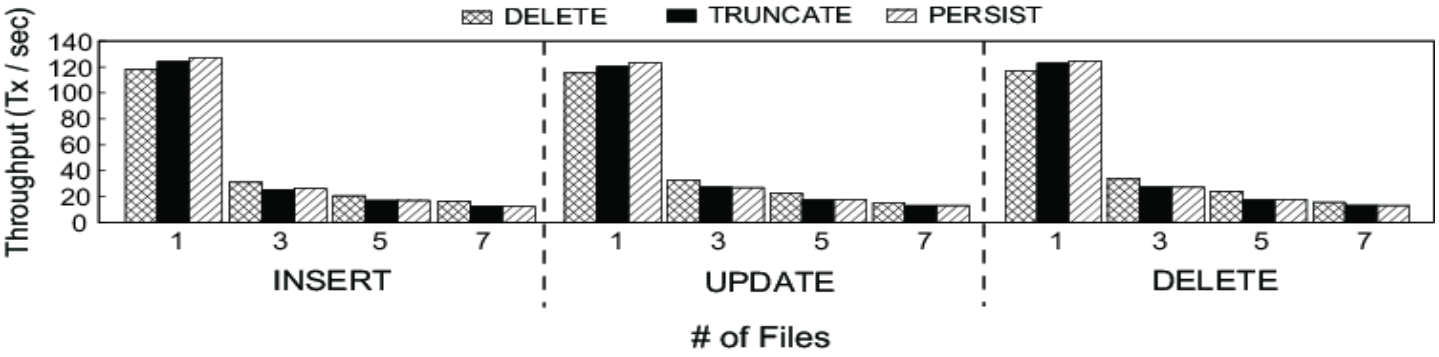

그림 1 multiple file transaction 성능 (INSERT, UPDATE, DELETE)

3.3 마이크로 벤치마크

우리는 현재 배포된 마이크로 벤치마크에서 multiple file transaction을 지원하는지 조사했다. RLbench, Androbench[4], Mobibench[5]를 조사해 본 결과 multiple file transaction을 지원해주는 벤치마크 도구가 없다는 것을 확인했다. 우리는 SQLite 의 multiple file transaction의 사용 유무를 분석한 내용의 부재로 인해 마이크로 벤치마크 도구에서 지원을 하지 않은 것이라고 생각한다. 따라서, 우리는 수집한 IO trace 분석한 결과를 바탕으로 실제 상황과 좀 더 유사하게 동작하도록 마이크로 벤치마크를 수정한다. 우리는 마이크로 벤치마크 중 Mobibench를 선정한다. Mobibench 같은 경우 mobile 내에서 SQLite 동작을 수행해 벤치마킹할 수 있는 open source이다. 벤치마크 중 가장 최근까지 소스코드 관리가 되고 있다. 우리는 인자로 file 개수를 전달받고 SQLite에서 제공하는 ATTACH / DETACH 동작을 이용해 multiple file transacton을 수행하는 코드를 Mobibench에 추가했다. File 개수의 기본 값은 하나로 설정했다.

4. 실 험

우리는 Galaxy S7에 수정된 Mobibench를 이용해 multiple file transaction의 성능을 측정하는 실험을 했다. 우리는 SQLite journal mode 중 multiple file transaction을 수행할 수 있는 Rollback journal mode 3가지를 이용해 실험했다. 테이블의 개수는 1개로 하였으며, 일관성을 위해 full sync 모드로 실험을 진행했다. Multiple file transaction을 발생시키기 위해 Mobibench를 수행할 때, 파일 개수를 1,3,5,7개를 늘려 실험을 진행했다. 총 10,000번의 트랜잭션을 수행한다. 그림 1. 은 multiple file transaction의 Insert, Update, Delete operation의 성능을 각각 나타낸다. 그림 1.은 multiple file transaction 성능을 보여준다. Persist journal mode 에서 single file transaction일 경우 성능이 약 130 transaction / sec를 출력한다. 하지만 file 3개에 대해 multiple file transaction을 수행할 경우 약 4배정도 감소된다. File이 늘어나면 늘어날수록 성능이 줄어든다. 우리는 실험 결과를 통해 single file transaction보다 multiple file transaction의 성능이 훨씬 좋지 않은 것을 알 수 있다. 향후 우리가 얻은 실험결과를 바탕으로 비효율 의 원인을 조사할 것이다.

5. 결 론

우리는 real IO trace를 통해 multiple file transaction 의 실제 디바이스에서 존재 유무를 파악했다. 현재 마이크로 벤치마크에서는 multiple file transaction을 지원하지 않는다. 우리는 마이크로 벤치마크를 수정해 좀 더 실제상황을 모사했다.

6. 사 사

본 연구는 한국연구재단 기초연구실 지원사업(No. 2017R1A4A1015498), IITP 전문연구실 지원사업(2018-0-00549) 그리고 미래창조과학부 및 정보통신기술진흥센터의 정보통신•방송 연구개발사업[R0601-16-1063, ICT 장비용 SW 플랫폼 구축] 의 지원을 받아 수행되었음.

참 고 문 헌

[1] Jeong, Sooman, et al. "I/O Stack Optimization for Smartphones." USENIX Annual Technical Conference. 2013.

[2] Owens, Mike, and Grant Allen. SQLite. Apress LP, 2010.

[3] Lim, Eunryoung, et al. "Androtrace: framework for tracing and analyzing IOs on Android." In Proc. Of INFLOW, ACM, 2015.

[4] Je-Min Kim, et al. Androbench: Benchmarking the storage performance of android-based mobile devices. In Frontiers in Computer Education, pages 667–674. Springer, 2012.

[5] Sooman Jeong, et al. Androstep: Android storage performance analysis tool. In Software Engineering (Workshops), volume 13, pages 327–340, 2013.