KIPS Trans. Comp. and Comm. Sys.
Vol.11, No.12 pp.429~436
ISSN: 2287-5891 (Print), ISSN 2734-049X (Online)

https://doi.org/10.3745/KTCCS.2022.11.12.429

Block Allocation Method for Efficiently Managing Temporary Files of Hash Joins on SSDs

Joontae Kim† · Sangwon Lee††

ABSTRACT

Temporary files are generated when the Hash Join is performed on tables larger than the memory. During the join process, each temporary file is deleted sequentially after it completes the I/O operations. This paper reveals for that the fallocate system call and file deletion-related trim options significantly impact the hash join performance when temporary files are managed on SSDs rather than hard disks. The experiment was conducted on various commercial and research SSDs using PostgreSQL, a representative open-source database. We find that it is possible to improve the join performance up to 3 to 5 times compared to the default combination depending on whether fallocate and trim options are used for temporary files. In addition, we investigate the write amplification and trim command overhead in the SSD according to the combination of the two options for temporary files.

Keywords : Hash Join, Temporary File, SSD, Trim, fallocate

SSD상에서 해시조인 임시 파일의 효과적인 관리를 위한 블록 할당 방법

김 준 태† · 이 상 원††

요 약

메모리보다 큰 대용량 테이블들에 대해 해시조인(Hash Join) 수행 시 임시 파일들을 생성해서 조인 과정에서 발생하는 임시 데이터 I/O를 수행하고 조인 종료 시에 그 파일들을 삭제한다. 본 논문에서는 해시조인용 임시 파일들을 하드디스크가 아닌 SSD상에서 관리할 때, 파일 생성 시 fallocate 시스템 콜 및 파일 삭제 관련 trim 옵션이 해시조인 성능에 큰 영향을 미치는 점을 밝힌다. 구체적으로 대표적인 오픈소스 데이터베이스인 PostgreSQL을 이용해서 다양한 상용 및 연구용 SSD 상에서 해시조인 수행 시, 임시 파일들에 대한 fallocate 및 trim 옵션 사용 여부에 따라 디폴트 조합에 비해 최대 약 3~5배 조인 성능 향상이 가능함을 보인다. 그리고, 임시 파일들에 대한 두 옵션의 조합여부에 따른 SSD내의 쓰기 증폭(Write Amplification)과 Trim 명령어 오버헤드가 조인 성능에 큰 영향을 미치는 점을 자세히 분석한다.

키워드 : 해시조인, 임시 파일, SSD, 트림, fallocate

1. 서 론

해시조인(Hash Join)은 조인의 대상이 되는 두 테이블 중에서 작은 테이블(Build Table)을 읽어 해시 영역(Hash Area)에 해시 테이블을 생성하고, 큰 집합(Probe Table)을 읽어 해시 테이블을 탐색하면서 조인하는 방식이다. 중첩 루프 조인(Nested Loop Join)은 조인 수행 시 무작위 접근으로 인한 성능 저하가 발생하고, 정렬 병합 조인(Sort Merge Join)의 경우 조인 수행 전의 정렬 작업으로 인한 성능 저하가 발생한다. 반면에 해시 조인의 경우, 그러한 성능 저하가 없기 때문에 동등 조인(Equal Join) 연산 수행에 널리 사용된다.

해시 조인은 해시 영역의 크기가 조인의 대상이 되는 테이블의 해시 테이블을 캐싱할 만큼 DRAM 크기가 크지 않을 때, 디스크에 임시 파일을 생성하여 조인 연산을 수행한다. 이때, 조인 연산을 마치는 임시 파일들은 순차적으로 삭제된다.

최근에는 많은 DBMS(DataBase Management System) 관리자들이 데이터베이스 스토리지로 HDD(hard disk drive)보다는 NAND 플래시 메모리 기반 저장 장치인 SSD(solid

※ 이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2022R1A2C2008225).
※ 이 논문은 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No.2015-0-00314, 비휘발성 메모리 기반 개방형 고성능 DBMS 개발).
† 준 회 원 : 성균관대학교 소프트웨어학과 석사과정
†† 정 회 원 : 성균관대학교 소프트웨어융합대학 교수
Manuscript Received : August 26, 2022
First Revision : September 22, 2022
Accepted : September 28, 2022
* Corresponding Author : Sangwon Lee(swlee@skku.edu)

state drive)를 선택하고 있다. 이는 SSD가 HDD와 비교하여 훨씬 높은 IO 성능 가지고 있을 뿐만이 아니라, SSD의 가격 또한 최근 몇 년 동안 계속해서 감소했기 때문이다.

SSD는 HDD와 몇 가지 다른 특성들을 가지고 있다. SSD는 여러 개의 블록 단위로 구성되어 있고, 이러한 블록은 또 여러 개의 페이지 단위로 구성된다. 또한, SSD에서 읽기/쓰기는 페이지 단위로 수행되지만 삭제는 블록 단위로 수행된다. 또한, HDD와 달리 SSD는 덮어쓰기(in-place update)가 불가능하다.

SSD가 어떤 페이지에 새로운 데이터를 쓸 때, 해당 페이지가 있는 블록의 전체가 먼저 지워져야한다. 만약 블록에 유효한 데이터가 있는 페이지가 남아있다면, 이 페이지는 바로 삭제되어서는 안 된다. 따라서 이러한 유효한 데이터가 있는 페이지들은 다른 블록으로 복사한 후 해당 블록의 삭제를 진행해야 한다. 이러한 데이터의 복사로 인해 호스트로부터 전달된 쓰기 이외의 추가적인 쓰기가 빈번하게 발생할 경우, SSD의 성능은 크게 저하될 수 있다.

불필요한 데이터의 복사를 막아 SSD의 IO 성능을 향상시키기 위해 운영체제와 파일시스템, SSD는 Trim 커맨드를 지원하기 시작하였다. Trim 커맨드는 ATA(Advanced Technology Attachment) 인터페이스에 대한 명령어로, 삭제한 파일의 데이터가 있는 LBA(Logical Block Address) 정보를 SSD에 전달하는 역할을 한다. 이러한 정보를 전달 받은 SSD 컨트롤러는 유효하지 않은 데이터의 불필요한 복사를 줄일 수 있게 되고, 이는 SSD의 성능을 크게 향상시킨다. 특히 랜덤 쓰기에서 Trim 커맨드의 효과가 두드러진다. Trim 커맨드를 사용한 경우, 그렇지 않은 경우보다 약 1.5배 정도 성능이 더 높았다[1,2].

그러나 이러한 Trim 커맨드도 워크로드의 특징에 따라 성능 저하를 발생시킬 수 있다. Trim 커맨드 또한 호스트에서 SSD로 전달되는 커맨드의 일종이기 때문에, IO가 활발히 일어나는 상태의 SSD에 계속해서 Trim 커맨드 또한 발생하면 심각한 IO 성능 저하를 발생시키게 되고 이는 SSD의 성능 저하로 이어지게 된다. Trim 커맨드의 개수가 많아지면 많아질수록 IO 커맨드와 Trim 커맨드 간 경합이 강해지기 때문에 성능이 더 크게 저하된다.

본 연구에서는 먼저 실험을 통해 해시조인에서 Trim 커맨드로 인해 발생하는 성능 저하를 검증한다. 또한, 이러한 성능 저하를 막기 위해 파일 데이터 블록의 평균 LBA 범위를 증가시킴으로써 Trim 커맨드의 개수를 줄일 수 있는 fallocate 커맨드를 제안하고 Trim 커맨드와 fallocate 커맨드의 사용 여부에 따른 성능 변화를 실험을 통해 비교하고 분석한다. 본 논문에서 수행한 실험에는 대표적인 OLAP (OnLine Analytical Processing) 벤치마크 중 하나인 TPC-H의 데이터 모델을 사용하였고, 워크로드는 해당 데이터 모델에서 두 테이블(orders, lineitem)의 해시조인 쿼리를 사용하였다.

본 논문의 구성은 다음과 같다. 2장에서는 해시조인의 알고리즘과 이로 인해 Trim 커맨드가 유발하는 성능 저하에 대해 설명한다. 3장에서는 이러한 성능 저하를 막을 수 있는 fallocate 커맨드의 사용과 그 원리에 대해 설명하고, 4장에서는 fallocate 커맨드와 Trim 커맨드 사용에 따른 성능 변화에 대해 여러 SSD를 사용하여 관찰하고 분석한다. 마지막으로 5장에서는 결론 및 향후 연구의 방향성을 제시하며 논문을 마친다.

2. 배 경

본 장에서는 논문의 배경과 실험에 사용할 워크로드의 특성에 관해 설명한다.

2.1 해시조인

해시조인은 빠른 검색 성능을 사용하여 동등 조인(equal join) 연산에서 가장 좋은 성능을 보여주는 것으로 알려져 있다. 해시조인은 조인 대상이 되는 두 테이블 중에서 작은 테이블 R(Build Input)을 읽어 해시 영역(Hash Area)에 해시 테이블(Hash Table)을 생성하고, 큰 테이블 S(Probe Input)를 읽어 해시 테이블을 탐색하면서 조인하는 방식이다.

해시조인은 조인 대상 테이블들이 해시 영역에 할당된 크기보다 클 때 스토리지에 임시 파일들을 생성하여 쿼리를 수행한다. 해시조인은 Fig. 1과 같이 크게 두 단계로 나뉘어 수행된다. 먼저, 첫 번째 단계인 파티션 단계에서는 조인 연산의 대상이 되는 두 테이블 중 먼저 작은 테이블 R을 스캔하고 분할하면서 해시 함수를 사용해 여러 파티션을 생성하여 임시 파일 형태로 디스크에 저장한다. 이후 큰 테이블 S 또한 스캔하고 분할하고 해시 함수를 사용해 이전과 같은 수의 파티션을 만들어 임시 파일 형태로 디스크에 저장한다. 두 번째 단계인 조인 단계에서는 파티션 단계에서 생성되었던 임시 파일들을 읽어 들이면서 해시 테이블과 매칭되는 튜플들을 찾아 조인 결과를 산출해낸다[3,4].

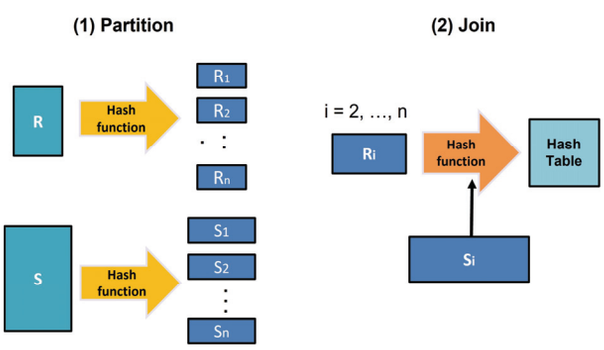

Fig. 1. I/Os in Two Phases of Hash Join Algorithm

2.2 가비지 콜렉션(Garbage Collection)

SSD는 블록 단위로 나뉘고 각각의 블록은 다시 페이지 단위로 나뉘게 되는데, 이 때 데이터 쓰기(write)는 페이지 단위로 수행되지만 데이터 삭제(erase)는 블록 단위로 수행된다. 또한 앞서 언급한대로 SSD는 HDD와 달리, 덮어쓰기가 불가능하다는 특징을 가진다. 따라서 SSD가 특정 데이터를 업데이트해야한다면, 새로운 데이터는 블록의 빈(free) 페이지에 쓰여져야 하고 기존의 데이터가 있던 페이지들은 유효하지 않은(invalid) 페이지로 마킹된다.

SSD가 데이터를 쓸 때, 페이지들은 먼저 삭제(erase)되어 빈 페이지가 되어야 한다. SSD의 데이터 삭제는 블록 단위로 수행되고, 삭제를 수행할 블록에는 유효한 페이지와 유효하지 않은 페이지가 섞여있을 수 있기 때문에, SSD 컨트롤러는 유효한 페이지들만 복사하여 다른 블록의 빈 페이지에 옮겨 쓴 후 블록을 삭제한다. 이러한 과정을 가비지 컬렉션(Garbage Collection)이라고 한다[5].

하지만 가비지 컬렉션은 유효한 페이지를 다른 블록에 복사해서 쓰는 과정에서 추가적인 쓰기를 수행해 성능 저하가 발생한다. 이를 쓰기 증폭(Write Amplification)이라고 한다. 이러한 쓰기 증폭의 크기를 나타내는 값으로 WAF(Write Amplification Factor)를 사용하는데, 해당 값은 호스트가 쓰는 데이터의 양에 대한 SSD 컨트롤러가 쓰는 데이터의 양으로 나타낸다. 높은 WAF는 실제 호스트에서 SSD로 전달되는 쓰기의 양에 비해 SSD의 쓰기 양이 많다는 것을 의미하므로, 이는 SSD의 전체적인 성능 저하를 유발한다.

WAF을 증가시키는 대표적인 원인이 바로 데이터 삭제이다. 파일을 삭제할 경우 삭제된 파일의 데이터는 필요 없음에도 불구하고 SSD 컨트롤러는 해당 데이터가 있는 페이지를 여전히 유효한 페이지인 것으로 인식한다. 따라서 가비지 콜렉션은 실제로는 유효하지 않지만 유효한 것처럼 보이는 페이지들을 계속해서 불필요하게 복사하고, SSD IO 성능의 큰 저하로 이어지게 된다.

2.3 트림(Trim)

가비지 콜렉션에서 발생하는 불필요한 페이지의 복사를 줄이기 위해 최근 Windows 10이나 kernel 2.6.33 버전 이상의 Linux와 같은 운영체제들은 Trim 커맨드를 지원하기 시작하였다. Trim 커맨드는 ATA(Advanced Technology Attachment) 명령의 일종으로, 운영체제는 Trim 커맨드를 사용해 어떤 데이터 블록이 더 이상 사용되지 않아 필요 없는지 SSD에 전달한다. 필요 없는 블록의 위치 정보를 전달 받은 SSD는 가비지 컬렉션 수행 시, 삭제된 파일의 데이터는 복사하지 않음으로써, 불필요한 쓰기를 줄이고 성능을 크게 향상시킨다.

최근 대부분의 SSD는 Trim 커맨드를 지원하지만, 사용하는 파일 시스템과 운영체제 모두 Trim 커맨드를 지원해야만 Trim 커맨드의 원활한 사용이 가능하다. 아직 모든 운영체제들이 Trim 커맨드를 지원하는 것은 아니지만, Windows 10이나 Linux kernel 2.6.33 버전 이상의 운영체제 등은 Trim 커맨드를 지원하고 있다. 또한 ext2, ext3 등 기존의 파일 시스템들은 Trim 커맨드를 지원하지 않았지만 ext4나 NTFS 등 최근의 파일 시스템들은 거의 모두 Trim 커맨드를 지원하고 있다.

하지만 특정 워크로드에서는 Trim 커맨드가 오히려 성능 저하를 유발한다. Trim 커맨드 또한 다른 읽기/쓰기 커맨드와 같이 호스트에서 SSD로 전달되는 일종의 커맨드로 간주될 수 있다. 다른 읽기/쓰기 커맨드가 자주 발생할 때, 즉 IO가 활발할 때 많은 수의 Trim 커맨드가 함께 발생한다면 IO 커맨드와 Trim 커맨드 간 경합이 심해지고, 이로 인해 성능 저하가 발생한다. 앞서 언급한 해시조인은 IO와 Trim 커맨드가 동시에 활발히 발생하는 대표적인 워크로드 중 하나이다[6,7].

Fig. 2a는 해시조인에서 발생하는 IO를 블록트레이스로 측정하여 나타낸 결과이다. 호스트는 파티션 단계에서 디스크에 생성되었던 임시 파일들을 조인 단계에서 순차적으로 읽어 들인 후 곧바로 삭제한다. 한 번 읽어 들인 임시 파일들은 다시 접근할 필요가 없기 때문이다. 최종적으로 해시 조인이 완료되고 조인 결과를 도출하는 시점에는 모든 임시 파일들의 삭제 또한 완료된다. 만약 Trim 커맨드가 활성화되어 있다면, 조인 단계에서 임시 파일들의 읽기와 삭제 모두 활발히 일어나게 된다.

앞서 언급한 대로, IO가 활발할 때 Trim 커맨드가 다수 발생할 경우 둘 간의 경합으로 인해 SSD의 IO 성능이 저하된다. Fig. 2b는 Trim 커맨드를 활성화하였을 때의 해시 조인 IO를 나타낸 것이다. 파티션 단계 수행 시간은 Trim 커맨드를 활성화하지 않은 Fig. 2a와 큰 차이가 없다. 그러나 읽기와 삭제가 동시에 활발하게 발생하는 조인 단계의 경우, Trim 커맨드를 활성화하지 않았을 때에는 읽기만 발생하여 약 120초 수행된다. 그러나 Trim 커맨드를 활성화한 경우에는 기존 읽기 수행 시간 120초보다 약 200초 더 수행되어, 약 2.7배 정도의 시간이 소요되었다. 이는 읽기와 Trim 커맨드가 동시에 발생할 때, 성능이 저하됨을 잘 보여준다.

3. 디자인

앞서 언급한대로, IO가 활발한 워크로드에서 Trim 커맨드의 발생은 성능 저하를 불러올 수 있다. 따라서 Trim 커맨드의 LBA 범위를 늘려 발생하는 Trim 커맨드의 개수를 줄임으로써 SSD의 성능을 향상시킬 수 있다. 이를 위한 방안으로 Linux에서 제공하는 fallocate 커맨드를 사용한다. fallocate

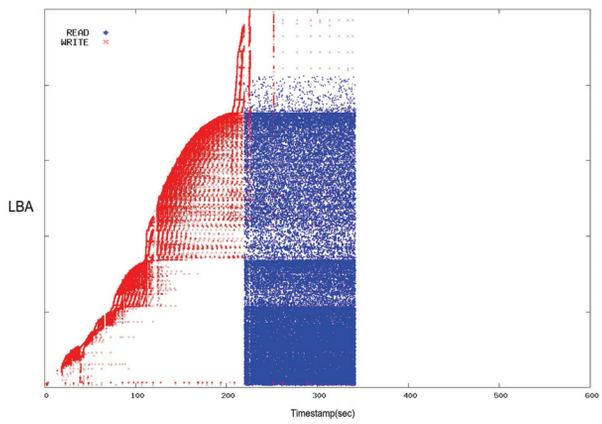

Fig. 2a. Hash Join IO Pattern

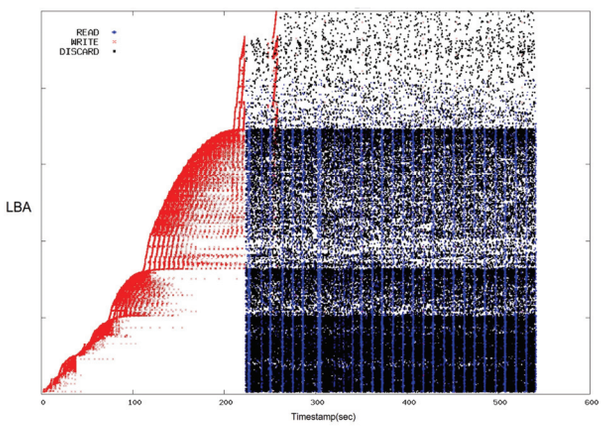

Fig. 2b. Trim Overhead during the Probing Phase in Hash Join

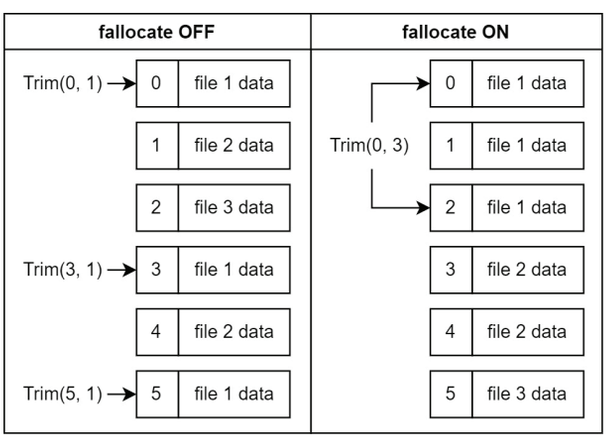

Fig. 3. The Effect of fallocate on Trim

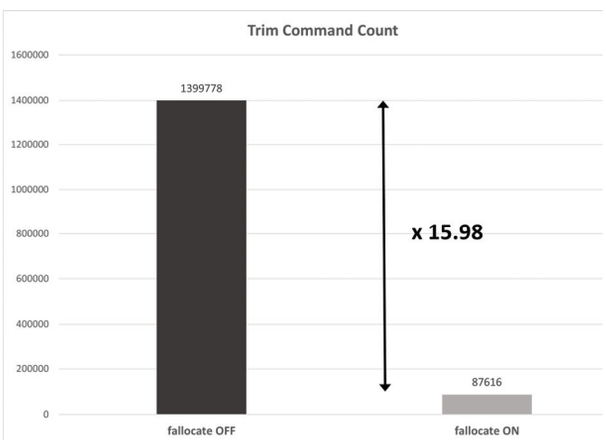

Fig. 4. Count Trim Counts: fallocate Off vs. On

커맨드는 특정 파일에 블록을 미리 할당하는 역할을 하는 명령어로, 파일을 열 때 특정 크기의 공간을 미리 할당한다.

Fig. 3은 fallocate 커맨드를 사용했을 때와 사용하지 않았을 때, 파일의 데이터 분포와 해당 파일의 삭제 시 발생하는 Trim 커맨드의 시작 LBA 및 길이를 예시로 비교한 것이다. fallocate 커맨드를 사용하지 않았을 때에는 파일 데이터가 저장될 때 작은 범위의 많은 LBA로 퍼져 저장된다. 따라서 LBA 범위가 작은 Trim 커맨드가 다수 발생하고 그러나 fallocate 커맨드 사용 시, 파일은 특정 개수의 블록을 미리 할당 받게 되고 해당 파일의 데이터는 연속된 큰 범위의 LBA에 저장된다. 이로 인해 파일 삭제 시, 파일 데이터가 저장되어 있는 LBA가 연속되어 있기 때문에 큰 LBA 범위의 적은 Trim 커맨드가 발생한다.

Fig. 4는 같은 수의 해시조인 쿼리를 수행했을 때, fallocate 커맨드 사용 여부에 따라 각각에서 발생한 Trim 커맨드의 수를 비교한 것이다. fallocate 커맨드를 사용하였을 때 발생하는 Trim 커맨드의 수보다, 사용하였지 않았을 때 발생하는 Trim 커맨드의 수가 약 16배 더 많음을 확인할 수 있다.

4. 실 험

4.1 실험 환경

실험은 페도라 기반 Linux CentOS 7.9에서 수행하였고 DBMS로는 PostgreSQL 12.11버전을 사용하였다. TPC-H 데이터 모델의 Orders와 Lineitem 테이블에 대해 다수의 해시조인 쿼리를 사용하여 실험을 수행하였다.

work_mem은 PostgreSQL에서 임시 파일을 쓸 때 사용 가능한 메모리의 최대 크기를 나타낸다. DRAM의 크기가 충분할 경우, work_mem을 크게 하여 임시 파일 처리의 성능을 향상시킬 수 있다. 그러나 반대로 work_mem 크기가 과도하게 커질 경우에는 PostgreSQL 서버에서 메모리 부족 문제가 발생할 수 있다.

Fig. 5는 work_mem 크기를 조정하며 해시조인을 10번 반복하여 수행해 각 반복마다 쿼리를 완료하는 데 걸린 시간을 측정한 결과이다. work_mem 크기를 충분히 할당하지 못하는 경우, 10회 반복 시 임시 파일 IO가 증가하여 최대 약 2.1배의 수행 시간 증가를 확인할 수 있었다. 본 연구는

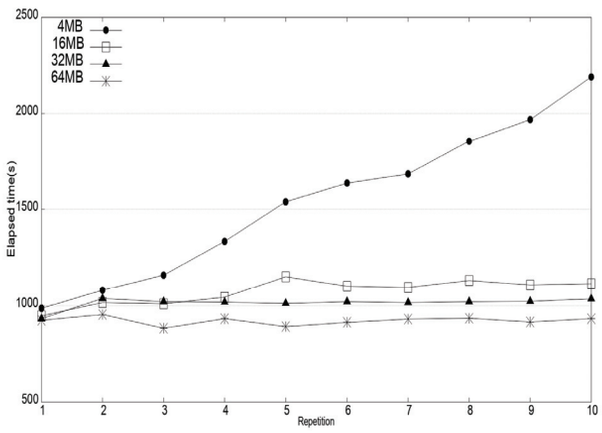

Fig. 5. Execution Time Varying work_mem Size

Trim과 fallocate를 이용하여 work_mem 크기가 충분하지 않은 환경에서 성능 저하를 막는 아키텍처를 제안한다.

앞서 언급한 대로, PostgreSQL은 해시조인 쿼리 수행 시 임시 파일 디렉토리에 다수의 임시 파일을 만든 후(파티션), 만든 파일들을 읽어 들이면서 동시에 삭제가 이루어진다(조인). 따라서 PostgreSQL의 임시 파일 디렉토리에 여러 SSD를 마운트해 각 SSD에서 실험을 수행하였다.

본 연구에서 수행한 실험의 구체적인 환경은 Table 1과 같다.

Table 1. Experiment Environment

OS	CentOS Linux 7.9
CPU	Intel(R) Xeon(R) CPU E5-2660 v4@ 2.00GHz (56 cores)
Kernel	3.10.0-1160.66.1.el7
DBMS	PostgreSQL 12.11
shared_buffer size	128MB(Cosmos+: 16MB)
work_mem size	4MB(Cosmos+: 512KB)
DB size	40GB(Cosmos+: 3GB)
DRAM size	1GB

본 연구에서 수행한 워크로드는 Table 2와 같다.

Table 2. Workload

Query	Hash Join
Table Scan	Sequential Scan
Repetition Count (Micron MX500)	30
Repetition Count(Samsung 840 pro, Cosmos+)	20

Table 3. Temporary File Storage

SSD	Size
Micron crucial MX500	250GB
Samsung 840 pro	250GB
Cosmos+ OpenSSD	32GB

또한 본 연구에서 PostgreSQL의 임시 파일 디렉토리의 스토리지로 사용한 SSD는 Table 3과 같다.

Table 3의 각 SSD를 임시 파일 디렉토리 스토리지로 사용해 다수의 조인 쿼리를 수행하며, 각 반복마다 모든 쿼리 시작부터 완료까지의 수행 시간을 측정하였다. 특히, Micron의 crucial MX500과 Cosmos OpenSSD의 경우, SSD 내부에서 발생하는 쓰기 양의 측정이 가능하여 WAF를 산출하였다. Cosmos+ Open SSD는 PCIe 인터페이스 기반의 연구 및 교육용 오픈소스 SSD 플랫폼이다[8,9].

4.2 성능 측정

본 연구에서는 성능 측정을 위해 각각의 SSD에 대해 4가지 조합으로 실험을 수행하였다. 4가지 조합은 fallocate 커맨드와 Trim 커맨드 각각의 사용 여부에 따른다.

4.3 실험 결과

1) Micron crucial MX500

Fig. 6은 Micron의 Crucial MX500을 임시 파일 스토리지로 사용하였을 때 각 반복마다 수행에 걸린 시간을 나타낸 것이다. Trim 커맨드와 fallocate 커맨드 모두 사용하지 않았을 때, 처음에는 좋은 성능을 보이지만 쿼리 수행을 반복할수록 수행 시간이 증가하며 성능이 저하되는 모습을 확인할 수 있다. 이는 Fig. 7의 WAF 그래프로 확인할 수 있듯이, 쿼리 수행을 반복할수록 SSD 내부 가비지 콜렉션 수행 시 발생하는 추가적인 쓰기가 계속해서 증가하기 때문이다. Trim 커맨드를 사용하지 않았기 때문에 실험을 반복하면서 조인 단

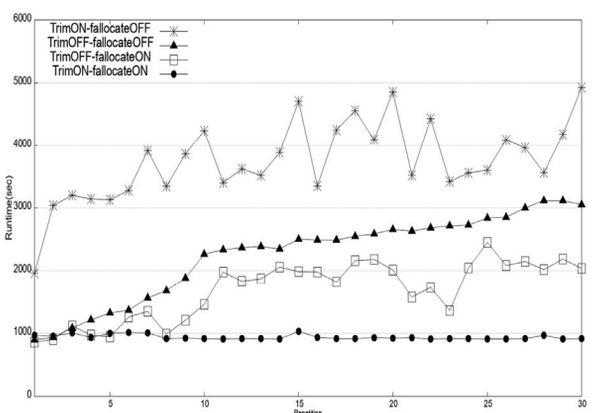

Fig. 6. Micron Crucial MX500 Execution Time

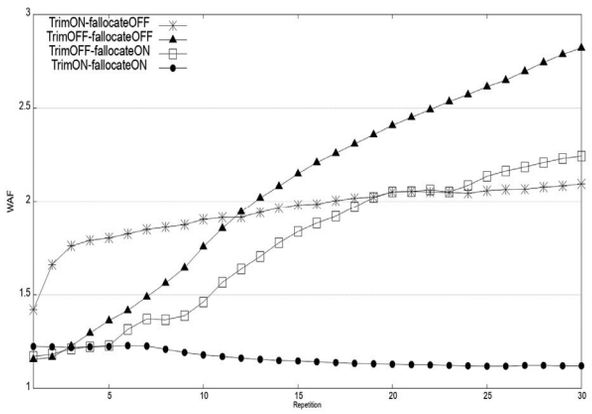

Fig. 7. Micron Crucial MX500 WAF

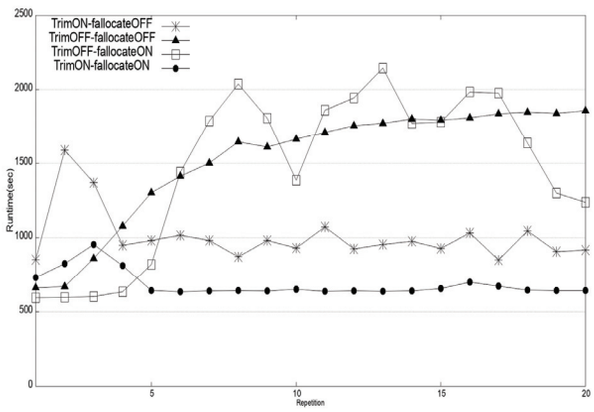

Fig. 8. Samsung 840 pro Execution Time

계에서 삭제된 파일의 데이터가 계속해서 축적되면서 필요 없는 복사가 발생해 WAF 증가를 촉진하고 점차적인 성능 저하로 이어진다.

fallocate 커맨드만 사용했을 때에도 마찬가지이다. 이 경우에도 Trim 커맨드는 사용하지 않았기에, 성능이 계속해서 저하된다. 그럼에도 불구하고, Trim 커맨드와 fallocate 커맨드를 모두 사용하지 않았을 때보다 전반적인 성능이 좋은 이유는 fallocate 커맨드를 통해 일반적인 IO의 LBA 범위가 커지고, 그에 따라 IO 수가 줄어들었기 때문이다.

반대로 Trim 커맨드만 사용한 경우에는 처음부터 수행 시간이 다른 경우보다 약 2배 이상으로 소요된다. 앞서 언급한 대로, 이는 조인 단계에서 발생하는 다수의 작은 Trim 커맨드로 인한 오버헤드로 보인다. 또한 이 경우에는 수행 시간의 변화가 매우 큰데, 이는 발생하는 Trim 커맨드의 수가 삭제되는 파일의 데이터 블록 분포가 반복 실험 수행마다 크게 변화하기 때문이다. 다만 WAF의 증가 속도는 전반적으로 Trim 커맨드를 활성화했을 때보다는 낮은데, 이는 조인 단계에서 삭제하는 임시 파일들의 LBA 정보를 운영체제가 Trim 커맨드로 SSD에 전달하여 불필요한 복사를 감소시키기 때문이다.

Trim 커맨드와 fallocate 커맨드를 모두 사용한 경우, 수행 시간과 WAF 모두 실험 수행 처음부터 반복 실험을 모두 완료할 때까지 안정적인 성능을 유지한다. 기존의 Trim 커맨드와 fallocate 커맨드를 사용하지 않은 경우보다 약 3배, Trim 커맨드만 사용한 경우보다는 약 5배 정도 성능이 더 좋음을 확인하였다.

2) Samsung 840 pro

Fig. 8은 Samsung 840 pro의 수행 시간 변화를 나타낸 것이다. 이 경우에도 전반적인 성능 변화의 양상은 Micron MX500의 결과와 크게 다르지 않다. Trim 커맨드를 사용하지 않았을 때, Micron MX500과 마찬가지로 실험을 반복할수록 수행 시간이 길어지면서 점차적인 성능 저하를 확인할 수 있다.

Trim 커맨드를 사용한 경우를 살펴보면, Trim 커맨드로 인한 성능 저하가 초반에는 매우 크게 나타나지만 전반적으로 MX500에서 확인했던 것보다는 그 정도가 적었다. 이는 840 pro Trim 커맨드의 레이턴시 크기가 MX500보다 작거나 내부 페이지 맵핑 알고리즘의 차이 때문일 것으로 보인다.

3) Cosmos+ OpenSSD

Fig. 9와 Fig. 10은 Cosmos+ OpenSSD에서 Trim 커맨드를 구현하여 앞과 같은 실험을 수행한 결과로, 각각 수행 시간과 WAF의 변화를 나타낸 것이다. Cosmos+에서도 Trim 커맨드를 사용하지 않은 경우, 반복 실험을 수행할수록 실험에 소요되는 시간이 점차적으로 증가함을 확인할 수 있다. 마찬가지로 WAF 또한 삭제된 파일의 불필요한 데이터 누적과 복사로 인해 계속해서 증가하였다.

Trim 커맨드를 사용하지 않을 때, fallocate 커맨드 사용 시 다른 SSD와는 달리 성능이 오히려 저하되었는데 이는 Cosmos+가 fallocate 커맨드에서 각 파일에 할당한 블록들을 모두 사용하지 않았을 때, 사용하지 않은 블록들의 반환 과정에서 오버헤드가 발생한 것으로 보인다.

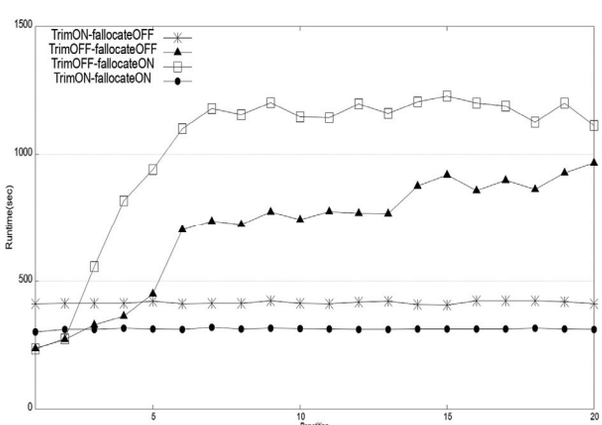

Fig. 9. Cosmos+ OpenSSD Execution Time

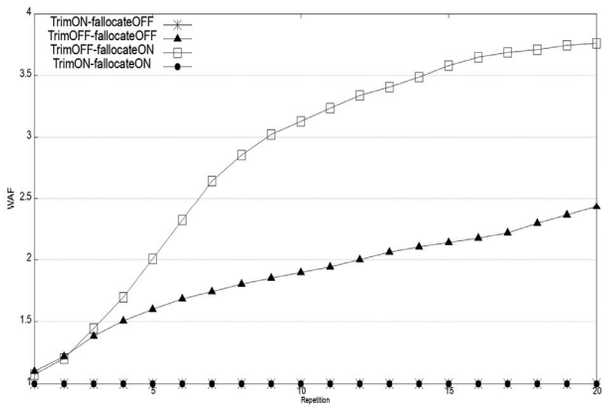

Fig. 10. Cosmos+ OpenSSD WAF

Cosmos+에서의 가비지 콜렉션 정책 특성 상, Trim 커맨드 사용 시 WAF는 증가하지 않는다. Fig. 10에서 이를 확인할 수 있다. 이로 인해 fallocate 커맨드 사용 여부와 상관없이 WAF가 1로 일정하게 유지되었다. 또한, Trim 커맨드 사용 시 fallocate 커맨드를 사용하지 않은 경우보다 사용한 경우 약 1.3배 성능이 더 좋음을 확인하였다.

5. 결론 및 향후 연구

Trim 커맨드는 가비지 콜렉션에서 데이터의 불필요한 복사를 막아 성능을 향상시킨다. 그러나 IO와 Trim 커맨드 모두 활발히 발생한다면, 이는 커맨드 간의 경합으로 이어져 오히려 SSD 성능을 저하시킨다. 본 논문에서는 fallocate 커맨드를 활용하여 Trim 커맨드의 LBA 범위 크기를 늘려 그 수를 줄이는 기법을 제안하였다. 실험 결과, fallocate 커맨드와 Trim 커맨드를 함께 사용한 경우, 그렇지 않은 경우보다 최대 약 3~5배의 성능 향상을 확인할 수 있었다.

해시조인은 임시 파일을 생성하는 다양한 워크로드 중 한 종류의 함수이다. 향후 본 논문에서 수행한 실험의 워크로드인 해시조인 이외에 다른 워크로드에서 Trim 커맨드가 유발하는 성능 저하와 fallocate 커맨드를 활용한 기법의 성능 향상을 확인하고 개선할 계획이다. 또한, 현재 상용 가능한 대부분의 SSD는 이러한 임시 파일이 아닌, 내구도가 높아야하는 일반적인 파일을 저장하고 처리하는 데에 최적화되어 있다. 따라서 임시 파일을 처리하는 워크로드에 최적화된 새로운 SSD 펌웨어 개발을 수행할 계획이다.

References

[1] J. Kim, H. Kim, S. Lee, and Y. Won, "FTL design for TRIM command," In *The Fifth International Workshop on Software Support for Portable Storage*, pp.7-12, 2010.

[2] G. Kim and D. Shin, "Performance analysis of SSD write using TRIM in NTFS and EXT4," *2011 6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT)*, pp.422-423, 2011.

[3] J. M. Patel, M. J. Carey, and M. K. Vernon, "Accurate modeling of the hybrid hash join algorithm," In *Proceedings of the 1994 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pp.56-66, 1994.

[4] S. W. Lee, B. Moon, and C. Park, "Advances in flash memory SSD technology for enterprise database applications," In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pp.863-870, 2009.

[5] K. Smith, "Garbage collection," SandForce, Flash Memory Summit, Santa Clara, CA, pp.1-9, 2011.

[6] H. Son, Y. Lee, Y. Kim, and J.-S. Kim, "An analysis on the performance of TRIM commands on SSDs and its application to the Ext4 file system," KIISE Transactions on Computing Practices, Vol.21, No.1, pp.52-57, 2015. https://doi.org/10.5626/ktcp.2015.21.1.52

[7] J. T. Kim, and S. W. Lee, "Analysis of the effect of TRIM on hash Join," 2021 KDBC. Korean Database Conference. https://dbsociety.kr/kdbc/kdbc2021/KDBC2021_Proceedings.pdf (pp.6-9), 2021.

[8] Y. H. Song, S. H. Jung, S. W. Lee, and J. S. Kim, "Cosmos openSSD: A PCIe-based open source SSD platform," *Proc. Flash Memory Summit*, pp.1-30, 2014.

[9] J. W. Kwak, S. J. Lee, K. B. Park, J. W. Jeong, and Y. H. Song, "Cosmos+ OpenSSD: Rapid Prototype for Flash Storage Systems," *ACM Transactions on Storage*, Vol.16, No.3, Article 15, pp.35, 2020. https://doi.org/10.1145/3385073.

김 준 태

https://orcid.org/0000-0003-2133-781X
e-mail : wojn1218@g.skku.edu
2021년 성균관대학교 소프트웨어학과(학사)
2021년 ~ 현 재 성균관대학교
소프트웨어학과 석사과정
관심분야 : 플래시 메모리 기반 DBMS

이 상 원

https://orcid.org/0000-0002-4206-3718

e-mail : swlee@skku.edu

1991년 서울대학교 컴퓨터공학과(학사)

1994년 서울대학교 컴퓨터공학과(석사)

1999년 서울대학교 컴퓨터공학과(박사)

1999년 ~ 2001년 한국오라클 기술과장

2001년 ~ 2002년 이화여자대학교 BK21 계약교수

2002년 ~ 2015년 성균관대학교 정보통신대학 교수

2015년 ~ 현 재 성균관대학교 소프트웨어융합대학 교수

관심분야 : 플래시 메모리 기반 DBMS