로그 구조 병합 트리의 복합 질의 처리를 위한 보조 인덱스 구조 및 수행 전략 분석

권도현, 최지헌, 정혁준, 김영훈, 오상윤*
아주대학교

{sylvester, unidev, jpo7173, yhoon, syoh*}@ajou.ac.kr

Secondary Indexes and Execution Strategies for Composite Query Processing in LSM-Tree

Dohyun Kwon, Jiheon Choi, Hyeokjun Jeong, Younghoon Kim, Sangyoon Oh*
Ajou University

요 약

로그 구조 병합 트리(LSM-Tree)는 우수한 쓰기 성능과 효율적인 기본 키 조회를 바탕으로 현대의 데이터 집약적 어플리케이션의 핵심 저장 구조로 채택되고 있다. 운영 환경에서 기본 키 조회를 넘어 여러 보조 속성 조건이 결합된 복합 질의를 처리해야 하는 요구가 빈번히 발생하게 되며, 이때 보조 인덱스는 보조 속성에 대한 효율적인 접근 경로를 제공하여 질의 탐색 범위를 좁혀 복합 질의를 최적화하는 데 사용될 수 있다. 본 연구에서는 기존의 LSM-Tree 독립형 보조 인덱스를 활용하는 상황에서의 복합 질의 처리 수행 전략을 분류 및 구현하고, 보조 인덱스 구조와 질의 수행 전략 조합에 따른 성능 특성을 비교 분석하였다.

Ⅰ. 서론

LSM-Tree 는 임의 쓰기 요청을 순차 쓰기 방식으로 변환하여 높은 쓰기 처리량을 보장하는 자료구조로, Bigtable [1], Cassandra [2] 등 현대의 데이터베이스 시스템의 핵심 저장 구조로 채택되고 있다.

그러나 기존의 LSM-Tree 읽기 최적화 메커니즘은 단일 키를 통한 접근에 한정되어 있다는 한계가 있다. 이에 대한 대안으로 별도의 LSM-Tree 를 역색인 구조로 사용하는 다양한 독립형 보조 인덱스 구조들이 제시되어 왔으며, Qader et al. [3] 는 이러한 기법들을 체계적으로 정형화 하여 성능 특성을 분석한 바 있다.

현대의 실제 서비스 환경에서는 단일 보조키 탐색 뿐만 아니라 여러 속성이 결합된 복합 질의에 대한 효율적인 처리 또한 요구된다. 일반적으로 관계형 데이터베이스에서는 특정 보조 속성 조합에 대한 질의에 최적화된 애드혹 (Ad-hoc) 복합 인덱스를 사전에 생성하여 관리할 수 있다. 그러나 이러한 방식은 가능한 복합 질의의 유형만큼 인덱스 관리가 필요하기 때문에, 쓰기 처리량이 저하되는 문제가 있다.

특히 LSM-Tree 환경에서 질의 유형별 애드혹 복합 인덱스를 모두 구축하는 경우 인덱스 관리 비용으로 인해 LSM-Tree 의 강점인 쓰기 성능이 크게 저해되는 문제가 발생한다. 따라서, 애드혹 인덱스를 추가하는 대신, 기존의 단일 보조 인덱스를 최대한 활용하여 복합 질의에 대응하는 전략을 우선적으로 고려할 필요가 있다.

이에 본 연구에서는 LSM-Tree 기반의 기존 보조 인덱스 구조들을 사용하였을 때 복합 질의 시 나타나는 성능 특성을 분석하고, 쓰기 성능을 보존하면서도 복합 질의 처리 효율을 최대화할 수 있는 보조 인덱스 구조의 설계 방향을 제시한다.

Ⅱ. 복합 질의 처리를 위한 인덱스 구조 및 수행 전략

본 장에서는 기존의 LSM-Tree 독립형 보조 인덱스와 본 연구에서 분류한 복합 질의 수행 전략들을 기술한다.

A. LSM-Tree 의 독립형 보조 인덱스 구조

LSM-Tree 독립형 인덱스 방식은 데이터 갱신 시점과 키-값 쌍 인코딩 방식에 따라 다음과 같이 분류된다.

1. *Lazy Updates*: 주 인덱스에 데이터가 삽입될 때 보조 인덱스를 즉시 갱신하지 않고, 인덱스의 변경분만 추가하는 방식이다. 변경분은 보조 인덱스의 SST 압축 과정에서 병합된다.
2. *Composite Keys*: 주 인덱스의 키를 보조 키와 기본 키 순서로 결합하여 저장하는 방식이다.

이 외에도 보조 인덱스의 상태를 항상 최신 상태로 동기화하는 Eager Updates 같은 방식도 존재하지만, LSM-Tree 의 핵심 이점인 높은 쓰기 처리량을 크게 저해하는 방식이므로 본 연구의 실험 분석 범위에서는 제외하였다.

B. 복합 질의 수행 전략

본 연구에서는 독립 보조 인덱스 환경에서 복합 질의 처리 기법을 다음과 같은 두가지 전략으로 분류 및 정의하고, 이를 구현하여 성능을 측정하였다.

1. *사후 필터링*: 단일 인덱스를 탐색하여 획득한 모든 기본 키들에 대하여 주 인덱스에 포인트 조회를 수행하여 나머지 보조 속성에 대한 조건 절 만족 여부를 사후 검사하는 방식이다.
2. *인덱스 병합*: 질의에 포함된 모든 보조 인덱스를 탐색한 뒤 얻은 기본 키 목록을 메모리상에서 교집합하여 최종 레코드에만 접근하는 방식이다.

Ⅲ. 실험 및 분석

본 연구의 실험은 Intel i5-11400, 64GB RAM, 1TB SSD 환경에서 수행되었으며, 실험을 위한 보조 인덱스는 RocksDB 상에 구현되었다.

또한, 표 1 과 같이 클러스터 서버 접근 로그를 모방한 데이터셋을 사용하며, 실제 I/O 비용을 모사하기 위해 인덱싱 되지 않는 256 바이트의 페이로드를 모든 레코드에 추가하였다. 또한, request_uri 속성에는 Zipfian 분포를 적용하여, 일부 보조 키에 다수의 레코드가 편향되는 환경에서의 질의 성능을 평가하였다.

표 1. 워크로드 명세

속성명	역할	카디널리티	데이터 분포
log_id	기본 키	10^9	Auto Increment
region_id	보조 키	10^1	Uniform
instance_id	보조 키	10^2	Uniform
request_uri	보조 키	10^6	Zipfian($\theta = 0.5$)

A. 인덱스 쓰기 오버헤드 분석

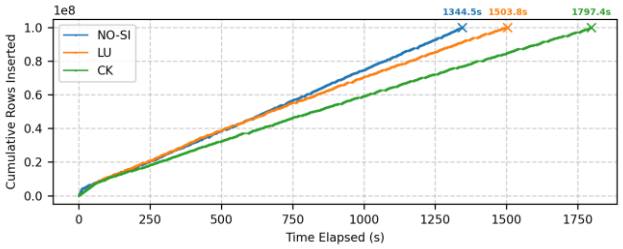
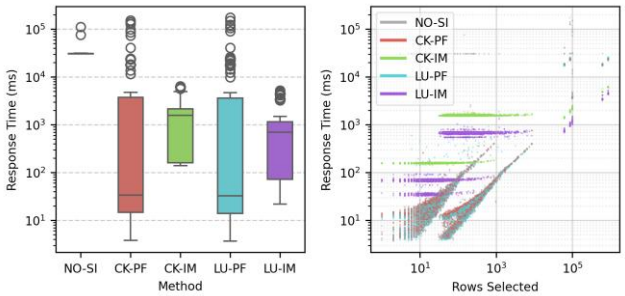


그림 1. 보조 인덱스 구조별 데이터 쓰기 소요 시간

총 1 억행의 레코드를 삽입하는데 소요된 시간을 측정한 결과, 보조 인덱스를 구축하지 않는 (NO-SI) 경우 대비 Lazy Updates (LU), Composite Keys (CK) 방식은 각각 약 1.12 배, 1.34 배의 시간이 소요되었다.

저장 공간 측면에서는 주 데이터의 크기 27GB 이고, 보조 인덱스의 크기는 LU 방식과 CK 방식이 각각 1.7GB, 1.9GB 를 차지하는 것으로 측정되었다.

B. 읽기 지연 시간 분석



(a) 지연 시간 분포 (b) 질의 결과 크기에 따른 지연 시간

그림 2. 보조 인덱스-복합 질의 수행 전략 조합별 읽기 지연 시간 비교

모든 레코드 삽입 이후 각 보조 인덱스와 복합 질의 수행 전략을 조합한 환경에서 각각 10,000 건의 복합 질의 응답 시간을 분석하였다. 각 질의는 보조 속성 중 임의의 속성 2 개를 무작위로 선택하여 구성하였다.

우선 NO-SI 방식은 테이블 전체 스캔이 불가피하여 질의 조건과 무관하게 일관적으로 30 초 이상의 높은 지연 시간을 보여 실시간 처리에 부적합함을 확인하였다.

사후 필터링 (PF) 전략을 사용하였을 때, 보조 인덱스 구조와 무관하게 중앙값 기준으로는 상대적으로 빠른 응답 속도를 보였으나, 선택도가 낮은 질의의 경우 과도한 무작위 포인트 조회로 인해 인덱스를 사용하지 않는 경우보다 더 느린 지연을 보이기도 하였다.

반면, 인덱스 병합 (IM) 전략은 PF 전략 대비 낮은 꼬리 지연을 가지지만, 질의를 구성하는 모든 보조 속성에 대해 인덱스를 탐색해야 하는 오버헤드로 인해 선택도가 낮은 경우에도 전반적인 응답 시간의 하한이 높게 형성되는 결과를 보였다.

보조 인덱스 구조간 비교에서는, CK 방식이 LU 방식 대비 질의 수행 전략과 무관하게 전반적으로 더 높은 지연 시간을 보였다. 이는 CK 방식이 SST 압축 과정에서 인덱스 엔트리를 병합하지 않는다는 구조적인 특성으로 인해, 읽기 시 더 많은 I/O 를 수행해야 하기 때문인 것으로 보인다.

Ⅳ. 결론

본 연구에서는 LSM-Tree 환경에서 복합 질의를 처리하기 위해 독립형 보조 인덱스 구조와 다양한 질의 전략을 적용하고 그 성능을 분석하였다. 실험 결과, 두 보조 인덱스 구조 모두 LU 방식과 CK 방식은 각각 12%, 34%의 쓰기 지연 시간 증가를 보였는데, 이는 다양한 유형의 질의를 효율적으로 처리할 수 있는 유연성을 확보한다는 측면에서 수용 가능한 수준의 비용임을 시사한다.

그러나 질의의 조건절에 포함된 보조 속성값의 분포가 편중되거나 카디널리티가 낮아 질의 결과의 선택도가 낮은 경우 그 지연 시간이 급격히 증가함을 확인하였다. 이를 해결하기 위해서는 주 데이터 접근 전 탐색 후보군을 최소화해야 하지만, 기존의 독립형 인덱스 조합을 통한 복합 질의 처리의 경우 각 속성에 대한 개별적인 인덱스 참조를 수행하는 방식 특성상 최종 결과와 무관한 데이터에 대한 I/O 및 연산 비용이 발생한다는 구조적인 한계가 존재한다.

따라서 이러한 한계를 극복하기 위해서는, 다중 속성 조건을 동시에 평가하여 부적합한 후보를 더 효율적으로 배제할 수 있는 경량화된 필터링 메커니즘이 요구된다. 이러한 분석을 바탕으로, 향후 연구에서는 SST 의 불변성을 활용하여 SST 단위의 비트맵 인덱스 구조를 설계 및 구현하여 LSM-Tree 의 강점인 쓰기 성능을 보존하면서도 인덱스 참조와 무작위 I/O 를 줄여 복합 질의 처리 성능을 개선할 수 있는 방안을 모색할 것이다.

ACKNOWLEDGMENT

이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (RS-2023-00283799)

참 고 문 헌

[1] Chang, Fay, et al. "Bigtable: A distributed storage system for structured data." ACM Transactions on Computer Systems (TOCS) 26.2 (2008): 1-26.

[2] Lakshman, Avinash, and Prashant Malik. "Cassandra: a decentralized structured storage system." ACM SIGOPS operating systems review 44.2 (2010): 35-40.

[3] Qader, Mohiuddin Abdul, Shiwen Cheng, and Vagelis Hristidis. "A comparative study of secondary indexing techniques in LSM-based NoSQL databases." Proceedings of the 2018 International Conference on Management of Data. 2018.