블록체인 기반의 지리 공간 포인트 데이터 인덱싱을 위한 공간 LSM 트리*

서민준[0] 권태현 정성원
서강대학교 컴퓨터공학과
soemj530@gmail.com, sungro96@gmail.com, jungsung@sogang.ac.kr

Spatial LSM Tree for Indexing Blockchain-based Geospatial Point Data

Minjun Seo[0], Taehyeon Kwon, Sungwon Jung
Department of Computer Science and Engineering, Sogang University

요 약

블록체인 기술은 IOT, 헬스케어 등 다양한 분야에서 높은 활용도로 주목받으며, 분산 데이터베이스에 대한 대안으로도 활용되고 있다. 블록체인에 대한 높은 활용성에도 불구하고, 블록체인 기반의 지리 공간 데이터를 효율적으로 인덱싱하는 기법은 지금까지 많은 연구가 진행되지 않았다. 이에 본 논문에서는 블록체인의 쓰기-집중적인(write-intensive) 특성을 반영하여 지리 공간 데이터를 블록체인 환경에서 삽입 시 I/O 비용을 감소시키는 공간 LSM 트리 인덱싱 기법을 제안한다. 제안 기법은 실시간으로 대량의 업데이트가 발생하는 블록체인 상에서 지리 공간 데이터를 Geohash를 통해 선형화하고, 데이터간의 공간적 인접성을 고려하여 데이터 삽입시 I/O 비용을 최소화한다. 또한 지리 공간 포인트 데이터 검색 질의 처리 시 I/O 비용을 최소화하는 공간 필터를 제안하여 공간 LSM 트리에 대한 불필요한 탐색을 줄인다.

1. 서 론

최근 블록체인 기술이 다양한 응용분야에 접목됨에 따라 블록체인 기반의 공간 정보 서비스가 많은 주목을 받고 있다. IBM에서는 소비자가 배송의 전 단계를 추적할 수 있는 블록체인 기반의 식품 유통 정보 시스템을 개발하였고[1], 공간 블록체인 시스템인 FOAM[2]은 블록체인 주소와 지리 정보를 결합한 거래 서비스를 제공할 수 있는 프로토콜을 개발하고, 영국의 대학에서는 블록체인 분산형 P2P를 이용해 드론과 자율주행차에 데이터를 전달해 재난에 대응하기 위한 개발[3]을 진행하였다. 이 밖에도 블록체인은 신뢰성과 개방성을 필요로 하는 다양한 분야에서 활용된다. 그러나 블록체인 기반의 위치 정보 서비스에 대한 관심에 비해, 블록체인에 저장된 지리 공간 데이터의 검색 및 분석 기법은 많은 연구가 진행되지 않았다.

블록체인은 기존의 공간 데이터베이스에서 가정하고 있는 환경과는 다른 특징을 갖는다. 첫 번째로, 블록체인은 실시간으로 대량의 업데이트를 발생시키는 쓰기-집중적인(write-intensive) 환경이다. 예를 들어 대표적인 블록체인 중 하나인 이더리움(Ethereum)의 경우 트랜잭션 데이터는 블록 단위로 전송되며, 14~15초 간격으로 지속적인 업데이트가 발생한다. 다양한 데이터베이스 시스템에서 사용되는 R-tree[4]와 같은 기법들은 인덱스의 유지 및 업데이트 비용이 크기 때문에, 업데이트가 잦은 환경에서는 병목현상을 유발할 수 있다. 두 번째 특징은 데이터의 불변성(immutability)이다. 블록체인에 저장된 데이터는 삭제나 변경이 되지 않는 것을 원칙으로 한다. 따라서 블록체인의 크기는 서비스가 지속됨에 따라 증가하게 되고, 데이터베이스의 인덱스를 메모리상에서 관리하는 것은 점차 어려워진다.

현재 다수의 블록체인에서 공간 정보에 대한 검색은 NoSQL DBMS의 보조 인덱스에 의존하고 있다. 그러나 NoSQL의 보조 인덱스는 시스템의 주 인덱스 및 디스크에 대한 반복적인 접근을 필요로 하기 때문에 데이터베이스에 따라 충분한 성능 향상을 얻기 힘들다. 따라서 본 논문에서는 블록체인 환경에서 지리 공간 데이터를 검색하는 방법으로, 대량의 업데이트에 최적화된 비 공간 데이터 인덱싱 기법인 LSM-tree[5]를 공간 데이터에 대한 인덱싱이 가능하도록 확장한 공간 LSM 트리 인덱싱 기법을 제안한다. 제안 기법은 공간 포인트 데이터의 좌표를 Geohash[6]를 통해 선형화하고, 데이터간의 공간적 인접성을 고려하여 LSM 트리의 컴포넌트에 배치해 질의와 연관이 없는 영역에 대한 탐색을 피한다. 본 논문의 기여는 다음과 같다.

- 일차원 데이터에 대한 인덱싱 기법인 LSM-tree를 확장하여 블록체인에 저장된 대용량 지리 공간 데이터에 대한 빠른 업데이트를 지원하는 인덱스 구조인 공간 LSM 트리 개발.
- 공간 LSM 트리에 기반한 포인트 데이터 검색 질의시 불필요한 탐색을 제거해 I/O 비용을 최소화하는 공간 필터 개발.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 공간 LSM 트리의 전체 구조 및 공간 필터를 설명하고 3장에서는 디스크 컴포넌트에 대한 플러시 기법과 병합을 차례대로 설명한다. 마지막 4장에서는 논문의 결론을 맺는다.

* 이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2021-0-00180, 다양한 산업 분야 활용성 증대를 위한 분산 저장된 대규모 데이터 고속 분석 기술개발).
이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. NRF-2019R1F1A1043854).
본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음 (IITP-2021-2017-0-01628).

2. 공간 LSM 트리

이 장에서는 공간 LSM 트리의 인덱스 구조와 공간 필터 그리고 컴포넌트 테이블을 설명한다.

2.1. 공간 LSM 트리의 Index 구조

공간 LSM 트리는 블록체인의 쓰기-집중적인(write intensive) 특성을 반영하여 지리 공간 데이터를 블록체인 환경에서 삽입 시 I/O 비용을 감소시키는 인덱싱 기법이다. 그림 1은 블록체인에 대한 공간 LSM 트리의 구조이다.

공간 LSM 트리는 메모리 영역에 하나의 메모리 컴포넌트와 컴포넌트 테이블을 저장하고, 디스크 영역에 디스크 컴포넌트를 저장한다. 먼저 메모리 컴포넌트는 블록체인에 가장 최근 저장된 포인트 데이터의 인덱스를 갖는 최상위 단계의 컴포넌트다. 디스크 컴포넌트 또한 데이터가 블록체인에 저장된 시간 순서대로 저장되며, Level 1에 최근의 데이터가, Level n에 가장 오래된 데이터가 저장되어 있다. 또한, 각 컴포넌트는 그림 2와 같은 데이터 저장 범위와 임계값을 갖는다. 예를 들어 메모리 컴포넌트는 위치에 관계없이 데이터가 저장 가능하고 x라는 임계값을 갖는다. 또한, Level 1 디스크 컴포넌트는 메모리 컴포넌트의 영역을 NW, NE, SW, SE의 4개 쿼드런트로 공간 분할 한 범위를 가지며, 그 예로 그림 2의 1_1은 SW, 1_2는 NW, 1_3은 SW, 1_4는 NE에 해당한다. 또한, 그림 1의 C1_2는 그림 2의 Level 1에 1_2와 같은 범위를 갖는다. Level 2 디스크 컴포넌트 역시 Level 1 컴포넌트가 임계값을 초과할 경우 4개의 영역으로 분할 해 Level 2에 저장하게 된다. 예를 들어 그림 1의 C1_2가 임계값 x를 초과할 경우 4개의 영역으로 분할 해 C2_5, C2_6, C2_7, C2_8에 저장하게 되며 그림 2로 확인해 보면 level 2에 5, 6, 7, 8과 같은 범위이다.

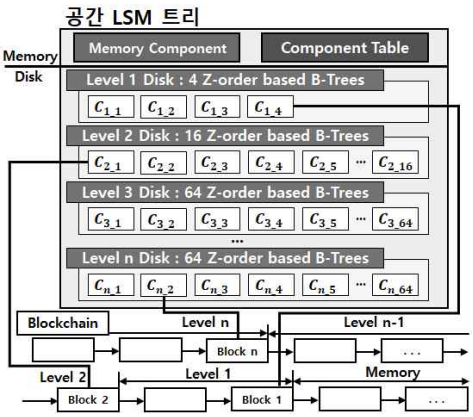


그림 1. 블록체인에 대한 공간 LSM 트리 구조

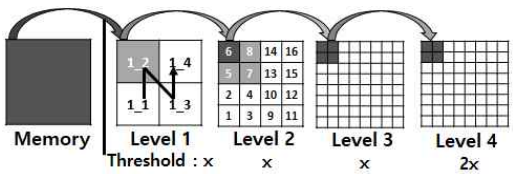


그림 2. 각 level별 컴포넌트의 개수와 임계값

하지만 지속적인 컴포넌트 증가와 저장 공간의 분할은 질의 처리 시 I/O증가로 인해 탐색 시간 증가와 질의 속도 저하를 불러온다. 따라서 그림 2와 같이 특정 level에서 컴포넌트의 증가와 공간 분할을 제한하고, 분할이 제한되는 Level부터 컴포넌트의 임계값을 2배로 증가시켜 지속적인 컴포넌트 증가를 막는다. 제한되는 Level은 데이터의 크기에 따라 유동적으로 조정이 가능하다. 컴포넌트 테이블은 2.2에서 자세히 설명한다.

컴포넌트는 Z-order based B-Tree를 통해 포인트 데이터를 저장하는 구조체이며, 데이터를 그림 3과 같이 (Geohash, BlockAddress) 형태로 저장하고, Geohash값을 기준으로 정렬된다. Geohash는 2차원 포인트 데이터를 1차원으로 나타내기 위해 (x, y) 쌍의 좌표 정보를 Z-ordering-value로 계산한 bit string이고, Block Address는 블록체인에서 정보가 저장된 블록의 주소이다. 각 컴포넌트의 데이터로 접근할 때 해당 포인트 데이터의 Geohash값을 찾은 뒤 대응되는 BlockAddress를 참조하여 데이터가 저장된 블록에 접근한다.

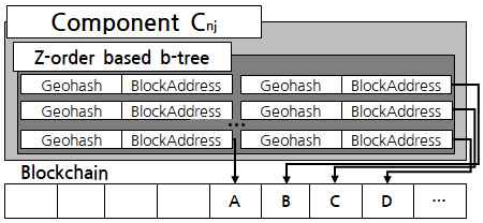
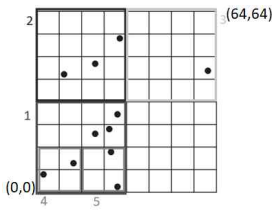


그림 3. 컴포넌트 구조

2.2. 공간 LSM 트리의 컴포넌트 테이블

컴포넌트 테이블은 메모리에서 관리하며, 공간 LSM 트리를 구성하는 모든 컴포넌트에 대한 정보를 저장한다. 컴포넌트 테이블에 저장된 정보를 통해 질의 처리 시 불필요한 컴포넌트의 탐색을 제거해 I/O비용을 줄이고 빠른 질의가 가능하다.



(a) 컴포넌트와 저장된 포인트 데이터의 예

Index	SLSM-tree Level	Key Range	Spatial Filter
0	0	0, 63	1101
1	1	0, 15	1011
2	1	16, 31	1011
3	1	48, 63	0010
4	2	0, 3	1001
5	2	8, 11	0011

Memory
Disk

(b) 컴포넌트 테이블의 예

그림 4. 컴포넌트 테이블

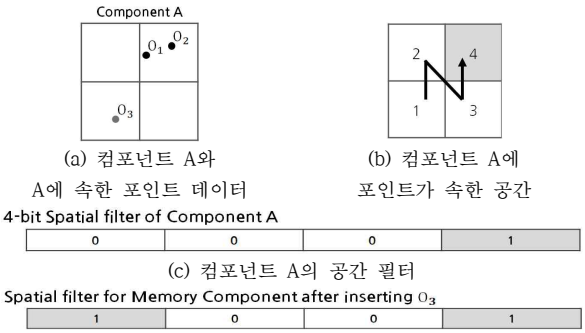
컴포넌트 테이블은 그림 4와 같이 컴포넌트의 level과 key range, 공간 필터를 저장한다. Level은 컴포넌트가 존재하는 단계이고, key range는 각 컴포넌트가 포인트 데이터를 저장할 수 있는 범위이다. 그림 4의 예를 보면 Index 0로 저장되는 메모리 컴포넌트는 전체 공간에 해당하는 key range를 가지므로 위치와 관계없이 저장할 수 있다. Level 1의 디스크 컴포넌트는 전체 데이터 공간을 4등분 한 0-15, 16-31과 같은 key

range를 갖는다. 마찬가지로 level 2의 디스크 컴포넌트는 level 1의 디스크 컴포넌트를 4등분 한 것 중 한 개, 즉 전체 영역을 16분할 한 0-3, 8-11과 같은 key range를 가진다.

그림 4(a)는 각 컴포넌트에 저장된 데이터를 표시한 예이다. 그림 4(b)는 공간 LSM 트리가 2.4(a)와 같이 주어질 때의 컴포넌트 테이블이다. 공간 LSM 트리에서의 질의 처리 시 컴포넌트 테이블을 통해 데이터 분포를 확인할 수 있다.

2.3. 공간 LSM 트리의 공간 필터

컴포넌트의 공간 필터는 질의 처리 시 불필요한 컴포넌트의 탐색을 제거해 I/O비용을 줄이고 빠른 질의 처리를 위해 활용한다. 공간 필터는 컴포넌트에 저장된 데이터의 공간적 분포를 나타내는 bit string이다. 공간 필터는 컴포넌트가 커버하는 영역을 4등분하여 데이터가 존재하면 대응되는 위치에 1, 존재하지 않으면 0을 표시한다. 그림 5는 컴포넌트 공간 필터의 예이다. 그림 5(a)와 같이 컴포넌트 A에 포인트 *O1, O2* 가 분포해 있을 때, 전체 데이터 영역을 4등분하여 그림 5(b)와 같이 인덱스를 부여하고, 그림 5(c)와 같이 공간 필터를 4자리 bit string으로 생성할 수 있다.



(a) 컴포넌트 A와 A에 속한 포인트 데이터

(b) 컴포넌트 A에 포인트가 속한 공간

(c) 컴포넌트 A의 공간 필터

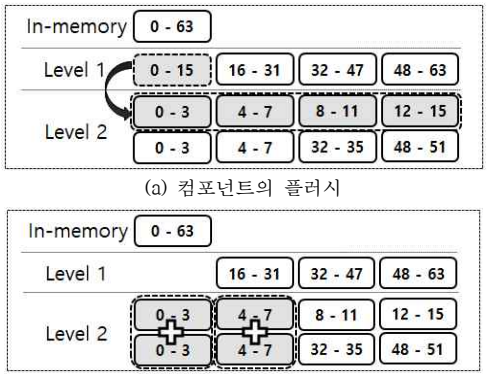
(d) O_8을 삽입한 후의 컴포넌트 A의 공간 필터

그림 5. 컴포넌트의 공간 필터

공간 LSM 트리는 각 컴포넌트에 새로운 포인트를 저장할 때 공간 필터를 업데이트한다. 예를 들어 컴포넌트 A에 포인트 *O3*를 저장하면 그림 5(d)와 같이 공간 필터를 업데이트한다.

3. 공간 LSM 트리의 포인트 데이터 삽입 방법

블록체인 환경에서 실시간으로 대량의 포인트 데이터가 블록에 추가될 때마다 새로운 블록에 있는 포인트 데이터들을 공간 LSM 트리에 인덱싱한다. 데이터 삽입 시 메모리 컴포넌트가 임계값을 초과하면 컴포넌트에 있는 Z-order based B-Tree를 플러시 한다. 플러시는 그림 6(a)와 같이 컴포넌트 단위로 수행되며 한 번의 플러시는 해당 컴포넌트의 Z-order based B-Tree를 그림 2와 같이 공간 분할 후 최소 1개, 최대 4개의 하위 level 컴포넌트에 추가하는 것이다. 플러시 진행 시 파라미터로 입력받은 특정 level까지는 Z-order based B-Tree를 4분할해 진행하지만, 그 이후의 level부터는 그림 2와 같이 컴포넌트의 증가가 없으므로 공간 분할 없이 하위 레벨에 있는 같은 공간으로 진행된다. 또한, 플러시 후 새로 생성되거나 업데이트된 컴포넌트의 데이터 분포를 확인하기 위해 공간 필터를 업데이트한다. 하지만, 플러시로 인한 컴포넌트 개수 증가는 LSM 트리의 탐색 시간을 증가시킨다. 따라서 동일 level에 속하는 컴포넌트 중 key range가 같은 컴포넌트를 하나의 컴포넌트로 병합하며, 그 예로 그림 6(b)에 있는 Level 2의 key range값 0-3, 4-7영역이 각각 병합되는 것을 확인할 수 있다. 컴포넌트 병합 후 컴포넌트의 크기가 임계값을 초과하면 추가적인 플러시가 일어난다.



(a) 컴포넌트의 플러시

(b) 컴포넌트 병합

그림 6. 컴포넌트 플러시와 병합

4. 결론

본 논문에서는 블록체인의 쓰기-집중적인(write-intensive) 특성을 반영하여 지리 공간 데이터를 블록체인 환경에서 삽입 시 I/O 비용을 감소시키는 공간 LSM 트리를 제안했다. 공간 LSM 트리의 메모리 컴포넌트와 하위 레벨의 컴포넌트는 내려갈수록 공간을 4등분 한 범위를 가지며, 이를 통해 계층적 구조를 띈다. 또한 제안 기법은 공간 필터로 각 컴포넌트에 데이터 분포를 파악할 수 있으며, 이를 활용해 질의 처리 시 불필요한 탐색을 줄여 I/O 비용을 최소화할 수 있다. 현재 제안한 공간 LSM 트리 기반의 효과적인 범위(Range)질의 및 kNN 질의 처리 알고리즘을 개발하고 있다.

참 고 문 헌

[1] Kamel Boulos, M.N., Wilson, J.T. & Clauson, K.A. Geospatial blockchain: promises, challenges, and scenarios in health and healthcare. Int J Health Geogr 17, 25 2018.

[2] https://foam.space/

[3] Boulos et al. “Geospatial blockchain: promises, challenges, and scenarios in health and healthcare” International Journal of Health Geographics, 2018.

[4] Antonin Guttman. R-trees: a dynamic index structure for spatial searching. SIGMOD Rec. 14, 2, 47-57, 1984.

[5] P. E. O’Neil et al., “The Log-Structured Merge Tree (LSM-Tree),” Acta Inf., vol. 33, no. 4, pp. 351-85, 1996.

[6] Jiajun Liu, Haoran Li, Yong Gao, Hao Yu and Dan Jiang, “A geohash-based index for spatial data management in distributed memory,“ 2014 22nd International Conference on Geoinformatics, pp. 1-4, 2014.