블록체인 기반의 지리 공간 포인트 데이터 인덱싱을 위한 공간 LSM 트리

서강대학교 | 컴퓨터공학과 | Bigdata Processing & DB LAB
서민준, 권태현, 정성원

- 2021.12.22 -

목 차

❖서론

- 배경지식
  - 블록체인 환경 특징
  - LSM 트리
  - Spatial Data Indexing Methods

❖본론

- 제안기법
- 공간 LSM 트리 인덱스 구조
  - 컴포넌트 구조
  - 컴포넌트 테이블
  - 공간필터
  - 활용 예시
- 공간 LSM 트리 Flush & Merge

❖결론

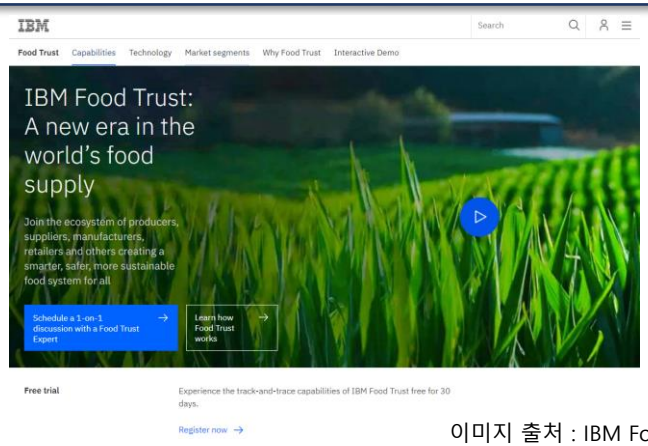
- 결론 및 향후 연구

배경지식

블록체인 환경 특징

- 블록체인 환경에서는 실시간으로 삽입되는 Write-intensive 환경

IBM
블록체인 기반 식품 유통 정보 시스템



이미지 출처 : IBM Food Trust

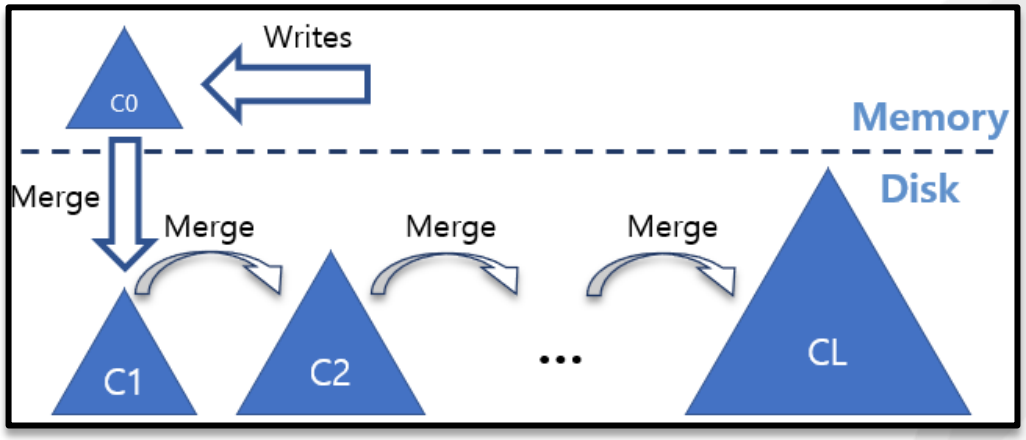
FOAM
블록체인 주소와 지리 정보를 결합한 거래 서비스

이미지 출처 : FOAM Space Youtube

➢ **<u>블록체인에 실시간으로 write-intensive하게 저장되는 공간 데이터에 대한 인덱싱 기술 부재</u>**

배경지식

- LSM 트리
  - 블록체인의 Write-intensive 환경에 최적화된 구조
  - Non-Spatial Data에 적합한 구조



➢ **Spatial Data에는 적합하지 않은 구조**

서론

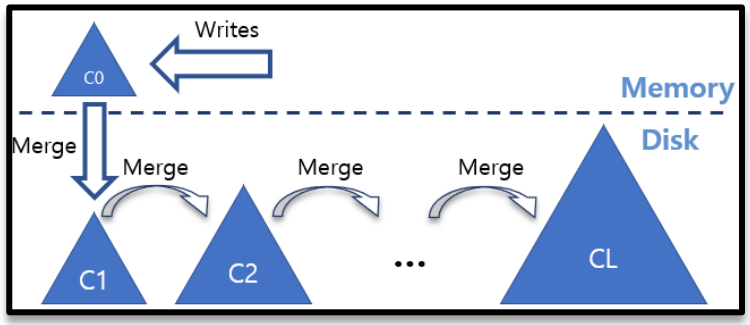
배경지식

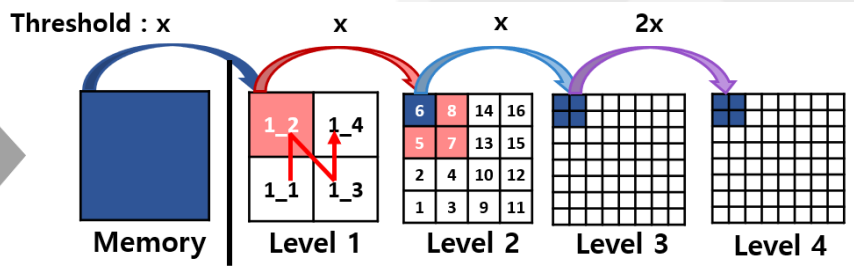
- Spatial Data Indexing Methods
  - **R-Tree 등의 Spatial Indexing Methods**
    - → 데이터 삽입 시 지속적인 Random I/O가 발생하여 적합하지 않음
    - → Write-intensive한 환경에서 많은 Disk I/O 발생

➢ **블록체인 환경에서 write-intensive하게 실시간으로 저장되는 공간 데이터에 대한 I/O비용이 최소화되는 효과적인 인덱싱 기술 및 이에 기반한 질의어 검색 기술 개발**

제안 기법

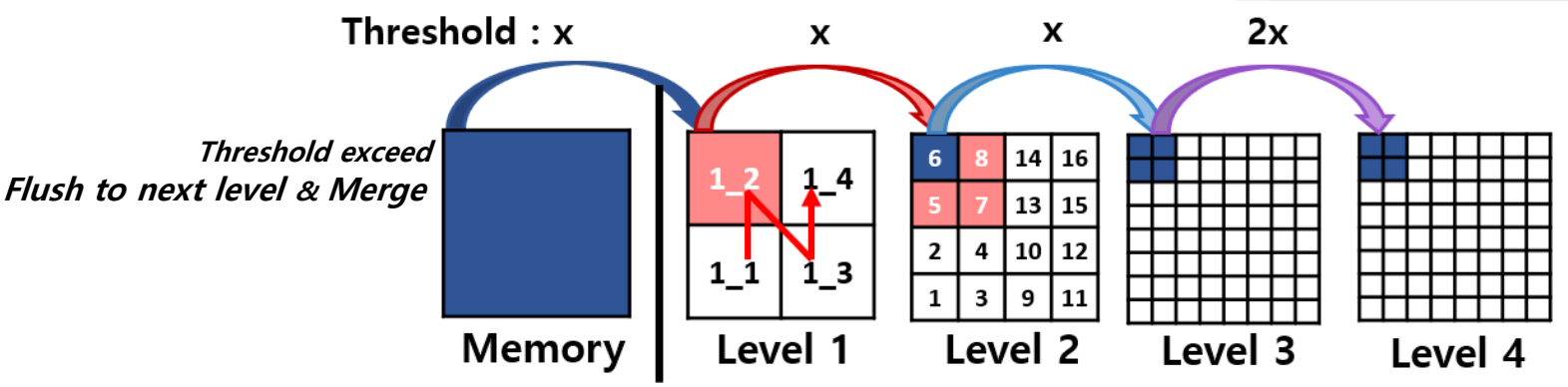
- 공간 LSM 트리
  - **LSM 트리의 장점과 기존의 Spatial indexing의 장점을 결합**
    - → Space Filling Curve를 활용해 포인트 데이터를 선형화
    - → 각 컴포넌트를 쿼드런트로 4분할한 후 관리함으로 인해 질의 처리나 Flush시 I/O를 줄이는 효과
    - → Write-intensive 환경을 반영하는 LSM 트리의 장점을 반영





제안 기법

- 공간 LSM 트리
  - 하위 레벨로 내려갈수록 컴포넌트 영역을 4분할하는 계층 구조
  - 특정 Level에서 컴포넌트의 증가와 공간 분할을 제한
  - Geohash를 통해 포인트 데이터를 선형화하고 공간적 인접성을 고려한 인덱싱
  - 각 컴포넌트마다 임계값을 설정해 전체 flush로 인한 I/O낭비 방지
  - 질의 처리 시 전체 탐색하지 않고 해당 영역만 탐색하여 효과적



본론

☑ 공간 LSM 트리 인덱스 구조

- **Memory Component**

  블록체인에 가장 최근 저장된 포인트 데이터의 인덱스를 갖는 최상의 단계의 컴포넌트

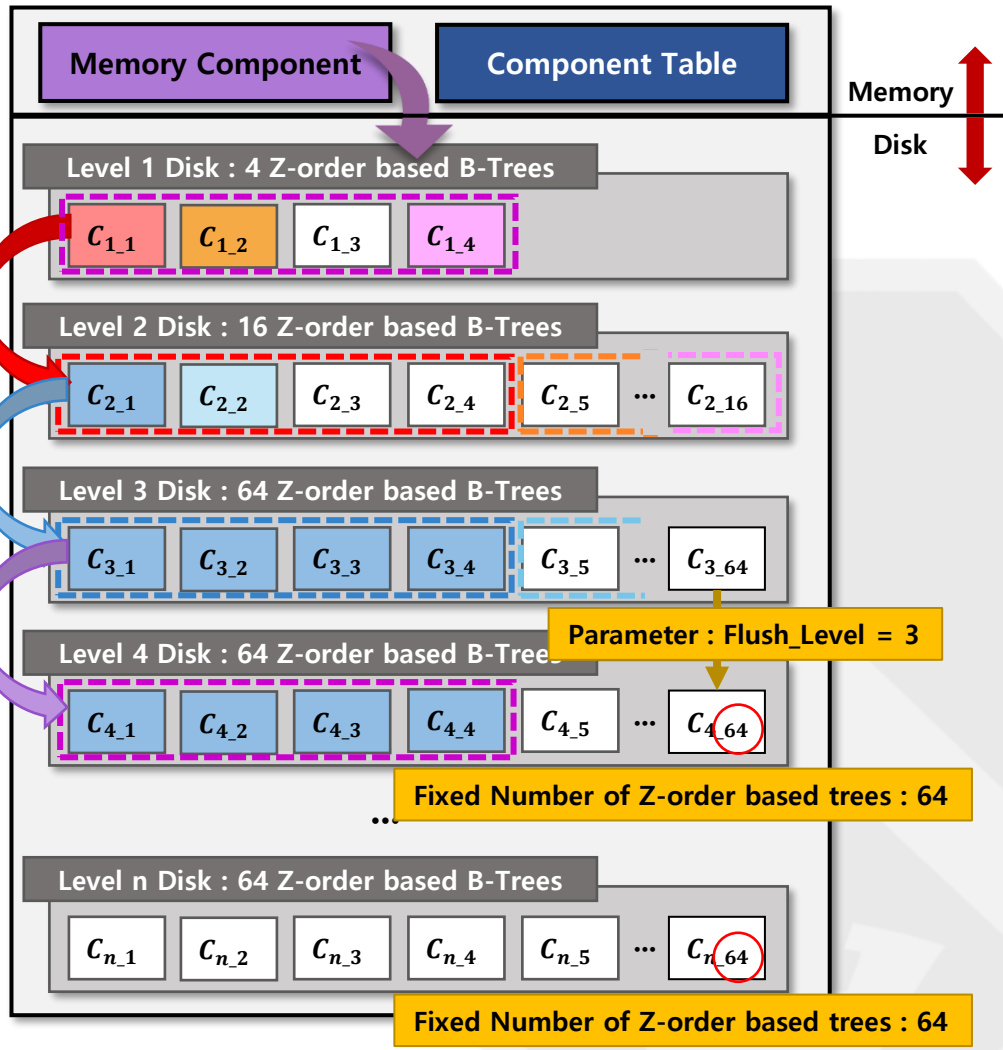
- **Disk Component**

  메모리 컴포넌트의 영역을 NW, NE, SW, SE의 4개 영역으로 공간 분할 한 범위를 갖고 데이터가 블록체인에 저장된 시간 순서대로 저장

- **Component Table**

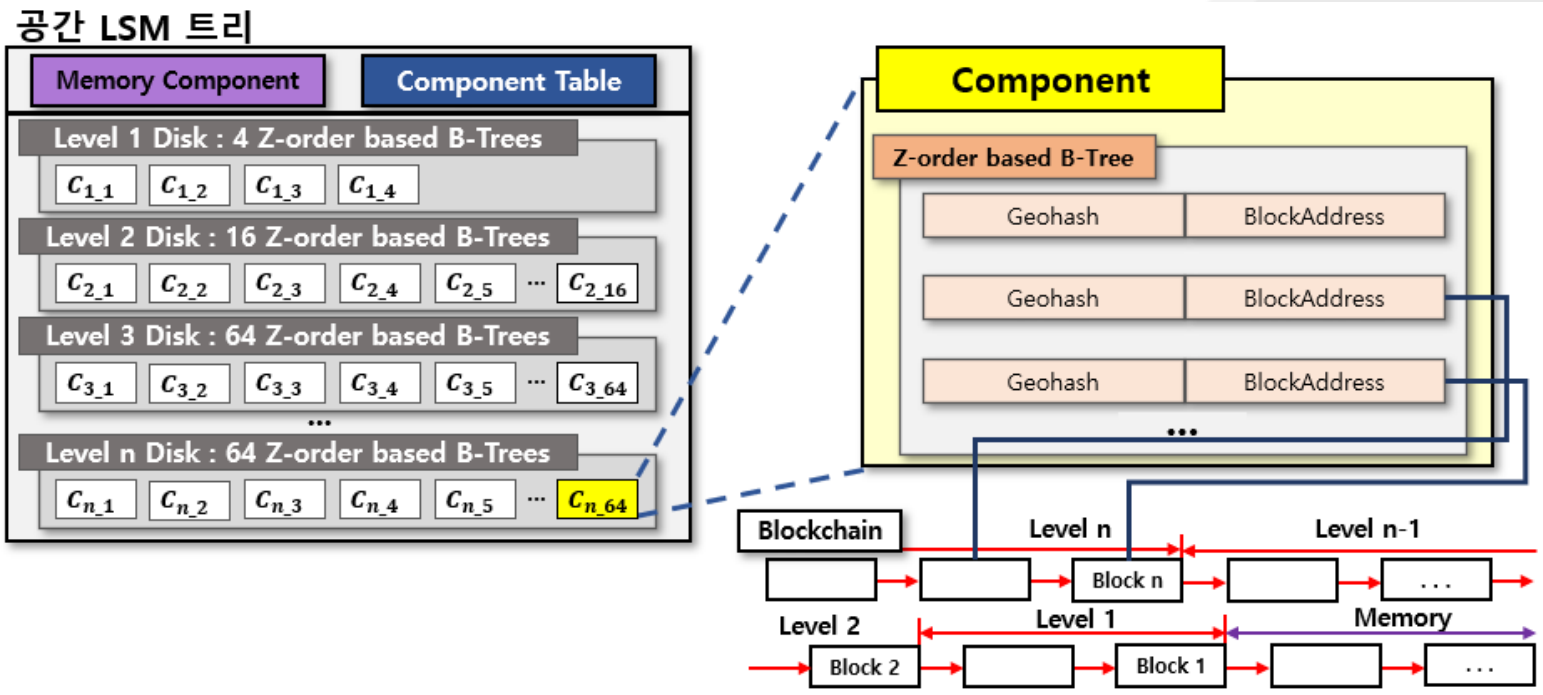
  컴포넌트 테이블은 메모리에서 관리하며, 공간 LSM 트리를 구성하는 모든 컴포넌트에 대한 정보를 저장

공간 LSM 트리



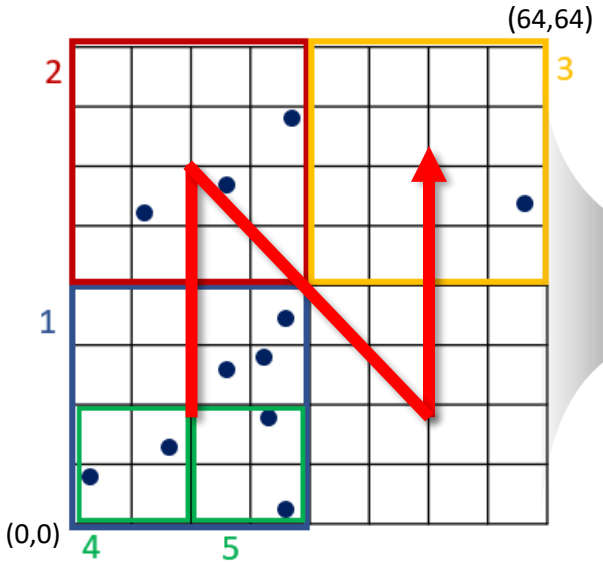
공간 LSM 트리 인덱스 구조

- 컴포넌트 구조
  - 각 컴포넌트는 해당 영역의 포인트 데이터를 인덱싱하는 Z-order based B-Tree
  - 포인트 데이터는 { Geohash, Block Address } pair로 저장



공간 LSM 트리 인덱스 구조

- 컴포넌트 테이블
  - 메모리에서 관리하며 모든 컴포넌트에 대한 정보 저장
  - 포인트 데이터 탐색 시 테이블에 저장된 정보 기반으로 불필요한 탐색 제거
  - I/O 비용을 줄이고 빠른 질의 가능



Component Table

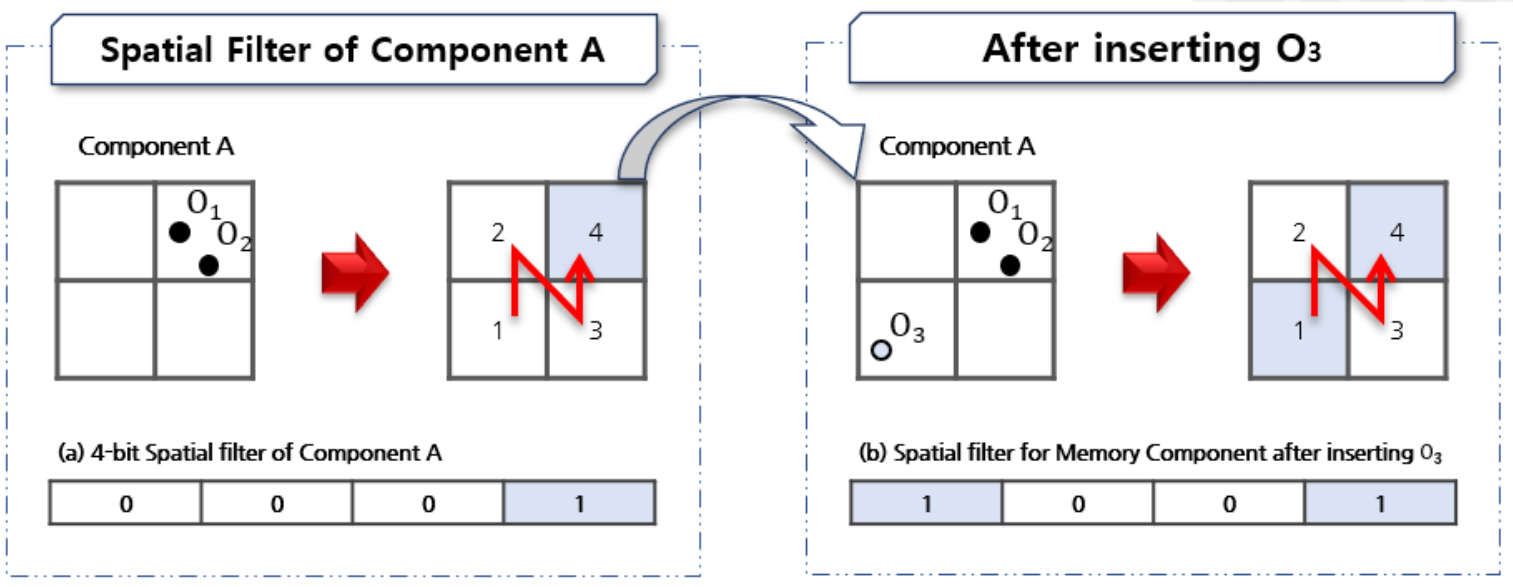
Index	SLSM-tree Level	Key Range	Spatial Filter
0	0	0, 63	1101
1	1	0, 15	1011
2	1	16, 31	1011
3	1	48, 63	0010
4	2	0, 3	1001
5	2	8, 11	0011

Memory

Disk

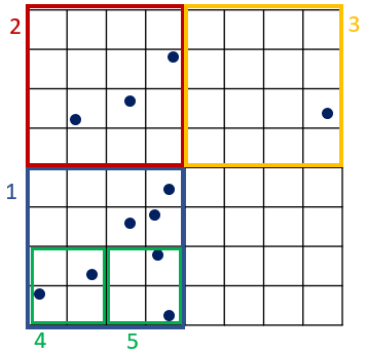
공간 LSM 트리 인덱스 구조

- 공간필터
  - 컴포넌트에 저장된 데이터의 공간적 분포를 나타내는 bit string
  - 질의 처리 시 불필요한 컴포넌트의 탐색을 제거
  - I/O비용을 줄이고 빠른 질의 처리를 위해 활용
    → 컴포넌트가 커버하는 영역을 4등분
    → 데이터가 존재하면 대응되는 위치에 1, 존재하지 않으면 0



본론

공간 필터와 컴포넌트 테이블 활용 예시



21	23	29	31	53	55	61	63
20	22	28	30	52	54	60	62
17	19	25	27	49	51	57	59
16	18	24	26	48	50	56	58
5	7	13	15	37	39	45	47
4	6	12	14	36	38	44	46
1	3	9	11	33	35	41	43
0	2	8	10	32	34	40	42

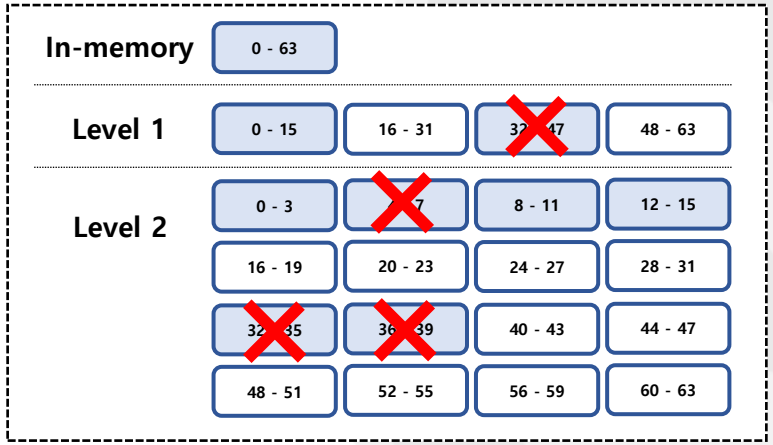
Query filter

2	3	6	7	...	14	15	32	33	36	37

Query Point ★
Query Range ○

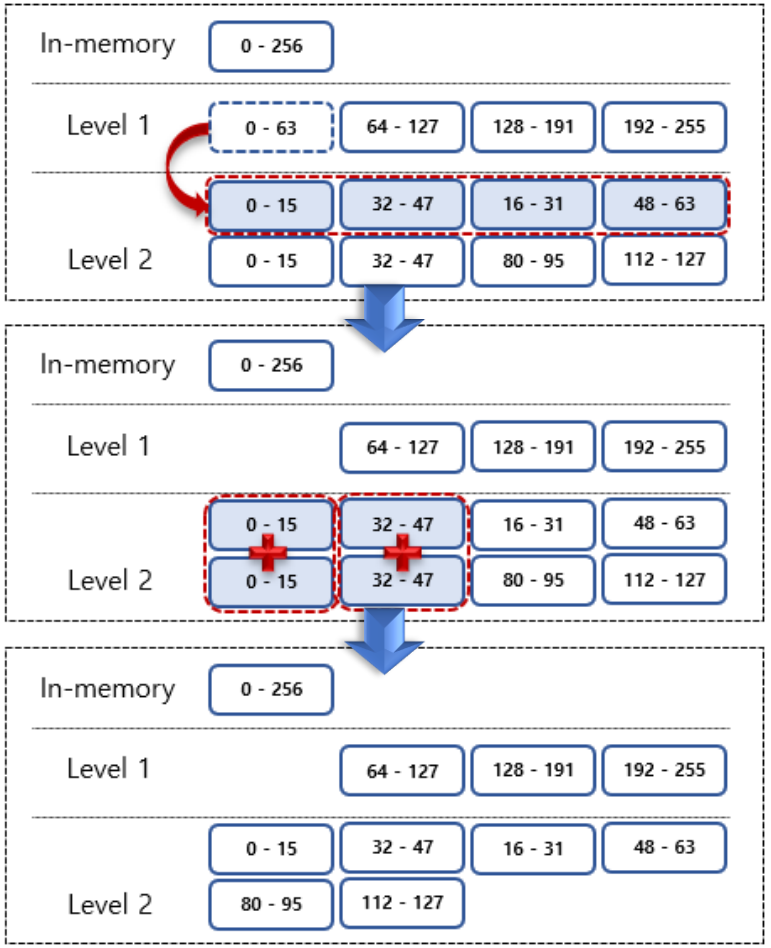
Component Table

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



공간 LSM 트리 Flush & Merge

- Flush
  - 각각의 Component가 임계값 초과 시 해당 Component만 Flush 진행
  - Level 0~3은 연쇄적으로 4분할
  - Level 4 이후부터 Level 3의 개수를 유지하며 각 임계값을 2배 증가
- Merge
  - Flush 발생 후 하위 Level의 동일한 영역의 컴포넌트와 병합
  - 병합 시 임계값을 초과하면 반복적으로 Flush & Merge 수행



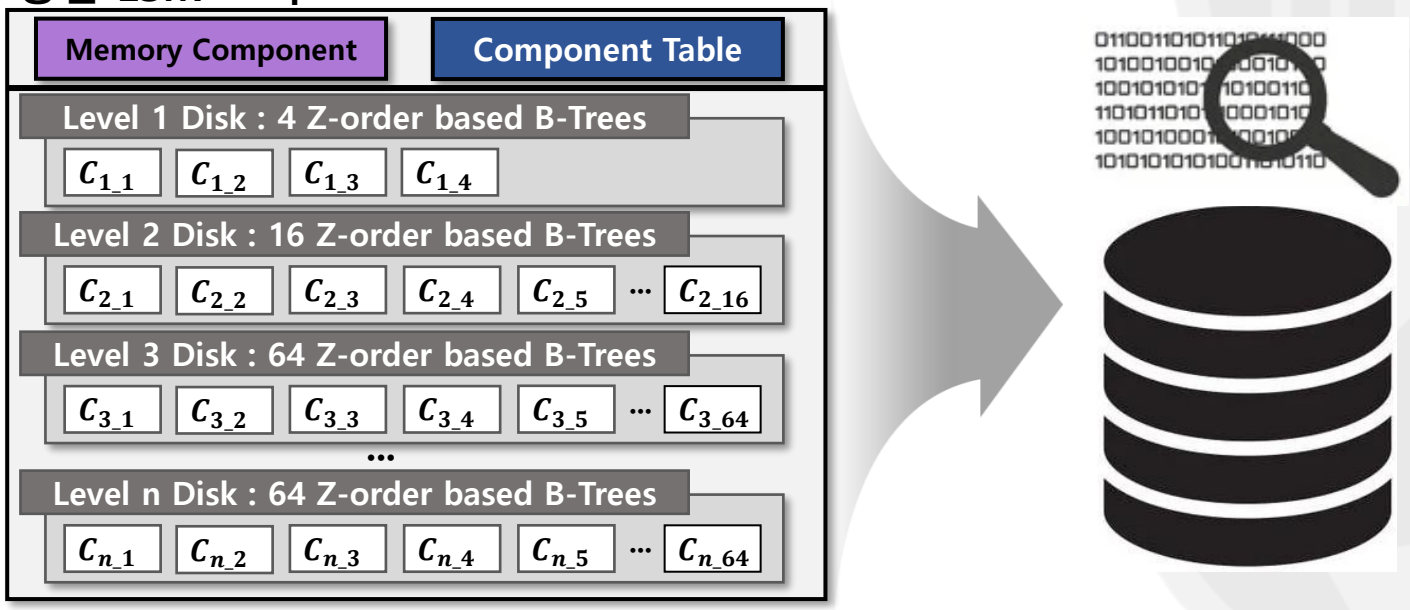
[Result of Flush & Merge]

결론

☑ 결론 및 향후 연구

- 블록체인에 공간 데이터 삽입 시 I/O비용 감소시키는 공간 LSM 트리 제안
  - 컴포넌트는 공간을 4분할 한 범위를 갖고, 이를 통해 계층적 구조를 이룸
  - 공간 필터로 각 컴포넌트 데이터 분포를 파악하며, 질의 처리 시 I/O비용 최소화
- 공간 LSM 트리기반 kNN query 개발
- 공간 LSM 트리기반 라인/폴리곤 데이터 인덱싱이 가능하도록 확장

공간 LSM 트리



감사합니다.