실시간 데이터를 이용한

ElasticSearch Indexing 성능 최적화

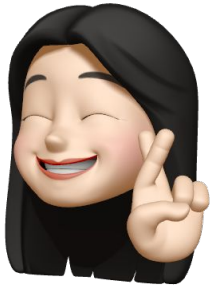
17th BOAZ Conference

중고책나라 팀

ABOUT TEAM

김건우

국민대
AI빅데이터융합경영학부
18기 엔지니어링



금나연

숙명여대 IT공학전공
18기 엔지니어링

박규연

국민대 소프트웨어학부
18기 분석

CONTENTS

01

Introduction

주제 및 주제 선정 배경 / 데이터 소개

Background of Topic

실시간 데이터
인덱싱 성능의 중요성

Elasticsearch 성능
최적화 관련 연구 부족

Elasticsearch 인덱싱 성능 최적화 가이드라인 마련

알라딘 API 데이터 스키마

제목, 리뷰랭킹 등 **중고책 관련 정보**
&
해당 책을 보유하고 있는 **알라딘 중고서점 정보**

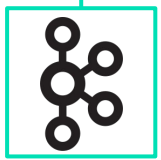
```
[
  {
    "book_id":155555,
    "title": "[중고] 밤의 피크닉",
    ...
    "customerReviewRank": 111000,
    "subInfo": {
      "usedType": "aladinUsed",
      "newBookList": {
        "newBook": {
          "itemId": 582029,
          "isbn": 8937830892,
          "priceSales": 10800,
          "link": "https://www.aladin.co.kr/shop/wproduct.aspx?ItemId=582029&partn
        }
      },
      "usedList": [
        {
          "shop_id":"aladin",
          "itemCount": 9,
          "minPrice": 4400,
          "link": "https://www.aladin.co.kr/shop/UsedShop/wuseditemall.aspx?ItemId=
        },
        {
          "shop_id":"Gwanghwamoon",
          "itemCount": 9,
          "minPrice": 4400,
          "link": "https://www.aladin.co.kr/shop/UsedShop/wuseditemall.aspx?ItemId=
        },
        ...
      ]
    }
  },
...
]
```

02

Architecture

사용 기술 및 아키텍처

Usage Skills



Kafka

분산형 데이터 스트리밍 플랫폼

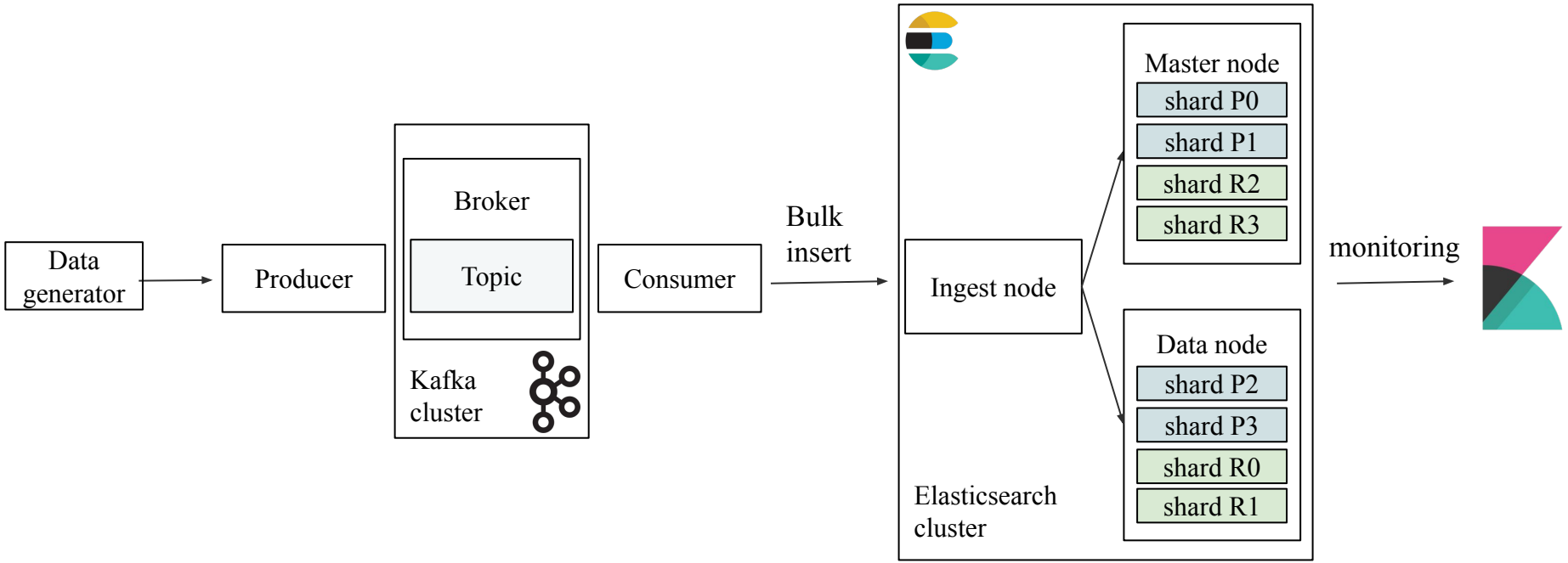
Elasticsearch

Apache Lucene 기반의 JAVA 오픈소스 분산 검색 엔진

Kibana

Elastic Stack 분석 및 시각화 플랫폼

Architecture



Elasticsearch

About Elasticsearch

Elasticsearch 란?

기반의 JAVA 오픈소스 분산 검색 엔진

- Lucene 라이브러리를 단독으로 사용
- 방대한 양의 데이터를 신속하고 거의 실시간(NRT, Near Real Time)으로 저장, 검색, 분석

Apache Lucene 이란?

검색 엔진의 시초

- 아파치 소프트웨어 재단의 회장 **더그 커팅**(Doug Cutting)이 고안한 역색인(Inverted Index) 구조인 아파치 Lucene을 기반으로 분산처리를 가능하게 한 아파치 Solr가 등장하여 검색엔진 시장을 장악
- 이후 Lucene을 기반으로 한 Elasticsearch가 등장하여 현재 지배적인 위치

Elasticsearch 사용 기업

- 검색 시스템이 존재하는 대부분의 서비스들이 Elasticsearch를 사용 중

Indexing 이란?

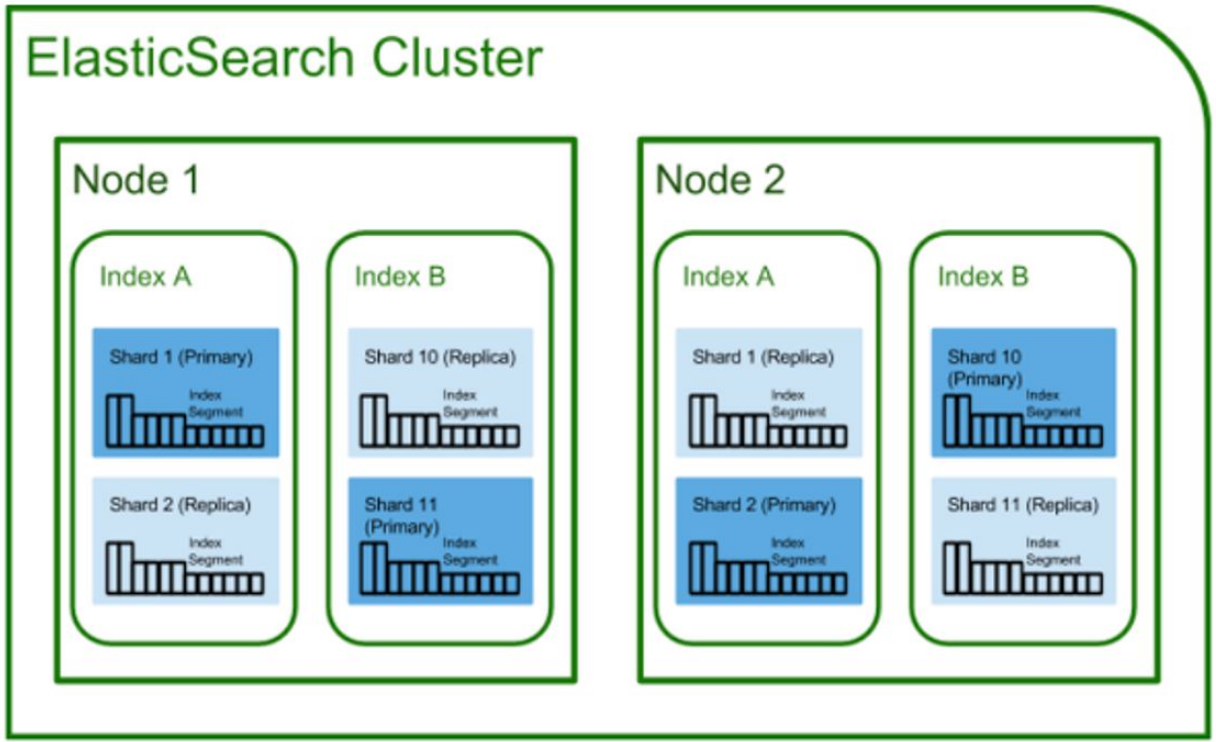
인덱싱의 목적

- 문서의 위치에 대한 index를 생성하여 빠르게 문서에 접근
- Elasticsearch는 inverted-index(역색인)의 구조로 구성

역색인

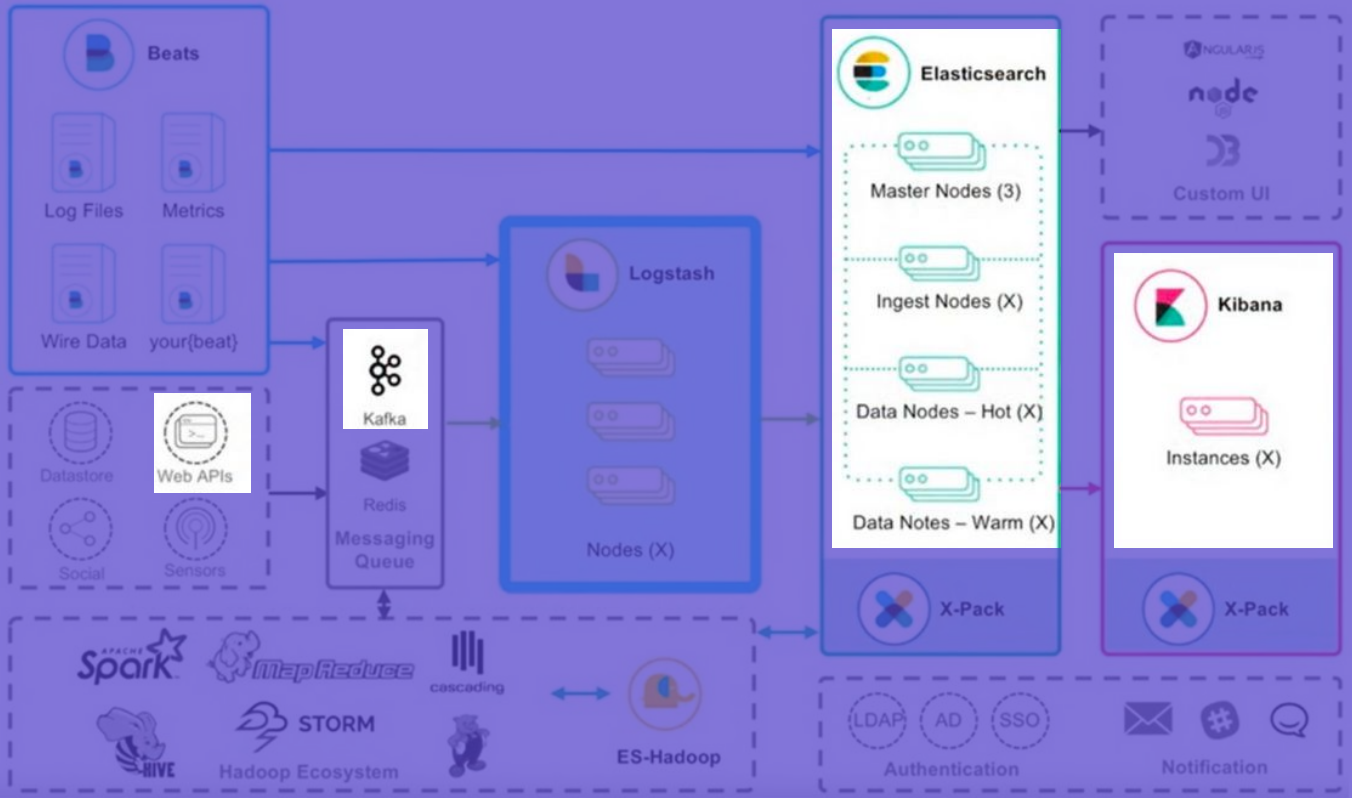
- 문서 내의 문자와 같은 내용물의 매핑 정보를 색인하는 전문 검색의 형태

Elasticsearch Architecture



https://velog.io/@koo8624/Database-Elastic-Search-2%ED%8E%B8-%EC%95%84%ED%82%A4%ED%85%8D%EC%B2%98Architecture

Elastic Stack



04

Optimization

인덱싱 최적화 실험

Optimization Experiment

실험 종속 변수(타겟)

- **Indexing Rate(/s, eps):** 초당 indexing된 document의 수
- **Latency(ms):** 노드에서 document의 indexing 처리시간

실험 환경

분류	항목	사양
Hardware	CPU	Intel(R) Core(TM) i5-1035G7 CPU @ 1.20GHz (4 core)
	RAM	8GB
	Storage	256GB SSD
Software	OS	Ubuntu 18.04
	JDK	OpenJDK 1.8.0
	Elasticsearch	7.17.7
	Kibana	7.17.8
	Apache Kafka	3.0.1
	Python	3.9

Optimization Steps

1. Static Mapping
2. Elasticsearch Shard 개수
3. Elasticsearch Data Node 개수
4. Nested to Unnested

Optimization Experiment

대조군

- Elasticsearch Ingest node **1**, Master node(Data node) **1**
- Primary shard **1**, Replica shard **1**, Dynamic mapping
- **Indexing Rate: 604.83 /s**
- **Latency: 0.32 ms**

1. Static Mapping

데이터 타입

- **Text:** 문자열을 term 단위로 쪼개어 역색인 구조를 생성. Full-text 검색
- **Keyword:** 문자열을 하나의 토큰으로 저장

→ Text 타입을 남용할 경우 쿼리로 인한 집계나 정렬 과정에서 메모리 사용량의 오버헤드 생성

- Aladin 데이터는 text, keyword, long, boolean 등 다양한 타입으로 이루어져 있어 text 형식의 데이터에 주목하여 keyword로 전환 후 성능 최적화

1. Static Mapping

실험군

- Elasticsearch Ingest node **1**, Master node(Data node) **1**
- Primary shard **1**, Replica shard **1**, **Static mapping**
- **Indexing Rate: 638.24 /s (↔604.83)** 약 1.06배 증가
- **Latency: 0.18 ms(↔0.32)** 약 1.78배 감소

2. Elasticsearch Shard 개수

Shard의 종류

- **Primary Shard:** 실제 CRUD를 제공하는 샤드, 증가 시 마스터 노드의 부하를 야기하여 색인과 검색 성능, 메모리에 악영향

- **Replica Shard:** 읽기 분산으로도 활용 가능한 장애 복구용 Primary Shard의 복제본.
  - 일반적으로 장애 대응을 위해 최소 한 개 이상의 replica shard를 두며, 운영 중 변경 가능
  - 인덱싱 성능과 trade-off 관계

2. Elasticsearch Shard 개수

실험군

1) Replica shard의 개수: 0개

- Elasticsearch Ingest node **1**, Master node(Data node) **1**
- Primary shard **1**, Replica shard **0**, Static mapping
- **Indexing Rate: 686.9 /s (↔638.24)** 약 1.07배 증가
- **Latency: 0.16 ms (↔0.18)** 약 1.3배 감소

2. Elasticsearch Shard 개수

실험군

1) Replica shard의 개수: 0개

- Elasticsearch Ingest node 1, Master node(Data node) 1
- Primary shard 1, Replica shard 0, Static mapping
- Indexing Rate: 686.9 /s (↔638.24) 약 1.07배 증가
- Latency: 0.16 ms (↔0.13) 약 1.2배 감소

→ Replica Shard의 수가 0개일 때는 1개에 비해 Indexing Rate이 증가하고 Latency이 감소하여 전체적인 성능이 향상되지만,

실시간 데이터의 특성 상 안정성을 위해 최소한의 replica shard 개수인 1개로 실험을 이어나간다.

2. Elasticsearch Shard 개수

2) Primary shard의 개수 : 2개

- Elasticsearch Ingest node **1**, Master node(Data node) **1, Data node 1**
- **Primary shard 2**, Replica shard **1,** Static mapping
- **Indexing Rate: 653.18 /s (↔638.24)** 약 **1.02배 증가**
- **Latency: 0.20 ms (↔0.32)** 약 **1.60배 감소**

2. Elasticsearch Shard 개수

2) Primary shard의 개수 : 4개

- Elasticsearch Ingest node **1**, Master node(Data node) **1, Data node 1**
- **Primary shard 4**, Replica shard **1,** Static mapping
- **Indexing Rate: 640.79 /s (↔638.24)** 약 1.004배 증가
- **Latency: 0.23 ms (↔0.32)** 약 1.39배 감소

2. Elasticsearch Shard 개수

2) Primary shard의 개수 : 8개

- Elasticsearch Ingest node **1**, Master node(Data node) **1, Data node 1**
- **Primary shard 8**, Replica shard **1,** Static mapping
- **Indexing Rate: 647.39 /s (↔638.24)** 약 1.01배 증가
- **Latency: 0.34 ms (←0.32)** 약 1.06배 증가

2. Elasticsearch Shard 개수

2) Primary shard의 개수 : 16개

- Elasticsearch Ingest node **1**, Master node(Data node) **1, Data node 1**
- **Primary shard 16**, Replica shard **1,** Static mapping
- **Indexing Rate: 640.91/s (↔638.24)** 약 **1.004**배 증가
- **Latency: 0.44 ms (↔0.32)** 약 **1.38**배 증가

2. Elasticsearch Shard 개수

2) Primary shard의 개수

- Primary shard 수를 늘리는 것은 Indexing rate 향상에 효과가 있으나 한계가 존재
- Primary shard 수는 Latency와 trade-off 관계
- 최대 성능: Primary shard 2개
  - Elasticsearch Ingest node **1**, Master node(Data node) **1**
  - **Primary shard 2**, Replica shard **1**, Static mapping
  - **Indexing Rate: 653.18 /s (↔638.24)** 약 1.02배 증가

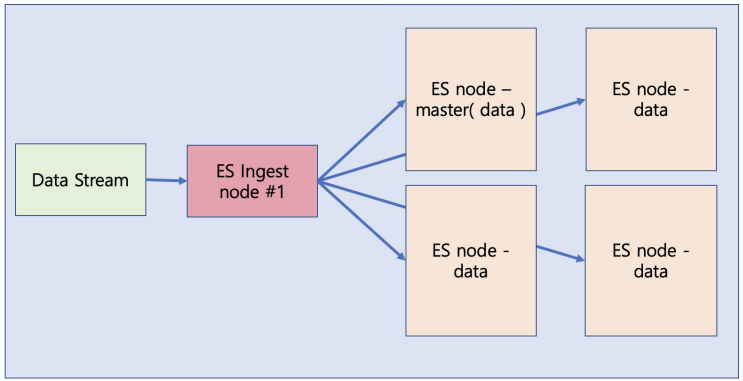
Primary shard	2	4	8	16
Indexing Rate	653.18 /s	640.79 /s	647.39 /s	640.91 /s
Latency	0.20 ms	0.23 ms	0.34 ms	0.44 ms

3. Elasticsearch Data Node 개수

Data node의 역할

- 데이터와 관련된 CRUD 작업을 수행하는 노드
- 실질적인 데이터 저장: 데이터가 실제로 분산 저장되는 물리 공간인 샤드가 배치
- 검색과 통계 등 데이터 관련 작업 수행

→ Data node의 개수가 데이터 indexing 작업에 미치는 영향을 실험하여 최적의 Data node 개수 설정



3. Elasticsearch Data Node 개수

Data node의 개수 : 3개

- Elasticsearch Ingest node **1**, Master node(**Data node**) **1, Data node 2**
- Primary shard **16**, Replica shard **1,** Static mapping
- **Indexing Rate: 590.86/s** (↔**653.18**) 약 **1.11**배 감소
- **Latency: 0.26 ms** (↔**0.20**) 약 **1.3**배 증가

3. Elasticsearch Data Node 개수

Data node의 개수 : 4개

- Elasticsearch Ingest node **1**, Master node(**Data node**) **1, Data node 3**
- Primary shard **16**, Replica shard **1,** Static mapping
- **Indexing Rate: 554.23/s (↔653.18)** 약 1.17배 감소
- **Latency: 0.32 ms (↔0.20)** 약 1.6배 증가

3. Elasticsearch Data Node 개수

Data node의 개수

- Data node의 수를 늘리는 것은 Indexing rate 향상에 효과가 없었음
- PC가 한 대라서 성능의 이득을 크게 볼 수 없었을 것으로 확인
  - 실제 클러스터를 구성할 시 I/O bottleneck 등이 일어날 가능성이 있어 Data node의 개수와 Indexing rate 사이에 trade-off가 존재할 것으로 예상되나, 여건 문제로 확인 불가
- 최대 성능(기존과 동일): Primary shard 2개
  - Elasticsearch Ingest node **1**, Master node(Data node) **1**
  - **Primary shard 2**, Replica shard **1**, Static mapping
  - **Indexing Rate: 653.18 /**
  - **Latency: 0.20 ms**

Data node	2	3	4
Indexing Rate	653.18 /s	590.86 /s	554.23/s
Latency	0.20 ms	0.26 ms	0.32 ms

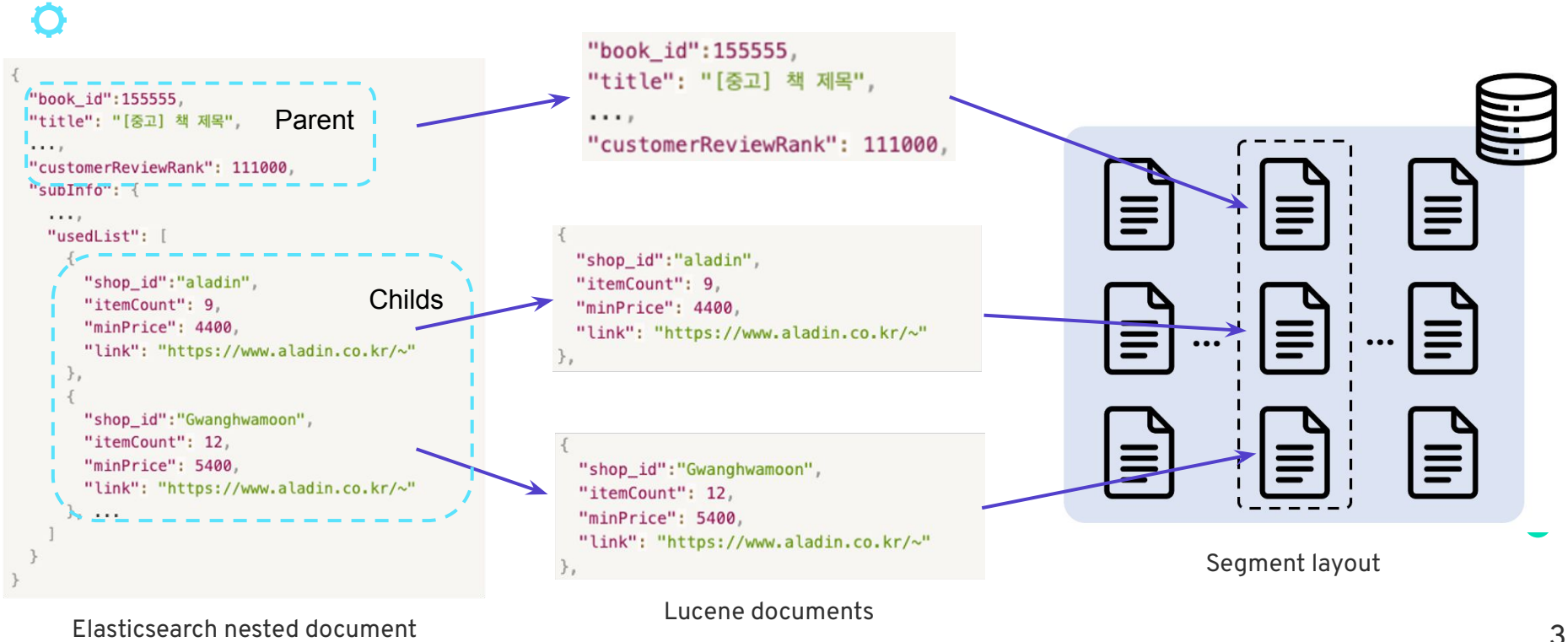
4. Nested to Unnested

문제 분석

데이터 모델에 **nested** 필드가 존재할 경우, 정보 변경에 큰 오버헤드 발생

```
{
  "book_id":155555,
  "title": "[중고] 책 제목",
  ...,
  "customerReviewRank": 111000,
  "subInfo": {
    ...,
    "usedList": [
      {
        "shop_id":"aladin",
        "itemCount": 9,
        "minPrice": 4400,
        "link": "https://www.aladin.co.kr/~"
      },
      {
        "shop_id":"Gwanghwamoon",
        "itemCount": 12,
        "minPrice": 5400,
        "link": "https://www.aladin.co.kr/~"
      }, ...
    ]
  }
}
```

4. Nested to Unnested



Elasticsearch nested document

Lucene documents

4. Nested to Unnested

문제 분석

데이터 모델에 nested 필드가 존재할 경우, 정보 변경에 큰 오버헤드 발생

- Parent, nested 문서는 segment 상에서 연속된 위치에 저장되어 연관 관계를 유지
- 정보 변경 시 update되는 것이 아닌 새 segment가 덮어 쓰임

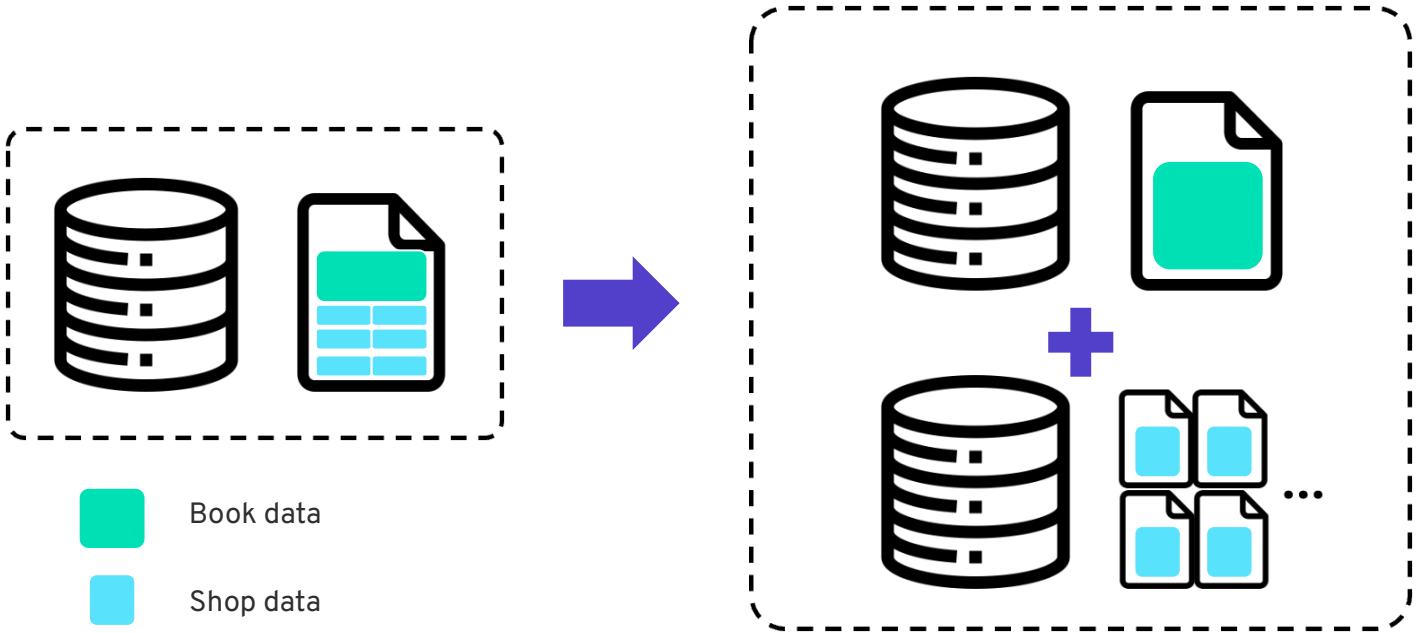
➜ **Parent** 정보가 변경될 때마다 모든 **nested** 문서 또한 덮어쓰이게 되고,

nested 필드의 개수만큼의 쓰기 증폭이 발생하게 되어 인덱싱 성능에 큰 악영향을 줄 수 있음

4. Nested to Unnested

데이터 구조 개편

데이터 모델을 재설계하여 인덱싱 성능 이슈 해결



4. Nested to Unnested

Unnested Data model

- Elasticsearch Ingest node **1**, Master node(Data node) **1, Data node 1**
- Primary shard **2**, Replica shard **1**, Static mapping
- **Indexing Rate: 1983.51 /s (↔653.18)** 약 **3.04**배 증가
- **Latency: 0.07 ms (↔0.20)** 약 **2.86**배 감소

05

Conclusion

결론 및 한계 / 참고문헌

성능 향상

대조군

- Elasticsearch Ingest node **1,**
- Master node(Data node) **1,**
- Data-only node **1,**
- Primary shard **1,**
- Replica shard **1,**
- Dynamic mapping
- Nested data
- **Indexing Rate: 604.83 /s**
- **Latency: 0.32 ms**

실험군

- Elasticsearch Ingest node **1,**
- Master node(Data node) **1,**
- Data-only node **1**
- Primary shard **2,**
- Replica shard **1,**
- **Static** mapping
- **Unnested data**
- **Indexing Rate: 1983.51 /s**
- **Latency: 0.07 ms**

Indexing Rate: 약 **3.28**배 증가

Latency: 약 **4.57**배 감소

Limits

한계점

- PC가 한 대라 구성의 이점을 누리기 어려움
- SSD를 이용하기 때문에 GC로 인한 성능 편차 존재
- 실험의 용이성을 위해 데이터 스트림을 일정하게 하여 실제 환경과 거리감 존재
- 검색 성능을 고려하지 않음

Reference

참고문헌

- Elastic 가이드북(https://esbook.kimjmin.net/)
- Elastic Discuss(https://discuss.elastic.co/)
- 실시간 인덱싱을 위한 Elasticsearch 구조를 찾아서 (https://techblog.woowahan.com/7425/)
- Feasibility Analysis of Big Log Data Real Time Search Based on Hbase and ElasticSearch, Jun Bai, 2013.7
- An Optimization Method for Elasticsearch Index Shard Number, Bizhond Wei, 2020 CIS
- Design and Implementation of Elasticsearch for Media Data, Lu Han, 2020 ICCEA

Q&A