Elasticsearch 성능 최적화와 모니터링(6회차)

목차

- Elasticsearch 색인 성능 최적화
- Elasticsearch 검색 성능 최적화
- Elasticsearch 모니터링
- Q & A

Elasticsearch 색인 성능 최적화

static mapping 적용하기 - 미리 정해놓은 스키마로 리소스 절약하기

_all 필드 - 니즈가 없으면 쓰지 말아야 할 기능들

refresh_interval 변경하기 - 색인 된 데이터를 검색결과에 반영하기

Elasticsearch 색인 성능 최적화

색인은 디스크에 write 하는 I/O job

어떤 방식으로 색인하는지가 I/O 의 빈도 혹은 부하를 결정

인덱스 mapping 스키마를 미리 적절히 정의하는 것만으로도 성능 향상

Elasticsearch 색인 성능 최적화 - static mapping 적용하기

Mapping

- 문서의 필드 데이터를 어떤 타입으로 저장할지 결정하는 과정
- string 데이터에 대해 text field 로 쓸지, keyword field 로 쓸지
- numeric type 은 short field 로 쓸지, long field 로 쓸지
- date type 은 어떻게 정의해서 쓸지
- ES 는 인덱스의 매핑을 미리 결정하지 않으면 알아서 문서의 필드를 읽어들여 dynamic 하게 매핑을 진행

Elasticsearch 색인 성능 최적화 - static mapping 적용하기 (text)

Field Datatype
- field type - text, keyword, date, long, double, boolean, …

Text field type
- email 본문 같은 full-text 로 색인 되는 필드타입
- 분석기를 통해 단어로 검색 가능하도록 색인됨
- sorting 및 aggregation 에 사용되지 않음
- analyzer 를 통해 생성된 토큰으로 검색 가능

Elasticsearch 색인 성능 최적화 - static mapping 적용하기 (text)

```
PUT text_index1
{
  "mappings": {
    "properties": {
      "title": {
        "type": "text"
      }
    }
  }
}
```

Elasticsearch 색인 성능 최적화 - static mapping 적용하기 (keyword)

Keyword field type

- email 주소나 호스트 네임같은 구조화된 내용으로 색인되는 필드 타입
- Analyze 되지 않음
- 필드의 전체 데이터가 하나의 토큰으로 생성됨
- sorting, aggregation 지원
- 색인 될 때 토큰이 전체 value 로 구성되어 Exact value 로만 검색됨

Elasticsearch 색인 성능 최적화 - static mapping 적용하기 (keyword)

```
PUT keyword_index
{
  "mappings": {
    "properties": {
      "title": {
        "type": "keyword"
      }
    }
  }
}
```

Elasticsearch 색인 성능 최적화 - static mapping 적용하기 (keyword)

text vs keyword

- dynamic mapping 으로 string 이 들어오면
  text 와 keyword 둘 다 사용 가능
- fields 로 multi field 세팅
- multi field 는 test.keyword 로 질의
- 색인 될 때 text 는 analyzer 를 거치게 됨
- keyword 로 사용할 의도였다면 이러한 분석은 낭비
- keyword 로 사용할 string 이라면 keyword type 으로 매핑

```
{
  "test": {
    "aliases": {},
    "mappings": {
      "_doc": {
        "properties": {
          "test": {
            "type": "text",
            "fields": {
              "keyword": {
                "type": "keyword",
                "ignore_above": 256
              }
            }
          }
        }
      }
    },
```

Elasticsearch 색인 성능 최적화 - static mapping 적용하기

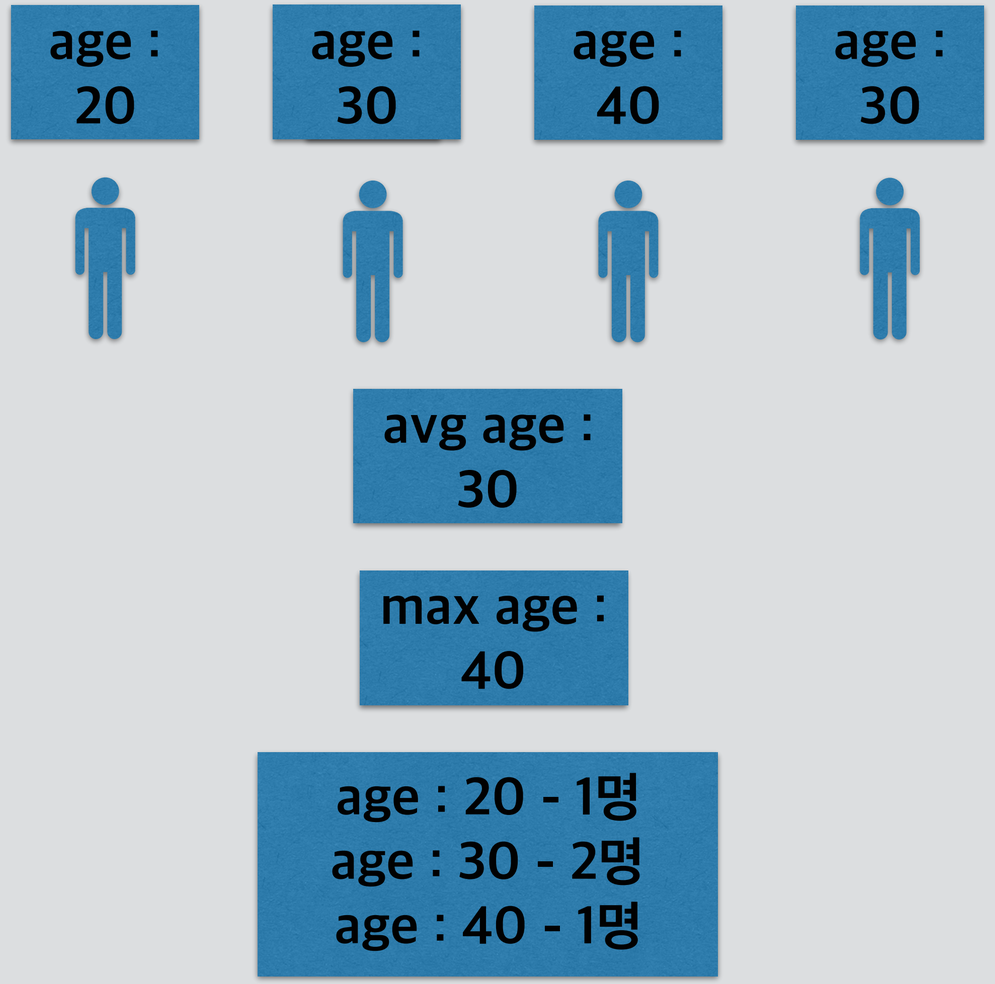
text vs keyword

항목	text	keyword
input	String	String
analyze	O	X
Full-Text Search	O	X
Sorting	X	O
Aggregation	X	O

Elasticsearch 색인 성능 최적화 - static mapping 적용하기

Aggregation

- 검색을 기반으로 수집된 데이터에 대해 집계/통계를 내는 프레임워크
- 메일 본문같은 긴 내용은 집계/통계를 내기 어려운 데이터
- 수치나 특정 문자로 한정된 데이터들의 집계를 내는 역할
- keyword 필드로 정의된 데이터에 대해 집계
- Kibana 에서 집계/통계를 낼 때 주로 사용
- Metric / Bucket Aggregation



Elasticsearch 색인 성능 최적화 - static mapping 적용하기 (date)

date field type

- 형식이 지정된 날짜가 포함 된 문자열 (예 : "2015-01-01"또는 "2015/01/01 12:10:30”)
- milliseconds 를 나타내는 epoch_millis (예 : 1420070400001)
- 허용할 타입만 지정도 가능

Elasticsearch 색인 성능 최적화 - static mapping 적용하기 (date)

date type 으로 매핑 생성

- Dynamic mapping 시 처음에 입력되는 형태로 인식
- 최초 epoch_millis 로 색인되면 long type 으로 Dynamic mapping이 되어 date 형태는 색인되지 않음

```
PUT date_index
{
  "mappings": {
    "properties": {
      "date": {
        "type": "date"
      }
    }
  }
}
```

Elasticsearch 색인 성능 최적화 - static mapping 적용하기 (date)

```
POST date_index/_doc
{ "date": "2015-01-01" }

POST date_index/_doc
{ "date": "2015-01-01T12:10:30Z" }

POST date_index/_doc
{ "date": 1420070400001 }

GET date_index/_search
{
  "sort": { "date": "asc"}
}
```

Elasticsearch 색인 성능 최적화 - static mapping 적용하기 (date)

사용자가 정의한 format 으로만 색인되도록 매핑 설정

```
PUT date_index_format
{
  "mappings": {
    "properties": {
      "date": {
        "type": "date",
        "format": "yyyy-MM-dd HH:mm:ss||yyyy-MM-dd||epoch_millis"
      }
    }
  }
}
```

Elasticsearch 색인 성능 최적화 - static mapping 적용하기

Numeric field type

- 숫자 형식의 필드 타입
- 적절하게 맞추면 좋으나 동일 타입으로 통일을 권고
- dynamic 매핑은 long

Value Type	supported
long	64-bit integer (-2^63 ~ 2^63 -1)
integer	32-bit integer (-2^31 ~ 2^31 -1)
short	16-bit integer (-32,768 ~ 32767)
byte	8-bit integer (-128 ~ 127)
double / float	64-bit / 32-bit IEEE 754 floating point number

Elasticsearch 색인 성능 최적화 - static mapping 적용하기

hierarchical nature of JSON field type

- object 나 list 같이 계층구조의 데이터는 properties 라 불리는 서브 필드에 포함됨

- properties 는 서브 필드가 사용되는 경우

1) 매핑을 정의해서 인덱스를 만들 때

2) 매핑 타입을 추가할 때(기존 필드 이름과 겹치지 않는 선에서만 지원)

3) 매핑을 정의하지 않고 문서를 색인 할 때 Elasticsearch 가 dynamic 하게 매핑을 정의할 때

4) 구조화된 object, list 같은 타입을 정의할 때

Elasticsearch 색인 성능 최적화 - static mapping 적용하기 (object)

```
PUT object_index
{
  "mappings": {
    "properties": {
      "region": {
        "type": "keyword"
      },
      "manager": {
        "properties": {
          "age": {
            "type": "integer"
          },
          "name": {
            "properties": {
              "first": {
                "type": "keyword"
              },
              "last": {
                "type": "keyword"
              }
            }
          }
        }
      }
    }
  }
}
```

Elasticsearch 색인 성능 최적화 - static mapping 적용하기 (object)

object type indexing

```
PUT object_index/_doc/1
{
   "region": "US",
   "manager": {
      "age":     30,
      "name": {
         "first": "John",
         "last":  "Smith",
         "full": "Smith John"
      }
   }
}
```

Elasticsearch 색인 성능 최적화 - static mapping 적용하기 (object)

object type searching

```
GET object_index/_search
{
   "query": {
      "match": {
         "manager.name.first": {
            "query": "John"
         }
      }
   }
}
```

Elasticsearch 색인 성능 최적화 - static mapping 적용하기 (nested)

nested type mapping

```
PUT nested_index
{
  "mappings": {
    "properties": {
      "user": {
        "type": "nested"
      }
    }
  }
}
```

Elasticsearch 색인 성능 최적화 - static mapping 적용하기 (nested)

- nested 로 정의한 매핑은 색인이 되는 순간 다시 dynamic mapping 으로 색인
- static mapping 으로 별도 지정

```
PUT nested_index/_doc/1
{
  "user" : [
      {
         "first" : "John",
         "last" :  "Smith"
      },
      {
         "first" : "Alice",
         "last" :  "White"
      }
   ]
}
```

Elasticsearch 색인 성능 최적화 - static mapping 적용하기 (nested)

- nested 필드는 nested query 로 검색

```
GET nested_index/_search
{
  "query": {
    "nested": {
      "path": "user",
      "query": {
        "bool": {
          "must": [
            {
              "match": {
                "user.first": "John"
              }
            },
            {
              "match": {
                "user.last": "Smith"
              }
            }
          ]
        }
      }
    }
  }
}
```

Elasticsearch 색인 성능 최적화 - static mapping 적용하기

Static Mapping more..
https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping-types.html

Elasticsearch 색인 성능 최적화 - _all field

_all field
- 도큐멘트의 모든 필드의 value 를 합치는 필드
- 검색을 할 때 사용되며, 세그멘트에 저장되지는 않지만 힙 영역에 올라가는 데이터
- text field 로 analyze 되는 필드
- 6.0 부터 deprecated

```
{
  "first_name": "John",
  "last_name": "Smith",
  "date_of_birth": "1970-10-24"
}
```

```
{
  "first_name": "John",
  "last_name": "Smith",
  "date_of_birth": “1970-10-24",
   "_all": "John Smith 1970-10-24"
}
```

Elasticsearch 색인 성능 최적화 - _all field

- _all field 를 5버전 이하에서 사용한다면 disable 권고
- 템플릿에 _default_ type 을 통해 전체 인덱스 disable

```
PUT allfield
{
   "mappings": {
      "_default_": {
         "_all": {
            "enabled": false
         }
      }
   }
}
```

- 7.x 버전에서 removed 됨
- _default_ field 도 함께 removed 됨

Elasticsearch 색인 성능 최적화 - _all field

- 니즈가 있다면 copy_to 기능 권고

```
PUT copy_index
{
  "mappings": {
    "properties": {
      "first_name": {
        "type": "text",
        "copy_to": "full_name"
      },
      "last_name": {
        "type": "text",
        "copy_to": "full_name"
      },
      "full_name": {
        "type": "text"
      }
    }
  }
}
```

Elasticsearch 색인 성능 최적화 - _all field

```
POST copy_index/_doc
{
    "first_name": "John",
    "last_name": "Smith"
}

GET copy_index/_search
{
    "query": {
        "match": {
            "full_name": "John Smith"
        }
    }
}
```

Elasticsearch 색인 성능 최적화 - _all field

More _all field..
https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping-all-field.html
https://www.elastic.co/guide/en/elasticsearch/reference/current/copy-to.html

Elasticsearch 색인 성능 최적화 - refresh_interval 변경하기

refresh_interval
- 메모리 캐시 버퍼 영역으로부터 세그멘트에 도큐멘트를 저장하는 주기
- refresh 가 되면 저장된 도큐멘트는 검색 가능한 상태로 변경됨
- 온라인상에서 인덱스별 세팅 가능

```
PUT refresh/_settings
{
   "index" : {
      "refresh_interval" : "1s"
   }
}
```

Elasticsearch 색인 성능 최적화 - refresh_interval 변경하기

- -1 로 설정하면 disabled, null 로 설정하면 1s 로 초기화

- interval 을 길게 가져갈수록 I/O 빈도가 낮아져 성능 향상

- 메모리 버퍼의 용량을 고려하여 interval 을 설정

```
PUT refresh/_settings
{
   "index" : {
      "refresh_interval" : -1
   }
}
```

Elasticsearch 색인 성능 최적화 - refresh_interval 변경하기

- refresh_interval 을 끄고 색인할 때 직접 제어도 가능
- refresh=true 로 설정한 문서만 세그먼트로 내림

```
PUT refresh/_settings
{
    "index" : {
        "refresh_interval" : -1
    }
}

PUT refresh/_doc/1?refresh=true
{ "msg": "first doc" }
```

Elasticsearch 색인 성능 최적화 - refresh_interval 변경하기

More refresh_interval
https://www.elastic.co/guide/en/elasticsearch/reference/current/index-modules.html#dynamic-index-settings
https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-refresh.html
https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-refresh.html

Elasticsearch 색인 성능 최적화 - 그 외..

Document id 없이 POST 로 색인 권고
- PUT 을 통해 doc id 를 설정한 채로 색인을 할 때 ES 는 해당 id 의 문서가 있는지를 먼저 체크
- 문서가 많아질수록, 인덱스가 커질수록 해당 부하가 커짐

Bulk Indexing 활용 권고
- 가능하면 단일 색인 요청보다는 한번에 많은 문서를 Bulk Indexing 하는 것이 더 효율적
- 단일 노드, 단일 샤드 환경에서 적절한 bulk size 측정 후 진행 권고
- request reject 나 gc 상황을 봐가며 적정 수치 확인

상황에 따른 Replica shard 를 0으로 설정
- 한 번에 많은 데이터를 적재해야하는 상황에서는 replica 를 0 으로 두고 색인
- 필요하면 색인이 전부 완료된 이후에 리플리카를 추가하는 형태로 작업하는 것을 권고

Heap 에 할당하고 남은 메모리가 버퍼 캐시로 활용될 수 있도록 충분히 확보

Index Buffer Size 를 상황에 맞게 설정 권고
https://www.elastic.co/guide/en/elasticsearch/reference/current/indexing-buffer.html

Elasticsearch 색인 성능 최적화 - 그 외..

More Indexing Performance..
https://www.elastic.co/guide/en/elasticsearch/reference/current/tune-for-indexing-speed.html

Elasticsearch 검색 성능 최적화

쿼리 튜닝 하기 - 검색에 유리한 튜닝방법

샤드 배치 결정하기 - 검색성능을 위해 샤드를 배치하는 노하우

Elasticsearch 검색 성능 최적화 - 쿼리 튜닝하기

multi field 로 검색을 해야할 때에는 가능한 적은 필드로 검색
- copy_to 를 이용하면 두개의 필드를 하나로 줄여 검색 가능

```
PUT copy_index
{
  "mappings": {
    "properties": {
      "first_name": {
        "type": "text",
        "copy_to": "full_name"
      },
      "last_name": {
        "type": "text",
        "copy_to": "full_name"
      },
      "full_name": {
        "type": "text"
      }
    }
  }
}
```

Elasticsearch 검색 성능 최적화 - 쿼리 튜닝하기

numeric field 에 대해 keyword field 로 색인 고려

```
PUT nonum
{ "mappings": { "_doc": { "properties": { "account_no": { "type": "keyword" } } } } }

POST nonum/_doc
{  "account_no": "12345" }
POST nonum/_doc
{  "account_no": "22345" }

GET nonum/_search
{ "query": { "term": { "account_no": "12345" } } }

GET nonum/_search
{ "query": { "range": { "account_no": { "gte": 12300 } } } }
```

Elasticsearch 검색 성능 최적화 - 쿼리 튜닝하기

- 집계 등의 수치 계산은 안 됨

```
POST nonum/_search?size=0
{
   "query" : {
      "constant_score" : {
         "filter" : {
            "match_all" : {}
         }
      }
   },
   "aggs" : {
      "hat_prices" : { "sum" : { "field" : "account_no" } }
   }
}
```

Elasticsearch 검색 성능 최적화 - 쿼리 튜닝하기

wildcard 는 꼭 써야하는지 검토

- token 으로 분리되는 용어는 wildcard 대신 match 를 써도 충분할 수 있음

```
GET bank/_search
{ "query": { "query_string" : { "query": "Flee*" } } }

GET bank/_search
{ "query": { "match": { "address": "Fleet" } } }
```

Elasticsearch 검색 성능 최적화 - 쿼리 튜닝하기

exact match 검색을 할 때에는 match 대신 term 쿼리 사용
- 불필요한 analyze 를 제외하여 성능 향상

```
PUT termquery
{ "mappings": {  "properties": {  "gender": {  "type": "keyword"  } } } }

POST termquery/_doc
{ "gender": "F" }

GET termquery/_search
{ "query": { "match": { "gender": { "query": "F" } } } }

GET termquery/_search
{ "query": { "term": { "gender": { "value": "F" } } } }
```

Elasticsearch 검색 성능 최적화 - 쿼리 튜닝하기

query context 와 filter context 를 적절히 활용
- filter context 쿼리는 스코어가 의미가 없는 쿼리
- filter 절에 넣어 스코어를 계산하는 단계를 없애면 성능 향상
- filter 절 안에 쿼리된 문서들은 캐싱됨
  (https://www.elastic.co/guide/en/elasticsearch/reference/current/query-cache.html)

```
GET bank/_search
{
   "query": {
      "bool": { "must": [ { "term":  { "gender.keyword": "F" } } ] }
   }
}
GET bank/_search
{
   "query": {
      "bool": { "filter": [ { "term":  { "gender.keyword": "F" } } ] }
   }
}
```

Elasticsearch 검색 성능 최적화 - 쿼리 튜닝하기

- bool 쿼리는 각 쿼리들이 interleave 하게 실행
- 쿼리 순서는 크게 상관이 없음

```
GET bank/_search
{
   "query": { "bool": {
         "must": [ { "match": { "state": { "query": "MI" }}}],
         "filter": [ { "range": { "age": { "lte": "30" }}}] }
   }
}

GET bank/_search
{
   "query": { "bool": {
         "filter": [ { "range": { "age": { "lte": "30" }}}],
         "must": [ { "match": { "state": { "query": "MI" }}}] }
   }
}
```

Elasticsearch 검색 성능 최적화 - 쿼리 튜닝하기

서비스 전 쿼리 캐시 warmup
- 처음 로딩되는 데이터는 Heap 에 올라와있지 않다면 버퍼 캐시에 데이터가 있는지 확인

- 버퍼 캐시에도 없다면 디스크에 저장된 segment 를 찾아
  1) query context 에서 사용되었다면 Heap 에 데이터를 올리고 리턴
     -> 이 경우는 gc age bit 에 의해 정리될 가능성이 높음
  2) filter context 에서 사용되었다면 자주 사용된 쿼리에 대해 버퍼 캐시에 캐싱
     -> 문서 10000개 미만을 포함하는 세그먼트, 혹은 전체 인덱스 크기의 3% 미만의 크기에 대해서는 캐싱 안함

- 자주 사용되는 쿼리로 서비스 전에 filter context를 사용해 미리 쿼리를 날려두어 버퍼 캐시에 데이터를 로딩

- 전체 데이터가 로딩될 만큼 캐시 사이즈가 크다면 filter context 에 와일드카드 쿼리로 전체 데이터를 로딩

- 세그먼트를 직접 올리는 것이 실효가 가장 크지만 최초 로딩 시 지연 발생, 검토 후 적용 필요
https://www.elastic.co/guide/en/elasticsearch/reference/current/preload-data-to-file-system-cache.html

Elasticsearch 검색 성능 최적화 - 쿼리 튜닝하기

가능하면 아래 구성은 사용하지 않는 것을 권고
- Parent / Child 구조로 join job 이 들어가는 구조
- nested datatype field 를 사용한 매핑을 사용하는 구조
- elastic search 에서 제공하는 스크립트를 사용하여 색인하고 검색하는 구조

range 쿼리에서 date 를 기준으로 범위를 정할 때 시간을 반올림하여 검색하는 것을 권고
- now-1h -> now-1h/m

replica 를 적절하게 두어 검색 성능을 높이도록 권고

Elasticsearch 검색 성능 최적화 - 샤드배치 결정하기

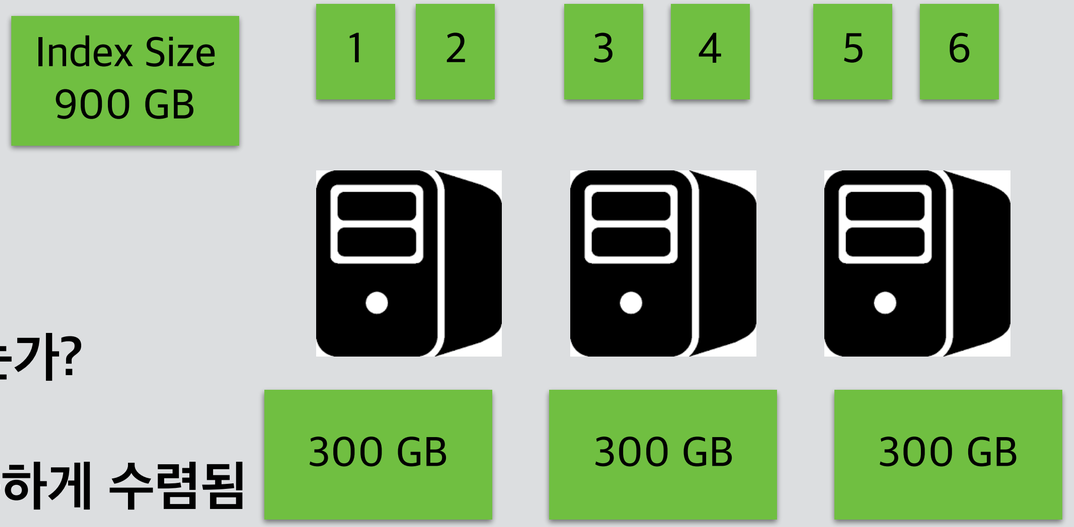
한 번 설정한 샤드 개수는 변경 불가

- 처음부터 샤드 개수를 잘 설정해야 클러스터 성능이 좋아짐

샤드 개수를 결정할 때 고려해야할 점

1) 인덱스가 생성될 때 전체 노드가 색인과 검색에 참여할 수 있는가?

- 분산 처리를 위해 샤드가 노드에 고루 분배되어야 함
- 노드 개수의 배수로 샤드 개수를 가져가면 노드 별 용량도 비슷하게 수렴됨
- 데이터 노드 증설 시 증설된 노드가 샤드를 받을 수 있어야 함
- 최소 공배수의 개념을 잘 활용하여 산정



Elasticsearch 검색 성능 최적화 - 샤드배치 결정하기

2) 하나의 데이터 노드에 적정 샤드 개수가 저장되도록 설계되었는가?

- Heap Size 1GB 의 Heap 당 20~25 개 정도의 샤드 설정

- 데이터노드에 Heap Size 를 16G 정도로 설정, 노드 당 320~400 정도의 샤드 설정

- ES 커뮤니티에서 권고하는 일반 설정

- 노드 당 300 개의 샤드를 넘지 않도록 권고(현실적으로 불가능한 케이스가 대다수)

- use case 별로 상이할 수 있음

Elasticsearch 검색 성능 최적화 - 샤드배치 결정하기

3) 하나의 샤드가 적절한 용량만 저장되도록 설계되었는가?
- 하나의 샤드 적정 크기는 보통 20~40GB 를 권고

- 작을수록 좋지만 너무 작게 나누면 전체 샤드개수가 증가 (2번 조건과 상충)

- 인덱스 전체 크기가 산정되면 하나의 샤드 크기가 적절하게 나눠질 수 있도록 설정

ex) daily 로 저장되는 인덱스의 총 크기가 500G
샤드 1개가 20G 만 가져갈 수 있도록 25개의 샤드 세팅

4) 클러스터의 전체 샤드 개수는 적절한가?
- 클러스터 전체 샤드 개수가 늘면 그만큼 클러스터 내 노드가 알고있어야 하는 정보들도 많아짐

- 실제 운영 환경에서 20,000 개가 넘어가는 순간부터 성능이 급격히 감소

Elasticsearch 검색 성능 최적화 - 샤드배치 결정하기

5) 꼭 실시간으로 replica shard 가 필요한가?

- replica shard 는 실제 문서 만큼의 추가 복제 비용이 발생

- 원본이 별도로 있다면 색인 될 때는 replica 를 없애면 성능 향상

- 데이터 유실의 위험이 있으니 상황을 잘 고려하여 판단

Elasticsearch 검색 성능 최적화 - 샤드배치 결정하기

클러스터 사이징 시나리오

1. daily 로 400G 정도로 색인되는 문서가 들어올 예정
2. 데이터 보존을 위해 replica 필요
3. 10일 정도 데이터 보존 희망

사이징 진행사항
- 먼저, 총 클러스터 인덱스 용량을 산정
- 400GB * 2(replica) * 10 days = 8 TB
- 하나의 샤드에 20G 적재로 산정, 20개의 primary 샤드 산정
- 클러스터 전체 샤드 개수는 20 * 2(replica) * 10 = 400 개
- 디스크가 SSD 2T 라고 가정하면 8TB 를 저장하고 20% 정도 여유 용량 확보를 위해 5대 구매
- 인덱스 샤드가 replica 포함 총 40개 이므로, 노드 당 샤드 8개 할당됨
- 하루 400GB 의 인덱스를 모두 힙과 버퍼에 담으려면 총 800G 의 메모리 필요
- 5대로 나누면 160G 의 메모리 필요 (이렇게 커지면 하루 데이터 모두를 메모리에 담는 것은 포기..)
- OOP 구조체를 고려하여 64G 메모리로 구매, Heap 을 30G 로 설정
- 노드 당 Heap 30GB 에 10일동안 노드 당 전체 샤드는 80개 (노드 1GB 당 3개 미만)

Elasticsearch 검색 성능 최적화 - _forcemerge

_forcemerge API
- segment 를 강제로 병합하는 API
- 색인 중인 인덱스에는 사용 비추
- 색인이 끝난 인덱스는 하나의 segment 로 merge 를 추천
- I/O 비용이 크기 때문에 색인이나 검색이 없는 시간대에 진행

```
POST shakespeare/_forcemerge?max_num_segments=1
```

Elasticsearch 검색 성능 최적화 - Routing

Routing
- routing key 를 통해 특정 샤드로 색인하는 방법
- routing key 를 색인 샤드 할당 알고리즘을 통해 할당

shard = hash(routing) % number_of_primary_shards

- routing key 에 의해 특정 샤드로 들어간 데이터는
  검색 시에 검색이 아닌 문서 가져오기 과정을 거쳐 성능이 향상됨

ex) 메일 서비스 검색엔진을 ES 로 사용하고, 사용자별로 메일을 검색
  사용자 id 를 routing key 로 주고 메일 본문을 색인
  검색시에 사용자 id 를 routing key 로 찾으면 특정 샤드만 검색

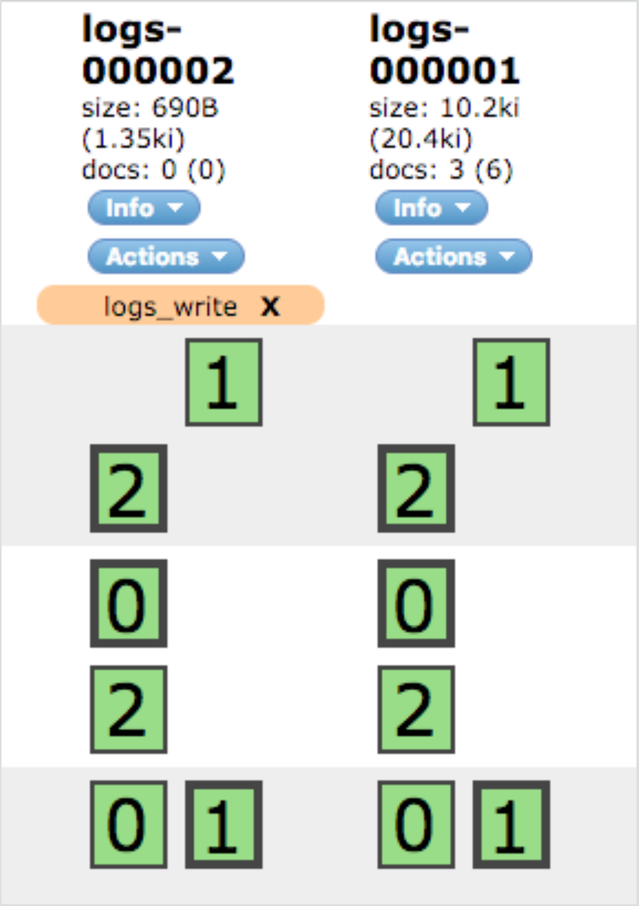
```
POST rindex/_doc?routing=user1
{  "title": "This is a document" }

POST rindex/_doc?routing=user1
{  "title": "This is a document for fastcampus" }

GET rindex/_search?routing=user1
{  "query": {  "match": {  "title": "fastcampus" } } }
```

Elasticsearch 검색 성능 최적화 - Routing

Rollover api

- 라우팅으로 커진 샤드를 캐어하는 방법
- 특정 조건이 맞으면 새로운 인덱스를 생성하고 alias 를 옮겨주는 기능

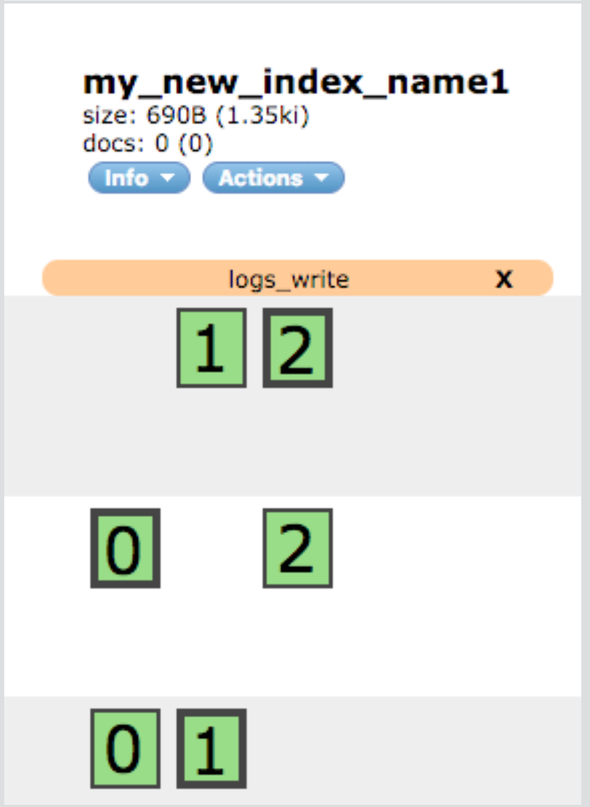
```
PUT logs-000001
{
   "aliases": {
      "logs_write": {}
   }
}
POST logs_write/_rollover
{
   "conditions": {
      "max_age":   "7d",
      "max_docs":  2,
      "max_size":  "5gb"
   }
}
```



Elasticsearch 검색 성능 최적화 - Routing

- 특정 이름을 지정하여 생성도 가능
- ?dry_run 을 실행하면 모의실행 가능

```
POST logs_write/_rollover/
my_new_index_name1
{
  "conditions": {
    "max_age":   "7d",
    "max_docs":  2,
    "max_size": "5gb"
  }
}
```



Elasticsearch 검색 성능 최적화 - Thread Pool Size

Thread Pool Size
- 헤비한 색인과 검색을 같이 해야하는 경우 색인에 Thread Pool 이 과하게 할당하여 검색 성능이 떨어짐
- Bulk job 에 할당되는 thread pool size 를 조절하여 검색성능 보장

```
# /etc/elasticsearch/elasticsearch.yml
thread.bulk.size: 16
```

Elasticsearch 검색 성능 최적화

More Searching Performance
https://www.elastic.co/guide/en/elasticsearch/reference/current/tune-for-search-speed.html

Elasticsearch 모니터링

rejected - 데이터의 누락이 발생하는 순간

Cat API - 클러스터의 여러가지 상태 살펴보기

Stats, Nodes API - 클러스터의 리소스 사용 지표 확인하기

Elasticsearch 모니터링 - rejected

rejected
- 노드 내에서 스레드의 메모리 사용을 관리하기 위해 여러개의 스레드풀 사용
- index, get, search, write(bulk) 등 여러개의 스레드 풀 존재
- 각각의 스레드풀은 대기열 큐를 두어 대기중인 요청을 보관

GET _nodes/thread_pool

구분	Index	get	search	write(bulk)
Thread pool size	Core 수	Core 수	(Core 수 * 3 / 2) + 1	Core 수
Queue size	200	1000	1000	200

Elasticsearch 모니터링 - rejected

- 대기열 큐가 꽉차면 그 이후부터는 요청이 reject 됨
- _nodes/stats 를 통해 모니터링

```
GET _nodes/stats
      "thread_pool": {
        "analyze": {
          "threads": 0,
          "queue": 0,
          "active": 0,
          "rejected": 0,
          "largest": 0,
          "completed": 0
...
```

Elasticsearch 모니터링 - rejected

- _cat/thread_pool?v 를 thread_pool 정보만 실시간으로 조회

```
GET _cat/thread_pool/search?v
node_name        name   active queue rejected
ben-tuto-data01  search      0     0        0
ben-tuto-master03 search     0     0        0
ben-tuto-master01 search     0     0        0
ben-tuto-master02 search     0     0        0
ben-tuto-data03  search      0     0        0
ben-tuto-data02  search      0     0        0
```

Elasticsearch 모니터링 - rejected

- /etc/elasticsearch/elasticsearch.yml 파일에 큐 사이즈 조절 가능

thread_pool.bulk.queue_size: 20000
thread_pool.search.queue_size: 20000

Elasticsearch 모니터링 - rejected

rejected more..
https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-threadpool.html

Elasticsearch 모니터링 - Cat API

Cat API
- Get Method 를 통해 클러스터, 노드, 인덱스 등의 상태를 확인해볼 수 있는 API

GET _cat/nodes
- 클러스터에 속한 node 들의 상태를 확인할 수 있는 명령

172.31.9.100 15 90 0 0.08 0.04 0.05 mdi * NXLq-qi
172.31.7.100 15 56 0 0.00 0.01 0.05 mdi - EWVXzG_

Elasticsearch 모니터링 - Cat API

- 뒤에 ?v 옵션을 추가로 주면 항목의 필드 이름 확인 가능

```
GET _cat/nodes?v
ip           heap.percent ram.percent cpu load_1m load_5m load_15m node.role master name
172.31.9.100           17          90   0    0.03    0.04     0.05 mdi       *      NXLq-qi
172.31.7.100           15          56   0    0.00    0.01     0.05 mdi       -      EWVXzG_
```

heap.percent - 사용중인 heap memory percentage
ram.percent - 사용중인 memory percentage
cpu - 사용중인 cpu 리소스 percentage
load_1m,5m,15m - 사용중인 load average 값
node.role - 노드의 role (m - master, d - data, i - injest)

Elasticsearch 모니터링 - Cat API

- 뒤에 &h= 를 통해 필요한 항목만 발췌 가능

```
GET _cat/nodes?v&h=ip,node.role
ip            node.role
172.31.9.100 mdi
172.31.7.100 mdi
```

- 뒤에 &s=ip:asc 를 통해 정렬 가능 (역순은 desc)

```
GET _cat/nodes?v&h=ip,node.role&s=ip:asc
ip            node.role
172.31.7.100 mdi
172.31.9.100 mdi
```

Elasticsearch 모니터링 - Cat API

- 뒤에 &format=json 을 통해 json 포맷으로 변경 가능

```
GET _cat/nodes?v&h=ip,node.role&s=ip:asc&format=json
[
  {
    "ip": "172.31.7.100",
    "node.role": "mdi"
  },
  {
    "ip": "172.31.9.100",
    "node.role": "mdi"
  }
]
```

Elasticsearch 모니터링 - Cat API

GET _cat/allocation?v
- 샤드 및 디스크 용량 관련 사항 조회

```
shards disk.indices disk.used disk.avail disk.total disk.percent host         ip           node
    33       97.4kb     1.2gb      6.7gb      7.9gb           15 172.31.7.100 172.31.7.100 EWVXzG_
    34        6.5mb     4.1gb     11.8gb     15.9gb           25 172.31.9.100 172.31.9.100 NXLq-qi
```

shards - 샤드 개수
disk.indices - 인덱스가 사용하고 있는 디스크 용량
disk.used - 실제 시스템에서 사용된 디스크 용량
disk.avail - 실제 시스템에서 사용 가능한 디스크 용량
disk.total - 실제 시스템에서 전체 디스크 용량
disk.percent - 실제 시스템의 디스크 용량 사용률

Elasticsearch 모니터링 - Cat API

GET _cat/shards?v
- 샤드 정보 조회

```
index                shard prirep state    docs  store ip           node
ttt                  2     r      STARTED     0   261b 172.31.9.100 NXLq-qi
ttt                  2     p      STARTED     0   261b 172.31.7.100 EWVXzG_
ttt                  3     p      STARTED     1  3.4kb 172.31.9.100 NXLq-qi
...
tdex1                2     r      STARTED     1  2.6kb 172.31.9.100 NXLq-qi
tdex1                2     p      STARTED     1  2.6kb 172.31.7.100 EWVXzG_
...
```

shard - 샤드 넘버
prirep - 프라이머리, 리플리카 샤드 여부
state - 샤드 상태
docs - 도큐멘트 개수
store - 저장된 사이즈

GET _cat/shards/ttt?v - 인덱스 별로도 샤드 정보 조회 가능

Elasticsearch 모니터링 - Cat API

GET _cat/indices?v
- 인덱스 정보 조회

```
health status index uuid                   pri rep docs.count docs.deleted store.size pri.store.size
green  open   ydex  gELeTBLXSYKM9rqdrOz_5Q   5   1          1            0     11.7kb         5.8kb
green  open   nomap   ykVP6PM6T-uUFGj_t4gEoA   5   1          1            0      8.9kb         4.4kb
```

health - 인덱스 헬스 상태 (green, yellow, red)
status - 인덱스 open 상태 (open, close)
pri - 프라이머리 샤드 개수
rep - 리플리카 샤드 개수
docs.count - 도큐멘트 개수
docs.deleted - 삭제된 도큐멘트 개수
store.size - 리플리카를 포함한 실제 저장된 사이즈
pri.store.size - 프라이머리 실제 저장된 사이즈

GET _cat/indices/ydex?v - 인덱스 별로 조회 가능

Elasticsearch 모니터링 - Cat API

GET _cat/health?v
- 클러스터 헬스체크 상태

```
epoch      timestamp cluster          status node.total node.data shards pri relo init unassign pending_task
max_task_wait_time active_shards_percent
1539602599 20:23:19  ben-lect-es-tmp green          6         3     98  49    0    0        0             0
100.0%
```

status - 클러스터 전체 상태(green, yellow, red)
node.total - 클러스터에 속해져있는 전체 노드 개수
node.data - 데이터노드 role 을 가지는 노드 개수
shards - 클러스터에 저장된 전체 샤드 개수
pri - 클러스터에 저장된 프라이머리 샤드 개수
relo - 현재 재 할당중인 샤드 개수
init - 샤드 상태 변경 전 초기화를 거치고 있는 샤드 개수
unassign - 할당되지 않은 샤드 개수
pending_tasks - 상태가 변경되는 과정에서 지연이 일어나고 있는 task 개수
active_shards_percent - 안정적인 상태에서 운영중인 샤드의 percentage

Elasticsearch 모니터링 - Cat API

GET _cat/templates?v
- 세팅된 템플릿 조회

```
name                        index_patterns             order  version
.monitoring-kibana          [.monitoring-kibana-6-*]   0      6040099
logstash-index-template     [.logstash]                0
template_2                  [te*]                      1
```

name - 템플릿 이름
index_patterns - 템플릿이 적용될 인덱스 패턴
order - 템플릿 오더 넘버
version - 템플릿 버전 정보

Elasticsearch 모니터링 - Cat API

Cat API More..
https://www.elastic.co/guide/en/elasticsearch/reference/current/cat.html

Elasticsearch 모니터링 - Stats API

Stats API

- Get Method 를 통해 전체 클러스터, 인덱스 별 프라이머리 샤드와 전체 샤드의 각종 지표를 확인하는 API 로 클러스터 모니터링 지표로 활용

- docs.count - 저장된 도큐멘트 개수
- docs.deleted - 삭제된 도큐멘트 개수
- store - 저장된 용량(bytes)
- indexing.index_total - 색인된 횟수
- indexing.index_time_in_millis - 색인 할 때 소요된 milliseconds
- get.total - get 한 횟수
- get.time_in_millis - get 할 때 소요된 milliseconds

Elasticsearch 모니터링 - Stats API

- search.(query/fetch)_total - search 에서 (query/fetch) 를 요청한 횟수
- (query/fetch)_time_in_millis - (query/fetch) 에 소요된 milliseconds

- A = 1초 전 search.query_total - 현재 search.query_total = 1초 간 search query total 건수
-> 초당 search query 의 rate
- B = 1초 전 search.query_time_in_millis - 현재 search.query_time_in_millis = 1초 간 query time_in_millis
-> 초당 search query 의 latency

그 외 merges, refresh 등의 수치도 제공
위 데이터를 primaries기준, total 기준으로 제공하고, 인덱스별로도 해당 데이터를 제공

Elasticsearch API 활용하기 - Nodes API

Nodes API
- 각 노드의 uuid 를 key 로 하여 elasticsearch.yml, jvm.options, plugin 정보 등을 제공

- 노드 이름, ip 정보, ES 버전, 노드 roles 정보

```
GET _nodes
  "nodes": {
    "NXLq-qizQNe8GX2cLqI2EQ": {
      "name": "NXLq-qi",
      "transport_address": "172.31.9.100:9300",
      "host": "172.31.9.100",
      "ip": "172.31.9.100",
      "version": "6.4.2",
...
      "roles": [
        "master",
        "data",
        "ingest"
      ],
```

Elasticsearch API 활용하기 - Nodes API

- elasticsearch.yml 에 세팅된 정보 외 추가정보

```
"settings": {
  "pidfile": "/var/run/elasticsearch/elasticsearch.pid",
  "cluster": {
    "name": "elasticsearch"
  },
  ...
  "path": {
    "data": [
      "/var/lib/elasticsearch",
   ],
    "logs": "/var/log/elasticsearch",
    "home": "/usr/share/elasticsearch"
  },
  ...
  "network": {
    "host": "0.0.0.0"
  }
```

Elasticsearch API 활용하기 - Nodes API

- jvm.options 에 세팅된 정보

```
"jvm": {
  …
  "version": "1.8.0_181",
  "vm_name": "OpenJDK 64-Bit Server VM”,
  …
  "input_arguments": [
    "-Xms1g",
    “-Xmx1g",
  …
},
"plugins": [],
```

Elasticsearch API 활용하기 - Nodes API

GET _nodes/stats
- _stats 에서 확인할 수 있었던 항목들을 노드별로 보여줌


```
"indices": {
  "docs": {
    "count": 13095,
    "deleted": 0
  },
  "store": {
    "size_in_bytes": 6836498
  },
  "indexing": {
    "index_total": 22,
    "index_time_in_millis": 85,
    ...
```

Elasticsearch API 활용하기 - Nodes API

- 노드의 시스템 지표도 함께 확인 가능


```
"os": {
  "timestamp": 1539609793534,
  "cpu": {
    "percent": 0,
    "load_average": {
      "1m": 0,
      "5m": 0.01,
      "15m": 0.05
    }
  },
  "mem": {
    "total_in_bytes": 3972685824,
    "free_in_bytes": 396218368,
    "used_in_bytes": 3576467456,
    "free_percent": 10,
    "used_percent": 90
  },
```

Elasticsearch API 활용하기 - Nodes API

- GC 에 대한 항목 확인 가능


```
"jvm": {
…
  "gc": {
    "collectors": {
      "young": {
        "collection_count": 708,
        "collection_time_in_millis": 4434
      },
      "old": {
        "collection_count": 3,
        "collection_time_in_millis": 190
      }
    }
  },
…
```

Elasticsearch API 활용하기 - Nodes API

- 파일시스템 사용량 확인 가능

```
"fs": {
        "timestamp": 1539609793535,
        "total": {
          "total_in_bytes": 17156800512,
          "free_in_bytes": 12746752000,
          "available_in_bytes": 12746752000
          ...
```

Elasticsearch API 활용하기 - Nodes API

Stats API More..
https://www.elastic.co/guide/en/elasticsearch/reference/current/cluster-nodes-stats.html#_nodes_stat

실습

Elasticsearch Monitoring
https://github.com/benjamin-btn/ES7-Tutorial/tree/master/ES-Tutorial-6

Q & A