



실무에 활용하는 Elasticsearch 검색엔진 구축

6일차 : ElasticSearch

ELK 구축하기

오늘의 아젠다



- 1. ELK 개요
- 2. ELK 설치
- 3. ELK 로그 분석
- 4. API 및 웹 활용 소스
- 5. 기타

1. ELK의 개요

ELK의 개요



실시간 로그 분석 기술 스택

Solr 진영에는 SILK와 BANANA (Solr + Logstash + Kibana)

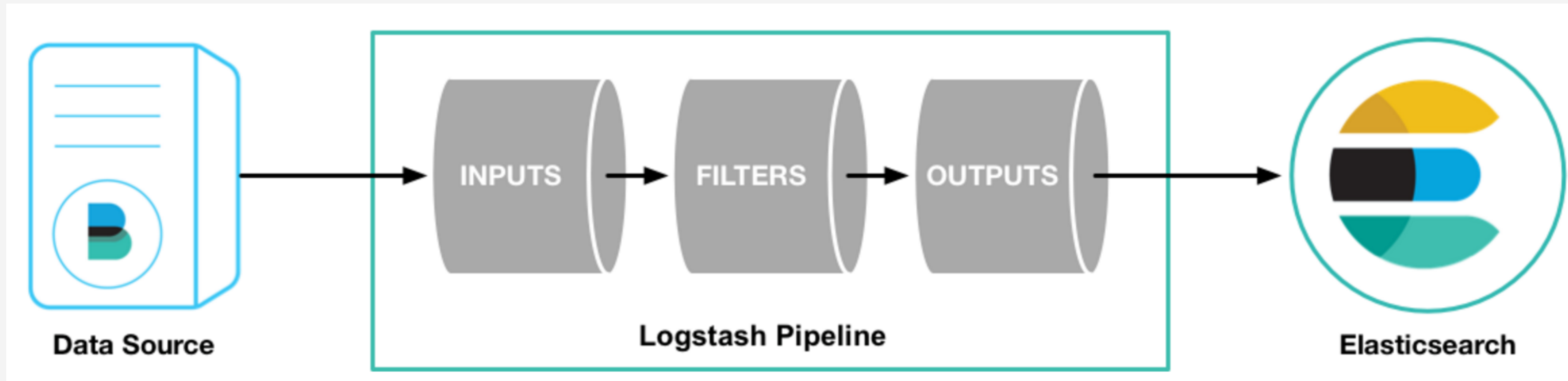
Logstash

- 데이터의 흐름을 관리하기 위해 개발된 오픈 소스 프로젝트
- 엘라스틱서치의 공식 패키지 구성 요소
- 아파치 라이선스 2.0 오픈소스
- JRuby로 작성 (자바 런타임 환경 필수, 1.7 이상)
- 다양한 방식으로 데이터 입/출력 가능

비슷한 기능의

Facebook의 스크라이브(Scribe)
아파치 프로젝트 Flume

Logstash



입력 : 다양한 경로로 부터 데이터를 읽어오는 작업

- 필터 : 읽어온 데이터를 가공하는 절차
- 출력 : 가공된 데이터를 다른 프로그램이나 채널로 입력

Logstash 요소들

Inputs

file : 파일시스템에 저장된 파일로부터 읽어옴.

읽어오는 형태는 UNIX tail -OF 의 결과물과 같음.

syslog : 보통 514번 포트로부터 들어오는 syslog 메세지들을 받아 RFC3164 형식으로 파싱함

redis : redis lists와 redis channels을 사용해서 redis server로부터 읽어옴

beats : Filebeat가 전송한 이벤트들.

이외에도 http, jdbc, log4j, s3, tcp 등 다양한 Input들이 존재

Filters

grok - 임의의 구조화된 텍스트로 파싱.

구조화되어 있지 않은 로그 데이터를 파싱할 때

Logstash에서 현재 사용할 수 있는 가장 최선의 방법

mutate - rename, remove, replace 등 일반적인 편집을 수행

drop - 완전히 삭제

clone - 복제본을 생성. 필드를 추가하거나 제거하는 것도 가능.

geoip - geo data, ip 주소 등 추가

Outputs

elasticsearch

file

graphite

statsd

Codecs

메세지를 손쉽게 구분하고, 전송할 수 있도록 도와주는 일종의 stream filters
어떻게 취급할 것인가?

json

msgpack

plain (text)

multiline -

java exception이나 stacktrace처럼 여러 줄로 이루어진
로그를 하나의 메시지로 취급하기 위한 코덱

Logstash Codec multiline

예시)

```
Exception in thread "main" java.lang.NullPointerException
    at com.example.myproject.Book.getTitle(Book.java:16)
    at com.example.myproject.Author.getBookTitles(Author.java:25)
    at com.example.myproject.Bootstrap.main(Bootstrap.java:14)
```

```
input {
  stdin {
    codec => multiline {
      pattern => "^\\s"
      what => "previous"
    }
  }
}
```

previous : 패턴과 일치하는 행이 이전행의 일부임

next : 패턴과 일치하는 행이 다음행의 일부임

주의점 : beat를 사용하는 경우 multiline 코덱을 사용하는 경우 logstash로 보내기 전에 처리해야함

Logstash 실행 방법

bin/logstash [options]

bin/logstash -f mypipeline.conf

nohup bin/logstash -f mypipeline.conf & (백그라운드 실행)

bin/logstash -f mypipeline.conf --config.reload.automatic (설정변경시 자동 리로드)

-f : 실행설정파일 경로

-e : 콘솔에서 바로 설정 정보 입력

예) bin/logstash -e input { stdin { type => stdin } }

-w : 파이프라인의 워커(스레드) 갯수를 지정(default CPU cores)

-b : 각 워커 스레드가 한번에 가져갈 수 있는 이벤트의 최대 양 지정 (default 125)

-u : 이벤트를 받아서 처리하는 사이의 시간 지정 (default is 250ms)

Logstash conf 파일

기본 골격

```
input {  
  # 입력 설정  
}
```

```
filter {  
  # 필터 설정  
}
```

```
output {  
  # 출력 설정  
}
```

조건식

```
if EXPRESSION {  
  ...  
} else if EXPRESSION {  
  ...  
} else {  
  ...  
}
```

==, !=, <, >, <=, >=

=~, !~

in, not in

예)

```
output {  
  # Send production errors to pagerduty  
  if [loglevel] == "ERROR" and [deployment] == "production"  
    pagerduty {  
      ...  
    }  
}
```

Logstash 값 형태

Array

users => [{id => 1, name => bob}, {id => 2, name => jane}]

Lists

path => ["/var/log/messages", "/var/log/*.log"]

uris => ["http://elastic.co", "http://example.net"]

Boolean

ssl_enable => true

Bytes

my_bytes => "1113" # 1113 bytes

my_bytes => "10MiB" # 10485760 bytes

my_bytes => "100kib" # 102400 bytes

my_bytes => "180 mb" # 180000000 bytes

Codec

codec => "json"

Hash

```
match => {  
  "field1" => "value1"  
  "field2" => "value2"  
  ...  
}
```

Number

port => 33

Password

my_password => "password"

String

name => "Hello world"

name => 'It\'s a beautiful day'

Logstash 공통 옵션

add_field : 필드를 추가함

add_tag : 태그를 추가함

enable_metric : 특정 플러그인의 메트릭정보를 수집하거나 수집하지 않음

id : 고유 아이디 값을 설정함

remove_field : 필드를 삭제함

remove_tag : 태그를 삭제함

Logstash Field vs Tag

Field : 문서나 이벤트의 하위 집합

필드가 정의 되면 E/S 입력시 필드가 생성됨

Tag : Filter 상의 판단을 하거나 분기를 하기 위한 속성

Logstash INPUT

```
input {  
  file {  
    path => "/var/log/nginx/access.log"  
    start_position => beginning  
  }  
  beats {  
    port => "5044"  
  }  
}
```

Logstash FILTER

Date 처리 필터

```
filter {  
  date {  
    match => [ "logdate", "MMM dd yyyy HH:mm:ss" ]  
  }  
}
```

Drop 필터

```
filter {  
  if [loglevel] == "debug" {  
    drop { }  
  }  
}
```

fingerprint 필터

```
filter {  
  fingerprint {  
    source => ["IP", "@timestamp", "message"]  
    method => "SHA1"  
    key => "0123"  
    target => "[@metadata][generated_id]"  
  }  
}
```

ruby 필터

```
filter {  
  ruby {  
    code => "event.cancel if rand <= 0.90"  
  }  
}
```

geoip 필터

```
geoip {  
  source => ...  
}
```

Logstash FILTER(mutate 필터 상세)

주요 옵션

convert : convert => { "fieldname" => "integer" }

split : split => { "fieldname" => "," }

strip : strip => ["field1", "field2"]

replace : replace => { "message" => "%{source_host}: My new message" }

rename : rename => { "HOSTORIP" => "client_ip" }

lowercase/uppercase : lowercase => ["fieldname"]

join : join => { "fieldname" => "," }

merge merge => { "dest_field" => "added_field" }

id : id => "my_plugin_id"

Logstash FILTER(geopip 필터 상세)

주요 옵션

database : path타입으로 GeoLite2 데이터 베이스 파일의 위치 지정

GeoLite2 City 기본 내장

source : ip주소가 담긴 필드

예제)

```
filter {  
  grok {  
    match => { "message" => "%{COMMONAPACHELOG}" }  
  }  
  geoip {  
    source => "clientip"  
    target => "geoip"  
  }  
}
```

Logstash FILTER(grok 필터 상세)

비정형 데이터를 파싱하여 정형데이터로 변형해주는 filter

예)

```
grok {  
  match => "${NAME:name}"  
  add_field => { "name" => %{name}  
  }  
}
```

<https://github.com/elastic/logstash/blob/v1.4.2/patterns/grok-patterns>

grok pattern 디버거

<https://grokdebug.herokuapp.com/>

Logstash OUTPUT

```
output {  
  elasticsearch {  
    hosts => ["http://localhost:9200"]  
    index => "인덱스명"  
    document_type => "타입명"  
  }  
}
```

주요 옵션

template : 템플릿의 파일시스템 경로

template_name : 템플릿 이름

action :

index : 문서를 색인

delete : 문서를 삭제

create : 문서를 색성 (이미 있으면 실패)

update: 아이디를 기준으로 문서를 업데이트

flush_size : 벌크 인트의 성능 좌우

routing : 특정 샤드로 색인하도록 라우팅

ssl : ssl 사용여부

Logstash Persistent Queues

로그스태쉬는 queue를 사용하여 이벤트를 받음.

기본값으로 memory에 queue가 존재 (시스템 다운시에 데이터 손실 발생)

다음 설정으로 파일 기반 queue로 운영 가능함.

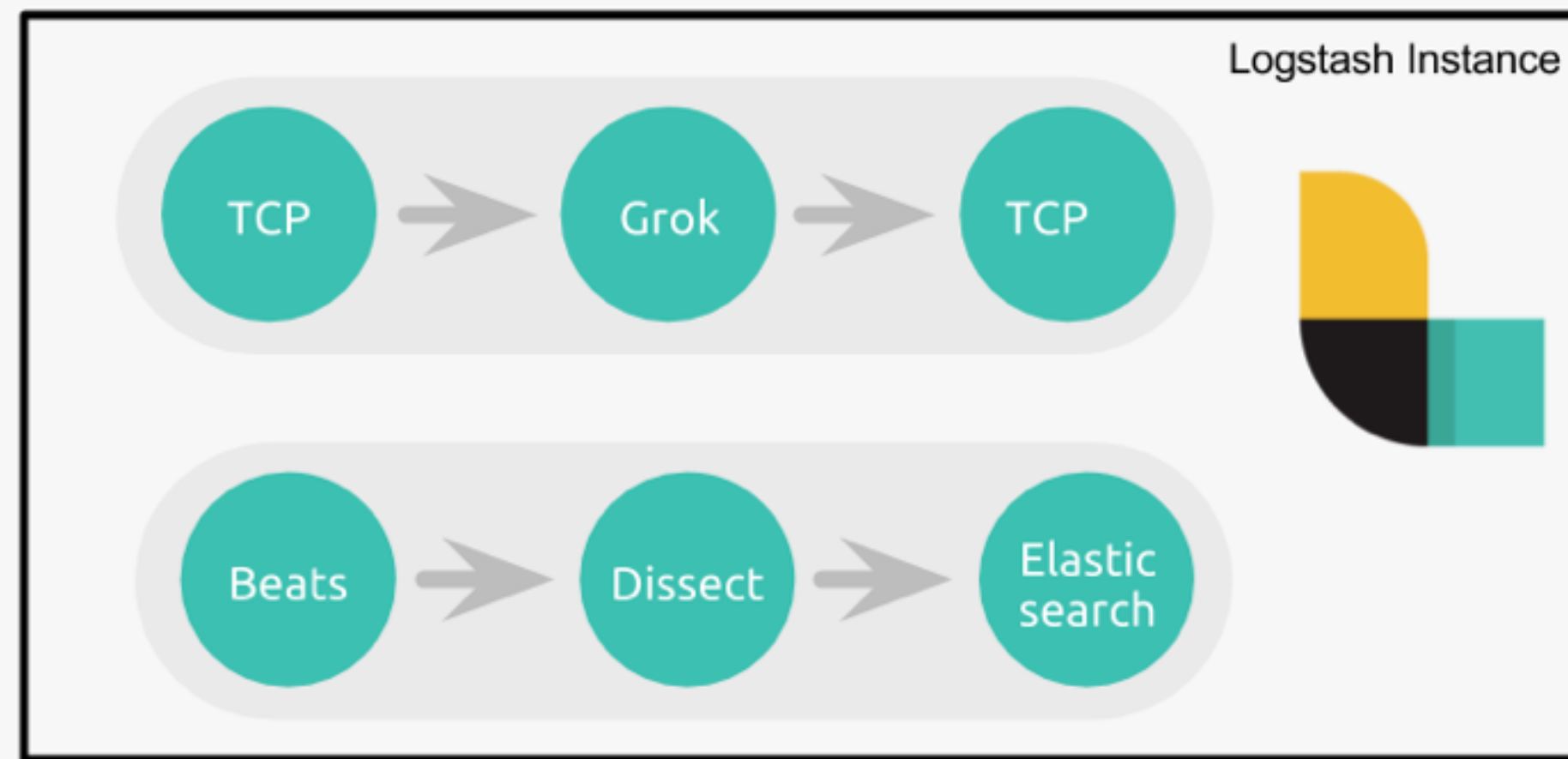
queue.type: persisted

queue.max_bytes: 4gb

path.queue: {저장 경로}

input → queue → filter + output

Logstash Multiple Pipelines



두 가지의 처리 흐름을 동시에 처리해야 할때

방법 1: 하나의 파이프라인에서 if else로 흐름을 제어

```
input {
  beats { port => 3444 tag => apache }
  tcp { port => 4222 tag => firewall }
}
filter {
  if "apache" in [tags] {
    dissect { ... }
  } else if "firewall" in [tags] {
    grok { ... }
  }
}
output {
  if "apache" in [tags] {
    elasticsearch { ... }
  } else if "firewall" in [tags] {
    tcp { ... }
  }
}
```

방법 2 : 로그스태쉬 인스턴스를 두개 실행

Logstash Multiple Pipelines

방법3 :

bin/logstash -f {설정 파일} **Single Pipelines**

bin/logstash -f 옵션으로 설정파일을 기술하면 config/pipeline.yml 을 무시하게 됨

특별한 config 파일을 지정하지 않으면 pipeline.yml을 읽어 실행함 **Multiple Pipelines 가능**

- pipeline.id: my-pipeline_1
 path.config: "/etc/path/to/p1.config"
 pipeline.workers: 3
- pipeline.id: my-other-pipeline
 path.config: "/etc/different/path/p2.cfg"
 queue.type: persisted

Logstash 튜닝

CPU, Memory, Disk I/O 추적

Worker 갯수 조절 : 기본값은 CPU core 수 (-w 옵션)

=> CPU 오버헤드 증가

배치 사이즈 조절 : 한번에 처리하는 크기 (pipeline.batch.size)

=> 메모리 오버헤드 증가

Kibana

- 데이터 분석 시각화 도구
- 엘라스틱서치의 복잡한 질의를 편하게 입력 가능
- 입력된 질의를 간편하게 시각화
- config.js 파일을 수정하여 간편하게 설정
- node.js로 작성
- 일부 설정은 엘라스틱서치 인덱스에 저장

Kibana 작업 순서

(1) Management

(2) Discover

(3) Visualize

(4) Dashboard

Warning No default index pattern. You must select or create one to continue.

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

☒ Index contains time-based events

☐ Use event times to create index names [DEPRECATED]

Index name or pattern

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

logstash-*

☐ Do not expand index pattern when searching (Not recommended)

By default, searches against any time-based index pattern that contains a wildcard will automatically be expanded to query only the indices that contain data within the currently selected time range.

Searching against the index pattern *logstash-** will actually query elasticsearch for the specific matching indices (e.g. *logstash-2015.12.21*) that fall within the current time range.

Unable to fetch mapping. Do you have indices matching the pattern?

(1) 대상이 되는 인덱스 패턴 설정

(2) Discover : 조건 필터

(3) Visualize : 시각화 방법 설정

(4) Dashboard : 대쉬보드 배치

TimeLion

시계열 데이터를 다양하게 활용하기 위한 키바나 플러그인
일정 시간 동안 고유한 사용자마다 각각 몇 개의 페이지가 노출되는가?
이번 주 금요일과 지난 주 금요일에는 어떤 차이가 있는가?
오늘 내 사이트를 방문한 사용자는 일본 인구의 몇 %인가?
S&P 500 지수의 10일 이동 평균은 얼마인가?
지난 2년 간 수행된 모든 검색의 누적 합계는 얼마인가?

문법

- 1) . es(*) 로 시작
- 2) 함수는 항상 .(도트)로 시작
- 3) 다른 연산은 , (콤마)로 분리

예제)

```
.es(timefield="date_seoul",index="index",metric=sum:sales,q=region:서울).cusum().label("목표매출")  
    , .es(timefield="date_seoul",index="index").cusum().label("실제매출")
```

timefield 값이 없으면 기본값은 @timestamp 값

metric의 기본값은 count

cusum() 누적값

q 쿼리로 조건 설정

TimeLion (2)

기본 문법

.es(*)
.es(q=*) 쿼리 지정
.es(q=*, index=logstash-*) 색인지정
.es(q=country:de).label(Germany) 라벨링
.es(q=de), .es(q=us) 멀티 타임시리즈
.es(split=country:4) 분할
.es(split=country:4, metric=sum:bytes)
.es(), .static(25, label='good visitor level') 상수값 출력
.es().bars(), .static(50).points(symbol=cross, radius=2), .static(60).lines(width=5) 라인에 대한 스타일 지정
.es(q=error).color(#FF0000), .es(q=warning).color(yellow) 라인에 대한 컬러지정
.es(), .es().derivative().label(derivative) 타임시리즈에서의 앞 데이터와 차이값
.es().color(#DDD), .es().mvavg(5h) 이동평균값

참고 URL : <https://github.com/elastic/timelion/blob/master/FUNCTIONS.md>

FileBeats

Elastic Stack의 beats 중의 한 종류
로그 및 파일을 경량한 방식으로 전달
프로세스 중단후 복귀시 마지막 중단점을 기억



2. ELK 설치

Logstash 설치

- 자바8 설치 필수

1) 일반적인 바이너리 다운로드

```
wget "https://artifacts.elastic.co/downloads/logstash/logstash-5.1.1.tar.gz"  
tar zxvf logstash-5.1.1.tar.gz  
cd logstash-5.1.1
```

2) yum으로 설치

```
rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch  
sudo yum install logstash
```

3) apt 설치

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -  
sudo apt-get install apt-transport-https  
echo "deb https://artifacts.elastic.co/packages/5.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-5.x.list
```

Logstash 디렉토리 구조

타입	설명	위치
home	홈디렉토리	{extract.path} 압축폴린폴더
bin	바이너리 스크립트 logstash logstash-plugin	{extract.path}/bin
settings	logstash.yml jvm.options startup.options	{extract.path}/config
logs	로그파일저장	{extract.path}/logs
plugins	플러그인 파일 저장	{extract.path}/plugins

Logstash logstash.yml

pipeline:

batch:

size: 125

delay: 5

혹은

pipeline.batch.size: 125

pipeline.batch.delay: 5

기타 세팅

node.name

path.data

pipeline.workers:1

pipeline.batch.size:125

pipeline.batch.delay:5

pipeline.unsafe_shutdown:false 메모리에 이벤트 있어도 종료

path.config : config 파일 위치

config.test_and_exit: false 구성이 정상인지 확인후 종료

config.reload.automatic: false

config.reload.interval:3

config.debug:false

queue.type

http.host

http.port

log.level

Kibana 설치

설치 및 실행

```
wget https://artifacts.elastic.co/downloads/kibana/kibana-5.4.3-linux-x86\_64.tar.gz
```

```
tar xvf kibana-5.4.3-linux-x86_64.tar.gz
```

```
cd kibana-5.4.3-linux-x86_64
```

```
bin/kibana
```

주요 옵션 설정

```
vi config/kibana.yml
```

```
server.port:
```

```
server.host:
```

```
server.name:
```

```
elasticsearch.url:
```

```
kibana.index:
```

```
server.ssl.enabled
```

```
server.ssl.certificate:
```

FileBeat 설치

설치 및 실행

```
wget https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-5.1.1-linux-x86_64.tar.gz
```

```
tar zxvf filebeat-5.1.1-linux-x86_64.tar.gz
```

```
cd filebeat-5.1.1-linux-x86_64
```

```
sudo ./filebeat -e -c filebeat.yml (데몬 실행)
```

주요 옵션 설정

```
vi filebeat.yml
```

```
filebeat.prospectors:
```

```
- input_type: log (logstash에서 변수로 사용가능)
```

```
paths:
```

```
- /var/log/*.log
```

```
output.elasticsearch:
```

```
hosts: ["192.168.1.42:9200"]
```

FileBeat 여러 파일을 동시에 처리

filebeats

filebeat.prospectors:

- input_type: log
path: - /path/to/apache/access.log
document_type: apache
- input_type: log
path: - /path/to/db/mydb.log
document_type: postgres

output.logstash:

hosts: ["localhost:5044"]

logstash

```
input {  
  beats {  
    port => "5044"  
  }  
}  
filter {  
  if [type] == "apache" {  
    # YOUR CUSTOM FILTERS FOR APACHE LOG ...  
  } else if [type] == "postgres" {  
    # YOUR CUSTOM FILTERS FOR POSTGRES LOG ...  
  }  
}  
output {  
  if [type] == "apache" {  
    # YOUR CUSTOM OUTPUT FOR APACHE LOG ...  
  } else if [type] == "postgres" {  
    # YOUR CUSTOM OUTPUT FOR POSTGRES LOG ...  
  }  
}
```

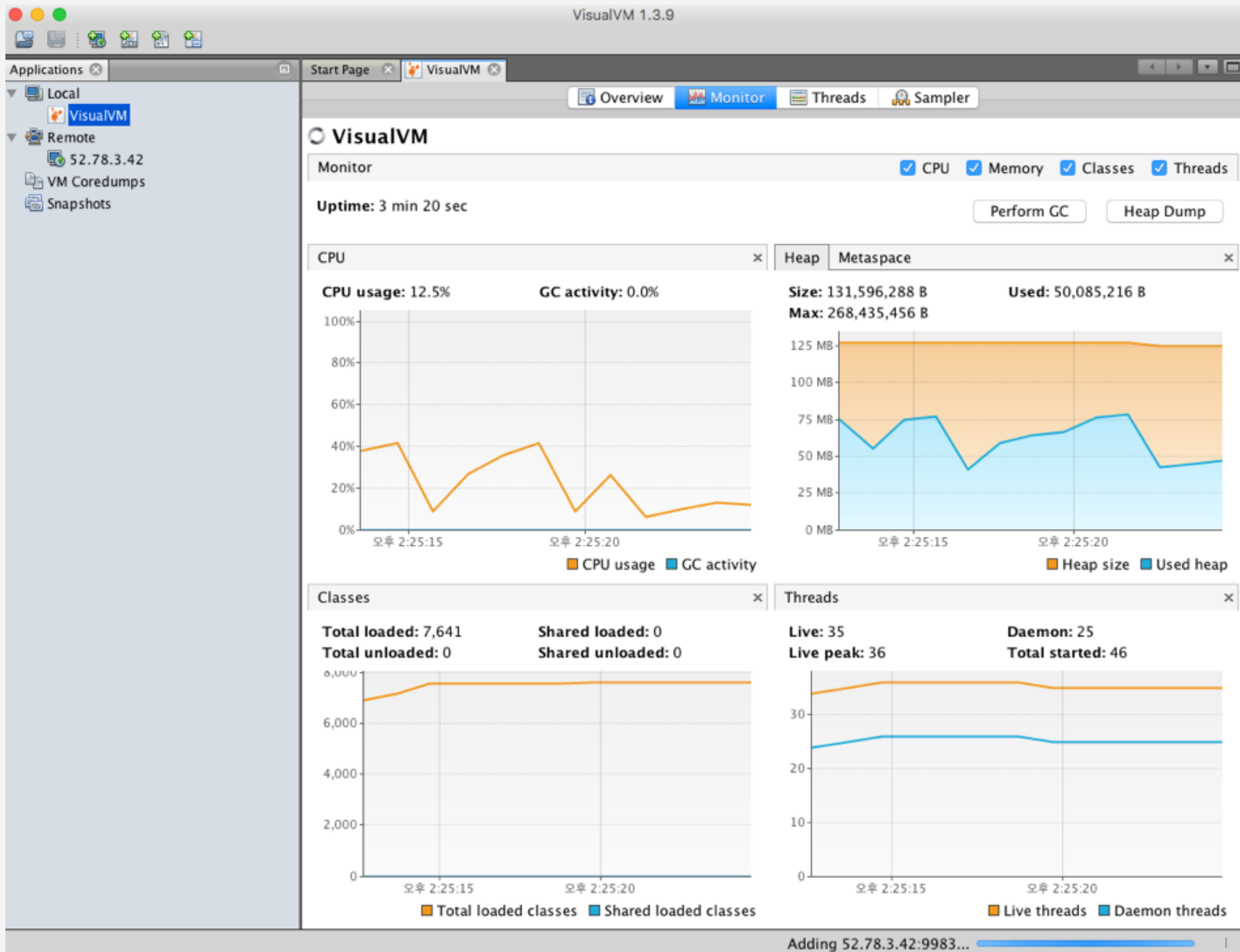
3. ELK 로그 분석

실습 1 : 서울시 지하철 탑승 정보 분석

실습 2 : nginx or apache 로그 분석

4. 기타 클러스터 관리 기법

Visual VM 설치후 JVM 모니터링



성능 극대화를 위한 고려 사항

어플리케이션의 복잡도 :

어플리케이션에서 색인 갱신 삭제 조회를 그룹화 함으로서
네트워크 오버헤드를 최소화

색인과 검색중 어느것에 초점인가?:

색인성능과 검색 성능의 양날의 검을 어떻게 조절할것인가?

메모리 :

성능의 중요한 요인인 캐싱을 어떤 전략으로 사용할 것인가?

루씬 세그먼트 관리 최적화

얼마나 자주 refresh와 flush를 수행하는가?

언제 refresh 할것인가? : `index.refresh_interval` : 5s (-1이면 refresh disable 수동처리)

언제 flush를 할 것인가?

메모리 버퍼가 찼을때 `indices.memory.index_buffer_size`

마지막 flush로부터 일정 시간 지났을때

`index.translog.flush_threshold_period` : 10m

트랜잭션 로그의 사이즈가 일정 임계치를 넘겼을때

`index.translog.flush_threshold_size` : 500mb

머지 정책은 어떻게 할 것인가?

`index.merge.policy.segments_per_tier` :

클수록 더 많은 세그먼트를 가짐. 색인 횟수가 적고 검색성능을 높인다면 이 값을 낮춤

`index.merge.policy.max_merged_segment` :

세그먼트 크기의 최대값, 빠른 색인을 원한다면 값을 낮춤.(머지작업이 잘 일어나지않게함)

`index.merge.scheduler.max_thread_count`:

백그라운드 머지 작업을 수행할 스레드 갯수 지정 : CPU와 I/O가 넉넉하다면 값을 높인다.

캐시의 최적화

엘리스틱 서치는 두번째 쿼리가 첫번째 쿼리보다 빠른 속도로 수행

1. 필터 캐시

필터 쿼리의 결과는 필터 캐시에 저장

필터를 자주 사용하는 경우 캐시 크기를 증가 고려

`indices.cache.filter.size:30%` : 30%를 필터 캐시에 할당 (정적인 값도 할당 가능)

`indices.cache.filter.expire:30m` : 캐시의 만료 시간 설정

2. 샤드 쿼리 캐시

샤드간의 요청에 대한 캐시

refresh가 일어나면 샤드 캐시는 무효화 (샤드가 변하지 않고 같은 요청이 반복되는경우 효과)

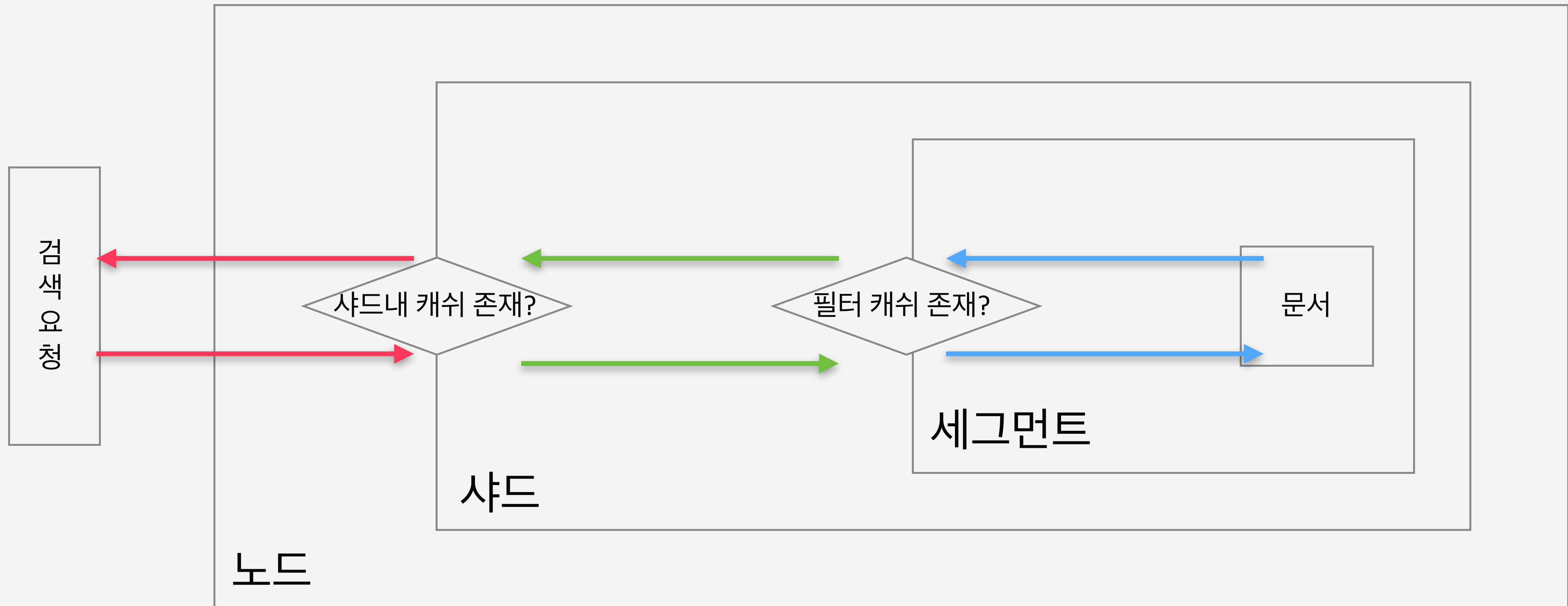
`index.cache.query.enable: true`

`index.cache.query.size: 10%` (기본값은 jvm 1%)

혹은

`GET /my_index/_search?request_cache=true`

캐시의 최적화



서킷 브레이커

필드 데이터 캐시는 OutOfMemory가 발생할 만큼 커질 가능성 존재
인위적 임계치 설정 (노드가 죽는 최악의 상황 방어, 쿼리를 중지하는 쪽으로 선택)

1. `indices.breaker.total.limit` -

필드 데이터와 요청 서킷 브레이커가 이 값을 넘을수 없음. 기본값 jvm 의 70%

2. `indices.breaker fielddata.limit` -

필드 데이터가 이 값을 넘을수 없음. 기본값 jvm 의 60%

3. `indices.breaker.request.limit`

집계 버킷 생성등의 작업에 할당할수 있는 힙 크기 기본값 jvm 의 40%

쓰레드 풀

엘리스틱 서치는 효율적인 CPU 메모리 사용을 위해 쓰레드풀을 사용

요청 종류에 따라 쓰레드 수 조절 가능

- bulk
- index
- search
- etc..

bulk 쓰레드 조정

threadpool.bulk.type: fixed (fixed 고정할당 bounded도 설정 가능)

threadpool.bulk.size: 40

threadpool.bulk.queue_size: 200

스냅샷 (백업)

처음 스냅샷을 사용하면 데이터 상태와 데이터를 복사
이후 사용하면 이전 것으로부터의 변경사항만 포함

path.repo: /home/ec2-user/my_backup

레파지토리 등록

```
-XPUT /_snapshot/my_repository
{
  "type": "fs",
  "settings": {
    "location": "/home/ec2-user/my_backup",
    "compress": true
  }
}
```

스냅샷 확인

```
-XGET _snapshot/_all
-XGET _snapshot/my_repository
```

```
-XPUT /_snapshot/my_repository/snapshot1?wait_for_completion=true
{
  "indices": "news,store",
  "ignore_unavailable": true
}
```

repository-s3 for S3 repository support

repository-hdfs for HDFS repository support in Hadoop environments

repository-azure for Azure storage repositories

repository-gcs for Google Cloud Storage repositories

스냅샷 (복구)

POST store/_close

POST _snapshot/my_repository/snapshot1/_restore

다중 백업 복구

POST _snapshot/my_repository/snapshot1/_restore

```
{  
  "indices": "store,news",  
  "ignore_unavailable": true,  
  "include_global_state": true,  
  "rename_pattern": "news(.+)",  
  "rename_replacement": "restored_index_$1"  
}
```

기타 성능에 관련한 Trade Off

refresh 빈도 : 빈도를 낮추어 색인 속도를 높이면 실시간성 손해

부정확한 매치 - 색인 시점에서 ngram을 사용하여 색인 시간을 늘리고 검색을 빠르게 할것인가?
fuzzy나 wildcard 검색을 사용하여 recall을 늘릴것인가?

분산 집계 : 더 정확한 집계를 위해서 aggregation의 round trip을 늘릴것인가?

성능 고려 사항 요약

벌크 API 사용

너무 많은 힙메모리를 사용하지 말고 노드를 나누자

한 서버에 여러개의 노드를 구성하는 것도 방법

최신 SSD를 사용하자

병합정책을 고려

Optimize를 실행하지 말자

_all 필드를 비활성화 하자 (_source 필드는 디스크 사용량에 영향)

색인 버퍼 사이즈 확인 (_stats 에서 index_writer_memory_in_bytes 확인)

자동ID를 사용하는 것이 좋다.

refresh_interval 조절

벤치마킹 예)

refresh_interval: 1s – 2.0K docs/s

refresh_interval: 5s – 2.5K docs/s

refresh_interval: 30s – 3.4K docs/s

[https://sematext.com/blog/2013/07/08/](https://sematext.com/blog/2013/07/08/elasticsearch-refresh-interval-vs-indexing-performance/)

elasticsearch-refresh-interval-vs-indexing-performance/

역할에 맞는 노드 구성

discover시의 핑 간격 조절하고 유니캐스트를 사용

리밸런싱을 사용하지 말자

라우팅을 적절히 사용

감사합니다.