ELK 스택을 활용한 로그 기반 콘텐츠 동시 시청 제어 실증

이주성, 신승호, 이상범
SK 브로드밴드

jusunglee@sk.com, sinius@sk.com, sb.lee@sk.com

A Demonstration of Log-based Simultaneous Viewing Block Method using ELK Stack

Lee Jusung, Shin Seungho, Lee Sangbeom
SK Broadband

요 약

본 논문에서는 오픈소스 솔루션인 ELK 를 사용하여 Log 기반 스트리밍 컨텐츠의 동시 시청 제어를 수행한다. 시청 Log 적재는 Kafka 를 사용하고, 시청 로그 분석에는 Elasticsearch 를 사용하여 동시 시청을 판별한다. 이를 실제 시스템에 적용하여 REST API latency 가 3ms 이하의 성능을 달성하였다.

Ⅰ. 서 론

OTT 서비스가 대중화 되면서, 스트리밍 컨텐츠 동시 시청 제어의 중요성이 대두되고 있다. 구독자의 도덕적 해이로 인해 계정 공유를 시도하는 사람들이 늘어나게 되고, 이 문제를 제어 하는 것은 CP(Contents Provider)에게 중요한 문제가 되었다.

기존의 스트리밍 컨텐츠 동시 시청 차단에는 RDBMS 이 많이 쓰였다. 기존의 데이터가 RDBMS 에 저장되는 경우가 많았고, 안정적인 서비스를 원하는 기업에선 계속 RDBMS 를 사용하는 경우가 많다. 그러나, RDBMS 는 수 백만명의 시청 로그를 저장하려면 상용 솔루션과 Scale Up 만 지원하여서 고사양의 서버를 필요로 한다.

이런 단점을 극복하기 위해 NoSQL 진영에서 빅데이터 저장에 용이하고 Scale Out 을 지원하게 하는 연구가 활발하게 진행되어 왔다. Scale Out 을 지원하면, 확장시 저사양의 서버를 cluster 에 추가하는 것만으로도 가능하다.

또한 NoSQL 진영에서는 고가의 상용 솔루션을 대체할 수 있는 많은 오픈소스 솔루션이 나오고 있다. 현재 Elasticsearch, MongoDB, hadoop 같은 오픈소스 솔루션이 대중적으로 사용되고 있다.

본 논문에서는 고가의 상용 솔루션을 사용하지 않아도, NoSQL 오픈소스 솔루션인 ELK stack 을 사용하여, 스트리밍 컨텐츠의 동시 시청 차단 서비스를 만들 수 있는 시스템을 제안한다.

Ⅱ. 본론

동시 시청 제어를 위해 스트리밍 시청 Log 적재와 스트리밍 시청 Log 분석이 각각 수행되어야 한다. 본 논문에서는 스트리밍 시청 Log 적재는 분산 환경에 특화된 Kafka 를 사용하여 수행하였다. ELK 에서 핵심인 Kafka 는 대용량 실시간 Log 에 특화되어, SSD 디스크 환경에 3 대의 Kafka cluster 로 설계 시 Log 처리량 초당 100 만개 레코드와 96MB/sec 까지 지원한다. 여기에 더불어 Scale Out 을 지원하기 때문에 확장이 필요할 시 cluster 에 Kafka 만 추가하면 된다.

Cluster	CPU	메모리	Replication	Partition
3 VM	6core	32G	2	18

표 1. Kafka 환경설정

Kafka 의 환경설정을 표 1 과 같이 하였다. Replication 을 2 (원본 1+복제본 2)로 설정하여 Kafka 서버에 물리적인 장애가 발생한 경우에도 topic 에 저장된 데이터 안정성을 올렸다. 또한 Partition 사이즈를 증가하여 시청 로그 Kafka producer 가 병렬적으로 메시지를 처리 가능하도록 하여 성능을 올렸다.

```
{
  “@timestamp”: “2021-05-15T00:00:00Z”,   #시간
  “customer_id”: "86C8D911-46EB-11EA",    #고객 ID
  “contents_id”: “cid00001”,              #컨텐츠 ID
  “pcid”: “20205150001”,                  #단말 ID
  “event”: “play_start”                   #시청시작/종료
}
```

그림 1. 시청 로그 규격

사용한 시청 로그의 규격은 그림 1 과 같다. 어떤 고객이 어떤 컨텐츠를 시청하는지를 로그에 기술하고 있다.

스트리밍 시청 Log 분석을 위하여 Elasticsearch 를 사용하였다. Elasticsearch 도 분산 환경에 최적화된 실시간 검색 엔진으로서, 대용량의 비정형 Log 데이터를

색인(Indexing)하여 빠른 검색이 가능하다. 특징으론 신규 WAS 구축 없이도, REST API 서버를 자체적으로 제공한다. 또한 빠른 response 를 위해서 자체적으로 Caching 기능을 제공한다.

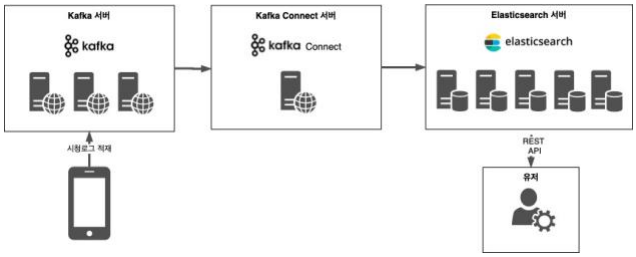

그림 2. 시스템 구성도

사용한 시스템 구성은 그림 2 와 같다. Kafka cluster 는 3 대로 구성했고, 시청 로그를 분석하기 위한 Elasticsearch 는 5 대로 cluster 를 구성했다. 상용서비스의 빠른 response 가 중요하기 때문에 Elasticsearch cluster 를 VM 5 대로 구성하였다.

```
{
  "query": {
    "bool": {
      "filter": [                          # MUST 조건
        {"term": {
            "customer_id": "86C8D911-46EB-11EA"
          }
        },
        {
          "term": {"contents_id": "cid00001"}
        },
        {
          "range": {
            "@timestamp": {
              "gte": "2021-05-15T00:00:00",
              "time_zone": "+09:00",
            }
          }
        }
      ],
      "must_not": [                        # MUST-NOT 조건
        {
          "term": {"pcid": "20205150001"}
        }
      ],
    }
  },
  "sort": [{"@timestamp": {"order": "desc"}}]
  "size": 1
}
```

그림 3. 시청 로그 검색 쿼리

동시 시청 차단을 위한 Elasticsearch Query 는 그림 3 과 같다. Elasticsearch 쿼리는 JSON 규격으로서 이 쿼리를 가지고 REST API 호출을 하게 된다. 그림 3 의 쿼리에서는 filter 부분에 MUST 조건을 나열하고, must_not 부분엔 MUST_NOT 조건을 나열한다.

동시 시청 차단의 개념은 등록한 고객(customer_id)이 동시에 여러 단말(pcid)로 동일한 컨텐츠(contents_id)를 시청을 하려고 할 때 차단하는 것이다. 이 쿼리는 본인(pcid: 20205150001)이 아닌, 다른 단말에서 동일 컨텐츠 (contents_id)를 시청하는 Log 가 있는지 검색한다. 만약 검색 결과가 없거나, 또는 시청 종료 Log(event: play_end)만 있다면 동시 시청 허용을 하면 된다.

Cluster	CPU	메모리	Replica	Shard
5 VM	8core	64G	1	5

표 2. Elasticsearch 환경설정

Elasticsearch 의 환경설정은 표 4 와 같다. REST API latency 를 줄이기 위해서, 서버는 5 대, Replica 1 (원본 1 + 복제본 1), shard 는 5 로 설정하였다. Replica 를 증가시키면 서버의 장애가 발생하는 경우에도 높은 고가용성을 제공합니다.

또한 Elasticsearch 는 분산 환경에 최적화된 검색엔진으로서, 서버 수와 shard 를 증가시키면, 검색 요청도 분산해서 처리하게 된다. 이런 환경설정에서 테스트 결과 Cache 서버 추가 없이도 검색 REST API 의 latency 가 3ms 이내에서 유지가 되었다.

Ⅲ. 결론

본 논문에서는 컨텐츠 동시 시청 제어 시 Kafka 와 Elasticsearch 를 사용하는 시스템을 제안하였다. 스트리밍 시청 로그 적재에는 Kafka 를 사용하였고, 스트리밍 시청 로그 검색에는 Elasticsearch 를 사용하였다. 이 Kafka 와 Elasticsearch 는 모두 오픈소스이지만 상용서비스 못지 않는 성능을 제공하고 있다. 본 논문에서 기술한 시스템 환경을 기준으로 동시 시청 제어를 위한 API 평균 응답시간이 3ms 이내에 제공이 가능한 것으로 확인되었으며, 고신뢰성 어플리케이션 및 서비스 개발이 가능한 성능 수준이라 볼 수 있다.

향후에는 고객과 서비스 사용량 증가로 인한 Scale-out 방안과 실제 효용에 대한 연구 분석이 진행될 예정이다.

참 고 문 헌

[1] 조진만, 김신효, “유료방송에서 Pay-Per-View 프로그램 시청시 중복과금 방지 알고리즘 고찰” (Anti-Double Charge Algorithm for Pay-Per-View Program on Pay-TV Service) (https://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE00219666)

[2] 최석환, 권준호, 최윤호 “모바일 앱을 이용한 실시간 사용자 계정 접근 관리” (https://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE07180858)