



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

碩 士 學 位 論 文

아파치 엘라스틱서치 기반 로그스테시를 이용한 보안로그분석 시스템

大田大學校大學院

情報通信工學科情報通信시스템工學專攻

李 相 龍

指導教授 李 奉 煥

2016年 2月

아파치 엘라스틱서치 기반 로그스태시를 이용한 보안로그분석 시스템

指導教授 李 奉 煥

이 論文을 工學碩士學位
請求論文으로 提出함

2015年 10月

大 田 大 學 校 大 學 院

情報通信工學科情報通信시스템工學專攻

李 相 龍

李相龍의 工學碩士學位
請求論文을 認准함

2015年 12月

學位論文審査委員會

委員長 _____

委員 _____

委員 _____

大田大學校大學院

목 차

제 1 장 서 론	1
1.1 연구 배경 및 목적	1
1.2 논문의 구성	4
제 2 장 관련 연구	5
2.1 보안로그 분석	6
2.1.1 로그	6
2.1.2 로그분석	7
2.1.3 RDBMS 방식의 로그 분석	8
2.1.4 NoSQL 방식의 로그 분석	8
2.1.5 분산검색엔진 방식의 로그 분석	9
2.2 분산검색엔진	10
2.2.1 엘라스틱서치(Elasticsearch)	10
2.2.2 엘라스틱서치의 특징	11
2.2.2.1 엘라스틱서치 대시보드 - 키바나(Kibana)	13
2.3 로그 컬렉터	14
2.3.1 로그스테시(Logstash)	14
2.3.2 아파치 플룸(Apache Flume)	14
2.3.2.1 아파치 플룸의 특징	16
2.4 기타연구사례	17
2.4.1 아파치 하둡	17
2.4.2 하둡의 특징	18
2.4.3 맵리듀스(MapReduce)	19
2.4.4 기타 관련 논문	21
제 3 장 엘라스틱서치를 이용한 보안로그 분석 시스템	23
3.1 엘라스틱서치를 이용한 로그분석 시스템 설계	23
3.2 엘라스틱서치를 이용한 보안로그 분석 시스템 환경구축	25

3.2.1 엘라스틱서치 멀티 태넌시 구축	27
3.2.2 엘라스틱서치 클러스터 환경 구축	29
3.2.3 엘라스틱서치 플러그인	30
3.3 로그스태시(Logstash)	34
3.4 로그스태시, 엘라스틱서치 및 키바나 연동	38
제 4 장 실험 및 성능평가	41
4.1 실험 환경 구성	41
4.2 실험 진행 방법	43
4.2.1 로그 수집 및 저장	43
4.2.2 보안로그 분석	44
4.3 성능 테스트	45
4.3.1 분석 성능 테스트	45
4.3.2 용량에 따른 로그데이터 처리 속도 비교	48
4.3.3 로그스태시 Worker 수에 따른 로그 처리량 테스트	50
제 5 장 결론 및 향후 과제	52
참고문헌	53

그 림 목 차

[그림 1-1] 전 세계 디도스(DDoS:분산서비스거부)공격	2
[그림 2-1] 메일 서버의 로그 생성 과정	7
[그림 2-2] 엘라스틱서치 클러스터 구성도	11
[그림 2-3] 로그스태시(Logstash) 입출력 과정	14
[그림 2-4] Flume NG의 기본 아키텍처	15
[그림 2-5] 아파치 플룸(Apache Flume)의 계층적 구조도	16
[그림 2-6] 맵리듀스(MapReduce) 처리 흐름도	20
[그림 3-1] 엘라스틱서치를 이용한 보안로그 분석 시스템 구성도	23
[그림 3-2] 엘라스틱서치를 이용한 보안로그 분석 시스템 아키텍처 ...	23
[그림 3-3] 노드 구성 및 로그 데이터 흐름도	26
[그림 3-4] 엘라스틱서치 플러그인 Marvel Query 상태 확인	31
[그림 3-5] 헤드 플러그인 구동 화면	32
[그림 3-6] 로그 데이터 수집 및 저장 구조도	39
[그림 4-1] 실험 환경 구성도	42
[그림 4-2] HTTP 요청에 의한 지속적인 GET 요청 발생 화면	43
[그림 4-3] Kibana를 이용한 로그 조회 시각화	44
[그림 4-4] 엘라스틱서치를 이용한 분석성능 비교 실험 구성도	45
[그림 4-5] 엘라스틱서치기반 시스템과 하둡기반 시스템 성능 비교 ...	46
[그림 4-6] HBase와 Elasticsearch 성능 비교	48
[그림 4-7] Logstash Worker 수 증가에 따른 로그 처리량	50

표 목 차

[표 2-1] 윈도우 계열 로그파일 저장위치와 설명	5
[표 2-2] 리눅스 계열 로그파일 저장위치와 설명	6
[표 2-3] 관계형 데이터베이스와 엘라스틱서치 데이터 구조 비교	12
[표 2-4] 기타 관련 논문	21
[표 3-1] 노드구성 사양정보 및 툴	26
[표 3-2] 엘라스틱서치 용어 설명	28
[표 3-3] Node Stats API 항목과 설명	33
[표 4-1] 엘라스틱서치기반 시스템과 하둡기반 시스템 비교	47
[표 4-2] Elasticsearch 와 HBase 데이터량에 따른 속도비교	49
[표 4-3] Logstash Worker 수에 따른 처리량 비교	51

제 1 장 서 론

1.1 연구 배경 및 목적

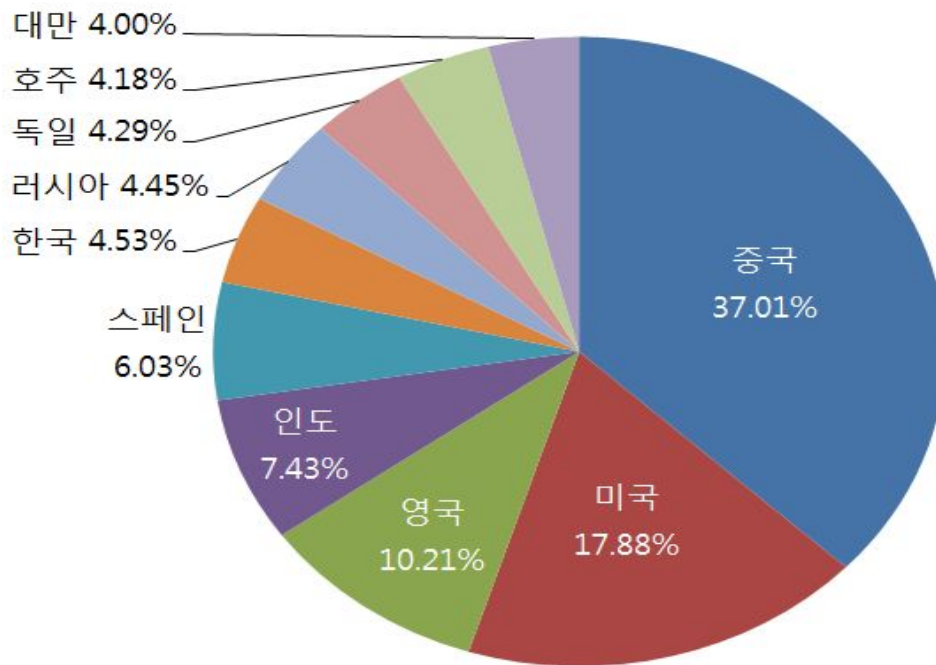
기업이나 공공기관, 학교 등 의 다양한 사이트에서의 네트워크 관리는 기존 네트워크를 통합하고 강화하는 절차로 인해 데이터양이 기하급수적으로 증가하고 복잡해지고 있다. 이 데이터들 중의 대부분을 차지하는 데이터는 로그데이터 이다. 로그데이터는 가동 중인 정보시스템 내에서 시스템 운영에 대한 전반적인 로그, 각종 서비스 내역을 기록한 로그 등 다양한 로그데이터가 대량으로 발생하고 있다.

일반적으로 시스템에서 일어나는 일들에 대한 로그데이터 정보를 파악하고 추출하여 보안적인 용도로 내부정보의 외부유출과 서버공격 및 서버무력화 등과 같은 악의적인 행위를 미리 예측하여 활용할 수 있다.

최근 미국의 보안업체 Proof Point 에서 조사한 자료에 의하면 2013년부터 2014년 까지 75만건의 피싱공격과 스팸공격 등의 악성 이메일 이 가정에 설치된 라우터, 스마트 텔레비전, 냉장고와 같은 스마트 가전 기기들에 의해 발송되었다고 한다. 악의적인 공격자들은 인터넷으로 연결되어있는 가전기기를 좀비 가전기기로 만든 후 피싱이나 스팸 메일을 보내게 한 것이다. 이처럼 사이버 공격 방식이 날이 갈수록 고도화 되고 있으며, 특히 개인정보 유출로 인해 금전적 사고로 이어진다면 그 피해는 더욱 커질 것이다.

빅데이터(Big Data) 시대에 직면한 지금 막대한 양의 데이터로 무엇을 할 다는 것 자체가 거대한 도전과제인데, 보안의 경우 로그 분석이 이에 해당된다. 로그 데이터는 시스템을 통과하는 데이터의 흐름을 추적하는 것으로, 다수의 사용자가 실시간으로 이용하는 시스템이 많을수록 방대한 로그 데이터가 기록이 되어 남게된다. 자동화된 로그 분석 도구들은 변칙적 트래픽과 보안에 관한 육감을 가지고 있는 로그 분석가들을 완전히 대체할 수 없는 것으로, 역량있는 로그 분석가는 기본적으로 로그

데이터를 바라보는데 매일 상당한 시간을 투자해야하는 필요성이 현재 대두되고 있다.[1]



[그림 1-1] 전 세계 디도스(DDoS:분산서비스거부) 공격

IoT(Internet Of Things) 시대에 직면한 현재 전 세계 디도스(DDoS: 분산서비스거부) 공격이 지난해보다 두 배 이상 증가하며, 분기 기준 최고치를 기록한 것으로 나타났다. 디도스 공격이란 용량을 초과하는 정보를 한꺼번에 보내 과부하로 서버를 다운시키는 공격 방식을 말한다. 국내에서 발생한 디도스 공격은 전체의 4.53%로 공격 발생 국가 중 6위로 조사됐다. 공격 기법 가운데는 각각 전체의 16%를 차지한 신(SYN)과 SSDP가 가장 빈번하게 나타났는데 특히 SSDP는 1년 사이 주요 공격 방법으로 떠올랐다. 네트워크 연결 기기의 보안이 특히 취약해 디도스 공격을 위한 SSDP 반사기로 악용될 수 있는 만큼 주의하고 있다.

네트워크의 규모가 점차 커짐에 따라서 방대한 양의 로그 정보가 발생하게 되는데, 이러한 로그 데이터 분석을 위해 빅데이터 분석 기술이 사용되고 있다. 빅데이터 분석 기술은 여러 분야에 광범위하게 사용되는데 빅데이터의 특성인 3V 중 속도(Velocity) 빠르게 누적되는 로그데이터를 분석하기에 탁월하며 정형화되어있지 않는 로그 데이터 분석이 가능하게 되었다.

로그 데이터가 수집되는 주요 소스로는 윈도우 서버와 방화벽 및 네트워크 장비가 주를 이루며, 모바일 기기, 건물관리 장비, 클라우드 등이 있다. 일반적으로 기업은 윈도우와 유닉스 등의 서버, 보안 기기, 스위치와 라우터, 침입탐지 시스템, 안티바이러스 및 기타 보안 애플리케이션과 같은 네트워크 장비, 가상화된 서버와 하이퍼바이저 뿐만 아니라 데스크탑과 노트북 등에서 로그 데이터를 수집한다.

정보보안에 있어서 로그 데이터의 빅데이터 분석을 중요하게 생각하고 있고, 보안 솔루션도 도입하고 있지만 기업 및 학교 공공기관 등은 아직 로그 분석에 충분한 시간을 투자하지 못하는 것으로 나타났다. 방대한 양의 로그 데이터들 중 필터링을 하고 필요로 하는 정보를 얻기 위해서는 이벤트 간 상관관계 분석 기능을 향상시킬 필요가 있는데, 이는 IT 및 보안 관리자들이 먼저 로그에 익숙해지고, 무엇이 정상이고 비정상인지에 대한 기준을 세우는 것이 필요한 시점이다.[1]

본 논문에서는 방화벽, 침입탐지시스템 등 막대한 양의 보안로그 빅데이터 분석을 위한 하둡 기반 환경을 사용하고 데이터의 수집 및 저장을 위한 도구인 검색엔진(Search Engine)을 이용하여 보안 로그데이터 분석 시스템을 제시한다. 제시하는 시스템은 다양한 네트워크 환경에서 검색엔진을 활용하여 보안로그를 수집하고 검색엔진에 저장한 후에 분석 과정을 거쳐서 원하는 데이터를 시각화하거나 조회가 가능하며 효율적인 보안로그 데이터 분석을 하는 것에 목적을 두었다.

1.2 논문의 구성

본 논문에서는 제안하는 하둡 기반 환경에서 검색엔진을 이용한 로그 데이터 분석 시스템은 다음과 같은 구성을 토대로 하여 작성되었다. 제2장에서는 관련 연구로 보안로그 분석과 빅데이터, 분산검색엔진에 대한 설명과 기술동향에 대해 기술하였다. 제3장에서는 검색엔진을 활용한 보안로그 분석 시스템을 설계하고 각각의 구성 모듈의 구현과 구축을 하였다. 제4장에서는 구축한 시스템에 대한 성능 평가를 위해 실험환경을 구축하여 실험을 진행하였고, 마지막 제5장에서는 결론과 향후 연구에 관하여 제시하였다.

제 2 장 관련 연구

본 장에서는 본 논문에서 제안하는 검색엔진을 이용한 보안로그분석 시스템과 관련한 연구에 관하여 서술하였다. 매우 빠르고 방대한 양의 로그 데이터로 인하여 로그 분석 기술 동향과 로그 데이터를 처리하기 위한 빅 데이터 기술, 분산 검색엔진 기술에 대하여 서술하였다. 끝으로 본 연구와 유사한 타 논문과 비교를 제시하였다.

2.1 보안로그 분석

2.1.1 로그

로그란 사용자의 행동패턴 및 이벤트 등을 저장하는 대상으로 보안사고 혹은 시스템장애 발생 시 중요한 증거자료로 사용되어지며 뿐만 아니라, 외부위협, 이상행위, 등을 탐지하기 위한 용도로 사용되어 지고 있다. 다음 아래의 [표 2-1]과 [표 2-2] 와 같이 시스템 내에서는 보안로그 기록이 외에도 많은 다양한 로그데이터를 기록하고 있다.

[표 2-1] 윈도우 계열 로그파일 저장위치와 설명

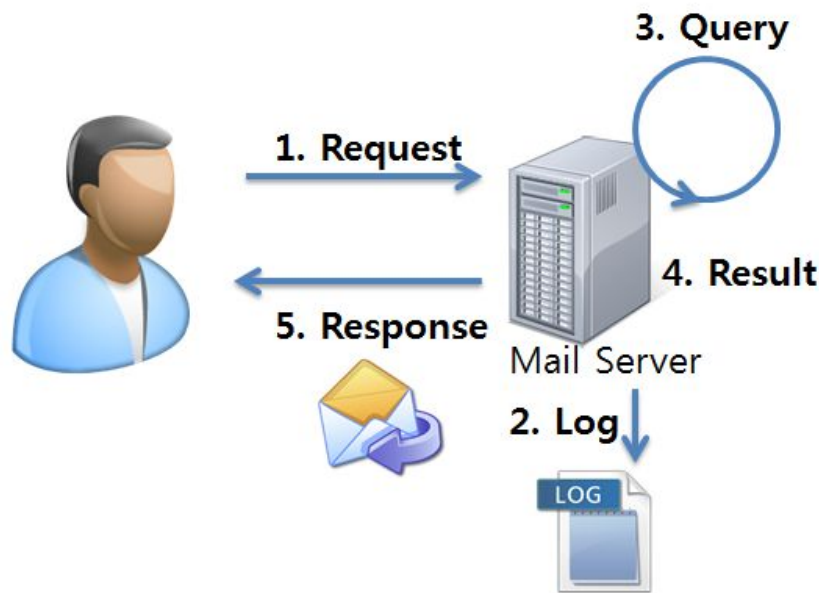
경로		설명
/var/log	btmtp/loginlog/failedlogin	로그인 실패 기록
	wtmp	사용자 접속 기록
	secure	운영체제 및 응용프로그램 동작 상태 기록
	message	su, 특정 데몬 및 부팅 시 발생한 에러 기록
	lastlog	모든 사용자 접속정보
/var/log/httpd	access_log	접속 요청 및 시도 로그
	error_log	접속 요청 및 에러 로그
/var/run	utmp	현재 로그인 사용자들 상태 로그

예를들어, 서버 관리자가 이 로그 파일을 분석하여 서버에서 발생한 악의적 행위를 해당 서버시스템에 문제가 생겼을 경우에 문제 해결방법을 찾기 위해서 먼저 로그파일을 이용하여 확인할 수 있다[2][4].

[표 2-2] 리눅스 계열 로그파일 저장위치 경로와 설명

로그파일	경로	설명
AppEvent.evt	C:\systemroot\system32\config	응용프로그램 실행 중 발생된 로그
SecEvent.evt	C:\systemroot\system32\config	보안관련 로그
SysEvent.evt	C:\systemroot\system32\config	시스템 가동 중 발생된 로그
IIS FTP log	\$IISROOT\log\MSFTPSVC1	FTP관련 로그기록
IIS Web log	\$IISROOT\log\W3SVC1	웹 서비스 로그

일반적으로 로그의 종류는 커널 로그, 시스템 로그, 보안 로그, 웹 로그 등 각각의 종류에 따라 세부적인 내용이 기록되어 있다. 예러 로그에는 웹 서버 운영에 발생하는 로그가 기록되며, 대부분의 접속 정보중의 하나인 액세스 로그 에는 서버로부터 전송되어 지는 정보가 기록되어 진다. 일반적으로 리눅스 시스템에서 기본적인 로그데이터는 syslogd에 의해 제어되며, syslogd 설정 파일인 /etc/syslog.conf 파일을 수정하여 로그 파일들의 저장위치와 저장파일명을 변경 할 수 있다. 일반적인 서버에서 사용자의 요청을 받아 서버가 응답을 하여 서버에서 로그가 생성되는 과정은 [그림 2-1]에 나타나 있다.



[그림 2-1 메일 서버의 로그 생성 과정]

2.1.2 로그분석

웹 로그는 웹상에서 사용자가 사이트에 접근하였을 때 서버에 저장되는 흔적으로 저장되는 내용으로는 접속시간, 접속한 사이트, IP(Internet Protocol) 주소, 브라우저 식별자, 운영체제 등의 정보가 포함되어 있다. 즉, 웹 서버 사용자의 이용경로를 가리킨다. 웹 사이트 개선 방향으로 웹 사용 형태를 분석하는 것은 웹 로그데이터 분석이라고 할 수 있겠다.

많은 조직에서 어려움을 겪고 있는 로그 관리의 문제는 빠른 속도로 증가하는 로그를 한정된 자원에서 효율적으로 저장, 분석, 관리해야 할 필요성이 있다는 것이다. 다양한 포맷으로 구성되어있는 로그 데이터의 내용을 일관성 있는 내용으로 저장을 위한 정규화 과정이 반드시 필요하게 되는게 현실이다. 정규화는 보안 시스템, 네트워크 관리자가 보안 로그 데이터를 효과적으로 분석 수행하도록 도와준다. 통상적인 로그 분석 도구는 사람이 쉽게 찾을 수 없는 동일한 이벤트에 관한 여러 로그 로직 패턴을 찾는데 그 의의를 둔다. 하지만 로그 분석은 다음과 같이 여러 문제점들이 있다. 로그 분석기술에서 이러한 문제점의 해결이 필요하다[3].

- ◆ 여러 종류의 형식으로 저장된 로그는 수집이 어려움
- ◆ 막대한 양의 로그에 대한 효율적 저장 관리가 어려움
- ◆ 이기종 보안 감사로그에 대한 분석이 어려움
- ◆ 단기 로그 및 장기간 로그에 대한 통합적 분석이 어려움
- ◆ 로그 분석으로 인한 비 연관 로그의 효과적 제거가 어려움

2.1.3 RDBMS 방식의 로그 분석

기존의 로그 분석들은 일반적으로 RDBMS 방식, 관계형 데이터베이스 방식을 기반으로 처리되었다. 관계형 데이터베이스란 일반적으로 키(Key)와 값(Value)들의 간단한 관계를 테이블화 시킨 매우 간단한 원칙의 전산 정보 데이터베이스 이다.[5] 하지만 기존의 방식은 빠르게 변화하고 있는 환경의 로그 데이터 특성을 반영하지 못하여 이를 분석하는 데에는 다음과 같은 한계점들이 있다. 먼저 성능의 문제가 있다. 기존 RDBMS 방식의 관계형 데이터베이스 방식은 자체적인 스케일 아웃 방식의 확장이 어려워 저장 공간 부족 현상이 나타날 수 있다. 이 문제점은 쌓여만 가는 많은 양의 로그 데이터를 수집하고 처리하는 데 있어 속도 저하 및 성능결함이 발생할 수 있다[3][5].

2.1.4 NoSQL 방식의 로그 분석 [4][15][16]

NoSQL 데이터베이스는 전통적인 관계형 데이터베이스 보다 덜 제한적인 일관성 모델을 이용하는 데이터의 저장 및 검색을 위한 매커니즘을 제공한다. 전자와 같은 접근은 동기는 디자인의 단순화와 수평적 확장성의 이유가 있다. NoSQL 데이터베이스는 단순한 SQL 검색이나 추가 작업을 위한 매우 최적화된 Key Value 저장 공간방식이다. Latency 와 Through put 과 관련하여 상당한 성능 이익을 내는 것이 목적이다. NoSQL 데이터베이스는 빅데이터와 실시간 웹 어플리케이션 이용에 널리 쓰인다. 또한 NoSQL 시스템은 SQL 계열 쿼리 명령어를 사용할 수 있다

는 사실을 강조한다는 면에서 “Not Only SQL” 이라고 불리기도 한다. 이러한 특성으로 NoSQL 데이터베이스 방식의 로그 분석은 수평적 확장을 극대화 시킬수 있어서 빅데이터 분석에 주로 쓰이곤 한다. 하지만 한가지 아쉬운 점이 있는데, 그것은 바로 단독적으로 쿼리를 보내어 검색 및 분석을 할수 없다는 단점이 있다. NoSQL 방식은 Apache Hive 와 같은 소프트웨어를 이용하여 하이브 가상 테이블로 테이블을 본떠서 R 과 같은 분석 소프트웨어를 통하여 분석을 하는 단계가 추가가 되어진다. 이러한 복잡한 단계로 인해 다양한 설정문제가 발생할 수 있다[4][6][8][16].

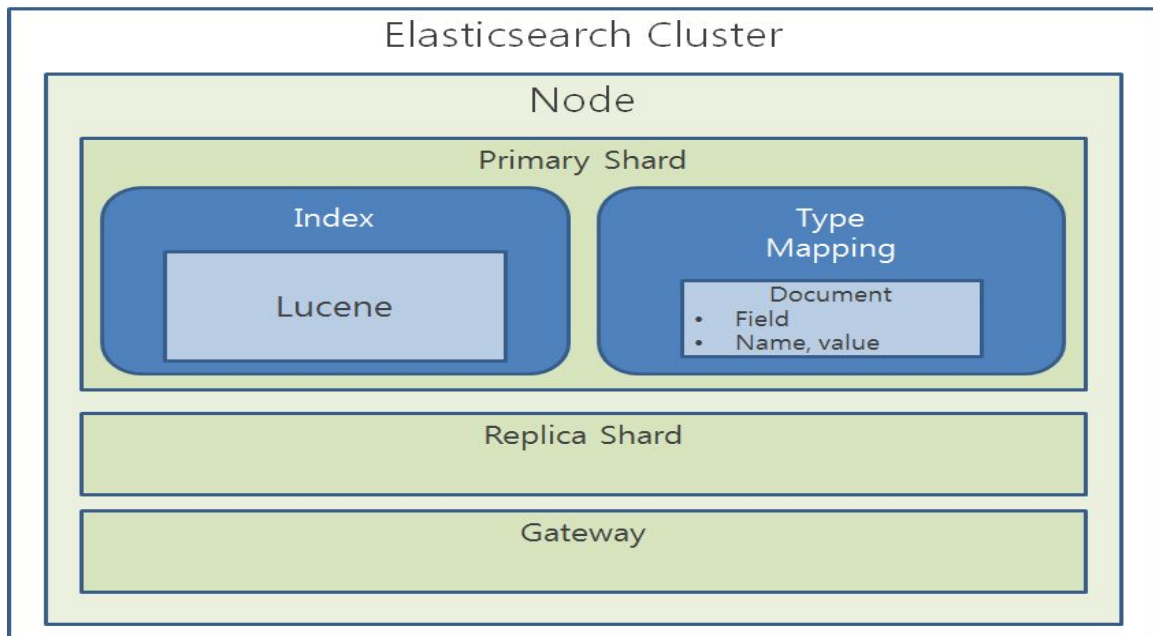
2.1.5 분산검색엔진 방식의 로그 분석

분산검색엔진은 데이터 분석, 문서 검색 등 모든 각종 서비스에 적용되어지는 필수 요소 중 하나이다. 분산검색엔진의 개발은 고급 기술 분야의 영역으로 자리 잡고 있어 국내 대형 기업이 아니라면 값 비싼 라이선스 비용과 운영, 유지보수 등을 감당하기 어려웠다. 하지만 빅데이터 시대가 열리면서 빅데이터에 대한 플랫폼 검색엔진들이 개발되어 출시됨에 따라, 오픈소스 검색 라이브러리인 아파치 루씬(Apache Lucene), 솔라(Solar), 오픈소스 분산 검색엔진인 엘라스틱서치 등이 나오면서 벤더 중심의 검색 솔루션을 사용하지 않아도 빅데이터 분석 시스템 구축이 가능하게 되었다. 엘라스틱서치는 샤이 베논(Shay Banon)이 2010년 처음 발표하여 아파치 루씬 기반으로 자바 환경에서 실행 가능할 뿐 아니라 RESTful API를 통해 데이터의 입력, 삭제, 검색 등을 할 수 있다. 자바, PHP, 펄, 자바스크립트, 파이썬, 루비 등의 라이브러리를 제공하여, 플러그인 설치도 지원한다. 여러가지 다양한 기능들을 쉽게 확장 설치 할 수 있는 점도 엘라스틱서치의 특징중 하나이다. 이러한 특징들로 로그 분석 시스템에 분산 검색엔진을 이용함으로써 더욱 강력한 기능들을 사용하여 로그분석이 가능하다.

2.2 분산검색엔진

2.2.1 엘라스틱서치(Elasticsearch) [7]

엘라스틱서치(Elasticsearch) 는 샤이 베논(Shay Banon) 이 루씬(Lucene)을 바탕으로 개발한 분산검색엔진 이다. 엘라스틱서치는 2010년에 처음 발표하여 지금까지 각종 커뮤니티에서 개발이 이루어지고 있으며, 2012년에는 Elasticsearch 법인을 설립하여 현재 데이터의 흐름을 보기 좋게 효과적으로 분석해주는 로그스태시(Logstash)와 시각화 해주는 키바나(Kibana)와 같은 다양한 데이터 관련 기술들이 있다. 엘라스틱서치는 설치와 서버확장이 매우 편리하기 때문에 개발하고 있는 시스템에 검색 기능이 필요하다면 엘라스틱서치를 적용하기가 원활하다. 분산 시스템이기 때문에 검색 대상 용량이 증가 했을 때 대응하기가 무척이나 수월하다는 것이 장점이다. 엘라스틱서치 클러스터는 다음 [그림 2-2] 와 같이 구성되어 있으며 Node 는 클러스터를 이루는 물리적인 서버, 각 Shard 는 Index의 subset 개념으로 Lucene을 사용하여 구성되어 있다[9]. 실제 데이터와 색인을 저장하고 있으며 Primary Shard와 Replica Shard로 분류 된다. Primary Shard 는 Shard를 구성하는 기본 인덱스 이며, Replica Shard는 분산된 다른 node에 저장된 Primary Shard 의 복제본이다. Type은 문서타입(Document Type)로 index 내에서의 논리적인 category/partition 이며 DBMS에서 테이블과 유사한 개념이다. Document 는 엘라스틱서치에서 관리하는 기본적인 데이터(정보)의 저장 단위이고, JSON(JavaScript Object Notation)으로 표현되어 있다. 또한 Field는 Document를 구성하고 있는 항목으로 name 과 value로 구성되어 있다. Gateway는 클러스터 상태 및 인덱스 설정 등의 정보를 저장하고 있다[7][10].



[그림 2-2] 엘라스틱서치 클러스터 구성도

2.2.2 엘라스틱서치의 특징 [7][16]

○ 분산병렬처리와 확장성

엘라스틱서치는 스케일 horizontally(수평적 확장) 하게 설계되어 있기 때문에 대용량의 환경에서는 그저 노드를 추가하여 클러스터가 인식할 수 있게 해서 추가적인 하드웨어로 사용할 수 있도록 해주기만 하면 된다. 같은 클러스터 내에서는 초기설정 그대로 각 노드끼리 연결되지만, 다른 클러스터에 있다면 설정이 필요하다[7].

○ 멀티테넌시

하나의 클러스터 내에서 Indices와 Document type을 활용하여 멀티클라이언트를 구성 및 운용 그리고 서비스를 할 수 있다. 예를 들면, ABC_mall 이라는 indice 에 각종 쇼핑물들을 Document type 으로 분리하여 생성하거나 쇼핑물들을 indice 별로 분리하여 ABC_mall 이라는

Alias을 생성할 수 있게 된다.

○ 실시간 검색, 분석 서비스

Real-Time 으로 발생하는 데이터 기반으로 검색했을 때 결과에 반영하거나 분석을 통한 결과를 실시간으로 제공받을 수 있다.

○ 플러그인 형태의 RESTful API 구현

엘라스틱서치를 직접 수정하지 않고 필요한 기능에 대한 플러그인을 설치하고 적용하여 기능을 확장 시킬 수 있다. 대부분의 명령들이 HTTP 프로토콜로 실행할 수 있다. 예를 들어, RESTful API를 이용해 'http://host:port///[id]' 형식으로 접근할 수 있다. 유닉스 환경에서는 curl 명령어로 접근이 가능하다. 또한 입출력과 환경설정까지 JSON 데이터로 처리 할 수 있다.

○ 문서 중심의 Document oriented

엘라스틱서치는 복잡한 현실세계의 엔티티들을 구조화된 JSON 문서 형식으로 저장 한다. 모든 필드는 기본적으로 인덱싱이되며, 모든 인덱스들은 단일 쿼리로 빠르게 사용할 수 있다는 장점이 있다. 그리고 사용자의 데이터가 어떻게 인덱싱 될건가에 관한 것은 사용자가 설정을 통하여 커스터마이징 될 수 있다.

다음 [표 2-3]은 관계형 데이터베이스와 엘라스틱서치의 데이터 구조 비교에 관한 것이다.

[표 2-3] 관계형 데이터베이스와 엘라스틱서치 데이터 구조 비교

관계형 데이터베이스	엘라스틱서치
------------	--------

데이터베이스(Database)	인덱스(Index)
테이블(Table)	타입(Type)
로우(Row)	도큐먼트(Document)
컬럼(Column)	필드(Field)
스키마(Schema)	맵핑(Mapping)

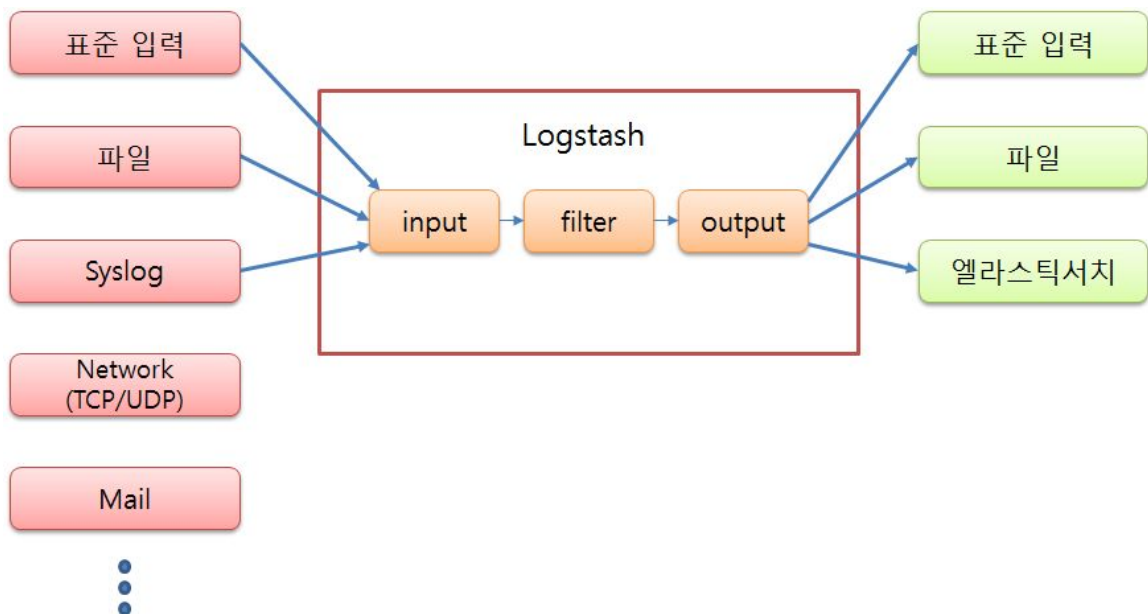
2.2.2.1 엘라스틱서치 대시보드 - 키바나(Kibana)

키바나(Kibana)는 엘라스틱서치의 대시보드 어플리케이션 중에 하나이다. 키바나3(Kibana3)의 경우는 순전히 웹앱(Web App) 으로 개발되어 어플리케이션 상에서 직접 엘라스틱서치의 HTTP API를 호출하였으나, 키바나4(Kibana4) 의 경우는 중간 프록시(Proxy) 서버를 실행하도록 만들어, 웹 클라이언트에서 직접 엘라스틱서치에 접근하지 않도록 하였다. 정형 및 비정형 데이터를 엘라스틱서치에 인덱스 및 엘라스틱서치와 함께 작동하도록 설계되어있고, 키바나는 모든 종류의 데이터에 시각화를 제공한다. 또한 지능적으로 데이터를 분석 및 수학적 변환을 수행하고 사용자 입장에서 이해하기 쉽게 막대 차트, 라인과 분산 형 그래프, 막대 그래프 등 다양한 차트 등을 제공해 준다. 또한 유연한 인터페이스와 쉬운 공유가 특징이다.

2.3 로그 컬렉터

2.3.1 로그스테시(Logstash)

로그스테시(Logstash)는 다양한 종류의 로그 데이터(System log, Webserver log, Error log, Application log)를 입출력 및 가공 처리 할 수 있는 엘라스틱서치의 수많은 플러그인 중 하나이다. 현재는 엘라스틱서치의 공식 패키지에 포함되어 있다. [그림 2-3] 은 로그스테시의 표준 파일 입출력 과정을 나타낸 그림이다.

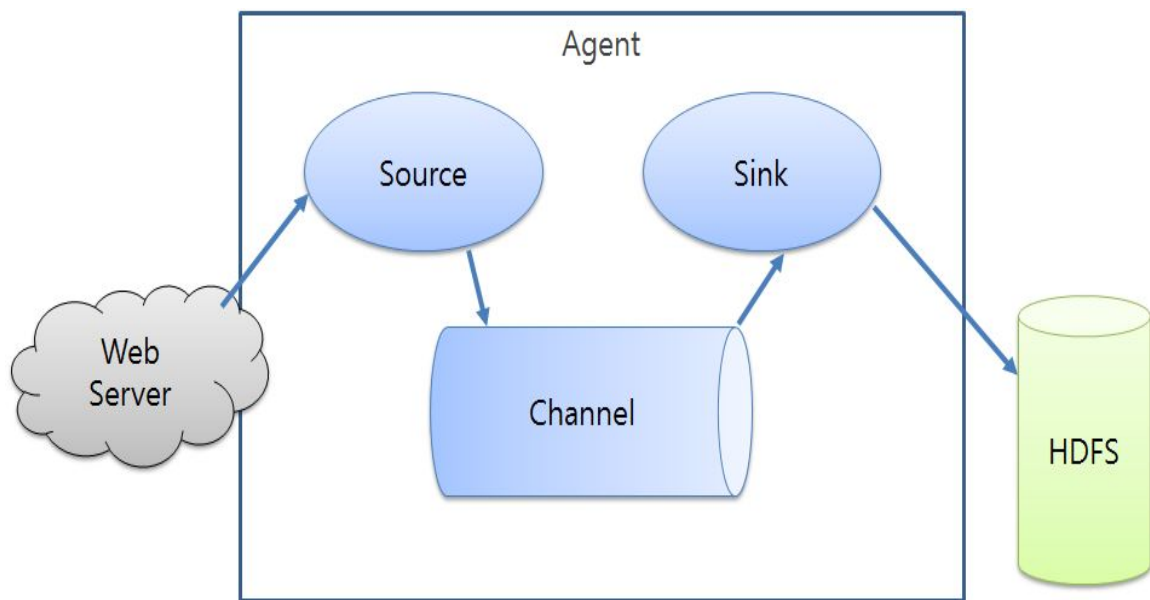


[그림 2-3] 로그스테시(Logstash) 입출력 과정

2.3.2 아파치 플룸(Apache Flume)

아파치 플룸(Apache Flume)은 대량의 로그 데이터를 수집하여 효율적으로 읽어 들이기 위한 분산프레임 워크 이다. 아키텍처는 데이터 소스(Data Source)의 데이터를 다른 저장소로 스트리밍하기 위한 심플한 구조를 가지고 있다. 아파치 플룸은 데이터 플로우 사이사이에 실패하는 상황에 대해 빠르게 복구하는 복구성에 중점을 두고 있어 대량의 로그 파일을 이동시키는데 적합하고 유용하다. 플룸(Flume)은 2011년 Cloudera CDH3에 처음으로 소개가 되었으며, 현재는 아파치(Apache)의 Top-Lev

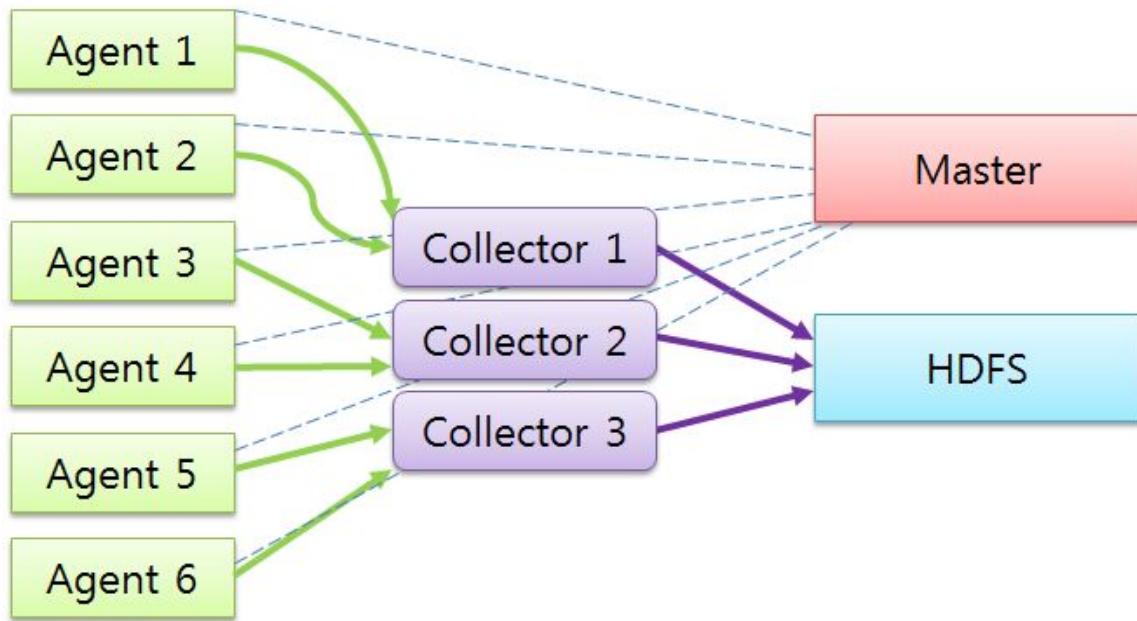
el Project로 이전 CDH3에서의 버전인 0.9 x버전은 Flume-OG라 명명하고 1.0.0 이후의 버전부터 Flumg-NG(Next Generation) 이라고 명명하며 현재 최신버전은 1.4.0 이다. [그림 2-4]는 Flume NG의 기본 아키텍처 이다.



[그림 2-4] Flume NG의 기본 아키텍처

플룸은 다음 [그림 2-5]와 같이 세 개의 계층으로 구성되어 있다. Agent 계층, Collector 계층, Storage 계층 3가지이다. 먼저 에이전트 계층에서 각 에이전트 노드들은 수집해야할 로그 데이터가 생성되는 머신에 설치가 하는 것 이 일반적이다. 에이전트에서 수집한 데이터는 각자의 컬렉터 노드로 전송이 되어 모아진다. 컬렉터 노드는 보통 다른 머신에 있으면서 여러개의 컬렉터 노드를 구성할 수 있다. 에이전트 노드에서 컬렉터 노드로 의 로그 데이터를 전송할 때에 목적지로 보내고 처리를 할 것인지에 대한 데이터 흐름 설정을 할 수 있으며, 설정한 대로 로그 데이터를 이동시켜 스토리지 계층에 저장하도록 한다. 데이터의 흐름을 담당하는 것이 바로

마스터 노드이며, 마스터 노드는 각 논리적 노드에서 실행해놓은 상태에서 마스터 노드를 이용해 설정을 변경할 수 있다. 즉, 데이터 플로우를 동적으로 지속적인 관리가 가능 하다.



[그림 2-5] 아파치 플룸(Apache Flume)의 계층적 구조도

2.3.2.1 아파치 플룸의 특징[11]

○ 신뢰성

플룸은 신뢰할 수 있는 데이터 전송을 위해 3가지의 수준 신뢰도를 지원한다. Flume이 이벤트를 받아 최종 목적지까지 보장하는 E2E(End-to-End) 신뢰도와 SoF(Store on Failure) 신뢰도는 데이터를 전송하다가 실패를 했을 시 디스크에 저장하여 다시 보낼 수 있게 하거나 다른 컬렉터를 선택할 때까지 기다린 후에 다시 보낸다. BE(best-effort) 신뢰도는 처리중인 데이터가 실패하였을 때 손실을 할 수 있는데, 이것은 가장 약한 신뢰도이지만 굉장히 낮은 확률이다.

○ 수평적 확장성

플룸은 수평적 확장을 지원한다. 이것은 포함되어 있는 시스템에 부가적인 머신을 추가함으로써 전체 Through put을 향상시킬 수 있다는 의미이다. 계층별로 부하량에 따라 다르게 노드를 추가선택 하여 전체적인 성능을 향상시킬 수 있다.

2.4 기타연구사례

2.4.1 아파치 하둡 [14]

○ 하둡 분산 파일 시스템

하둡 분산 파일 시스템은 일반적인 분산 파일 시스템과 같이 Master-Slave 구조로 구성되어 있다. 하둡 분산 파일 시스템에서의 마스터노드의 역할은 Namenode 라 불리우는 프로그램이 담당하고 있고, Slave node의 역할은 Datanode라 불리우는 프로그램이 담당하고 있다. Namenode 는 마스터 노드와 같이 Datanode 들을 관리하며 Datanode 에 저장되어 있는 사용자 데이터의 Metadata를 관리 한다. Datanode는 Slave node와 같이 사용자 데이터를 저장하고 사용자의 요청에 의해 데이터를 저장, 복제, 보관 하는 등의 역할을 한다.

○ 마스터 노드

분산 파일 시스템에서 마스터 노드가 하는 역할은 크게 두 가지가 있다. 첫 번째로 현재 분산 파일 시스템에서 사용하고 있는 슬레이브 노드 관리 이다. 일반적인 분산 파일 시스템에서는 몇십, 몇백, 몇천 대의 슬레이브 노드들이 동작하고 있다. 마스터 노드는 각각 슬레이브 노드의 상태를 체크하고 있어야 하지만 사용자에게 의해 데이터 저장요청이 오거나 데이터 다운로드 요청이 왔을 때 어떤 슬레이브 노드에 데이터를 저장 혹은 다운로드 할지 결정 할

수 있다. 마스터 노드의 가장 중요한 역할은 어느 슬레이브 노드가 현재 분산 파일 시스템에서 동작 중인지, 아니면 어떤 슬레이브 노드가 작동하지 않는지에 대한 정보를 실시간적으로 파악하고 있는 것이다.

○ 슬레이브 노드

슬레이브 노드가 사용자 데이터를 저장하는 역할을 하는데, 여기에서 슬레이브 노드는 파일이 업로드 되면 사용자의 파일을 저장하고, 사용자가 데이터 다운로드를 했을 시에 데이터를 전달 해 주는 역할을 한다. 기본 역할 이외에도 분산 파일 시스템 환경에서 슬레이브 노드가 하나의 파일을 여러 개의 슬레이브 노드에 복제해서 관리하는 역할도 한다. 이러한 기능은 분산 파일 시스템의 가장 강력한 기능이기도 하다. 일반적인 데이터 디스크는 자주 고장을 일으키는 반면 이 기능을 이용하면 사용자의 데이터를 보다 안전하게 보관 가능하고, 서버장애 시에도 언제든지 데이터를 사용 가능하게 할 수 있다.

○ 하둡 맵리듀스

맵리듀스는 분산 클러스터 환경에서 대용량 데이터를 병렬로 처리하기 위해 개발되었다. 하둡에서 데이터 분석을 위해 사용되어지는 맵리듀스는 마스터-슬레이브 구조로 되어있다. 마스터 노드의 역할은 잡 트래커라 불리는 프로그램이 담당하고 있다. 잡 트래커는 맵리듀스가 수행할 때 전체 작업을 중앙관리 하는 역할을 한다. 슬레이브 노드는 태스크 트래커라 불리는 프로그램이 담당하고, 맵 리듀스 작업은 태스크 트래커가 수행하게 된다.

2.4.2 하둡의 특징 [4]

○ 선형적인 확장성

일반적인 저장 공간은 초기에 어플리케이션에서 사용할 스토리지 용량을

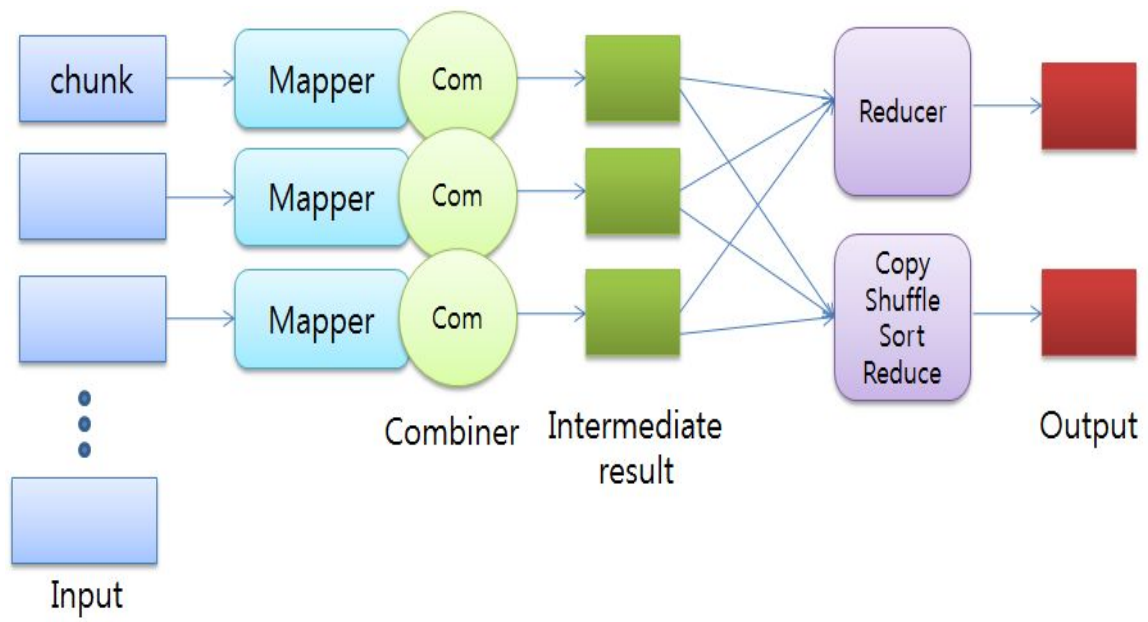
미리 확보한 상태에서 시스템이 동작 한다. 반면에, 하둡 파일 시스템을 이용할 경우에는 서비스 초기에 필요한 저장 공간을 확보해 시스템을 가동한 후 스토리지 증가 상태를 파악하면서 추가하는 방식으로 진행 가능하다.

○ 데이터 분석 처리에 활용

방대한 양의 분석용 데이터를 하둡 파일 시스템에 저장을 하여 MapReduce 라는 분산 병렬 처리 프레임워크를 이용해 레코드 집합으로 하여 각 레코드는 Key 와 Value를 갖는다. MapReduce 작업은 Map 작업과 Reduce 작업으로 나뉘는데 하나의 데이터 셋을 다른 데이터 셋으로 변환시키는 과정에서 마지막으로 하나의 형태의 Key 와 Value 의 쌍으로 다른 형태의 Key 값과 Value 값의 집합으로 변환하게 된다. 이러한 작업은 레코드 별로 이루어짐에 따라 병렬성이 높아지고 다수의 서버에서 다루기 쉽다는 장점이 있다.

2.4.3 맵리듀스(MapReduce)

맵리듀스(MapReduce)는 구글(Google)에서 제작한 대용량 데이터를 위한 분산 병렬 컴퓨팅 환경에서 데이터 처리를 하기 위한 목적으로 2004년에 제작하여 발표한 소프트웨어 프레임워크이다. 이 맵리듀스 프레임워크는 페타바이트(peta byte) 이상의 대용량 데이터를 신뢰도가 낮은 클러스터 환경에서 병렬처리를 지원하기 위하여 개발 되었다. 일반적으로 함수형 프로그래밍에서 일반적으로 사용되어 Map 과 Reduce 라는 함수 기반으로 구성이 되어 진다[17].



[그림 2-6] 맵리듀스(MapReduce) 처리 흐름도

2.4.4 기타 관련 논문 [4][12][13][14]

로그의 수집 및 분석 그리고 보안관리 시스템 성능 향상을 위해 다양한 연구가 이루어지고 있다. 표[2-4]의 논문들은 하둡을 이용한 데이터 저장 및 맵리듀스의 활용과 보안 장비에서 발생하는 로그 데이터를 통합적으로 분석하였다. 또한 빅데이터 플랫폼 기술들을 활용하여 보안로그를 분석하기 위한 시스템 구현에 대한 모델을 연구하였다.

[표 2-4] 기타 관련 논문

관련 논문	내용	
로그 분석을 통한 침입 대응 시스템 설계 및 구현	설명	보안로그 데이터를 패킷 필터링 모듈과 필터링 데이터 저장 모듈, 패킷 처리 모듈을 통하여 방화벽에서 발생하는 보안로그를 분석하여 모니터링
	특징	침입대응시스템 환경에서 로그파일을 XML 형식의 변환을 통해 데이터 구조화에 따라 시스템 로그 파일을 분석 및 관리할 수 있도록 하였고, 로그 데이터의 상호 처리 능력 향상.
NoSQL 기반의 MapReduce를 이용한 방화벽 로그 분석 기법	설명	NoSQL 기반의 맵리듀스 설계 방식을 이용하여 방화벽 로그 분석 기법을 통해 대용량 로그 분석 효율 및 분석 기법 성능 입증
	특징	다양한 공격별 유형을 탐지 목적에

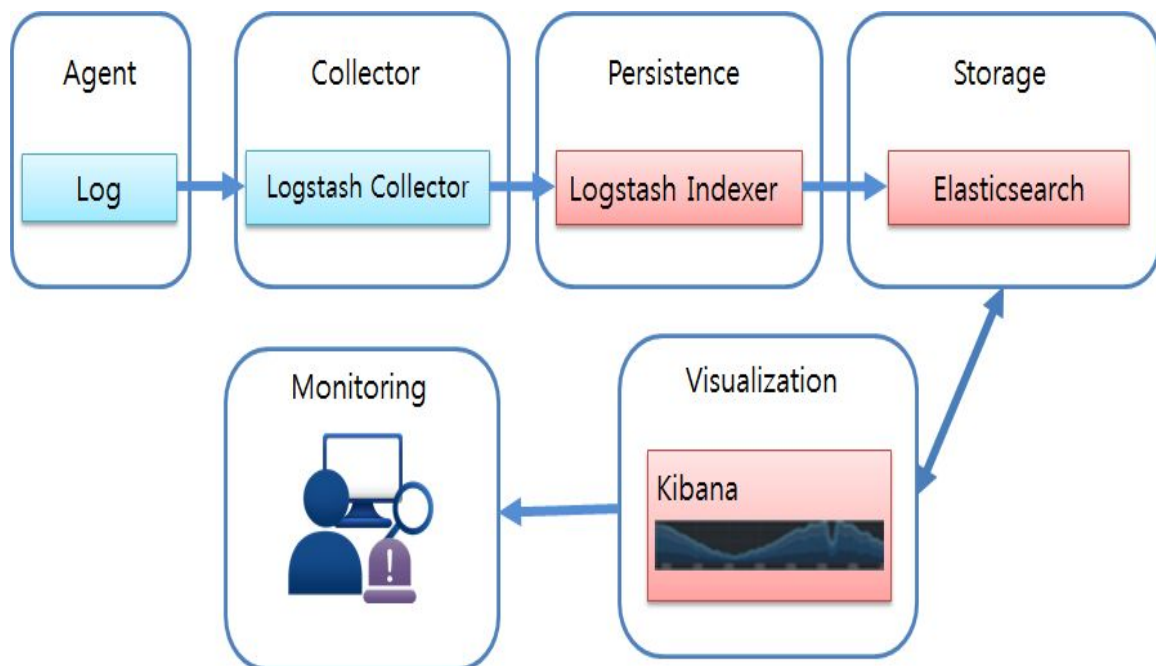
		따라 동적으로 맵(map)의 Key를 설정하는 방식과 이를 통해 다양한 공격 패턴에 대응 할 수 있음.
효율적인 빅데이터 분석을 위한 로그 데이터 기반의 로그분석 지원 시스템	설명	로그 분석 지원 시스템은 수집되는 로그 데이터를 미리 분석하고, 분석 결과에서 의미있는 결과를 추출하여 분석가를 지원할 수 있는 시스템
	특징	빅데이터 관리 시스템에서 발생할 수 있는 오버헤드를 최소화하고, 샘플링을 통한 R 기반의 분석 지원 환경을 분석가에게 제공할 수 있다.
하둡 기반의 정규식을 이용한 효율적인 보안 로그분석시스템 설계 및 구현	설명	정규식을 이용하여 수집되는 로그 데이터를 R과 Hive를 이용하여 분석 및 시각화를 하여 효율적인 보안 로그 분석 시스템
	특징	로그 데이터를 정규식을 이용하여 사용자가 필요한 로그를 추출하여 효율적인 보안 로그 분석 시스템 이다.

제 3 장 엘라스틱서치를 이용한 보안로그 분석 시스템

본 논문에서 제안하는 엘라스틱서치를 이용한 보안로그 분석 시스템을 구축하기 위한 사항들을 설명하였다. 이 시스템을 기반으로 보안장비에서 발생하는 로그 데이터 수집, 분석, 저장 및 데이터 시각화 등 전체적인 구성 요소들을 설계 및 구현 하였다. 기존 하둡 기반의 보안로그 분석 시스템과 비교하여 테스트 및 환경설정을 수행하였다.

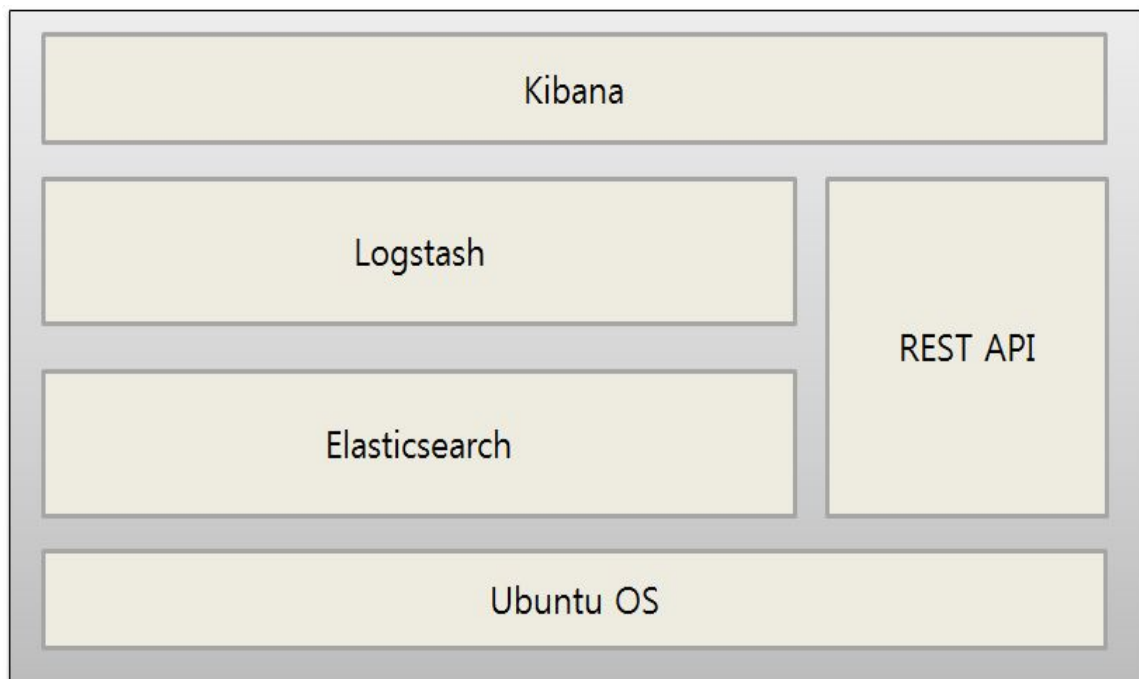
3.1 엘라스틱서치를 이용한 보안로그 분석 시스템 설계

엘라스틱서치를 이용한 보안로그분석 시스템 전체 구성도는 아래 [그림 3-1]과 같다. 에이전트와 보안 디바이스에서 발생하는 보안 로그 데이터를 로그스태시 컬렉터를 이용하여 데이터를 수집 후 로그스태시 인덱서와 엘라스틱서치 싱크를 이용해 엘라스틱서치에 저장하여 시각화 도구인 Kibana를 이용해 시각화 및 사용자 모니터링을 확인할 수 있도록 구성을 하였다.



[그림 3-1] 엘라스틱서치를 이용한 보안로그 분석 시스템 구성도

제안하는 시스템은 크게 4가지 부분으로 구성된다. 먼저, 로그스태시 컬렉터에서 에이전트 부분에 존재하는 방화벽, 웹서버 등의 보안 장비에서 나오는 로그데이터를 수집한다. 다음으로 로그스태시 컬렉터에서 로그스태시 인덱서로 로그데이터를 모은후에 엘라스틱서치에 데이터를 저장한다. 하둡 파일시스템(HDFS)과 분산검색엔진인 엘라스틱서치를 이용해 로그 데이터를 인덱싱하여 필요한 때에 실시간으로 데이터를 분석하고 검증할 수 있다. 설정부분에서는 각 인덱스를 여러 개의 부분(Shards)로 나눌 것인지와 이 부분(Shards)을 얼마나 복제 할 것인지에 대해 설정하여 관리한다. 마지막으로 제안하는 시스템은 웹브라우저 기반의 분석 및 검색엔진 인터페이스를 제공하는 Kibana를 이용하여 시각적으로 로그 통계 및 검색 리포트를 생성하고 그 결과를 시각화 할 수 있게 된다. [그림 3-2]는 본문에서 제안한 시스템에 대한 전체 시스템 아키텍처를 표현한 아키텍처 그림 이다.

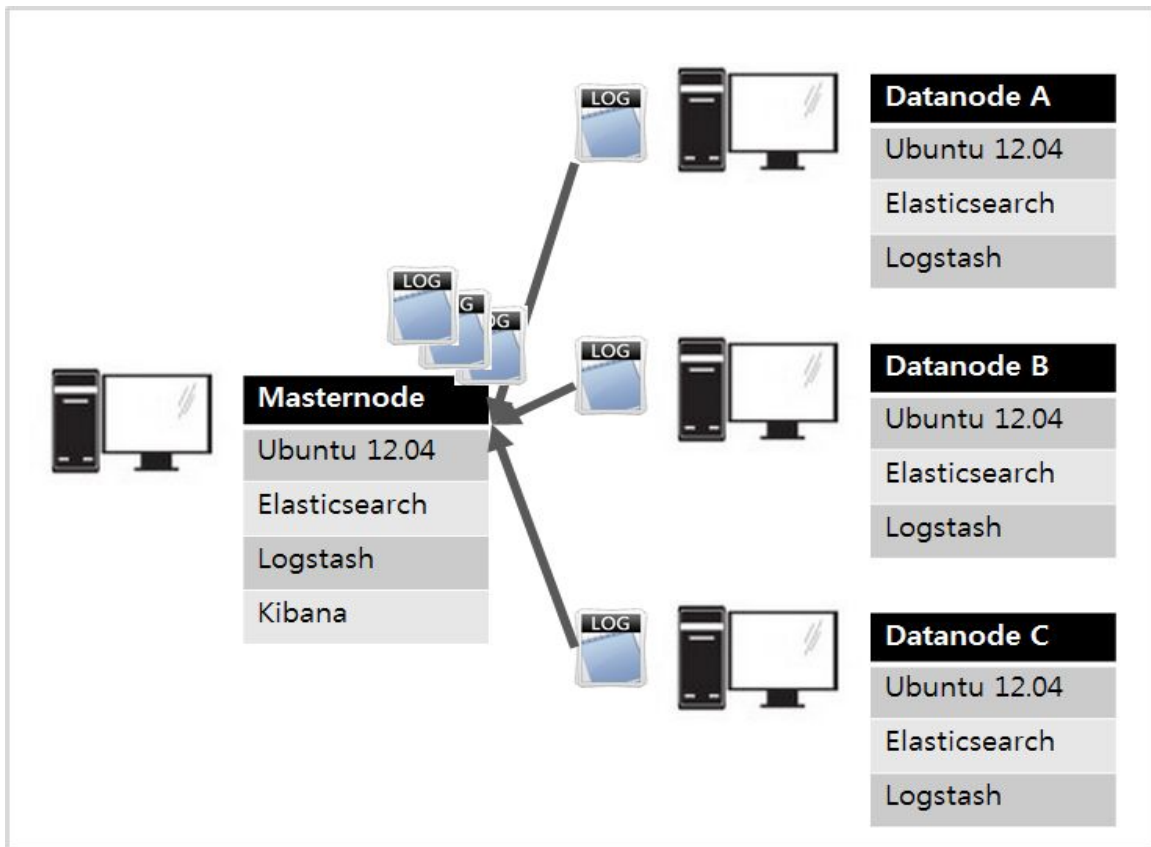


[그림 3-2] 엘라스틱서치를 이용한 보안로그 분석 시스템 아키텍처

3.2 엘라스틱서치를 이용한 보안로그 분석 시스템 환경 구축

엘라스틱서치를 이용한 보안로그 시스템 환경구축을 하는데 총 4대의 서버가 사용되었다. 네임노드 1대 와 3대의 데이터 노드로 구성하였으며 네임노드의 운영체제는 Ubuntu 12.04 기반으로 하였다. 실험 및 비교를 위해 분산 파일 시스템과 맵리듀스를 위해 하둡을 설치하였다. 로그 데이터가 발생하면 로그스태시 Collector 에서 로그 데이터를 전송하고, 오픈소스 검색엔진인 엘라스틱서치를 이용하여 저장을 한 후에 Kibana 를 이용하여 로그 통계, 모니터링을 통해 확인을 가능하게 하였다. 분산검색엔진인 엘라스틱서치를 설치하여 실행하면 하나의 노드가 마스터 노드로 뽑히게 된다. 마스터 노드가 작동을 멈추게 되면 또 다른 노드가 새로운 마스터 노드가 된다. 각자의 노드는 여러 개의 Shard 로 데이터를 분산 저장하며 Shard는 복사본을 갖는다. 노드가 정지하게 될 경우 정지된 노드의 데이터를 다른 노드에 복사를한 후에 자동으로 정렬된다.

데이터노드의 운영체제는 Ubuntu 12.04 운영체제를 설치하였으며, 각 노드에는 엘라스틱서치 노드 분산검색엔진 환경을 구성하였다. 대용량의 로그 데이터를 여러 대의 데이터 노드에 분산 저장을 위한 서버 이다. 데이터 블록을 저장하기 위해 사용되는 하둡과 엘라스틱서치의 Shard 기술을 사용할 수 있는 서버를 구성하기 위한 엘라스틱서치 클러스터로 구성되어 있다. 본 논문에서 제안하는 시스템의 노드 구성도는 [그림 3-3]과 같으며 시스템 사양 및 정보는 아래 [표 3-1]과 같다.



[그림 3-3] 노드 구성 및 로그 데이터 흐름도

[표 3-1] 노드구성 사양정보 및 툴

	CPU	RAM	HDD	OS	Tools
Masternode	Quad Core 2.80GHz	16.0GB	1TB	ubuntu 12.04	Elasticsearch-1.5.1 Logstash-1.4.2 Kibana-4.0.2
Datanode A	Quad Core 2.80GHz	16.0GB	1TB	ubuntu 12.04	Elasticsearch-1.5.1 Logstash-1.4.2
Datanode B	Quad Core 2.80GHz	16.0GB	1TB	ubuntu 12.04	Elasticsearch-1.5.1 Logstash-1.4.2
Datanode C	Quad Core 2.80GHz	16.0GB	1TB	ubuntu 12.04	Elasticsearch-1.5.1 Logstash-1.4.2

3.2.1 엘라스틱서치 멀티 테넌시 구축

엘라스틱서치는 검색엔진이지만, NoSQL처럼 이용할 수 있다. 데이터 모델을 JSON을 사용하고 있기 때문에, 요청 및 응답을 모두 JSON 문서로 주고 받을 수 있고 저장 형태도 JSON 형태로 저장할 수 있다. schema를 미리 정의하지 않아도, JSON 문서를 통해 넘겨주면 자동으로 인덱싱할 수 있다. 숫자 또는 날짜 등의 타입은 자동으로 매핑한다. 엘라스틱서치는 multi-tenancy를 지원한다. 하나의 엘라스틱 서버에 여러 개의 인덱스를 저장하고 여러 인덱스의 데이터를 하나의 쿼리로 검색할 수 있다. 예를 들어 날짜별로 인덱스를 분리하여 로그를 저장할 수 있고, 검색 할 때에는 검색 범위에 포함되는 날짜의 인덱스를 하나의 쿼리로 요청 할 수 도 있다.

○ 분산 저장

엘라스틱서치는 분산 검색엔진이다. Key 에 따라 다수의 Shard 가 구성되는 형식으로 데이터를 분산 저장 한다. 인덱스는 각 Shard 마다 구성되며, 각각의 Shard 는 0개 이상의 복사본을 가지게 된다. 엘라스틱서치는 클러스터링을 지원하여 클러스터가 가동될 때에 다수의 노드 중 하나에서 메타데이터 관리를 위한 마스터 노드가 선출이 되는 방식이다. 마스터 노드가 동작을 정지 하였을 시에, 클러스터 내의 노드중 하나가 마스터가 된다. 노드 추가가 간단하며 네트워크 상에 노드를 추가하는 경우에 추가된 노드가 멀티캐스트를 사용해 자동으로 클러스터를 찾아가 자신을 추가하게 되는 방식이다. 다음의 표는 엘라스틱서치의 용어와 관련한 표 이다.

[표 3-2] 엘라스틱서치 용어 설명

용어	상세내용
클러스터(Cluster)	<ul style="list-style-type: none"> Node 의 집합으로 유일한 이름을 가진다.
노드(Node)	<ul style="list-style-type: none"> Cluster를 이루는 물리적인 서버이다.
인덱스(Index)	<ul style="list-style-type: none"> 유사한 특징을 가진 문서들의 모음으로 DBMS에서 데이터베이스와 유사한 개념 Term, Count, Docs 로 구성된다.
샤드(Shard)	<ul style="list-style-type: none"> Index의 subset 개념으로 Lucene을 사용하여 구성 실제 데이터와 색인을 저장하고 있으며 Primary Shard 와 Replica Shard 로 분류 Primary Shard : Shard 를 구성하는 기본 인덱스 Replica Shard : 분산된 다른 node 에 저장된 Primary Shard의 복제본 이다. 서비스 장애시 서비스의 연속성을 보장한다.
타입(Type (Document Type))	<ul style="list-style-type: none"> 데이터(Document)의 종류로 index 내에서의 논리적인 category/partition DBMS에서 테이블과 유사한 개념
매핑(Mapping)	<ul style="list-style-type: none"> DBMS에서 테이블 스키마와 유사한 개념
Route	<ul style="list-style-type: none"> 색인 필드중 unique key 에 해당하는 값을 routing path로 지정한 후, 이 path를 사용하여 인덱싱과 검색에 사용할 shard를 지정하여 성능을 향상 시킬 수 있다. Routing Field : store 옵션을 yes로 index not_analyzed 로 설정
Document	<ul style="list-style-type: none"> Elasticsearch 에서 관리하는 기본적인 데이터(정보)의 저장단위 JSON(JavaScript Object Notation)으로 표현 DBMS에서 레코드와 유사한 개념
Field	<ul style="list-style-type: none"> Document를 구성하고 있는 항목으로 name과 value 로 구성 DBMS에서 컬럼과 유사한 개념
Gateway	<ul style="list-style-type: none"> Cluster 상태, Index 설정 등 정보를 저장
Query	<ul style="list-style-type: none"> 검색어
TermQuery	<ul style="list-style-type: none"> 검색어의 종류
Term	<ul style="list-style-type: none"> 검색어의 항목
Token	<ul style="list-style-type: none"> 검색어의 항목을 구성하는 요소

3.2.2 엘라스틱서치 클러스터 환경 구축

본 논문에서 제안하는 보안로그분석 시스템에 효율적인 로그 수집을 위해서 관리자는 로그 수집을 위해 Logstash 의 설정을 통하여 다양한 종류의 로그데이터를 수집할 수 있다. 또한 Kibana를 통한 로그 데이터를 검색 및 분석 하여 통계 그래프로 시각화해서 사용자에게 분석을 더욱 용이하게 만들 수 있다. 검색엔진의 최대장점인 실시간 검색 능력과 강력한 REST API 를 바탕으로 로그 데이터를 분석하여 에러 혹은 악의적인 공격 여부를 판단할 수 있다. Kibana를 사용하여 엘라스틱서치에 저장된 로그 데이터를 Query 문을 사용하여 검색이 가능하다. Kibana 와 연동하여 검색엔진인 엘라스틱서치의 Document 형식으로 저장된 Key 와 Value 값들을 효율적이게 검색하여 사용자에게 검색의 결과를 시각적으로 보여 주게 된다.

엘라스틱서치는 하나의 Shard 가 깨지거나 사용불가 상태가 되었을 때에, 다른 Replica 로 자동적으로 복구되어 오랜 시간 지속을 위한 지속성 및 신뢰적인 , 비동기 쓰기가 가능하다. 또한 실시간 검색기능과 다양한 API 및 클라이언트 모듈 지원으로 사용자가 사용하기 간편한 이점도 지니고 있다. 엘라스틱서치 클러스터 환경을 구축하기 위해 각 노드에 엘라스틱서치의 /elasticsearch/config/elasticsearch.yml 파일을 수정하여야 한다.

○ 마스터 노드의 elasticsearch.yml 파일 수정

Elasticsearch.yml 설정 파일에서 확인해야 할 점은 로그 기본 저장 위치를 바꿀 수 있다는 점이다. 기본적으로 날짜순으로 로그 데이터가 나뉘긴하지만 Rotating 이 되지는 않기 때문이다. 메모리 스왑 방지를 위한 옵션도 매우 중요하다. 엘라스틱서치를 설치한 후에 Config 파일을 수

정하였다면, 다음으로 기본적으로 서버가 실행되는 것을 확인하기 위해 `curl 'http://localhost:9200'` 으로 확인이 가능하다. 노드와 클러스터의 상태를 확인하는 API 이며 상세한 정보도 확인이 가능하다.

```
# Cluster
node.rack: ${RACK_ENV_VAR}
# 지정한 이름으로 노드들이 클러스터가 된다.
cluster.name : elasticsearch_test
#각 샤드의 파일 저장위치 설정
path.data: /mnt/elasticsearch/data
#노드의 이름
node.name : "Master node"
#마스터 노드 여부 확인
node.master: true
node.data: false
...
...
#메모리 스왑 방지 옵션
bootstrap.mlockall: true
#포트 넘버
http.port: 9200
#flush 값을 1GB 로 늘려서 인덱싱 향상에 도움을 준다.
index.translog.flush_threshold_size: 1GB
...
```

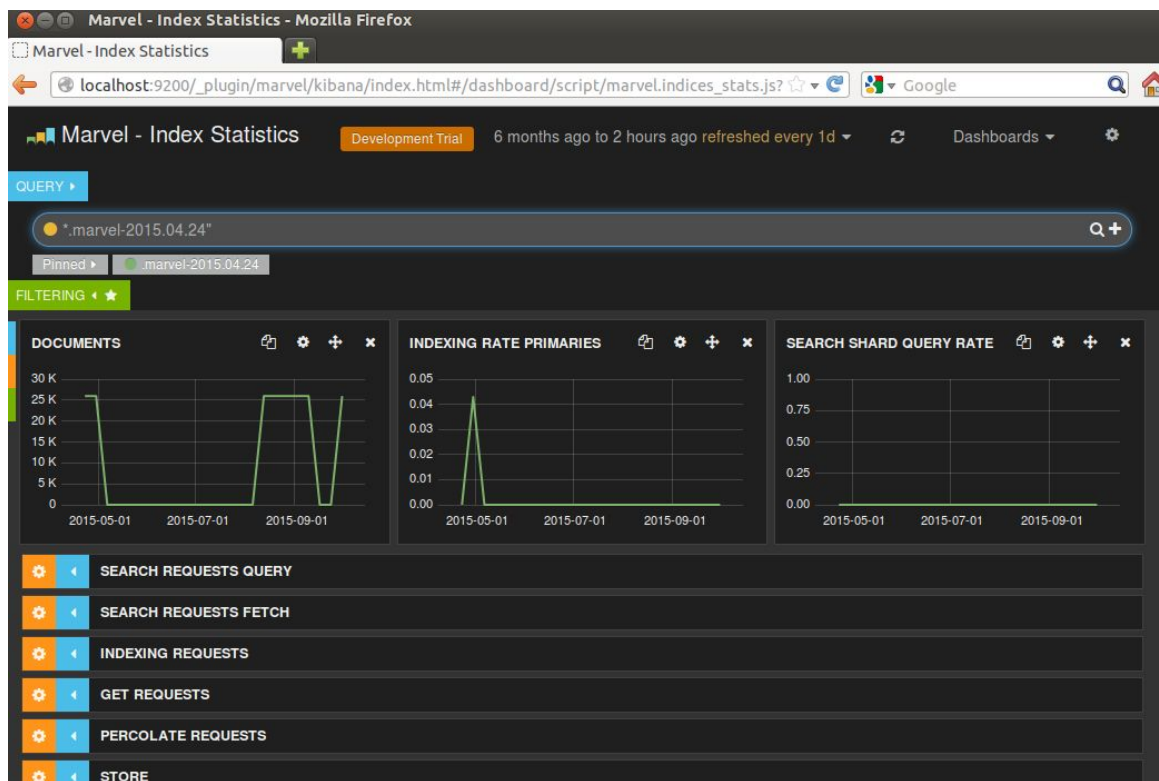
3.2.3 엘라스틱서치 플러그인

엘라스틱서치는 다양한 Plugin 을 제공하며 각각의 기능은 엘라스틱서치 클러스터와 노드들의 상태를 자세하게 보여주며 저장된 데이터 검색 및

통계 수치를 그래프화하여 표현한다.

○ Marvel Plugin

Marvel Plugin은 엘라스틱서치 클러스터 전체 노드들에 대한 상태정보를 인덱스에 색인하고 검색 및 대시보드 형태로 제공한다. [그림 3-4] 는 엘라스틱서치의 Plugin 인 marvel 의 대시보드 화면이며, 간단한 Query 문을 서버에 보내어 현재 서버의 Document 와 Indexing rate 등을 확인할 수 있다.

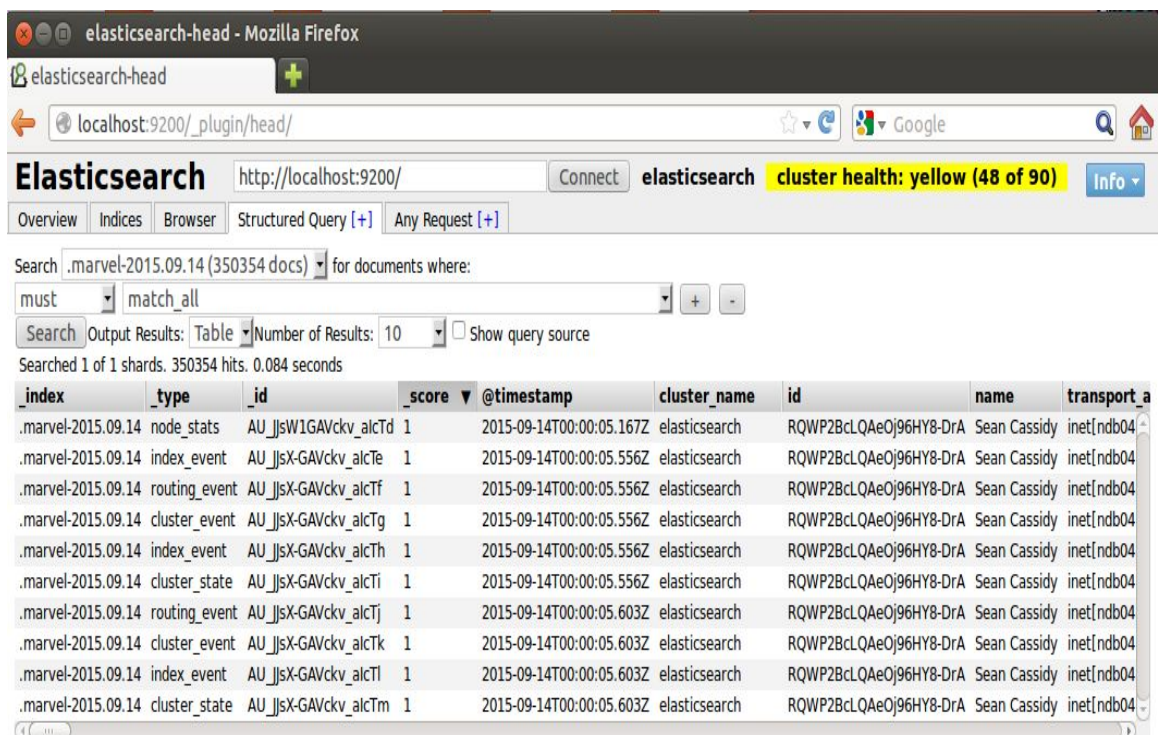


[그림 3-4] 엘라스틱서치 플러그인 Marvel Query 상태 확인

○ Head Plugin

엘라스틱서치 Plugin 인 Head Plugin 은 Index 와 Shard , Document 에 관한 관리 및 상태를 점검할 수 있는 Plugin 이다. 기본적인 검색 질의 기능을 제공하며 설치 명령어로는 \$ /bin/plugin -install

mobz/elasticsearch-head 이다. 각각의 탭으로 Overview는 기본적인 노드 리스트와 노드에 생성되어 있는 Index 리스트를 자세하게 나타내며, Index의 meta data와 관리를 위한 기본 정보들이 포함된다. Brower는 생성한 전체 Index 리스트와 Field , 저장한 데이터 리스트를 보여준다. Structured Query 부분은 Bool Query 형태의 질의 가능한 웹 화면을 제공한다. Any Request 부분은 JSON Query를 생성하고 질의할 수 있는 기능을 제공한다. [그림 3-5]은 기본적인 Head Plugin의 상태 화면이다.



[그림 3-5] 헤드 플러그인 구동 화면

○ Bigdesk Plugin

Elasticsearch 의 검색엔진 상태와 서버의 자원 현황을 실시간 모니터링 할 수있게 해주는 Plugin 이다. marvel 과의 다른점은 각각 인덱스에 상태 정보를 저장하지 않는다는 것이다. 실시간 적으로 검색엔진의 상태를 대쉬보드로 구성하여 실시간 정보외의 다른 과거 데이터 조회는 불가능하다. Cluster API 중에서 Node Stats API 를 사용한다. 설치 명령어는 `/bin/plugin -install mobz/bigdesk` 이다. [표 3-3] 은 Node Stats API 의 항목과 설명에 관한 표이다.

[표 3-3] Node Stats API 항목과 설명

항목	설명
Fs	• 파일 시스템 정보와 Disk 의 읽기/쓰기 통계 정보
Jvm	• JVM 관련 통계와 메모리 풀 정보
Os	• 운영체제의 CPU , MEM , SWAP , LOAD 통계 정보
Thread_pool	• 검색엔진에서 사용하는 Thread_pool 의 통계 정보
Breaker	• JVM 메모리에 등록되는 필드 데이터 통계 정보
Indices	• Index 크기 , Document 수 , 검색 수행시간 , 색인 수행시간 등에 대한 정보
Network	• TCP 통신 정보
Process	• 프로세스에서 사용하는 CPU , MEM 통계 정보
Transport	• Cluster 에서 통신에 주고받은 통계정보

○ 그 외의 기타 Plugin

엘라스틱서치의 그외에 기타 플러그 인 으로 Sense, Elasticsearch HQ, Hammer Plugin, Inquisitor Plugin, SegmentSpy Plugin 등이 있다. 각각의 기능은 REST API를 실행하여 결과를 확인하는 기능과 Head 와 Bigdesk 의 기능이 합쳐 기능이 개선되거나 비슷한 기능을 제공한다.

3.3 로그스태시(Logstash)

로그스태시는 실시간 로그 데이터 수집 도구로, 분산 환경에서 쏟아내는 로그 데이터 및 여러 에이전트에서 나오는 데이터들을 보낼 때 사용하는 도구이다. Collector의 역할일 하는 Shipper 와 Indexer 의 개념이 있으며 이 두가지를 통하여 사용자가 원하는 Collector Server로 전송 시켜 적절한 input을 가해 filter를 거친 후 output으로 나오게 되는 작업을 수행한다.

로그스태시의 이러한 과정을 정의하는 설정 configure 파일은 크게 세 가지로 나뉘는데 Input, filter 그리고 output 이다. 각 항목들은 선 정의된 다양한 기능들로 로그 데이터를 편리하게 분석할 수 있도록 되어 있다. input 설정의 경우 파일로 만들어지는 로그 데이터가 파일 소켓으로 데이터를 받을 수 있고 그 종류는 무려 20~30 가지이다. 사용하는 솔루션의 형태가 각각 다르다면 filter를 사용하여 로그 데이터 분석 및 재정의가 가능하게 만들 수 있다. 분석이 필요한 서버들에 로그스태시를 설치하여 만족하는 데이터를 모아 분석할 수 있게 된다. 각 에이전트에서 발생하는 로그들은 실시간으로 로그스태시에 의해 수집되어 저장된다. 로그스태시의 구성은 세부분으로 나뉘는데 먼저, Collector 부분은 로그를 수집하여 Indexer 로 보내게 된다. 로그스태시 Forwarder를 이용하여 로그를 tailing 하여 보내게 되면 서비스 중인 서버에 부담을 덜 줄수 있기 때문에 Forwarder를 이용하였다. 다음으로 Indexer 는 전달 받은 로그를 분석하여 의미있는 필드로 추출한 후에 Elasticsearch 에 저장을 하게 된다. 마지막으로 데이터 저장소인 Elasticsearch 에 데이터가 저장되고 Kibana를 이용하여 분석 및 시각화를 할수 있게 된다. 로그스태시는 기본적으로 로그스태시 와 로그스태시 forwarder 간 통신을 위해 인증서를 생성한다. Indexer 서버의 '/etc/pki/tls/openssl.cnf' 파일을 수정하여 모든 서버의 IP를 등록한 후 인증서를 생성한다. 인증서 키 생성 명령어와 서버 IP 등록은 다음과 같다.

○ 서버 IP 등록

[openssl.cnf 파일]

```
...
subjectAltName = IP:192.168.170.182, IP:192.168.170.183,
IP:192.168.170.184, IP:192.168.170.185
...
```

○ 키생성 명령어

```
$ openssl req -x 509 -batch -nodes -newkey rsa:2048
-keyout logstash-forwarder.key -out logstash-forwarder.crt
```

키생성을 한 후에 로그스테시, 로그스테시-Forwarder 에 설정 사항을 모두 적용한 후 설정파일에 파일경로를 기술하여야 한다. 이후 로그스테시-Forwarder 설정과 Logstash 에러로그 수집 설정은 다음과 같다.

○ Logstash-Forwarder 설정

```
{
  "network" : {
    # indexer ip list
    "servers":["192.168.170.182:2999", "192.168.170.183:2999"
              "192.168.170.184:2999", "192.168.170.185:2999"
    "ssl certificate":
    "/logstash-forwarder/cert/logstash-forwarder.crt",
    "ssl key": "/logstash-forwarder/cert/logstash-forwarder.key",
    "ssl ca": "/logstash-forwarder/cert/logstash-forwarder.crt",
    "timeout": 15
  }, "files": [
    { "paths": [ "/var/log/snort/portscan.log" ]}]}
```

○ Logstash.conf 파일 설정

```
input {
  lumberjack {
    port => 2999
    ssl_certificate => "/logstash/cert/logstash-forwarder.crt"
    ssl_key => "/logstash/cert/logstash-forwarder.key"
    type => "errorlog"
  }
}

filter {
  if [type] == "errorlog" {
    multiline {
      pattern => "(\^d+\serror)|(\^.+Exception: .+)|(\^s+at
.+)|(\^s+... \d+ more)|(\^s*Caused by:.)|(\^D.+)"
      what => "previous"
    }
    grok {
      patterns_dir => "./patterns"
      match          => [ "message",
"^%{TIMESTAMP_ISO8601:timestamp}          %{LOGLEVEL:loglevel}
\[%{WORD:classname}\] - %{GREEDYDATA:description}" ]
    }
    ruby {
      code => "
          event['@timestamp']
event['@timestamp'].localtime('+09:00')
          "
    }
  }
}
```

```
filter {
  if [type] == "errorlog" {
    if [loglevel] != "ERROR" and [loglevel] != "WARN" {
      drop { }
    }
  }
}
output {
  elasticsearch {
    protocol => "node"
    index => "logstash-error-%{+YYYY.MM.dd}"
  }
}
```

3.4 로그스태시, 엘라스틱서치 및 키바나 연동

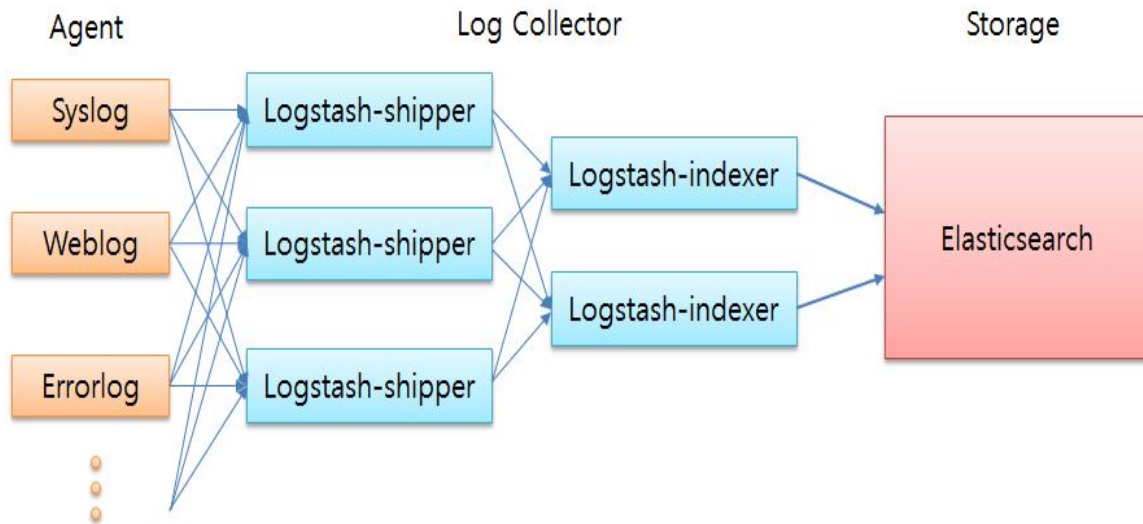
로그 수집기인 Logstash, 분산검색엔진인 Elasticsearch , 그리고 데이터 시각화 도구인 Kibana 세가지를 서로 연동시킴으로써 로그데이터를 검색 및 통계처리가 가능한 시스템을 구축 할 수 있다.

Elasticsearch 의 상태 모니터링을 위한 Plugin 인 Elasticsearch-head 를 설치하여 현재의 상태를 모니터링 할 수 있다. Logstash 를 이용하여 로그를 수집하고 Logstash 의 shipper 들을 통해 로그 데이터를 전달 받아 Elasticsearch 에 저장을 한 후에 Kibana 를 이용하여 시각화를 할 수 있다. 먼저, Logstash Config File 을 생성하여 다음과 같이 저장한다.

○ Logstash Config File

```
input {
  file {
    path => "/var/log/apache2/access.log"
    type => "apache"
  }
}
filter {
  grok {
    type => "apache"
    pattern => "%{COMBINEDAPACHELOG}"
  }
  date {
    type => "apache"
    match => [ "timestamp", "dd/MMM/yyyy:HH:mm:ss Z" ]
  }
}
output {
  stdout { debug => true debug_format => "json"}
  elasticsearch { host => "localhost" }
}
```

각기다른 디바이스 및 에이전트에서 Logstash 설정파일에 해당되는 로그가 발생하여 Logstash를 통해 로그가 수집되고, 실시간으로 Elasticsearch 에 저장시키는 방법으로 로그 데이터 저장 설계를 하였다. 그림 [3-4]는 로그 데이터 수집 구조 이다.



[그림 3-6] 로그 데이터 수집 및 저장 구조도

Logstash 설정 파일을 관리자 요구 사항에 알맞게 생성한 후 설정 파일을 실행시키는 Logstash 명령어를 입력하여 각각의 Agent 에서 로그 데이터를 보낼 수 있는 상태로 만든다.

○ Logstash 실행 명령어

```
bin/elasticsearch -d
bin/logstash agent -f logstash.conf
bin/logstash-forwarder -config logstash-forwarder.json
```

Logstash 는 체인처럼 설정이 서로 연결되어 있기 때문에 (input -filter-output) logstash 와 별도의 호스트를 지정하지 않고, 클러스터

로 포함시키려면 Elasticsearch 설정 파일을 logstash 설정 디렉토리와 함께 포함 시켜주어야 한다. 다음으로 Elasticsearch 를 실행시킨 후에 logstash 와 logstash-forwarder을 실행하는 연동구조 이다.

제 4 장 실험 및 성능 평가

4장에서는 제안한 시스템인 엘라스틱서치기반의 로그분석 시스템의 성능을 평가하기 위한 실험환경을 구축하여 성능평가를 하였다. 엘라스틱서치 기반의 보안로그 분석시스템은 약 40GB의 로그데이터를 Logstash로 수집하여 분산검색엔진인 엘라스틱서치에 저장한 후에 분석통계 및 시각화 도구인 Kibana 를 이용해 로그 검색 및 분석 처리시간을 하둡 기반의 NoSQL을 이용한 시스템과 성능을 비교하는 실험을 진행 하였다. 기존의 하둡 기반으로 로그 데이터를 수집하여 이를 검색하고 분석하여 하둡 시스템에서 HBase에 저장하고 R 과 Hive와의 연동을 통하여 데이터 분석 및 모니터링을 제공을 비교하는 실험하였다. 두 가지 시스템의 성능비교 테스트 환경을 구축하고 성능 평가를 진행하였다.

4.1 실험 환경 구성

실험을 위한 환경 구성을 위해 엘라스틱서치 클러스터링 된 4대의 Node와 로그 생성을 위한 Snort 기반의 1대의 Masternode로 구성하였다. 총 4대의 노드중 1대는 Masternode로 분석 및 통계 서버의 역할을 하며 나머지 3대는 Datanode 이다. 시스템 구성은 다음과 같이 구성하였다.

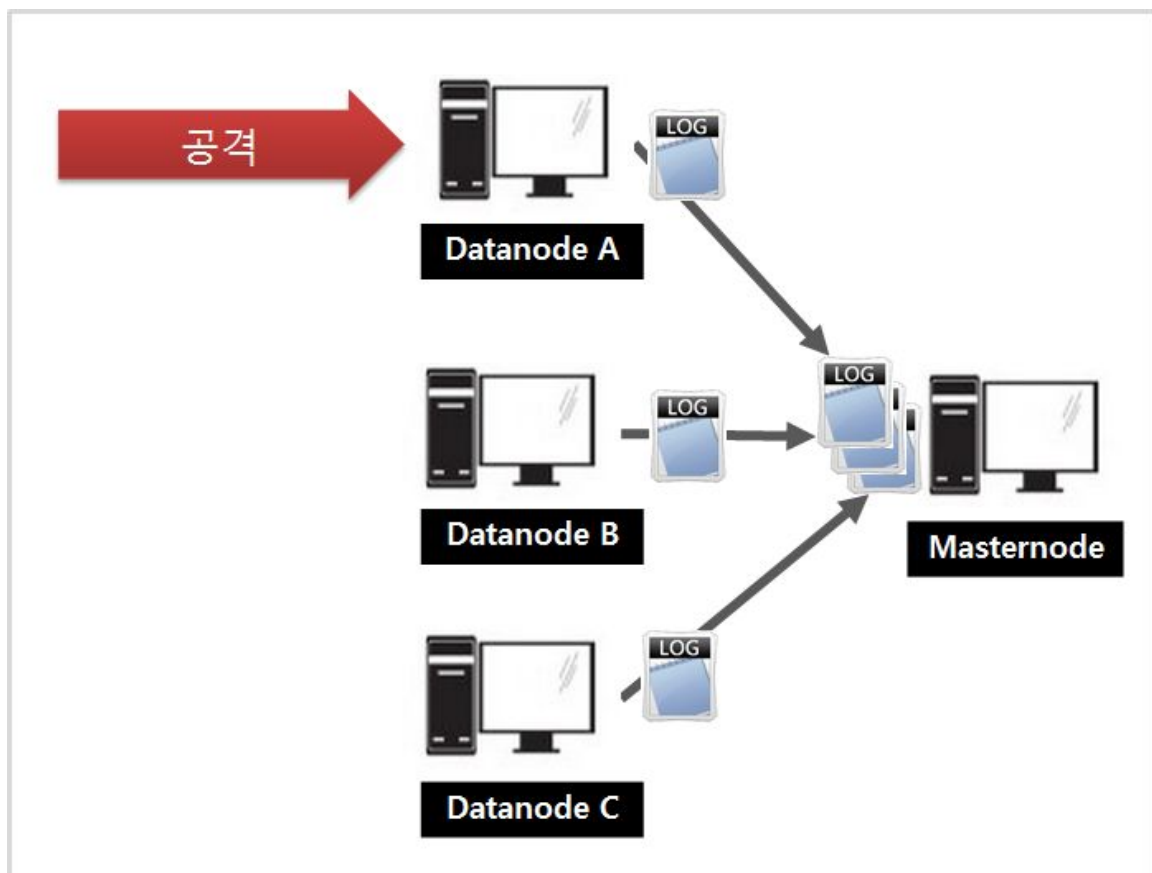
- Masternode Elasticsearch Cluster Master 구성
 - Elasticsearch, Logstash, Kibana
- Datanode Elasticsearch Cluster 구성
 - Elasticsearch, Logstash, Snort
- Node System 사양

CPU : QuadCore 2.80GHz, RAM : 16GB

HDD : 1TB, Ethernet : 1Gbps

4.2 실험 진행 방법

실험을 시작하기에 앞서 보안로그를 발생시키기 위해 해당 Datanode 서버에 Apache 웹서버를 설치하였다. 이후 HTTP 요청을 발생시켜 서버의 CPU와 Memory를 가득 채워 간단히 서버를 마비시킬 수 있는 매우 심각한 서버자원 고갈 공격을 하였다. 해당 공격은 서버 자원을 낭비시키기 때문에 서버에 영속성에 치명적인 공격이다. [그림 4-1]은 실험 환경 구성도이다.

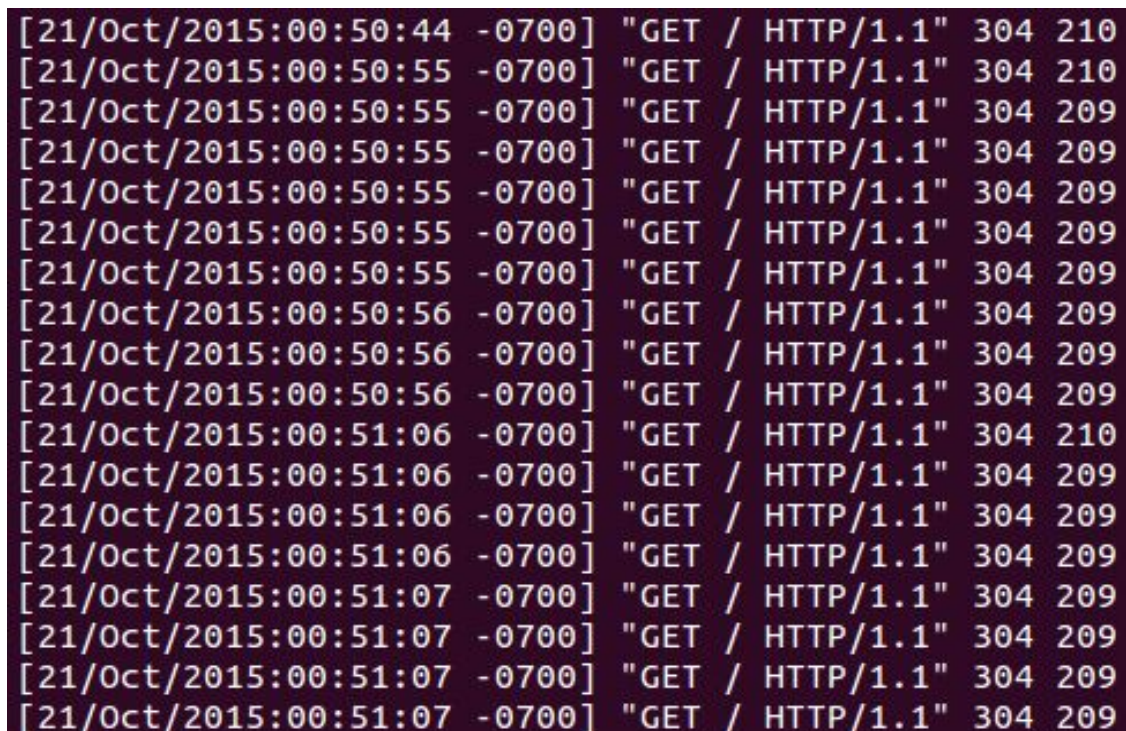


[그림 4-1] 실험 환경 구성도

Datanode 에서 현재 서버에서 이루어지고 있는 HTTP 요청에 의한 Access.log 로 하여금 저장이 된다. 이후 Datanode 에서 검출된 기록을 Logstash 가 해당되는 로그를 엘라스틱서치 저장소에 저장을 하게 된다. 그리고나서 저장된 로그 데이터를 바탕으로 Kibana 를 이용하여 분석 및 통계화 그래프화 하여 확인하는 방식으로 진행하였다.

4.2.1 로그 수집 및 저장

Datanode 에 설치한 Logstash를 통하여 해당 공격이 발생할 때의 Access 로그를 검출하여 엘라스틱서치에 저장하게 된다. 이때 저장된 로그 데이터는 209로 응답하여 마치 GET 플로딩 공격을 받을때 처럼 동일한 URL에 대한 GET 요청이 지속적으로 발생하는 것을 확인할 수 있다. 다음 [그림 4-2]는 해당 로그 데이터가 저장되어 GET 플로딩 공격을 받을 때 처럼 동일한 URL에 대한 GET 요청이 지속적으로 발생한 화면이다.



```
[21/Oct/2015:00:50:44 -0700] "GET / HTTP/1.1" 304 210
[21/Oct/2015:00:50:55 -0700] "GET / HTTP/1.1" 304 210
[21/Oct/2015:00:50:55 -0700] "GET / HTTP/1.1" 304 209
[21/Oct/2015:00:50:55 -0700] "GET / HTTP/1.1" 304 209
[21/Oct/2015:00:50:55 -0700] "GET / HTTP/1.1" 304 209
[21/Oct/2015:00:50:55 -0700] "GET / HTTP/1.1" 304 209
[21/Oct/2015:00:50:55 -0700] "GET / HTTP/1.1" 304 209
[21/Oct/2015:00:50:56 -0700] "GET / HTTP/1.1" 304 209
[21/Oct/2015:00:50:56 -0700] "GET / HTTP/1.1" 304 209
[21/Oct/2015:00:50:56 -0700] "GET / HTTP/1.1" 304 209
[21/Oct/2015:00:51:06 -0700] "GET / HTTP/1.1" 304 210
[21/Oct/2015:00:51:06 -0700] "GET / HTTP/1.1" 304 209
[21/Oct/2015:00:51:06 -0700] "GET / HTTP/1.1" 304 209
[21/Oct/2015:00:51:06 -0700] "GET / HTTP/1.1" 304 209
[21/Oct/2015:00:51:07 -0700] "GET / HTTP/1.1" 304 209
[21/Oct/2015:00:51:07 -0700] "GET / HTTP/1.1" 304 209
[21/Oct/2015:00:51:07 -0700] "GET / HTTP/1.1" 304 209
[21/Oct/2015:00:51:07 -0700] "GET / HTTP/1.1" 304 209
```

[그림 4-2] HTTP 요청에 의한 지속적인 GET 요청 발생 화면

로그 데이터를 지속적으로 발생시킨 후 생성된 로그 데이터 파일의 용량이 약 1GB가 넘어가게 되면 access.log 에서 access.log.1 로 이름이 바뀌어 저장된다. 로그 데이터 총 40GB 중에 10GB 로그데이터와 20GB 바이트 그리고 30GB 로 나누어 실험을 진행 하였다. 이러한 용량이 다른 로그데이터를 바탕으로 엘라스틱서치 기반의 검색 및 분석 시스템에 적용하여 검색 및 분석시간을 테스트 하였다.

4.2.2 보안로그 분석

보안로그 분석에서 Kibana 를 이용하여 엘라스틱서치 저장소에 있는 데이터를 인덱싱 및 검색하여 로그 조회, 및 명령어 Query()를 이용하여 해당 날짜별 로그 데이터 조회, 특정 동일한 URL에 대한 악의적 로그 조회등 다양한 대용량 로그데이터를 분석할 수 있다. [그림 4-3] 은 Kibana 를 이용해 지속적인 GET 플로딩 공격 로그가 발생한 시간에 따른 로그 건수를 카운트한 그래프이다. 해당 공격이 08시에 시작되어 18시 까지 로그가 남겨진 것을 확인할 수 있다.



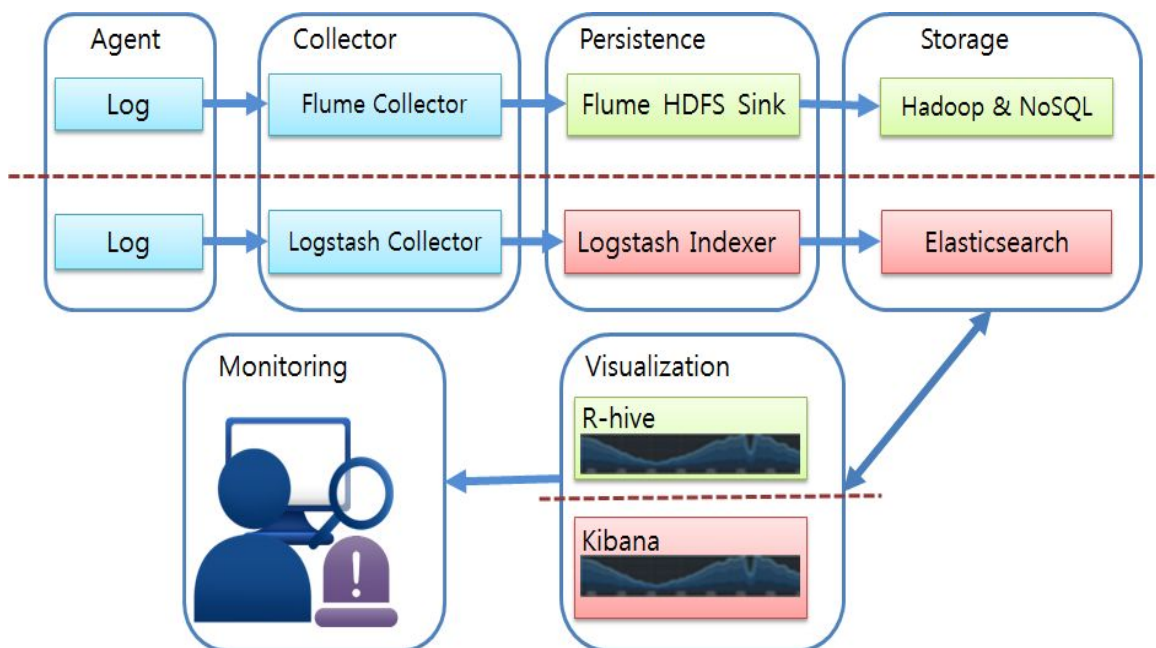
[그림 4-3] Kibana 를 이용한 로그 조회 시각화

4.3 성능 테스트

4.3.1 분석 성능 테스트

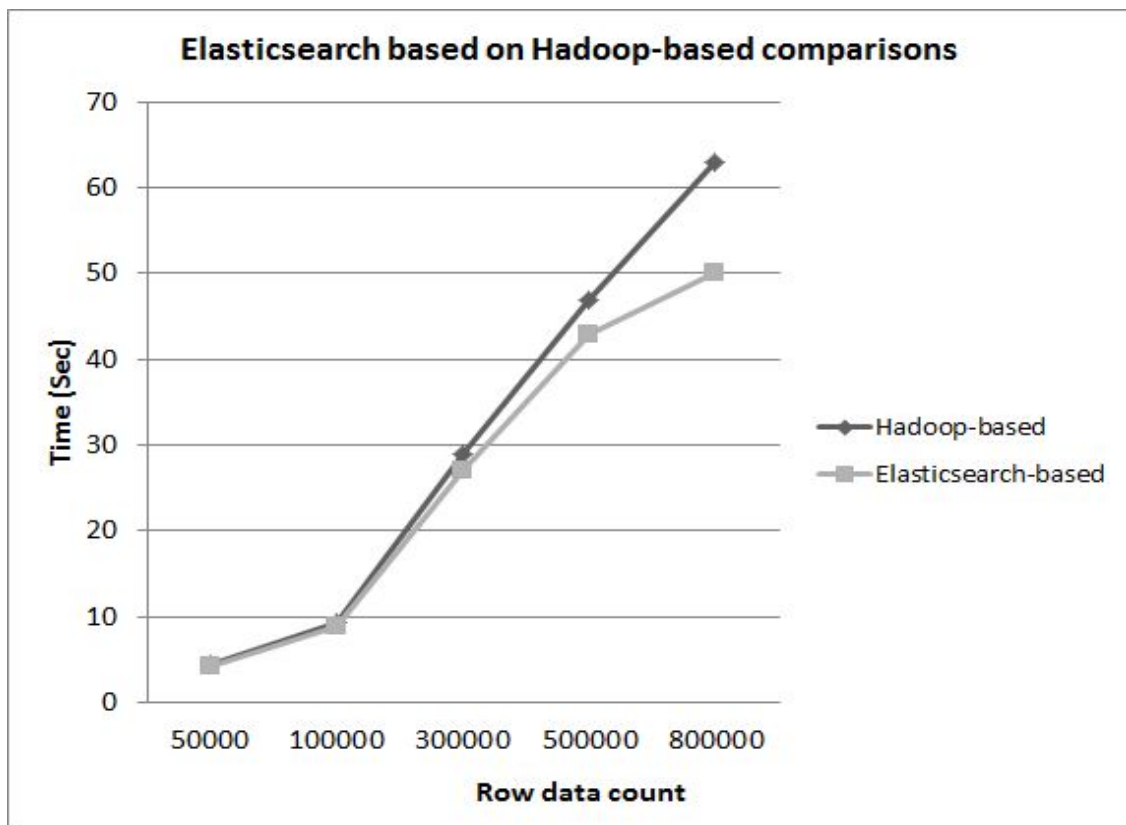
엘라스틱서치를 이용하여 로그분석 성능 평가를 하기 위해 테스트 환경을 구축하여 성능을 평가하였다. 제안한 시스템의 로그 수집 단계에서 데이터 저장 및 속도 성능을 평가하기 위하여 하둡 기반의 보안 로그 데이터 분석 시스템과 비교하여 실험 환경을 구축 하였다. 또한 로그 데이터의 용량을 증가시켜 각 테이블에 데이터를 저장 하였다.

로그분석 방법에는 엘라스틱서치를 이용한 로그 분석시스템에서는 Kibana를 이용하였고, 비교 환경인 하둡기반 로그 분석 시스템에서는 HBase에 저장된 로그에서 데이터를 조회하기 위해 쿼리문의 필요성 때문에 Hive로 가상 테이블을 생성하여 SQL이 가능하도록 하였다. 저장된 같은 로그 데이터를 통하여 결과값을 얻기 위한 쿼리문을 보내어 처리속도를 비교하는 실험을 하였다. 실험환경에서 사용한 Row 데이터는 데이터 베이스에서 테이블의 단일 구조 데이터 항목을 말하며, GET플로딩 공격 데이터를 사용하였다.



[그림 4-4] 엘라스틱서치를 이용한 분석성능 비교 실험 구성도

[그림 4-4] 는 비교 실험 환경인 하둡 기반의 보안로그 분석 시스템 과 분산검색엔진인 엘라스틱서치를 이용한 보안로그 분석 시스템의 구조도 이다. 먼저 비교환경인 하둡기반 환경의 에이전트에서 쌓인 로그를 콜렉터인 플룸 콜렉터가 보안 로그를 모아 HDFS 플룸 싱크를 통하여 분산파일 환경인 하둡에 저장을 한 후에 NoSQL인 HBase 에 분산 저장이 된다. Hbase 에 저장한 로그 데이터를 토대로 R-Hive를 통해 컬럼으로 분석을 하게 된다. 분석을 할 때에는 HBase에 저장되어진 로그 데이터를 Hive 를 통해 가상 테이블을 생성한 후에 SQL이 가능하도록 환경을 구성한다. 이후 결과 값을 얻기 위해 쿼리문을 사용하여 처리 속도를 산출 하였다. 제안하는 시스템은 에이전트단에 쌓인 로그 데이터를 Logstash 콜렉터가 수집하여 인덱서에게 전달하고 이를 분산검색엔진인 엘라스틱서치에 저장이 된다. 이후 Kibana와 엘라스틱서치 Plugin을 통하여 로그 분석 및 통계가 가능하게 된다. Kibana 에서 인덱스를 생성한 후에 앞에서 실행한 동일한 쿼리문을 사용하여 처리 속도를 비교하였다.



[그림 4-5] 엘라스틱서치기반 시스템과 하둡기반 시스템 성능 비교

[그림 4-5]는 본 논문에서 제안하는 엘라스틱서치기반 시스템과 하둡기반 시스템의 Row수에 따른 요청한 쿼리의 처리 속도를 비교한 그래프이다. 실험 결과는 쿼리처리 5만건 에서 10만건 사이는 비교적 차이가 심하지 않았지만, 50만건 부터 데이터 수가 증가할 수록 급격한 차이를 확인하였다. 이를 토대로 데이터가 더욱 많아지면 많아질수록 검색엔진기반 시스템과 하둡기반 시스템의 성능 차이는 더 커질 것이다.

[표 4-1]은 Row 수를 증가시키면서 엘라스틱서치기반 시스템과 하둡기반 시스템의 처리 속도를 비교한 표 이다. 5~10만건의 처리 속도는 검색엔진기반 시스템이 하둡 기반 시스템보다 0.3초 정도 속도가 더 빨랐지만 데이터의 수가 급격히 커질수록 처리 소요 시간 차이가 커지는 것을 확인할 수 있다. 50만건에서 80만건 부터 검색엔진을 기반으로 한 시스템이 하둡 기반의 시스템 보다 5만건에서 0.2초 10만건에서 0.3초 그리고 마지막으로 80만건에서 13초 차이를 보였다. 데이터의 양이 많아질수록 검색엔진기반 시스템이 하둡기반 시스템보다 빠른 처리 속도를 내는 것을 확인하였다. 처리 속도가 빠르게 될수록 분석 및 검색의 속도도 빨라지게 되기 때문에 로그 데이터를 분석함에 있어서 검색엔진을 기반으로 한 시스템이 처리 시간을 줄이는 효과가 있는 것을 확인하였다.

[표 4-1] 엘라스틱서치기반 시스템과 하둡 기반 시스템 비교

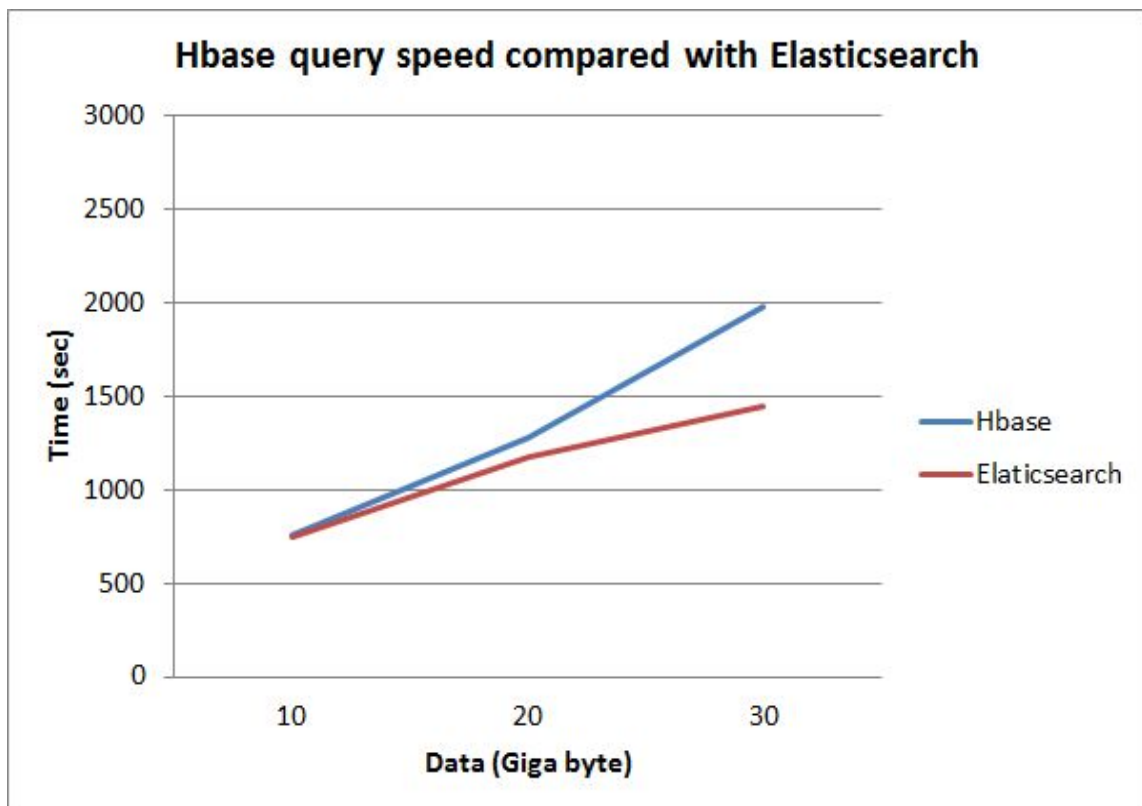
구 분	데이터 수				
	5만	10만	30만	50만	80만
엘라스틱서치 기반	4.3	9	27	43	50
하둡기반	4.5	9.3	29	47	63

(단위 시간 : 초)

4.3.2 용량에 따른 로그데이터 처리 속도 비교

제안하는 엘라스틱서치기반 보안 로그 데이터 분석 시스템과 하둡기반 보안 로그 분석 시스템의 처리 속도를 비교 하였다. 기존의 하둡기반 방식의 NoSQL 기반 방식인 HBase와 Elasticsearch의 성능을 비교 하였다. 처리속도 비교를위해 성능 테스트에 사용된 로그 데이터는 10GB의 약 1,000만건의 로그 데이터 와 20GB 의 약 2,000만건 데이터 마지막으로 30GB 의 약 3,000만건 데이터로 구성하였다.

[그림 4-6]은 처리 속도에 관한 테스트 결과이다. 해당 쿼리 처리 속도는 로그 데이터를 처리하는데 걸리는 시간에 대한 성능 비교 테스트 이다.



[그림 4-6] HBase와 Elasticsearch 성능 비교

실험 결과로 보아 로그 데이터의 증가에 따라 HBase 와 Elasticsearch의 차이는 더욱 커질 것으로 보인다. 처리속도가 10GB 로그 데이터에서 처

리속도의 차이는 10초 가량 차이가 나지만 로그 데이터량 20GB 에서의 처리속도는 110초 30GB 에서는 530초 가량 차이가 나는 것을 확인 할 수 있다. 결과값을 볼때 데이터량이 증가하면 증가 할수록 Elasticsearch의 처리속도가 HBase 보다 더욱 빨라지는 것을 유추할 수 있다. 아래 [표 4-2] 는 데이터량의 증가에 따른 Elasticsearch와 HBase의 처리 속도 비교 표 이다.

[표 4-2] Elasticsearch 와 HBase 데이터량에 따른 속도비교

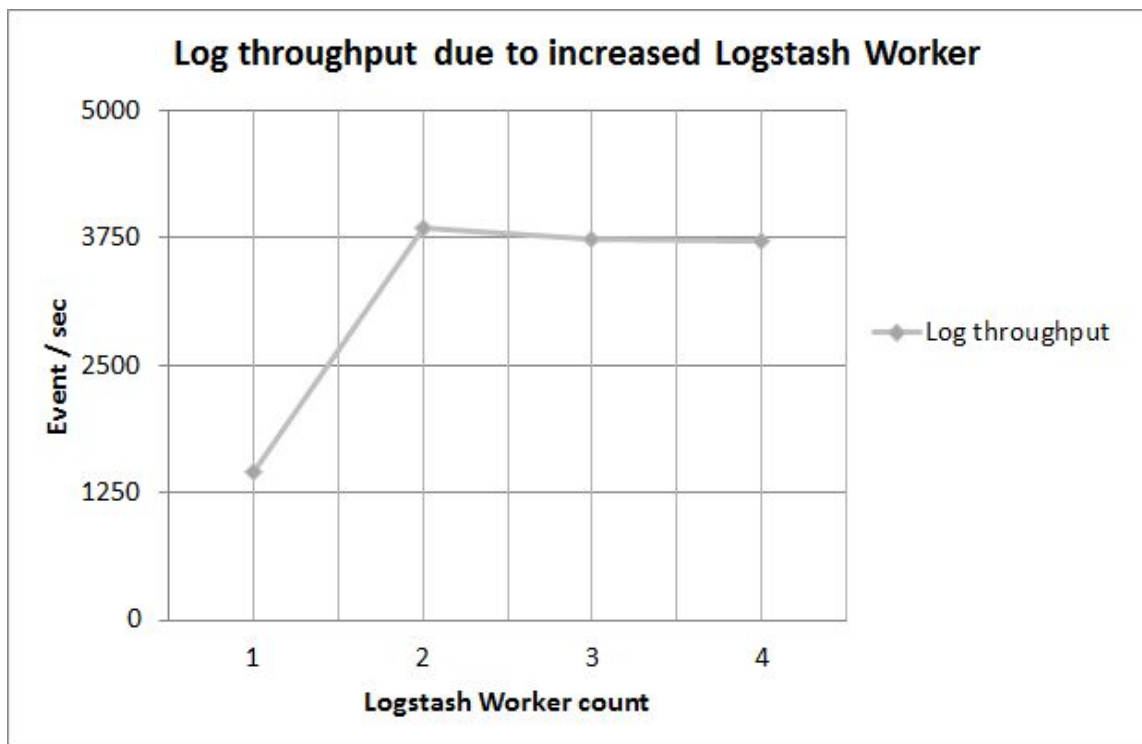
구 분	로그 데이터량		
	10Gbyte	20Gbyte	30Gbyte
HBase	760	1,280	1,980
Elasticsearch	750	1,170	1,450

(단위 시간 : 초)

4.3.3 로그스태시 Worker 수에 따른 로그 처리량 테스트

제안하는 분산검색엔진을 이용한 시스템의 성능 검증을 위해 테스트는 Logstash Worker 수를 1대부터 4대까지 늘려서 로그 처리량을 테스트 하였다. 시스템 성능 테스트를 위해 로그 데이터는 20GByte로 구성하여 실험하였다.

[그림 4-7] 은 Logstash Worker 수에 따른 초당 로그 처리량에 관한 그래프이다.



[그림 4-7] Logstash Worker 수 증가에 따른 로그 처리량

테스트 결과를 그래프로 보았을 때 Logstash Worker 수를 1대에서 최대 4대까지 늘려 실험을 진행하였다. Logstash Worker의 수가 2대 부터 4대까지 증가 하였지만 초당 처리량 변화가 적었다. 그리고 Elasticsearch 클러스터를 구성 하였을 때 초당 처리량이 증가하였

다.

[표 4-3] Logstash Worker 수에 따른 처리량 비교

구 분	Logstash Worker 수			
	1대	2대	3대	4대
초당 로그 처리량	1,450	3,840	3,740	3,720

(단위 : Event/sec)

[표 4-3]에서 보는바와 같이 Logstash Worker 수가 1대부터 2대까지는 초당 로그 처리량이 급격하게 증가하였지만, 2대부터 4대까지의 로그 처리량 차이는 미비하게 나타났다. 결과로 유추하여 볼때 최적의 Logstash Worker 수는 2대가 가장 효율적인 것으로 나타났다.

제 5 장 결론 및 향후 과제

IT 시스템에서 모든 행동과 보고의 발자취인 로그데이터는 네트워크가 존재하는 다양한 곳에서 계속적으로 발생되고 있다. 이 로그데이터들을 수집하고 해당 시스템에서 분석을 통해 미래에 유사시 활용 가능한 유용한 정보를 얻을 수 있는 중요한 정보이다. 이 자료를 활용하여 예상치 못한 보안 위협을 방어하고 해당 시스템의 연속성에 기여할 수 있는 보안로그분석 시스템이 필요하다. 이러한 시스템은 실시간적 대용량 데이터 분석을 필요로하기 때문에 막대한 량의 로그 데이터를 수집하여 저장하고 분석하는데 걸리는 시간을 최대한 줄이기 위한 노력이 현재까지도 이루어지고 있다. 기존의 하둡 기반의 로그분석시스템에 관한 연구에서는 로그 데이터의 수집 및 저장방식 그리고 분석을 위한 의미 있는 데이터를 추출하는 과정에서 정규식이라는 특수한 방법이 필요하여 이를 이용해 시간 소요가 늘어난다는 점이 있었다. 본 논문에서 제안한 방식인 분산 검색엔진을 이용한 로그분석 시스템 구축하여 사용자 편의에 맞는 인덱싱된 로그 데이터의 수집 및 저장을 통하여 필요한 시간을 최대한 줄일 수 있었다. 제안하는 시스템에서 로그 데이터를 수용하고 처리하기 위해 오픈소스분산검색엔진인 Elasticsearch 를 이용하여 로그 데이터의 검색과 분석을 시스템의 보안 정책에 따라 판단할 수 있는 기존 시스템보다 처리 속도가 대상 로그데이터가 30Gbyte 였을 시에 500초 가량 차이를 보였으며, 응답시간과 데이터 분석능력에서 Kibana를 이용하여 기존의 하둡 기반 시스템 보다 효율적임을 확인하였다.

향후에는 빅데이터를 단순히 수집 축적하는 것에 목적을 둔 것이 아니라 숨겨진 패턴에 초점을 두어 현재의 로그 데이터 패턴과 과거의 로그 데이터 패턴을 분석하여 미래의 데이터를 미리 예측하여 사고에 대비할 수 있는 자가 진단이 가능한 지능형 보안시스템의 연구가 필요하다.

참고문헌

- [1] 정경원, “보안 분야의 빅데이터, 로그 분석 시간 부족”, 정보통신산업진흥원, 정보통신 기술, 시장, 정책 주간기술동향 2012. 05.
- [2] 김태훈, "Nosql기반 보안로그 분석시스템", 정보보호학회논문지 제23권 제4호, 2013.8, 667-677 (11 pages).
- [3] 김진홍, "빅데이터 기반의 로그데이터 분석을 통한 시스템 장애 감지 기법 연구" 2014. 08 건국대학교 정보통신대학원 석사학위논문.
- [4] 안광민, 이종윤, 양동민, 이봉환, “하둡 기반의 효율적인 보안로그 분석시스템 설계 및 구현” 한국정보통신학회논문지 제19권 제8호, 2015.8, 1797-1804 (8 pages).
- [5] RDBMS, <https://ko.wikipedia.org/wiki/RDBMS>
- [6] NoSQL, <https://ko.wikipedia.org/wiki/NoSQL>
- [7] Elasticsearch, <http://www.jopenbusiness.com/mediawiki/index.php?title=ElasticSearch>
- [8] 조성룡, “Big Data, Environmental Changes and Distributed Database System”, 정보과학회지 제30권 제5호, 2012.5, 21-28(8 page)
- [9] 이동환, 박정찬, 유찬곤, 윤호상, “빅데이터 기반의 실시간 네트워크 트래픽 분석 플랫폼 설계”, 정보보호학회논문지 제23권 제4호, 2013.8, 721-728 (8 pages)
- [10] Flume, <https://flume.apache.org>
- [11] Apache Hadoop, <http://wiki.apache.org/hadoop>
- [12] 김남용, 김석훈, 손우용, 송정길, “로그 분석을 통한 침입대응 시스템 설계 및 구현” 한국 인터넷 정보 학회 2004. 11, 123-126 (4 page)
- [13] 최보민, 공종환, 홍성삼, 한명목, “NoSQL기반의 MapReduce를 이용한 방화벽 로그 분석 기법”, 한국정보보호학회논문지 제23권 제4

호, pp. 667-677, 2013

- [14] 김용현, 허의남, “효율적인 빅데이터 분석을 위한 로그 데이터기반의 로그분석 지원 시스템” 한국정보과학회 학술발표논문집, 2014.12, 936-938(4 page)
- [15] 김명진, 한승호, 최운, 이한구, “클라우드 환경에서 MongoDB 기반의 비정형 로그 처리 시스템 설계 및 구현”, Journal of Internet Computing and Services(JICS) 2013. Dec: 14(6) : 71-84 ISSN 1598-0170
- [16] Bomin Choi, Jong-Hwan Kong, and Myung-Mook Ha, “The Model of Network Packet Analysis based on Big Dat”, Journal of Korean Institute of Intelligent Systems, Vol. 23, No. 5, October 2013, pp. 392-39
- [17] 최대수, 문길종, 김용민, 노봉남, “MapReduce를 이용한 대용량 보안 로그 분석”, 한국정보기술학회논문지 제9권 제8호, 2011.8, 125-132 (8 pages)
- [18] 강만모, 김상락, 박상무, “빅 데이터의 분석과 활용”, COMMUNICATIONS OF THE KOREA INFORMATION SCIENCE SOCIETY 30(6), 2012.6, 25-32 (8 pages)

A Security Log Analysis System using Log Stacy based on Apache ElasticSearch

Sang Yong Lee

Dept. of Information and Communications Engineering

Graduate School, Daejeon University

(Advised by Prof. Bong-Hwan Lee, Ph.D.)

Abstract

Recently cyber attacks can cause tremendous damage on various information systems. Log data analysis could be able to resolve this problem. Security log analysis system allows to cope with security risk properly by collecting, storing, and analyzing log data from information systems.

In this thesis, a security log analysis system is designed and developed in order to analyze security log data using the Log Stacy implemented in the ElasticSearch, which is a distributed search engine.

The proposed system makes use of ElasticSearch cluster

method, and processes various log data using Log Stacy, a log collector. The experimental results show that the proposed system reduces log data analysis time by 500 seconds compared to the Hadoop-based system in case that log data size is 30 Gbytes.