빅데이터 클러스터 기반 검색 플랫폼의 실시간 인덱싱 성능 최적화

[1]금나연, [2*]박동철

Real-Time Indexing Performance Optimization of Search Platform Based on Big Data Cluster

[1]Nayeon Keum, [2*]Dongchul Park

요 약

정보기술의 발달로 모든 데이터는 데이터베이스화 되어 빅데이터 시대를 맞이하였으며 방대한 양의 데이터에 대한 접근성과 활용 가능성을 높이고자 빅데이터 검색 플랫폼의 필요성이 증가되었다. 검색 플랫폼은 기본적으로 효율적인 검색을 위해 인덱스를 빠르게 생성하고 저장하는 인덱싱 (indexing) 과정과 생성된 인덱스를 활용하여 필요한 정보를 찾는 검색 (searching) 과정으로 구성된다. 빅데이터 시대를 지나 초빅데이터 시대를 맞이하여 데이터의 용량이 거대해짐에 따라 데이터 인덱싱 성능이 검색 플랫폼의 매우 중요한 성능문제로 대두되고 있다. 많은 기업들이 효율적인 빅데이터 검색을 위해 검색 플랫폼들을 도입하고 있으나, 검색 효율성 및 검색 정확도 관련 연구에 비해 검색 성능의 핵심이 되는 인덱싱(indexing)의 성능을 최적화하는 연구는 상대적으로 미흡한 실정이다. 또한 인덱싱(indexing) 기본 단위인 샤드(Shard) 수와 크기를 최적화하는 연구에 비해 검색 플랫폼을 클러스터 기반으로 운영하기 위한 다양한 성능 비교 관련 연구는 미흡하다. 이에 본 연구에서는 대표적인 엔터프라이즈 빅데이터 검색 플랫폼인 Elasticsearch 클러스터를 구성하여 확장성 높은 검색 환경을 위해 최적의 인덱싱 성능을 낼 수 있는 구성을 제안한다. 본 논문은 클러스터와 검색 플랫폼의 다양한 구성 변경을 통해 최고의 인덱싱 성능을 낼 수 있는 구성을 도출하여 최적 구성에서 기본 구성보다 평균 3.13배 높은 인덱싱 성능의 향상을 확인하였다

Abstract

With the development of information technology, most of the information has been converted into digital information, leading to the Big Data era. The demand for search platform has increased to enhance accessibility and usability of information in the databases. Big data search software platforms consist of two main components: (1) an indexing component to generate and store data indices for a fast and efficient data search and (2) a searching component to look up the given data fast. As an amount of data has explosively increased, data indexing performance has become a key performance bottleneck of big data search platforms. Though many companies adopted big data search platforms, relatively little research has been made to improve indexing performance. This research study employs Elasticsearch platform, one of the most famous enterprise big data search platforms, and builds physical clusters of 3 nodes to investigate optimal indexing performance configurations. Our comprehensive experiments and studies demonstrate that the proposed optimal Elasticsearch configuration achieves high indexing performance by an average of 3.13 times.

Keywords: *Elasticsearch, Search Engine, Big Data, Indexing, Distributed Systems, Clusters*

[1] *숙명여자대학교 IT 공학전공(ryann3@sookmyung.ac.kr)*
[2*] *교신저자 중앙대학교 산업보안학과 교수(dongchul@cau.ac.kr)*

Received: Oct 04, 2023, Revised: Nov. 25, 2023, Accepted: Nov. 26, 2023

I. 서론

빅데이터 시대를 맞이하여 데이터의 양이 폭발적으로 증가함에 따라 데이터베이스에 저장되어 있는 정보를 빠르고 정확하게 검색하여 사용자들에게 제공하는 것이 주요한 문제로 자리잡았다. 이에 정보에 대한 접근성과 활용 가능성을 높이고자 검색 플랫폼의 필요성이 증가되었고, 다양한 검색 엔진이 오픈소스 소프트웨어로 개발되었다[1]. 그 중 Elasticsearch 는 근실시간(Near Real-Time, NRT)으로 페타바이트 규모의 데이터를 검색할 수 있으며 수평 확장이 가능하고 가용성이 높은 NoSQL (Not Only Structured Query Language) 기반 분산 시스템으로써 기존의 관계형 데이터베이스가 지원할 수 없었던 확장성, 빅데이터 검색 및 성능 문제를 명시적으로 해결할 수 있게 되었다 [2][3]. 이렇게 Elasticsearch 는 대표적인 Java 기반 오픈소스 검색 엔진으로 자리잡았으며, 이러한 특징들로 인해 많은 기업들이 효율적인 검색을 위해 Elasticsearch 를 사용하고 있다. 이에 검색 효율성 및 검색 정확도 향상, 사용자 검색 기록 및 활용도 분석을 기반으로 한 Elasticsearch 기반 개인화 검색 엔진 관련 연구 등이 활발하게 진행되고 있다 [6].

그러나 대용량 데이터를 다루는 만큼 Elasticsearch 에서 데이터 검색 성능 뿐만 아니라 데이터를 인덱스로 변환하여 저장하는 인덱싱(Indexing)의 성능을 최적화하는 것이 중요한 문제로 대두되고 있지만, 아직까지는 사용자와 직접적으로 연관된 검색 알고리즘에 비해 검색 플랫폼에서의 인덱싱과 관련된 연구는 상대적으로 미흡한 실정이다 [4]. 또한 Elasticsearch 의 전체 텍스트(full-text) 검색 기술과 데이터 특성을 기반으로 인덱스 샤드(Shard) 수와 크기를 최적화하는 연구가 존재하지만, 검색 플랫폼으로써 Elasticsearch 를 클러스터 기반으로 운영하기 위한 다양한 성능 비교 관련 연구는 미흡하다 [7][8].

검색 플랫폼에서 효과적인 인덱싱 생성은 검색 성능 향상과 직결되며, 빅데이터 시대를 지나 초빅데이터 시대에 접어들면서 대용량 데이터의 인덱스 변환과 저장 작업은 더 많은 컴퓨팅 자원과 저장시간을 필요로 함에 따라 검색 플랫폼의 핵심적인 문제로 부각되고 있다. 가령, 1 대의 고성능 서버 (2 Socket CPUs (24 cores), 256GB RAM, 24TB HDD (RAID 5))를 이용하여 1TB 실제 기업 로그 데이터를 Elasticsearch 플랫폼 기반의 인덱싱 작업 시, 약 30 시간의 인덱스 생성 및 저장 소요시간과 평균 114GB 메모리가 소요됨을 산업체와 본 연구팀의 공동 연구를 통해 확인하였다.

이에 본 논문에서는 Elasticsearch 의 다양한 구성 변경을 통해 최고의 인덱싱 성능을 낼 수 있는 구성을 도출하고자 한다. 본 연구는 국내 중견 보안기업과 산학공동 연구를 통해 진행되었으며 실험에 사용된 모든 데이터는 개인정보보호를 위해 기업측에서 완벽하게 익명처리가 된 실제 보안로그 데이터를 제공받아 수행되었다. 연구 실험결과 Elasticsearch 기본 구성보다 본 연구 결과로 도출된 최적 구성에서 평균 3.13 배 인덱싱 성능 향상을 달성하였다. 또한 본 연구에서 사용한 동일한 클러스터 구성이 아닌, 향후 확장성이 고려된 다른 컴퓨팅 환경에서도 본 논문에서 제공한 연구결과를 적용할 수 있도록 다양한 Elasticsearch 환경 구성별 성능 실험 결과 분석자료를 제공하였다. 이를 통해 미래에 달라질 수 있는 여러 환경 (가령, 데이터용량 증대, 클러스터 노드 개수 증가, 클러스터 노드 컴퓨팅 리소스 증대, 등) 에서도 본 논문에서 제공한 연구 결과를 통해 최적의 인덱싱 구성과 성능을 예측할 수 있는 참조 역할을 할 수 있을 것으로 기대한다.

본 연구의 핵심 요약 결과는 다음과 같다.

- Elasticsearch 에서 인덱싱 성능은 물리적 Elasticsearch data node 개수에 큰 영향을 받는다. 따라서 클러스터의 물리적 노드 개수가 증가할수록 인덱싱 성능이 비례하여 증가한다. 그러나 물리적 data node 의 개수가 증가할수록 master node 를 담당하는 노드의 컴퓨팅 성능이 bottleneck 이 되므로, 미래 확장성을 고려해야 할 경우 고사양의 Elasticsearch master node 서버를 도입하는 것이 유리하다.
- 클러스터내 여러 물리적 노드에서 master node 의 역할을 data node 와 최대한 분리 구성하는 것 (즉, 같은 서버내에 master node 와 data node 를 함께 구성하지 않는 것)이 인덱싱 성능 향상에 큰 영향을 준다.

- Master node 도 data node 들에게 데이터를 분산하여 전달 (ingesting)할 수 있으나 별도로 ingest node 를 구성하여 data ingesting 기능만을 전담하도록 하는 것이 성능 향상에 유리하다.
- 논리적 Elasticsearch data node 개수에는 큰 영향을 받지 않는다. 오히려 개수가 증가할 수록 성능이 감소하는 경향이 있다.
- 일정 수준까지 Logstash 노드 개수가 많아질수록 병렬 분산처리가 가능하므로 인덱싱 성능이 향상된다.
- Disk I/O 성능이 인덱싱 성능에 매우 큰 영향을 준다. 따라서 SSD 를 사용하거나, HDD 의 경우 RAID 를 이용하여 I/O 성능을 높이면 인덱싱 성능이 매우 높아진다[5].

본 논문의 2 장에서는 Elasticsearch 와 관련된 기존 연구들을 소개한다. 3 장에서는 본 연구에서 사용한 Elasticsearch 와 Elastic Stack 에 대해 설명하고, 장에서는 사용한 데이터와 실험 장비의 사양을 소개한다. 4 장에서는 다양한 성능 실험 결과에 대하여 설명한다. 먼저 Elasticsearch node 개수와 shard 개수에 따른 성능 변화를 비교하고, Elasticsearch 에서 master node 의 역할을 분리하였을 때의 결과를 보여준다. 마지막으로 Logstash node 의 개수에 따른 성능 변화를 분석하고 가장 높은 성능을 보였던 구성의 시스템 환경과 실험 구성을 설명한다. 6 장에서는 결론을 맺는다.

II. 관련 연구

E. Lee [6]는 차세대 대용량 영구 메모리인 인텔 옵테인 영구 메모리의 효용성을 검증하기 위해 옵테인 메모리 기반 시스템에 Elasticsearch 를 설치하여 다양한 성능 평가 및 분석을 진행하였다. 그러나 클러스터가 아닌 단일 Elasticsearch 노드 환경에서 성능 평가를 수행하였으며 클러스터 환경에서 인덱싱 성능 최적화를 위한 본 연구와는 차이가 있다.

J. He [7]는 검색 효율성 및 검색 정확도 향상, 사용자 검색 기록 및 활용도 분석을 기반으로 Elasticsearch 기반 개인화 검색 엔진을 제안하였고, 검색 엔진을 다방면에서 테스트하고 분석하였다. 그러나 Elasticsearch 의 검색 성능이 아닌 인덱싱 성능 분석이나 클러스터 구성을 통한 성능 분석 연구는 확인할 수 없었다.

B. Wei [8]는 Elasticsearch 의 전체 텍스트(full-text) 검색 기술과 데이터 특성을 기반으로 실제 적용 시 Elasticsearch 인덱스 샤드(Shard) 수를 최적화하는 방법을 제안했다. 이 연구는 Elasticsearch 의 남은 저장 공간과 분산 클러스터 내 각 노드의 인덱스 샤드 크기를 종합적으로 분석 및 계산하여 시스템 내 최적의 인덱스 샤드 수를 결정함으로써 데이터 검색 효율성을 향상시킬 수 있음을 보여주었다. 그러나 인덱스 샤드의 수나 크기를 제외하고 Elasticsearch 의 다른 구성이나 클러스터 구성에 대한 성능 최적화 연구는 존재하지 않았다.

이와 같이 Elasticsearch 의 성능 분석과 관련된 여러 선행연구를 조사하였으며, 영구 메모리 기반 성능 분석, 검색 성능 최적화, 인덱스 샤드 수/크기 최적화 등 다양한 연구가 존재했다. 그러나, 본 연구는 여러 대의 PC 로 구성한 클러스터 환경에서 빅데이터 검색 플랫폼으로 각광을 받고 있는 Elasticsearch 의 인덱싱 성능을 최적화하는 연구로서 기존 관련 연구와 차별점이 있다.

III. 배경지식

3.1 Elasticsearch

Elasticsearch 는 Apache Lucene 라이브러리를 기반으로 개발된 오픈소스 검색엔진이다. NoSQL 의 일종으로 내용 전체를 색인해서 특정 단어가 포함된 문서를 검색하는 고차원적인 전문 검색(Full Text)이 가능하며, 분산 처리를 통해 대량의 데이터에 대해 실시간에 준하는 빠른 검색이 가능하다. 뿐만 아니라 높은 확장성을 가지고 있어 근래에는 이미지 검색을 위한 검색

엔진으로도 사용되고 있다[9]. 기본적으로 HTTP 를 통해 JSON 형식의 RESTful API 를 이용하여 검색을 수행하며 개발 언어, 운영체제, 시스템에 관계없이 이기종 플랫폼에서도 이용 가능하다[10].

Elasticsearch 의 클러스터 기본적인 개념 구조는 [그림 1]과 같다. Elasticsearch 를 분산 환경으로 구성하면 데이터 저장 공간인 인덱스(Index)가 여러 노드에 분산 저장되어 관리된다. 색인된 문서는 하나의 인덱스에 담기는데, 이 때 인덱스 내부에 색인된 데이터는 물리적인 공간에 여러 개의 파티션으로 나뉘어 구성되고, 이 파티션을 샤드(Shard)라고 부른다[10].

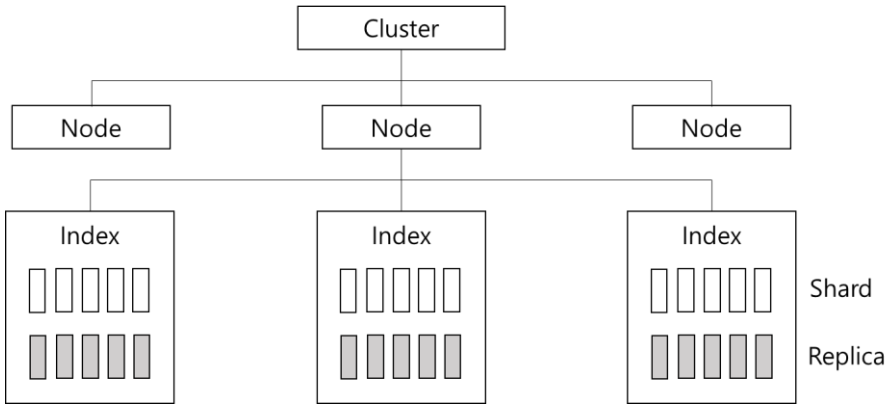

Figure 1. Elasticsearch Cluster Architecture
그림 1. Elasticsearch 클러스터 구조

샤드는 원본 데이터를 저장하고 요청을 받아들이는 역할을 하는 프라이머리 샤드(Primary Shard)와 그의 복제본인 레플리카 샤드(Replica Shard)로 구분된다. [그림 2]와 같이 레플리카 샤드는 프라이머리 샤드의 데이터를 복제하여 장애에 대응할 수 있도록 한다. 프라이머리 샤드와 레플리카 샤드는 서로 다른 인덱스에 저장되어 데이터의 안정성을 보장하며, 데이터는 빠른 검색을 위한 역색인(Inverted Index) 구조로 저장되어 있다. 인덱스를 만들 때마다 샤드의 수를 조절할 수 있으나, 한 번 생성된 인덱스의 프라이머리 샤드 개수를 변경할 수는 없다. 이처럼 샤드를 사용하여 데이터 분할을 수행하면 확장성, 일관성, 내결함성 문제를 향상시킬 수 있다[11][12].

노드가 여러 개 존재하는 경우 인덱스는 여러 노드에 분산 저장된다. 노드는 역할에 따라 master 노드, data 노드, ingest 노드, coordinate 노드 등으로 구분할 수 있다[13]. Master 노드는 전체 클러스터를 관리하고 data 노드에는 색인된 데이터가 저장된다. Ingest 노드는 색인을 하기 전, 데이터를 변환하는 역할을 수행한다[14]. 환경 설정 시 노드에게 부여할 역할을 설정할 수 있으며 하나의 노드에 여러 역할을 할당하는 것도 가능하다[10].

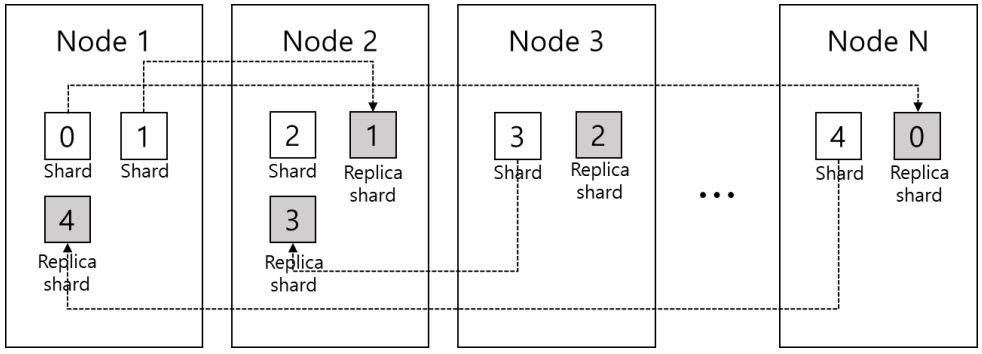

Figure 2. Elasticsearch index distribution
그림 2. Elasticsearch 인덱스 분배

3.2 Elastic Stack

데이터를 원천으로부터 수집하여 적재하고 시각화하는 과정은 여러 오픈 소스들을 이용하여 수행할 수 있는데, 이를 [그림 3]과 같이 Elastic Stack 으로 구현할 수 있다. Elastic Stack 은 검색 및 분석 엔진인 Elasticsearch 와 더불어 여러 소스에서 데이터를 수집할 수 있는 Logstash 등의 로그 수집기와 Kibana 와 같이 Elasticsearch 에서 차트와 그래프를 이용해 데이터를 시각화할 수 있게 만드는 오픈소스 프로젝트들의 모음 지칭하는 단어로, 대표적인 소프트웨어들의 앞 글자를 딴 ELK Stack 은 Elastic Stack 의 일종이다[15][16]. 본 연구에서는 ELK Stack 을 구축하여 실험을 진행하였다.

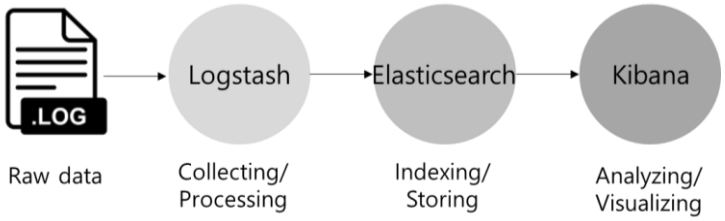

Figure 3. Elastic Stack Architecture
그림 3. Elastic Stack 구조

3.2.1 Logstash

Logstash 는 실시간 파이프라인 기능을 가진 오픈 소스 데이터 수집 엔진으로, 소스의 종류, 사이즈 및 형태에 관계없이 다양한 소스에서 데이터를 동시에 수집하여 변환하고 저장소로 전달하는 역할을 한다. 즉, Logstash 는 데이터 소스로부터 데이터 입력, 필터, 출력의 세 단계의 파이프라인으로 구성된다. 입력은 파이프라인 가장 앞부분에 위치해서 데이터 소스로부터 원본 데이터를 입력 받는 단계이며 직접 데이터 원본에 접근해 읽어 들이거나 서버를 열어놓고 받아들이는 형태로도 가능하다. 입력 받은 데이터가 Logstash 를 거칠 때 각 이벤트를 구문 분석하고 명명된 필드를 식별하여 구조를 구축하며 이를 공통 형식으로 변환 통합해주는 필터링 과정을 거칠 수 있으며 이와 같은 필터 단계는 필수 단계인 입력, 출력 단계와는 달리 선택적으로 적용 가능하다. 출력은 파이프라인의 마지막 단계로서, Elasticsearch, 파일, Kafka 와 같은 원하는 저장소 (리파지토리)에 데이터를 전달하는 과정을 담당한다.

이와 같이 Logstash 는 서로 다른 다양한 소스의 데이터를 탄력적으로 통합하고 선택한 대상으로 데이터를 정규화 할 수 있으며 다양한 입력, 필터, 출력을 통해 ETL(Extract-Transform-Load) 작업을 수행할 수 있다[17][18].

3.2.2 Kibana

Kibana 는 웹 기반 시각화 도구로 Elasticsearch 에서 색인된 데이터를 쉽게 검색하고 시각화하는 기능을 제공한다[17]. 다양한 그래프 및 차트 작성 도구로 대시보드를 통해 Elastic Stack 클러스터를 모니터링, 관리 및 보호하기 위한 사용자 인터페이스의 역할과 중앙 집중식 허브 역할을 하며[19], 여러 사용자가 공유하는 Elasticsearch 서비스의 경우 Kibana 에 액세스할 수 있는 모든 사용자는 Elasticsearch 에서 다른 사용자의 정보를 검색할 수 있다[20]. Kibana 는 대시보드 생성, 시각화, 검색 및 필터링, 매핑 관리, 플러그인 지원, 보안과 권한관리와 같은 6 가지 주요한 기능을 제공한다. Kibana 를 사용하여 다양한 시각화 요소를 포함한 대시보드를 만들 수 있으며, Elasticsearch 에 저장된 데이터를 시각화 할 수 있다. Elasticsearch 의 강력한 검색 엔진을 활용하여 데이터를 검색하고 필터링 가능하며 데이터 스키마를 기반으로 인덱스의 데이터 구조를 구성 및 관리가 가능하다. 마지막으로 Kibana 는 다양한 플러그인을 지원하여 기능 확장이 가능하고, 사용자 및 역할 기반 액세스 제어를 통해 데이터에 대한 접근 권한을 관리할 수 있다[18].

IV. 실험 구성

본 절에서는 클러스터 환경에서 작동하는 다양한 Elasticsearch 실험 구성 및 구조에 대해서 설명한다. 또한 산학공동연구를 통해 기업측으로부터 제공받은 실험 데이터 (실제 로그데이터)의 각 항목구성에 대해서 알아본다.

4.1 실험 환경 및 구성

[그림 4]에서 보듯, 본 연구 실험을 위해서 세대의 물리적 노드로 클러스터를 구성하였으며 각 노드들은 2.5GbE 네트워크 스위치로 연결이 되어 있다. 각 노드들은 i5-9600K CPU (6 cores), 16GB RAM, 2TB HDD, 500GB SSD 로 동일하게 구성되어 있다. 각 노드에는 Ubuntu v22.04, OpenJDK v1.11, Elasticsearch/Logstash/Kibana v7.17.7 등이 설치되어 있으며, 실험에 사용한 모든 소프트웨어는 오픈소스를 사용하였다. 참고로 본 실험에 사용된 클러스터의 노드 개수는 3 대이지만 노드 개수와 최적화 인덱싱 성능이 거의 비례 (초당 평균 인덱싱율: 30,216 → 64,554 → 80,745 eps) 하는 결과를 실험을 통해 확인하였으므로 노드 세 대로 구성된 현 클러스터의 실험결과로도 확장된 클러스터의 성능을 유추하기에 유의미하다고 생각된다.

Table 1. System configurations
표 1. 시스템 구성

Hardware/Software	Items	Specifications
Hardware (3 Servers)	CPU	Intel(R) Core(TM) i5-9600K CPU @ 3.70GHz (total 6 cores)
	RAM	16GB
	Storage	2TB HDD for storage 500GB SSD x 1 for OS
	Network	2.5GbE network switch, adapter
Software (Open Source)	OS	Ubuntu v22.04.1 LTS (64-bit)
	JDK	OpenJDK v1.11.0
	Elasticsearch, Logstash, Kibana	v7.17.7

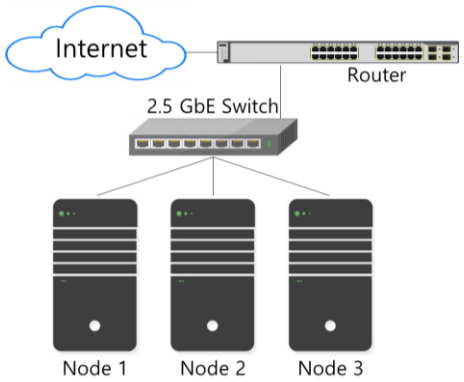

Figure 4. Overall Cluster Architecture
그림 4. 전체 클러스터 구조

4.2 실험 데이터

실험에 사용된 데이터는 산학 공동 연구를 진행한 기업체로부터 제공받은 실데이터로, 보안관제 서버로부터 수집된 실제 대용량 보안 로그 데이터이다. 또한 개인정보 보호를 위해 본 연구실에 데이터 제공 전 모든 보안처리와 익명처리를 기업체를 통해 완벽하게 거쳤다. 로그데이터는 저장된 디렉토리로부터 Logstash 를 통해 Elasticsearch 에 인덱싱하였다. 제공받은 전체 보안 로그데이터의 크기는 4.8 GB 이며 다음과 같은 총 56 가지 필드로 구성되어 있다.

Table 2. Experiment data fields
표 2. 실험용 데이터의 각 필드명

1	rdate	20	src_country	39	score
2	sender_type	21	dst_country	40	raw_packet
3	ip	22	device_id	41	src_info
4	group_id	23	node	42	dst_info
5	device_type	24	category	43	opt1
6	ckc	25	minor	44	opt2
7	sensor_id	26	flag	45	attack_category_code
8	stime	27	msg	46	signature_code
9	etime	28	pool_id	47	packet_size
10	src_ip	29	pool_group_id	48	drop_packet_size
11	dst_ip	30	cve_code	49	raw_packet_origin
12	proto	31	data_type	50	pool_info
13	src_port	32	is_profile	51	datasize
14	dst_port	33	profile_log_id_list	52	filter
15	action	34	profile_id	53	ip_version
16	signature	35	profile_name	54	raw_packet_length
17	highlight	36	ai_flag	55	malware_id
18	priority	37	ip_type	56	Interface
19	cnt	38	type		

4.3 실험 방법

실험마다 Elasticsearch node 와 Logstash node 를 구성에 맞게 필요한 개수만큼 동시에 실행하여 전체 데이터(4.8GB) 인덱싱이 완료될 때까지의 인덱싱 시간과 인덱싱 성능을 Kibana 를 이용하여 측정한다. 이와 동시에 Elasticsearch 클러스터가 구성된 각 노드들에서 Elasticsearch resource 사용량(CPU, memory), 시스템 전체 resource 사용량(CPU, memory), 디스크 I/O, Network I/O, 그리고 시스템 iowait 을 10 초 간격으로 측정한다. Logstash node 가 여러 대일 경우 각 node 에 4.8GB 의 데이터를 균등하게 분할하여 인덱싱하고, 매 실험마다 기존에 인덱싱 된 데이터, 로그들을 수동으로 삭제하고 새로 실행한다.

가령, PC1 과 PC2 에 Elasticsearch node 가 각각 한 개씩, 총 두 개 존재하고 primary shard 가 Elasticsearch node 마다 각각 두 개씩, 총 네 개 있다고 가정하자. 이때 데이터를 인덱싱하기 위해서 데이터의 위치로부터 stdout(standard output)을 이용해 Logstash node 가 Elasticsearch node 로 데이터를 전달하고, Elasticsearch node 안의 shard 에 데이터가 저장된다. 이 때, Logstash node 가 세 개라면 각 node 마다 전체 데이터 4.8GB 의 1/3 인 1.6GB 만큼의 데이터를 Elasticsearch node 로 전달하고, Elasticsearch 의 Master node 가 이를 클러스터 내의 네 개의 shard 에 균등하게 분배한다. 이 과정에서 Kibana 로 인덱싱 성능을 측정하고, PC1 과 PC2 각각에서의 Elasticsearch resource 사용량, 시스템 전체 resource 사용량, 디스크 I/O, Network I/O, 그리고 시스템 iowait 을 10 초 간격으로 측정한다.

V. 실험 결과 및 성능 분석

본 절에서는 Elasticsearch (ES)의 인덱싱 성능 최적화를 위한 각 주요 요소들의 구성 변경을 통한 다양한 실험결과 및 분석을 제공한다. 다양한 클러스터별 구성과 Elasticsearch 구성 요소들의 변화에 따른 인덱싱 성능 결과와 리소스 분석을 수행하고 최적의 성능을 보여주는 각 노드들의 구성과 성능 결과 및 분석을 통해 향후 클러스터 노드의 scale-out 또는 scale-up 시스템 확장에 따른 최적화 방향과 구성에 대한 중요한 단서 및 정보를 제공하는 것을 목적으로 한다. 각 실험 설정표는 다소 중복되는 정보가 있더라도 실험 재연의 용이성을 위해 전체 설정 정보를 그대로 제공하고자 하였다.

5.1 Primary shard 개수 별 성능 변화

본 절에서는 Elasticsearch 의 primary shard 개수 변화에 따른 인덱싱 성능 변화를 살펴본다.

5.1.1 Elastic Stack 실험 설정

본 실험에서는 아래 [표 3]와 같이 기존 최적화 설정을 유지하고 primary shard 개수를 1 개에서 4 개까지 변화시켜가면서 인덱싱 성능의 변화를 살펴본다.

Table 3. Elastic Stack Configurations
표 3. Elastic Stack 설정

Source	Configuration	Value	Remarks
Elasticsearch	Master-only node	true	Is the Elasticsearch master node the master-only node?
	Ingest node	true	Does the ingest node exist?
	Java heap size	1.5g	Java heap size per node
	Data node number	3	Elasticsearch node count
Logstash	Java heap size	1g	Java heap size per node
	Pipeline.batch.size	125	Batch size per node
	Node number	3	Logstash node count
Index	Primary shard	1 ~ 4	Primary shard count
	Replica shard	0	Replica shard count
	mappings	keyword	Field type of Elasticsearch index

5.1.2 인덱싱 성능 (인덱싱율과 인덱싱 시간)

[그림 5]는 싱글 노드와 두 개 노드로 구성된 클러스터 환경에서 Elasticsearch 의 primary shard 개수의 변화에 따른 인덱싱율과 인덱싱 시간을 보여준다. 참고로 Elasticsearch 에서 샤드라고 부르는 것은 primary shard 를 의미한다. 싱글 노드의 경우 primary shard 개수를 1 개에서 4 개로 증가시켜도 인덱싱 성능의 큰 변화는 없으며 오히려 4 개가 되었을 경우 1 개일때보다 0.95 배의 성능이 되었다. 이는 Elasticsearch 에 존재하는 모든 샤드는 master node 에서 관리를 하며 shard 개수가 증가할수록 master node 의 부하도 증가하기 때문이다. 즉, master node 의 부하로 인해 인덱싱 성능이 상대적으로 5% 감소되었다. 이런 현상은 두 개 노드로 구성된 클러스터에서도 유사하게 나타난다 ([그림 14]-(b)). Primary shard 개수가 증가하면 인덱싱 성능이 감소하는 경향이 있다. 또한 [그림 14]은 data node 개수의 변화에 따른 인덱싱 성능의 변화도 함께 보여준다. 기본적으로 data node 에 샤드가 저장되며 data node 당 샤드를 각각 1 개 (node*1)와 두 개(node*2)씩 저장하도록 변경하였다. 그림에서 보듯 data node 에 샤드를 두 개 할당할 때 1 개를 할당하는 경우보다 평균 25%의 성능이 하락함을 확인하였다. 또한 data node 개수가 증가할수록 인덱싱 성능이 하락하였다. [그림 14]-(c) 와 (d)는 인덱싱을 완료하는 시간을 나타낸다. 인덱싱 시간은 인덱싱 비율과 정확하게 반비례하므로 동일한 현상을 보여준다.

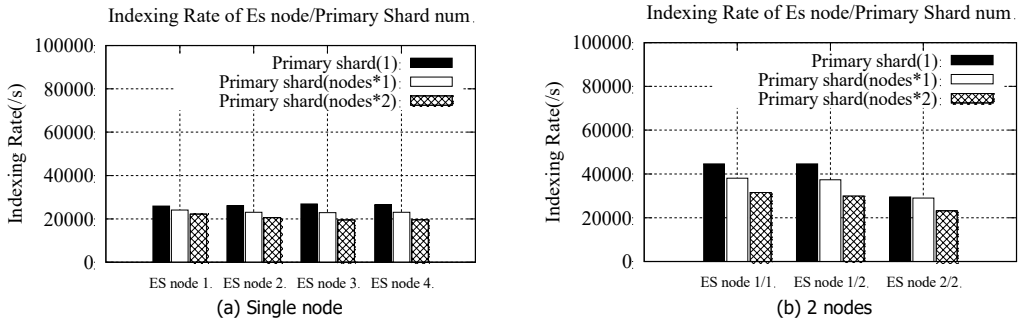

Figure 5. Indexing rate over the # of primary shard
그림 5. Primary shard 개수에 따른 인덱싱율

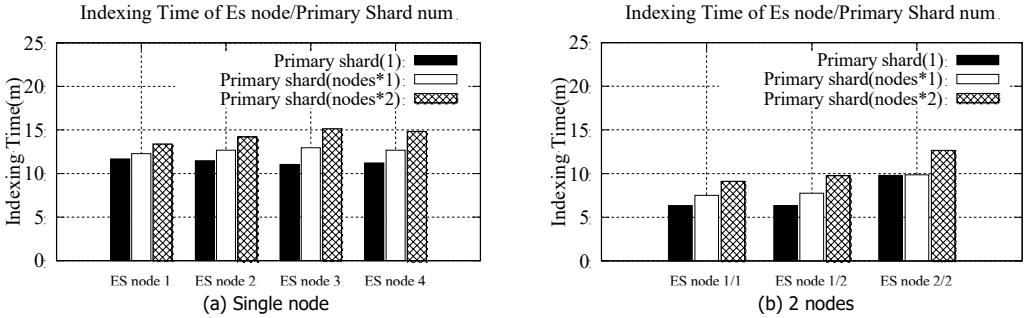

Figure 6. Indexing time over the # of primary shard
그림 6. Primary shard 개수에 따른 인덱싱 시간

5.2 Master node decoupling (Master-only, Ingest)

Elasticsearch 클러스터에는 master node 역할을 하는 노드가 반드시 하나 존재해야 하며, 그 master node 는 클러스터 상태관리, 샤드 할당, 인덱스 생성 및 수정, 노드 추가 및 삭제 등 다양한 역할을 수행한다. 따라서 처리할 데이터의 양이 방대하거나 클러스터의 사이즈가 확장될수록 master node 의 오버헤드가 급격히 증가하므로 master node 의 역할을 적절히 분담할 수 있도록 Elasticsearch 클러스터의 노드를 분배하는 것이 매우 중요하다. 이번 절에서는 Elasticsearch 의 master node 의 기능을 독립적으로 운영할 때와 다른 노드와 배분할때 성능 변화를 알아보고자 한다. 또한 클러스터 물리적 노드 사이즈를 싱글 노드부터 2 대, 3 대로 증가하면서 인덱싱에 미치는 영향을 알아본다.

5.2.1 Elastic Stack 실험 설정

본 실험 구성을 위해서 아래 [표 4]에서 보듯 master-only node, ingest node 할당 유무, 그리고 data node 개수를 1 개부터 3 개까지 변경하면서 실험을 진행한다.

Table 4. Elastic Stack Configurations
표 4. Elastic Stack 설정

Source	Configuration	Value	Remarks
Elasticsearch	Master-only node	True / False	Is the Elasticsearch master node the master-only node?
	Ingest node	True / False	Does the ingest node exist?
	Java heap size	1.5g	Java heap size per node
	Data node number	1 ~ 3	Elasticsearch node count
Logstash	Java heap size	1g	Java heap size per node
	Pipeline.batch.size	125	Batch size per node
	Node number	3	Logstash node count
Index	Primary shard	1	Primary shard count
	Replica shard	0	Replica shard count
	mappings	keyword	Field type of Elasticsearch index

5.2.2 인덱싱 성능 (인덱싱율과 인덱싱 시간)

Elasticsearch 클러스터에서 master node 는 인덱싱된 샤드를 저장하는 data node 의 역할과 Logstash node 로부터 수집 및 처리된 데이터를 클러스터에 있는 data node 로 분산 전송하는 ingest node 역할을 모두 수행할 수 있다. 소규모 데이터의 경우 별도로 master node 의 기능을 분리할 필요가 없으나, 대용량 데이터를 처리하거나 클러스터가 확장될 경우 master node 가 성능 bottleneck 이 되므로 기능을 별도로 분리할 필요가 있다. [그림 7]은 싱글 노드 (서버) 환경에서 (1) master 노드 기능 비분리 (Master Node), (2) data node 기능 분리 (Master-only Node),

(3) data node 기능 분리 + ingest node 기능 분리 (Ingest Node) 와 같이 세 가지 구성으로 실험한 결과를 보여준다. 그림에서 보듯, master node 에서 data node 를 분리시키고 master-only 노드로 구성을 했을 경우 인덱싱 성능이 평균 1.15 배 증가하였다. 또한 기존 master node 에서 data node 로 데이터를 전송하는 역할을 분리하여 독립적인 ingest node 로 구성 시 인덱싱 성능이 평균 1.14 배 향상되었음을 확인하였다.

[그림 8]와 [그림 9]은 동일한 실험을 클러스터 상에서 수행한 결과를 보여준다. 싱글노드 환경과 비교하여 물리적 노드 세 대로 구성한 클러스터 환경에서 인덱싱 성능은 평균 2.3 배가 향상됨을 확인하였다. 또한 [그림 9]에서 보듯, ingest node 기능을 분리시킬 경우 (Ingest Node), 물리적 노드 개수가 많아질수록 (즉, 클러스터 규모가 커질수록) ingest node 기능 분리의 성능 향상 효과 (평균 1.53 배 증가)가 뚜렷함을 알 수 있다. 따라서 클러스터 규모가 크거나 향 후 확장성을 고려할 경우에는 독립된 ingest node 를 구성하는 것이 성능 향상에 큰 도움이 된다.

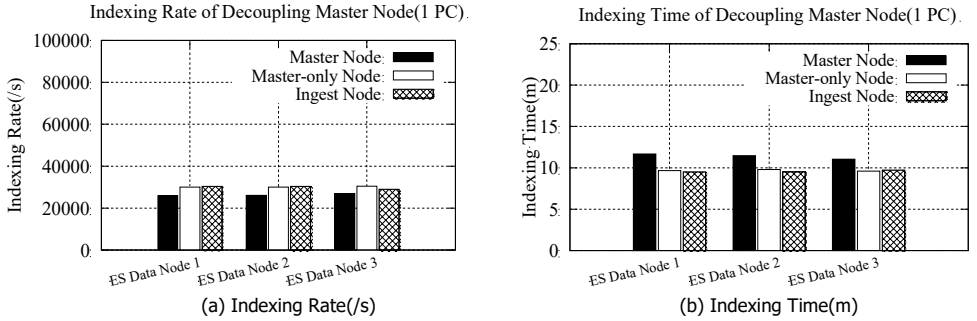

Figure 7. Indexing performance (rate and time) on a single node
그림 7. 싱글 노드 환경에서 인덱싱 성능 (인덱싱율과 시간)

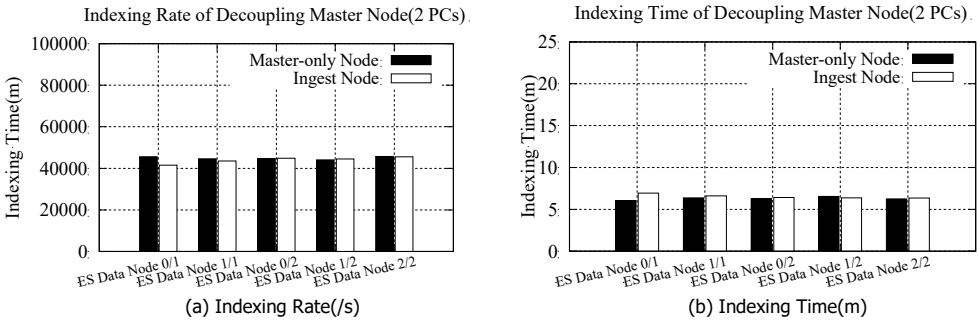

Figure 8. Indexing performance (rate and time) on a cluster of two nodes
그림 8. 두 노드 클러스터 환경에서 인덱싱 성능 (인덱싱율과 시간)

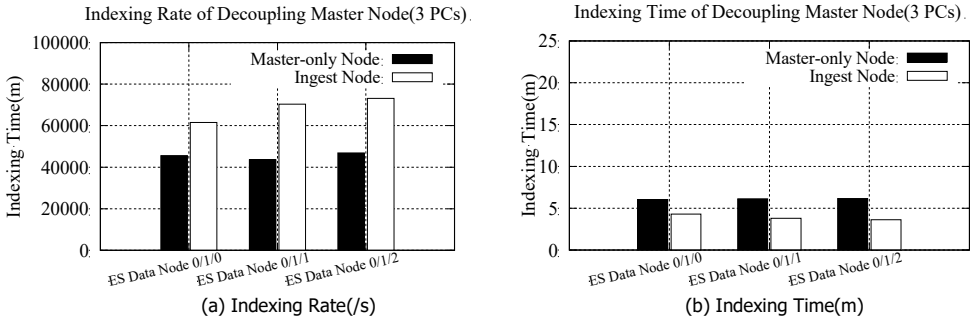

Figure 9. Indexing performance (rate and time) on a cluster of three nodes
그림 9. 세 노드 클러스터 환경에서 인덱싱 성능 (인덱싱율과 시간)

5.3 Logstash node 개수 별 성능

Logstash 노드는 데이터를 동시에 수집하여 변환하고 저장소로 전달하는 역할을 하므로 Elasticsearch 의 인덱싱 성능에 매우 큰 영향을 미친다. 본 절에서는 Logstash 노드의 개수를 1 개에서 7 개까지 차례로 변경하면서 인덱싱 성능 변화를 살펴본다.

5.3.1 Elastic Stack 실험 설정

아래 [표 5]에서 보듯, 기본 최적화 설정은 유지한 채 세 개 노드로 구성된 Elasticsearch 클러스터에서 Logstash node 개수만 변경하며 실험한다. 참고로 Logstash node 는 master node 인 물리 노드 1 에 존재한다.

Table 5. Elastic Stack Configurations
표 5. Elastic Stack 설정

Source	Configuration	Value	Remarks
Elasticsearch	Master-only node	true	Is the Elasticsearch master node the master-only node?
	Ingest node	true	Does the ingest node exist?
	Java heap size	1.5g	Java heap size per node
	Data node number	3	Elasticsearch node count
Logstash	Java heap size	1g	Java heap size per node
	Pipeline.batch.size	125	Batch size per node
	Node number	1 ~ 7	Logstash node count
Index	Primary shard	1	Primary shard count
	Replica shard	0	Replica shard count
	mappings	keyword	Field type of Elasticsearch index

5.3.2 인덱싱 성능 (인덱싱율 및 시간)

[그림 10]는 Logstash node 개수 변화에 따른 인덱싱 성능을 보여준다. 또한 싱글 노드 구성 (PC1)부터 2 개 (PC2), 3 개 (PC3) 노드로 클러스터 사이즈 증가에 따른 성능 변화도 함께 보여주고 있다. 그림에서 보듯 Logstash node 가 7 개로 증가함에 따라 인덱싱 성능은 평균 3.22 배나 증가했고 그에 반비례해서 인덱싱 완료시간 역시 평균 0.3 배로 감소하며 인덱싱 성능이 크게 향상되었다. 이는 Logstash node 개수가 증가함에 따라 각 노드가 병렬적으로 데이터를 수집/처리하기 때문이다. 그러나 Logstash node 의 처리 작업은 상대적으로 많은 컴퓨팅 자원을 필요로 하므로, 특히 싱글 노드 구성의 경우, Logstash node 가 5 개까지만 구성이 가능하며 6 개 이상은 컴퓨팅 자원 부족으로 구성이 불가능하여 그래프에는 나타나지 않는다. 클러스터 구성 시, data node 들을 별도의 물리 노드로 분리하면서 Logstash node 의 확장성도 증가함을 알 수 있다. 또한 물리적 노드를 1 대에서 3 대로 증가 시, 인덱싱 성능이 평균 1.74 배 향상되어 Logstash node 개수와 더불어 클러스터 사이즈 확장도 인덱싱 성능 향상에 큰 영향을 미침을 확인하였다. 그러나 Logstash node 의 경우 일정 개수 (여기서는 5 개 정도)부터는 인덱싱 성능 향상이 이루어 지지 않고 오히려 컴퓨팅 자원만 소모할 수 있으므로 반드시 컴퓨팅 환경에 맞는 최적 개수를 선정하여 구성하는 것이 바람직하다.

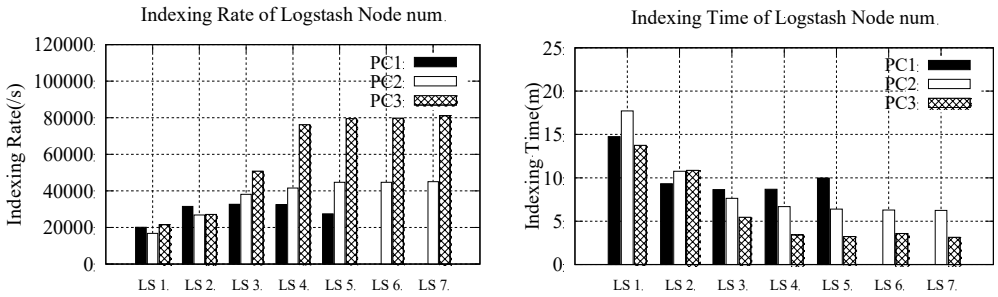

Figure 10. Indexing performance (rate and time) over # of Logstash node
그림 10. Logstash node 개수별 인덱싱 성능

5.4 최적화 결과

5.4.1 최적화 구성

Elasticsearch 의 최적성능을 위한 구성은 단순히 클러스터내 물리적 노드 개수뿐만 아니라 Elasticsearch 플랫폼에서 제공하는 논리적 노드들 (master node, ingest node, data node, logstash node, 등)의 구성이 매우 큰 역할을 한다. 클러스터의 물리적 노드 개수가 많더라도 논리적 노드 구성의 최적화가 이루어지지 않으면 성능 향상이 이루어지지 않음을 연구결과 확인했으며 또한 클러스터 물리적 노드의 컴퓨팅 성능이 높더라도 Elasticsearch 논리적 노드들의 최적화가 미진할 경우에는 심지어 컴퓨팅 성능이 낮은 물리적 노드보다 인덱싱 성능이 더 낮게 나타나는 것을 확인했다. 따라서 Elasticsearch 검색 플랫폼의 인덱싱 성능 최적화를 위해서는 단순히 물리적인 클러스터의 확장뿐만 아니라 논리적 노드들의 확장 및 최적 구성이 매우 중요함을 확인했다.

[그림 11]는 세 대의 물리적 노드로 구성된 Elasticsearch 클러스터에서 본 연구의 실험결과 최적의 인덱싱 성능을 보여주는 구성을 보여준다. 그림에서 보듯, 여러 개 (여기서는 7 개)의 Logstash (LS) node 를 구성하여 동시에 병렬적으로 데이터를 입력 받도록 구성해야 하며, 입력된 데이터를 각 data node 들로 분산 저장하기 위해서 ingest node 를 별도로 구성하도록 해야한다. Master node 역시 ingest node 의 역할을 대신할 수 있으나, master node 의 경우 전체 클러스터를 coordinate 해야하는 역할을 하므로, 클러스터가 확장됨에 따라 master node 가 성능의 bottleneck 가 될 수 있으므로 master-only node 로 구성하는 것이 성능 및 향후 확장성에 유리하다. Elasticsearch 의 data node 는 실질적으로 인덱스 샤드를 저장하는 역할을 담당하므로 많은 컴퓨팅 성능을 요구한다. 따라서 [그림 11]에서 보듯 data node 는 master node 와 반드시 별도의 물리적 클러스터 서버 노드에 구성을 해야한다. 본 실험에서는 두 개의 논리적 data node 를 구성했을 때 최적의 인덱싱 성능을 보여주었다. 또한 Logstash node 와 ingest node, master node 는 클러스터가 작을 경우 하나의 물리적 노드에 모두 할당해도 되며, 향후 물리적 클러스터 확장 시 master node 를 별도로 독립시키는 것이 유리하다.

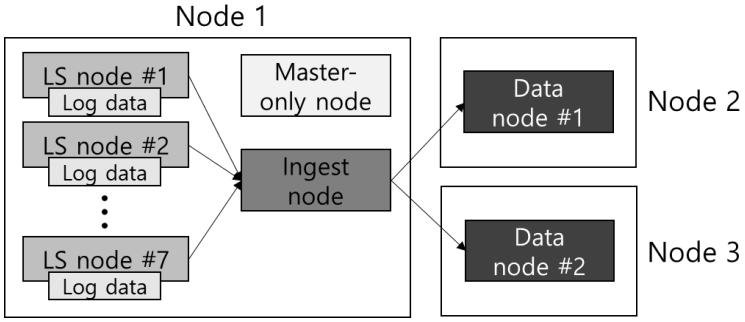

Figure 11. Optimal architecture (cluster of 3 nodes)
그림 11. 최적 아키텍처 (세 대 노드 클러스터 환경)

5.4.2 실험 설정

[표 6]은 세 대 노드로 구성된 본 실험 클러스터 환경에서 최적 인덱싱 성능을 보여주는 Elasticsearch, Logstash, 인덱스 각 설정 변수들의 값을 보여준다. 성능에 큰 영향을 미치는 주요 변수들 (노드 개수, primary shard 개수, Logstash 노드 개수, master node, Ingest node 유무, 등)의 설정값 변화에 따른 인덱싱 성능 변화는 5 장 실험 구성별 성능 분석에서 더 자세하게 다루도록 한다.

Table 6. Optimal configurations (cluster of 3 nodes)
표 6. 최적 설정 (세 대 노드 클러스터)

Sources	**Configurations**	**Values**	**Remarks**
Elasticsearch	Master-only node	true	Is the Elasticsearch master node the master-only node?
	Ingest node	true	Does the ingest node exist?
	Java heap size	1.5g	Java heap size per node
	Data node number	3	Elasticsearch node count
Logstash	Java heap size	1g	Java heap size per node
	Pipeline.batch.size	125	Batch size per node
	Node number	7	Logstash node count
Index	Primary shard	2	Primary shard count
	Replica shard	0	Replica shard count
	mappings	keyword	Field type of Elasticsearch index

5.4.3 인덱싱 성능 (인덱싱율 및 시간)

본 실험에 사용된 클러스터 기반 Elasticsearch 기본 구성하에서 인덱싱 성능은 평균 25,937.77 eps (event per second)로 측정이 되었다. 그러나 [그림 11]와 [표 6]의 최적 구성에서 4.8GB 실데이터의 인덱싱 성능은 80,745.07eps 로 기본 구성보다 평균 3.13 배 높은 성능을 보여주었다 ([표 7]). 또한 인덱싱 완료 소요 시간은 3 분 12 초였으며 인덱싱 시간은 인덱싱 비율과 비례하므로 향 후 인덱싱 성능 지표는 인덱싱 비율로 표기하도록 한다.

Table 7. Optimal Indexing Performance
표 7. 최적 인덱싱 성능

Indexing Rate(/s)	Indexing Time(m)	Indexing Size(GB)
80,745.07	03:12	4.8 GB

[그림 12]은 Kibana 대시보드에서 추출한 시간에 따른 인덱싱 성능 변화를 보여준다. 초기 30 초 (1 구간) 동안 인덱싱 성능이 서서히 증가하는 이유는 [그림 11]에서 설정한 모든 Elasticsearch 논리적 노드들이 한꺼번에 동시에 작동하지 않고 하나씩 active 해지면서 실행되기 때문이다. 그리고 30 초 이후부터는 전체 논리적 노드들이 모두 병렬적으로 작동하면서 최적(최고)의 성능을 지속함을 알 수 있다. 반대로 마지막 종료 구간에도 작업을 모두 마친 논리적 노드들이 하나씩 작업을 종료하면서 인덱싱 성능이 순차적으로 낮아지는 모습을 보여준다. Kibana 대시보드에서 보여주는 최적 인덱싱 성능은 평균 80,745.07eps 이지만 이는 시작과 끝부분의 낮은 인덱싱 성능이 모두 포함되었기 때문이며 실질적인 지속 인덱싱 평균 성능은 100,000eps 에 근접한다. 따라서 데이터 용량이 더욱 증가할 경우 최적 평균 인덱싱 성능은 현재 80,745.07eps 보다 훨씬 높은 90,000eps 후반 ~ 100,000eps 근접한 성능이 나올 것으로 기대한다.

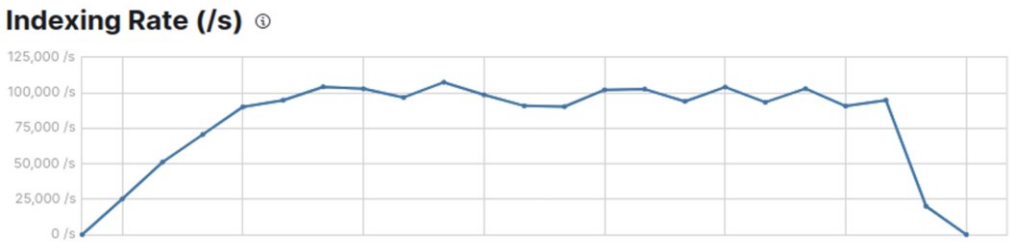

Figure 12. Indexing rate over time (each section: 30 seconds)
그림 12. 시간에 따른 인덱싱 성능 변화 (구간별: 30 초)

5.4.4 Resource 측정

[그림 13]은 클러스터 내 node 1 에서 측정한 인바운드 및 아웃바운드 네트워크 트래픽 양을 보여준다. [그림 11]에서 보듯 node 1 은 Logstash node 들과 master node, ingest node 가 동시에 존재하는 물리적 노드로서, 여러 Logstash 노드들을 통해 읽은 실험용 로그 데이터를 ingest node 가 Elasticsearch 의 data node 들이 존재하는 node 2 와 3 에 네트워크를 통해 전달하는 역할을 한다. 따라서 node 1 의 아웃바운드 네트워크 트래픽이 평균 48.351MB/s 으로 인바운드 트래픽보다 22.8 배 더 많은 네트워크 트래픽을 보여줌을 알 수 있다.

실험용 클러스터 노드들은 2.5GbE 스위치와 NIC (Network Interface Card)으로 연결되어 있으므로 네트워크 대역폭이 Elasticsearch 클러스터 성능의 bottleneck 이 될 가능성은 없으나 (현재 네트워크 대역폭의 평균 19% 정도를 사용중) 클러스터가 확장되거나 I/O 성능이 높아질수록 네트워크 대역폭이 성능 문제가 될 수 있으므로 10GbE 네트워크나 Infiniband 와 같은 고대역폭 네트워크 연결이 좋은 대안이 될 것이다.

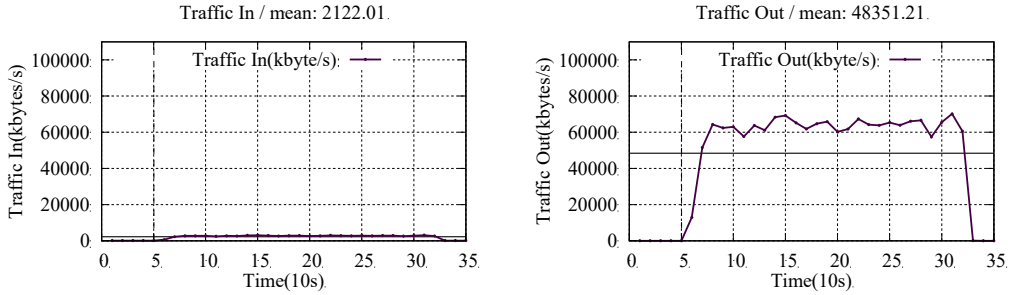

Figure 13. Inbound/outbound network traffic (node 1)
그림 13. 인바운드/아웃바운드 네트워크 트래픽 (node 1)

[그림 14] 은 각 클러스터 노드들의 다양한 시스템 리소스 사용량을 보여준다. Logstash node 들과 master node, ingest node 가 모두 존재하는 물리적 node 1 (즉, PC1)에서 CPU 와 메모리 사용량이 가장 높음을 알 수 있다. 특히 CPU 사용량은 초기에 100%에 육박하는 모습을 보여주고 있으며 메모리 사용량 역시 전체 16GB 중 11GB 를 사용할 정도로 리소스 사용량이 높다. 이는 데이터를 읽어서 네트워크를 통해 data node 로 전송하는 I/O 오버헤드에 기인한 것으로 [그림 14]-(c)와 (d)의 disk I/O 값을 통해서 확인 가능하다. [그림 14]-(c)를 보면 Logstash node 와 ingest node 를 통해 데이터의 수집과 처리 역할을 담당하는 node 1 에서 높은 disk read I/O 가 일어나고, [그림 14]-(d)를 통해 [그림 11]와 같이 Elasticsearch data node 각각 설정되고 primary shard 가 존재하는 node 2 (PC2)와 node 3 (PC3)에서 인덱스 저장으로 인한 높은 disk write I/O 가 일어나는 것을 볼 수 있다.

이 실험 분석을 통해 향 후 클러스터 확장 시 master node 역할을 담당하고 있는 노드 (여기서는 node 1)가 성능 bottleneck 이 될 가능성이 매우 높음을 알 수 있으며, 이에 따라 더 많은 컴퓨팅 자원 (또는 고성능 서버)을 master node 역할을 하는 노드에 할당하거나 논리적 노드들

(Logstash node, ingest node, master node)을 각 물리적 노드 (서버)로 독립시키는 것이 바람직하다.

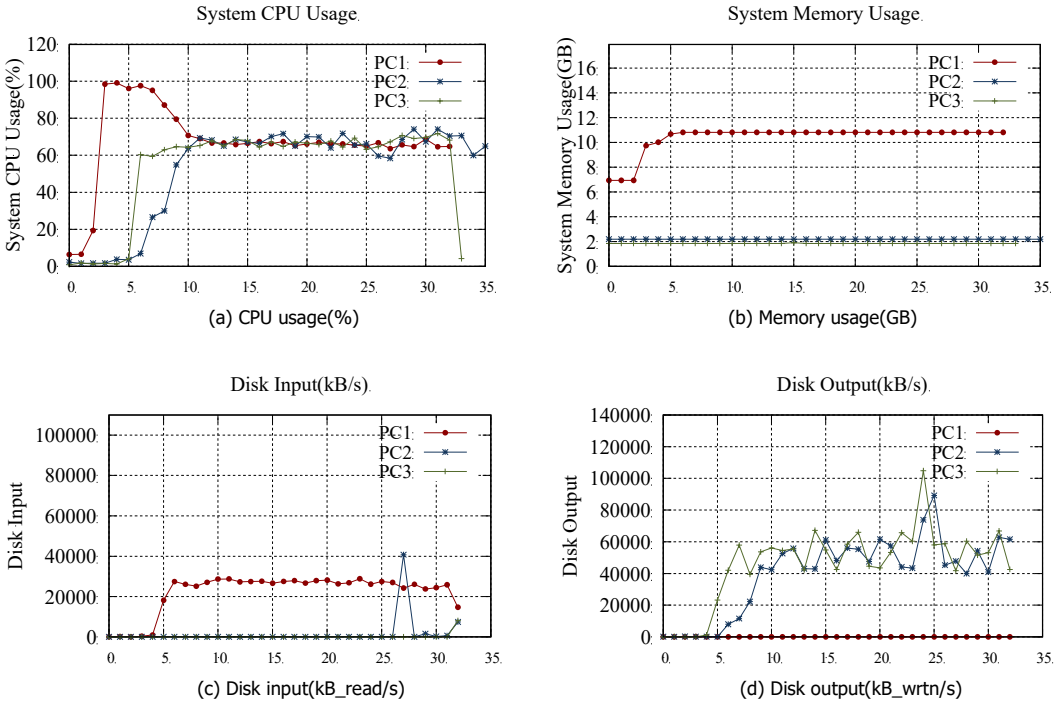

Figure 14. System resource usage
그림 14. 전체 리소스 사용량

VI. 결론

본 논문에서는 여러 대의 물리 서버에서 Elasticsearch cluster 를 구성할 때, 최적의 인덱싱 성능을 내기 위한 구성을 제공하였다. 클러스터의 물리 서버의 개수를 증가시키고, master node 의 역할을 data node 와 ingest node 로 분리하고, Logstash node 의 개수를 증가시킬수록 인덱싱 성능을 높일 수 있음을 다양한 실험 결과 분석을 통해 확인했다. 3 대 노드 Elasticsearch 클러스터 환경하에서 최적 구성 연구 실험 결과 Elasticsearch 기본 구성의 인덱싱 성능보다 약 3.13 배 더 높은 성능을 도출할 수 있었다.

또한 다양한 클러스터 구성과 Elasticsearch 환경 구성별 다양한 성능 실험 결과 분석을 제공함으로써 단순히 본 연구 실험에 사용된 3 대 노드 클러스터 구성의 환경이 아닌, 향후 달라질 수 있는 여러 환경 (가령, 데이터용량 증대, 클러스터 노드 개수 증가, 클러스터 노드 컴퓨팅 리소스 증대, 등) 에서도 본 논문에서 제공한 연구 결과를 통해 최적의 인덱싱 구성과 성능을 예측할 수 있는 훌륭한 참조 역할을 할 수 있을 것으로 기대한다.

VII. 감사의 글

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학 ICT 연구센터지원사업의 연구결과로 수행되었음 (IITP-2023-2018-0-01799).

VIII. 참고문헌

[1] B. Jin and Y. Ji, “A Study about Search Engine Interface Design including User’s Search Goal,” The Journal of Society for e-Business Studies, Vol. 13, No. 4, pp111-124. February 2008.

[2] K. Kim and Y. Cho, “Improving Elasticsearch for Chinese, Japanese, and Korean Text Search through Language Detector,” Journal of information and communication convergence engineering (Journal of Information and Communication Convergence Engineering), Vol. 18 No. 1, pp 33–38. 2020.
[3] O. Kononenko, O. Baysal, R. Holmes and M. W. Godfrey, "Mining modern repositories with elasticsearch." Proceedings of the 11th working conference on mining software repositories, pp 328-331, 2014.
[4] D. Sharma, R. Shukla, A. K. Giri and S. Kumar, "A Brief Review on Search Engine Optimization," 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, pp. 687-692, 2019.
[5] E. Tomes and N. Altiparmak, "A Comparative Study of HDD and SSD RAIDs' Impact on Server Energy Consumption," 2017 IEEE International Conference on Cluster Computing (CLUSTER), Honolulu, HI, USA, pp 625-626, 2017.
[6] E. Lee and D. Park, “Performance Analysis of Real-Time Big Data Search Platform Based on High-Capacity Persistent Memory,” Journal of Platform Technology, Vol. 11, No. 4, August 2023.
[7] J. He, "Research on Personalized Search Based on ElasticSearch," 2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT, Weihai, China, 2020, pp. 572-575
[8] B. Wei, J. Dai, L. Deng and H. Huang, "An Optimization Method for Elasticsearch Index Shard Number," 2020 16th International Conference on Computational Intelligence and Security (CIS), Guangxi, China, 2020, pp. 191-195
[9] S. Iyer, S. Chaturvedi, T. Dash, “Image Captioning-Based Image Search Engine: An Alternative to Retrieval by Metadata.” Soft Computing for Problem Solving, vol. 817, pp 181-192, 2019.
[10] T. Kwon, D. Kim, H. Kim, J. Park, Y. Choi, and H. Hwang, “Elasticsearch practical guide,” Wikibooks, pp 29-33, 2019.
[11] P. M. Dhulavvagol, V. H. Bhajantri, and S. G. Totad, “Performance analysis of distributed processing system using shard selection techniques on elasticsearch”. Procedia Computer Science, 167, pp 1626-1635, 2020.
[12] Elastic Guide book, Chapter 3.2. Index & Shards, https://esbook.kimjmin.net/03-cluster/3.2-index-and-shards
[13] S. Park and J. Kang. Elasticsearch operation know-how, Insight, pp 1-20, pp 75-107, 2021.
[14] Elasticsearch Best Practice Architecture, https://www.elastic.co/kr/webinars/elasticsearch-architecture-best-practices
[15] What it ELK Stack?, https://www.elastic.co/kr/what-is/elk-stack
[16] The Complete Guide to the ELK Stack, https://logz.io/learn/complete-guide-elk-stack/#what-elk-stack
[17] J. Hamilton, B. Schofield, M. G. Berges, and J. C. Tournier. “SCADA Statistics monitoring using the elastic stack (Elasticsearch, Logstash, Kibana).”, International Conference on Accelerator and Large Experimental Physics Control Systems, January 2018.
[18] Introduction to Logstash, https://www.elastic.co/guide/kr/logstash/current/introduction.html
[19] What is Kibana?, https://www.elastic.co/kr/what-is/kibana
[20] W. Takase, T. Nakamura, Y. Watase, and T. Sasaki. “A solution for secure use of Kibana and Elasticsearch in multi-user environment.” arXiv preprint arXiv:1706.10040, 2017.

저자소개

금나연 (*Nayeon Keum*)

2024 년 2 월 숙명여자대학교 IT 공학전공 학사

관심분야 : 플랫폼 엔지니어링, 클라우드, 오픈소스

박동철 (*Dongchul Park*)

2017 년 8 월 미국 인텔 연구소 수석연구원
2023 년 2 월 숙명여자대학교 소프트웨어학부 조교수
2023 년 3 월 ~ 현재: 중앙대학교 산업보안학과 부교수

관심분야 : 빅데이터 플랫폼, 스토리지 시스템, 분산 컴퓨팅