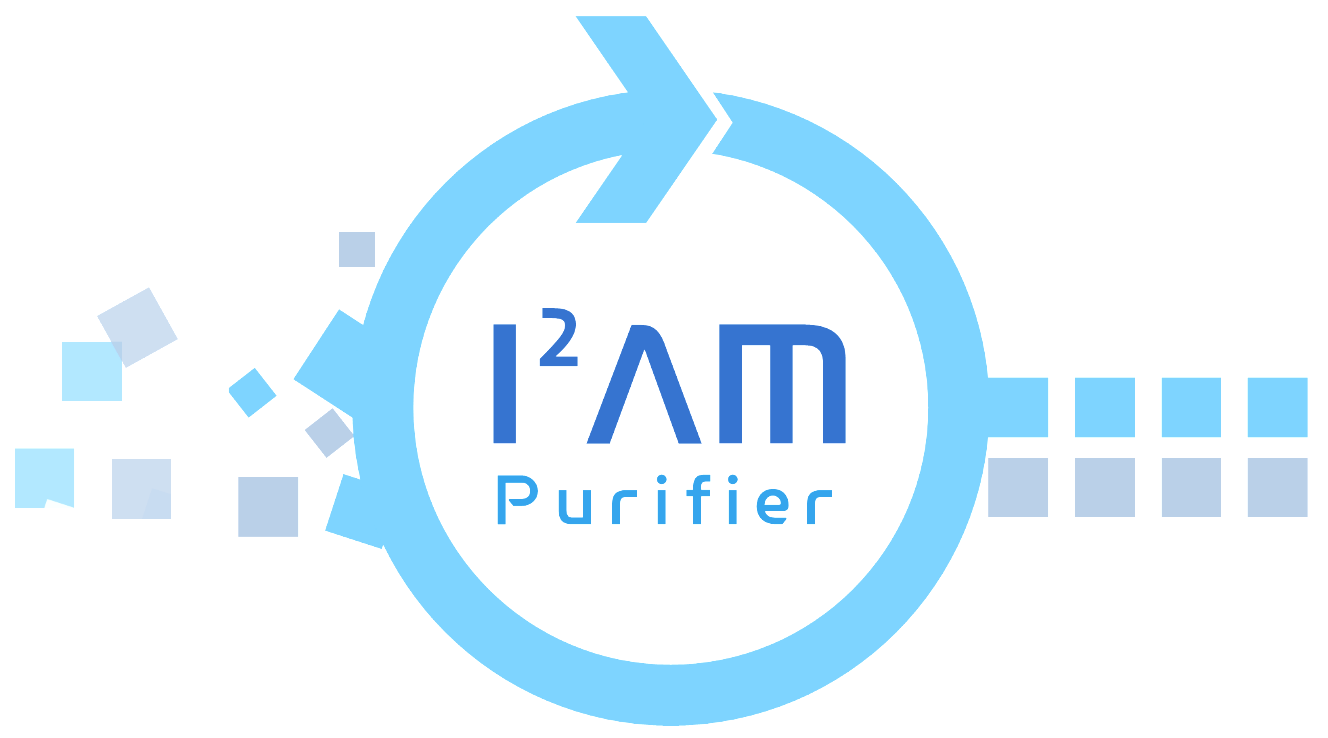
****

2016. 11.

강원대학교, 연세대학교

시스템 분석서

- 실시간 이상 탐지 -

시스템 분석서

- 동적 태그 클라우드 -

문서 정보

|  |  |
| --- | --- |
| **버 전** | 0.1 |
| **작성일** | 2016-11-14 |
| **상 태** | 🞎 완료 🞎 진행 중 🗹 초안 |
| **작성자** | 손시운(강원대학교) |
| **검토자** | 이름(소속) |
| **승인자** | 이름(소속) |

**목 차**

[1. 개요 1](#_Toc467351492)

[1.1. 시스템 배경 및 목적 1](#_Toc467351493)

[1.2. 관련 연구 2](#_Toc467351494)

[1.2.1. Apache Storm 2](#_Toc467351495)

[1.2.2. Apache Kafka 3](#_Toc467351496)

[1.2.3. Apache Flume 4](#_Toc467351497)

[1.2.4. 이동 평균 및 3-시그마 4](#_Toc467351498)

[2. 실시간 이상 탐지 시스템 5](#_Toc467351499)

[2.1. 시스템의 전체 구조 5](#_Toc467351500)

[2.2. 로그 데이터 수집을 위한 Flume 및 Kafka 6](#_Toc467351501)

[2.3. 이상 탐지를 위한 Storm 토폴로지 7](#_Toc467351502)

[2.4. 시각화를 위한 웹 및 데이터베이스 8](#_Toc467351503)

[3. 결론 9](#_Toc467351504)

# 개요

## 시스템 배경 및 목적

최근 소셜 네트워크 서비스(SNS, social network service), 사물 인터넷(IoT, Internet of Things)등 대용량 데이터를 발생시키는 서비스가 급증함에 따라 빅데이터가 화두에 올랐다. 빅데이터란 다양한 형태로 빠르게 발생하는 대용량 데이터, 또는 이러한 데이터를 저장·가공하기 위한 기술을 의미한다. 대부분의 빅데이터 기술은 단일 서버보다 여러 노드로 구성된 클러스터를 통해 데이터를 관리하며, 가장 대표적인 기술은 하둡(Apache Hadoop)이 있다. 하지만 하둡은 대용량의 데이터를 일괄적으로 저장 및 처리하며, 이는 실시간이 아닌 배치 작업에 적합하다. 이에 따라 실시간으로 끊임없이 발생하는 스트리밍 데이터를 분석하기 위한 많은 기술이 연구되었다. 대표적인 분산 실시간 분석 프레임워크인 Apache Storm은 스트리밍 데이터를 분산 환경에서 실시간으로 처리할 수 있으며, 하둡과 매우 유사한 구조를 갖는다.

이러한 분산 환경에서는 각 서버의 자원 활용을 위한 상태 모니터링과 관리가 반드시 필요하다. 로그 데이터는 시스템 자원이나 응용 프로그램의 상태를 기록하는 데이터로써, 각각은 작은 양이지만 매우 빠르게 다양한 형태로 생성되므로 대표적인 빅데이터에 해당된다. 다수의 서버를 구축하여 사용하는 분산 시스템은 서버의 증가에 따라 관리해야 할 로그 데이터는 폭증하게 되며, 로그 데이터를 기반으로 한 모니터링에 많은 비용이 요구된다. 시스템 모니터링의 주요 목적은 분석을 통한 이상 탐지, 시스템 사용량 파악 등이 있으며, 특히 이상 탐지의 수행을 위한 여러 가지 방법들이 연구되어 왔다.

본 시스템은 분산 환경에서 발생하는 대량의 로그 데이터로부터 급격히 변화하는 이상치를 탐지하고자 한다. 이를 위해 먼저 분산 수집 프레임워크인 Apache Flume을 사용하여 분산된 서버의 시스템 자원을 로그 데이터 형태로 실시간으로 수집한다. 다음으로 수집된 로그 데이터는 분산 메시징 시스템인 Apache Kafka를 통해 Storm으로 전달되며, Storm은 이를 이상 탐지 기법인 이동 평균 및 3-시그마를 사용하여 이상치를 데이터베이스에 저장한다. 마지막으로 이상 탐지 결과를 웹 인터페이스로 시각화하여 이상치가 올바로 탐지되었음을 확인한다. 본 시스템을 활용하면 실시간으로 수집된 로그 데이터에서 이상치를 탐지하고 그 결과를 직관적으로 확인할 수 있다.

## 관련 연구

### Apache Storm

실시간 데이터 처리 기술은 실시간으로 발생하는 스트림 데이터를 수집 즉시 처리하는 기술이며, 대표적으로 Apache Storm, Apache Spark, EsperTech Esper 등이 있다. 이 중 Storm은 분산된 다수의 서버에서 빠르게 데이터를 처리하는 실시간 분산 처리 기술이며, 필요에 따라 클러스터를 확장(scale-out) 가능하다는 장점이 있다. Storm은 Apache Hadoop과 유사한 마스터/슬레이브 구조를 갖는다. 그림 1은 이와 같은 Storm의 동작 구조를 나타낸다. 그림에서, 님버스(Nimbus)는 마스터, 슈퍼바이저(Supervisor)는 슬레이브에 해당하며, 이들은 주키퍼(Zookeeper)를 통해 관리가 된다. 님버스는 슬레이브 역할을 수행하는 슈퍼바이저에 작업을 배포/할당하며, 주기적으로 슈퍼바이저의 상태를 파악하고 장애에 대응한다. 슈퍼바이저는 님버스가 할당한 작업을 직접 수행하며 님버스에 상태를 보고한다.

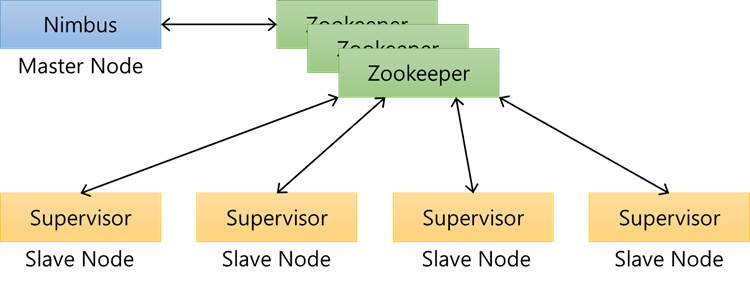


그림 1. Apache Storm 동작 구조.

Storm은 작업을 처리할 때 데이터의 입력부터 출력까지 일련의 작업을 토폴로지(Topology)로 표현한다. 토폴로지는 스파우트(Spout)와 볼트(Bolt)로 구성된다. 먼저, 스파우트는 스트리밍 데이터의 발생지로부터 데이터를 입력 받아 Storm에서 사용하는 데이터 형태인 튜플(Tuple)로 바꾸어 볼트에게 전달한다. 다음으로, 볼트는 스파우트로부터 전달 받은 데이터를 처리하여 다음 볼트로 전달하거나, 결과를 출력 또는 저장 장치로 보낸다. 본 시스템에서는 이상 탐지를 위해 Storm 상에서 이동 평균 및 3-시그마 기법을 설계 및 구현한다.

### Apache Kafka

Kafka는 LinkedIn에서 대용량의 실시간 로그 데이터의 처리를 위해 개발한 분산 메시징 시스템으로 실시간으로써, 서로 다른 시스템 간에 무한히 발생하는 데이터를 전달한다. 기존 ActiveMQ, RabiitMQ 등의 메시징 시스템은 단일 서버에서만 동작이 가능하였으나, Kafka는 분산 시스템에서 동작이 가능하도록 설계하여 분산 및 복제의 구성이 매우 쉽다.

그림 2는 Kafka의 동작 구조를 나타낸 것으로, 이는 Kafka가 발행-구독 모델을 기반으로 동작함을 보인다. 먼저 좌측의 프로듀서(producer)는 다른 시스템에서 발생한 메시지를 브로커(broker)로 발행(publish)하는 역할을 하며, 우측의 컨슈머(consumer)는 다른 시스템에서 브로커로부터 메시지를 구독(subscribe)하는 역할을 한다. 또한, 중앙의 브로커는 프로듀서로부터 전달 받은 메시지를 토픽(topic) 별로 분류하여 적재해나는데, 이때 토픽이란 kafka에서 메시지를 분류하기 위한 키의 역할을 한다.



그림 2. Apache Kafka 동작 구조.

Kafka는 메시지를 분산 환경에서 분배 및 복제하여 관리한다. Kafka는 메시지를 파티션(partition) 개념으로 나누어 각 서버에 분배하는데, 여기서 프로듀서가 메시지를 어떤 파티션에 전송할 지는 파티션 분배 알고리즘에 의해 결정된다. 다음으로 Kafka는 고가용성을 위해 정의된 복제 수만큼 복제하여 저장한다. 만약 복제 수를 N으로 설정하였을 때 한 개의 리더(leader)와 N-1개의 팔로워(follower)로 구성되며, 모든 읽기 및 쓰기 작업이 리더에서만 이루어지므로 N-1개의 장애까지 극복이 가능하다.

### Apache Flume

Flume은 다수의 서버로부터 발생하는 로그 데이터를 수집하여, 하둡 HDFS 등의 중앙 저장소로 데이터를 저장하는 시스템에 적합하다. 그림 3는 Flume의 가장 기본적인 구조도이며, 웹 서버로부터 발생하는 로그 데이터를 수집하고 HDFS에 저장한다. Flume은 소스(source), 채널(channel), 싱크(sink)로 구성된다. 먼저 소스는 이벤트로부터 데이터를 읽어 들이고, 다음으로 채널이 데이터를 저장하며, 마지막으로 싱크가 결과를 내보낸다.

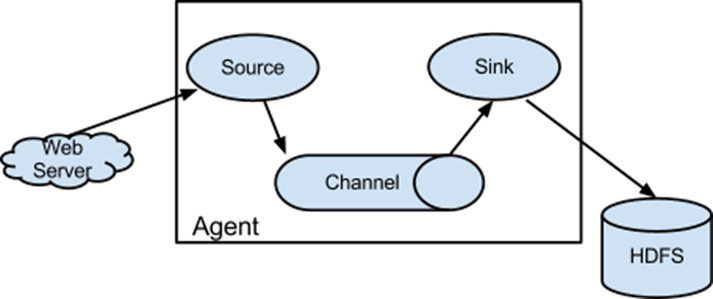


그림 3. Apache Flume 동작 구조.

### 이동 평균 및 3-시그마

이동 평균은 각 시간대에서 일정 기간(window)의 평균을 계산하는 방법으로 불규칙한 변동을 제거하여 데이터의 추세를 파악할 수 있다. 이동 평균은 단순 이동 평균, 가중 이동 평균, 지수 가중 이동 평균(지수 평활 법) 등으로 나누어진다. 먼저 단순 이동 평균은 각 구간에서 일반적인 평균을 계산한다. 다음으로, 가중 이동 평균은 각 구간의 데이터에 서로 다른 가중치를 부여하여 평균을 계산한다. 마지막으로, 지수 가중 이동 평균은 이전 시간의 이동 평균에 가중치인 평활 계수를 곱하여 계산한다. 3-시그마는 정규 분포에서 평균으로부터 ±3 표준편차의 범위 내에 99.7%의 거의 모든 값이 포함된다는 경험적인 규칙을 뜻한다. 본 시스템은 이러한 세 가지 기법 중 단순 이동 평균 및 3-시그마 기법을 사용하여 변화율이 급증하는 로그 데이터를 이상치로 간주하고 이를 탐지한다.

# 실시간 이상 탐지 시스템

본 장에서는 제안하는 실시간 이상 탐지 시스템에 대하여 설명한다. 먼저 제2.1절에서 전체 구조를 살피고 제2.2절부터 제2.4절에 걸쳐 시스템에 사용한 각 모듈을 자세히 설명한다.

## 시스템의 전체 구조

그림 4는 실시간 이상 탐지 시스템의 전체 구조도이다. 가장 하단에는 시스템에 사용한 하드웨어를 표현하였으며, 마스터 노드 한 대와 슬레이브 노드 여덟 대로 구성되어있다. 본 시스템은 아홉 대의 서버에서 로그 데이터를 수집하고 이상치를 탐지하였다. 이때 로그 데이터를 수집하는 Flume을 다른 서버에 추가한다면, 해당 서버의 이상치도 탐지할 수 있다. 다음으로 Flume과 Kafka는 로그 데이터를 수집하여 Storm 클러스터로 전달하는 역할을 한다. 그리고 Storm 클러스터는 전달 받은 로그 데이터로부터 이상치를 탐지하여 데이터베이스(MariaDB)에 전달한다. 마지막으로 오픈 소스 웹 시각화 도구인 D3.js를 사용하여 이상 탐지 결과를 실시간으로 시각화한다. 각각의 모듈은 다음의 제2.2절부터 제2.4절에 걸쳐 자세히 설명한다.

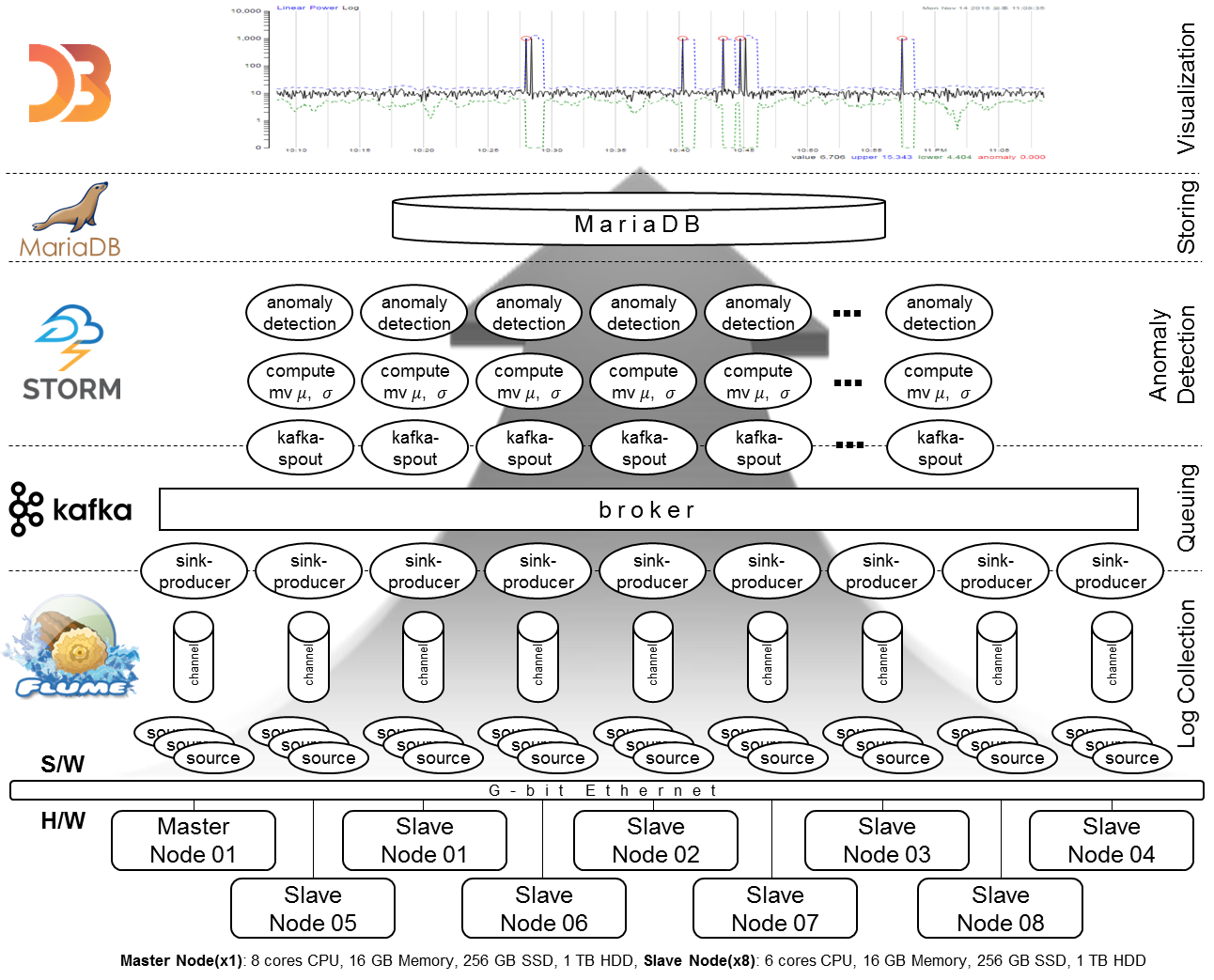


그림 4. 실시간 이상 탐지 시스템의 전체 구조도.

## 로그 데이터 수집을 위한 Flume 및 Kafka

본 절에서는 다수의 시스템으로부터 로그 데이터를 수집하는 Flume과 수집된 로그 데이터를 전달하는 Kafka에 대해 설명한다. 먼저 Flume은 다수의 서버로부터 이상 탐지의 대상이 되는 로그 데이터를 실시간으로 수집한다. 로그 데이터의 종류는 매우 다양하나 본 시스템에서는 네 개의 로그 데이터를 수집하며, 각각의 로그 데이터는 전체 디스크의 사용량(MB), 메모리의 사용량(KB), CPU의 사용률(%), 네트워크의 5초당 출력량(KB)이다.

먼저, Flume의 소스는 이러한 로그 데이터를 수집하기 위해 각각의 로그 데이터를 모니터링하며 결과를 채널로 전송한다. 따라서 본 시스템에서는 아홉 대의 서버에 각각 Flume이 동작하며, 총 36 개의 소스가 로그 데이터를 수집한다. 이때, 로그 데이터의 형태는 [{클러스터 명}, {호스트 명}, {로그 키}, {로그 값}, {로깅 시간}]으로 정의하였다. 이렇게 수집된 로그 데이터는 싱크에서 Kafka의 브로커로 전송된다.

본 시스템에서는 Kafka를 위해 한 대의 마스터 노드에 주키퍼를 설치하고, 여덟 대의 각 슬레이브 노드에 Kafka를 동작시킨다. 이는 관리하는 노드와 데이터를 처리하는 노드를 구분하기 위함이다. 또한, 로그 데이터를 전달하기 위해 사용한 토픽은 세 개의 복제 수와 여덟 개의 파티션으로 정의하였다. 그리고 Flume의 싱크가 Kafka의 브로커 파티션으로 데이터를 분배할 때, 로그 데이터로부터 클러스터 명, 호스트 명, 로그 키를 기준으로 해시(hash)하였다. 이를 통해, 같은 서버에서 수집된 동일한 로그 데이터는 반드시 같은 파티션으로 전달되며, 또한 로그 데이터의 순서도 유지될 수 있다.

## 이상 탐지를 위한 Storm 토폴로지

Storm에서 이상 탐지를 수행하기 위해 토폴로지를 정의해야 한다. 본 시스템에서 사용한 토폴로지는 한 개의 스파우트와 네 개의 볼트로 구성되어 있다. 먼저 스파우트는 Kafka에 저장된 로그 데이터를 읽어 들이기 위해 KafkaSpout을 사용하였다. KafkaSpout은 Storm의 내부 프로젝트 중, Kafka와의 쉬운 연동을 위해 구현된 프로젝트에 정의되어있다. 하지만 KafkaSpout이 Kafka로부터 수집한 로그 데이터는 하나의 문자열 객체이다. 따라서 이상 탐지에는 바로 적용할 수 없으므로 이를 나누어야 한다.

KafkaSpout은 다음 볼트인 ADParserBolt로 튜플을 전달한다. ADParserBolt는 하나의 문자열 객체인 로그 데이터를 이상 탐지에 사용할 수 있도록 파싱(parsing)한다. 앞서 로그 데이터는 [{클러스터 명}, {호스트 명}, {로그 키}, {로그 값}, {로깅 시간}]의 형태로 정의하였다. 따라서 이를 파싱하기 위해, 문자열을 콤마(,)로 나누었고 다섯 개의 값을 순서대로 다음 볼트인 ADGroupingBolt에게 전달한다.

본 시스템에서는 이상 탐지를 위해 이동 평균 및 3-시그마를 사용한다. 따라서 이동 평균과 이동 표준편차를 계산하여야 하는데, 이를 위해 로그 데이터를 윈도우 단위로 나누어야 한다. 현재 Storm에서는 튜플의 윈도우 연산을 위해 IWindowedBolt를 지원한다. 하지만, 이 방법으로 다음 볼트에게 윈도우를 전달하게 되면 어떠한 그룹핑 방법으로도 데이터가 뒤섞이는 것을 방지할 수 없다. ADGroupingBolt는 이러한 문제를 해결하기 위해 정의한 볼트이다. 이 볼트에서는 전달 받은 튜플로부터 클러스터 명, 호스트 명, 로그 키를 기준으로 맵을 구성하고, 로그 값을 리스트 형태로 저장한다. 이때, 로그 값의 리스트는 미리 정의된 윈도우 크기를 넘어갈 경우 가장 오래된 로그 값은 삭제한다. 이를 통해 로그 데이터는 클러스터 명, 호스트 명, 로그 키로 구분된 윈도우가 만들어지며, 이 윈도우는 다음 볼트인 ADMovAvgStdBolt로 전달된다.

ADMovAvgStdBolt는 이상 탐지를 위해서 이동 평균과 이동 표준편차를 계산한다. 이 볼트에서는 앞서 전달받은 윈도우에 저장된 로그 값을 사용하여 평균과 표준편차를 계산한다. 그리고 그 결과를 다음 볼트인 AD3SigmaBolt에게 전달한다.

마지막 볼트인 AD3SigmaBolt는 전달받은 이동 평균과 이동 표준편차로 3-시그마 밴드를 계산하고 로그 값이 이 밴드를 벗어나는 이상치인지 판단한 후, 그 결과를 데이터베이스에 저장한다.

## 시각화를 위한 웹 및 데이터베이스

본 시스템에서는 이상 탐지의 결과를 저장하고 시각화하기 위해 웹 및 데이터베이스 서버도 함께 구현하였다. 먼저, 데이터베이스는 오픈 소스 프로그램 관계형 데이터베이스인 MariaDB를 사용하였다. 그림 5는 데이터베이스에 이상 탐지 결과를 저장하기 위한 테이블 스키마이다. 테이블은 여덟 개의 애트리뷰트(attribute)로 정의하였으며, 각각은 순서대로 클러스터 명, 호스트 명, 로그 키, 로그 값, 3-시그마의 상한, 3-시그마의 하한, 이상 유무, 로깅 시간이다.



그림 5. 이상 탐지 결과가 저장될 데이터베이스의 테이블 스키마.

다음으로 이상 탐지 결과를 실시간으로 시각화하기 위한 웹 서버에 대해 설명한다. 이상 탐지의 결과를 그래프 형태로 나타내기 위해 자바스크립트 기반의 오픈 소스 시각화 도구인 d3.js를 사용하였다. [그림 x]는 시각화를 위해 구현한 웹 화면이다. 상단의 체크 버튼은 시각화할 대상을 정하며, 데이터베이스로부터 불러온 클러스터 명, 호스트 명, 로그 키이다. 사용자가 원하는 로그 키를 선택한 후, apply 버튼을 클릭하면 아래에 실시간 그래프가 나타난다. 이때, 버튼을 클릭하면 데이터베이스로부터 10분 간의 이상 탐지 결과를 읽어 들여 그래프에 반영한다. 또한 5초가 지나면 새로 분석된 이상 탐지 결과를 읽어 들여 그래프에 반영함으로써 실시간으로 그래프를 나타낸다.

# 결론

본 시스템은 다수의 서버로부터 로그 데이터를 수집하고, 수집된 로그 데이터로부터 이동 평균 및 3-시그마 기반의 이상 탐지를 수행한다. 그리고 이상 탐지의 결과를 데이터베이스에 저장한 후, 웹 화면으로 시각화하였다. 본 시스템을 사용하면 다수의 서버에서 발생하는 대용량의 로그 데이터를 실시간으로 이상 탐지하여 시각화할 수 있다.