다양한 스트리밍 데이터의 실시간 처리를 위한 프레임워크 비교

김대광, 권영우
경북대학교
kdk0811@knu.ac.kr, ywkwon@knu.ac.kr

A Comparison of frameworks for real-time processing of various streaming data

Dae Gwang Kim, Young-Woo Kwon
Kyungpook National University

요 약

IoT 디바이스 센서와 금융 거래 정보 등은 실시간으로 여러 소스에서 대량의 스트리밍 데이터를 지속적으로 생성한다. 이러한 스트리밍 데이터는 발생하는 상황에 빠르게 대응하기 위하여 스트림 처리 플랫폼에서 실시간으로 처리된다. 스트림 처리를 위한 다양한 프레임워크들이 존재하고 있으며 목적에 따라 선택되어 사용되고 있다. Apache Kafka, Apache Spark, Apache Flink 및 Apache Storm 등과 같은 대표적인 오픈소스 플랫폼들이 존재한다. 따라서 본 논문에서는 스트림 처리를 위한 다양한 프레임워크들을 비교하고자 한다.

Ⅰ. 서론

스트리밍 데이터는 여러 데이터소스로부터 연속적으로 생성되는 데이터 레코드 형태로 동시에 전송된다. IoT 센서 데이터나 금융 거래 정보 및 서버 로그 등과 같이 지속적으로 전송되는 스트리밍 데이터는 실시간으로 처리 및 분석 되어야하기 때문에 시간에 민감한 데이터이다. 따라서 지속적으로 생성되는 데이터 스트림을 실시간으로 분석함으로써 여러 상황에 적절하게 대응할 수 있어야 한다[1]. 이와 같이 실시간 스트리밍 데이터 처리를 위해서 다양한 프레임워크들이 사용되고 있는데 Apache Kafka, Spark, Flink, Storm[2-4] 등과 같은 오픈소스 기반의 스트림 처리 프레임워크가 있으며 지원하는 언어, 지원하는 연결 소스, 확장성 등의 사용 기준에 따라 선택해서 사용할 수 있다. 본 논문에서는 스트림 처리 프레임워크들을 분석하고 비교를 위한 실험을 설계한다.

Ⅱ. 본론

2.1 Spark Streaming과 Kafka Streams

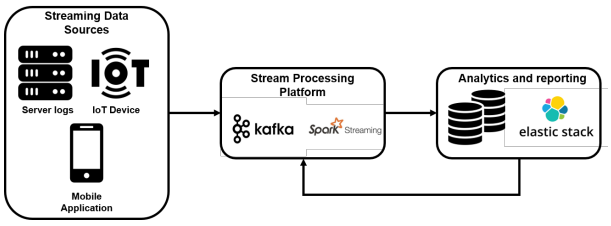

그림1. 스트림 처리 아키텍처

애플리케이션에서 Spark API의 확장 기술인 Spark Streaming을 이용하여 실시간 데이터 처리가 가능하다. 그림1은 스트림 처리 아키텍처를 나타내고 있다. Kafka와 같은 소스로부터 데이터를 가져오면 map, join 혹은 windowing 같은 알고리즘을 이용해서 데이터를 처리할 수 있다. Spark Streaming은 데이터 소스로부터 실시간으로 데이터 스트림을 받아서 Spark Core의 스케줄링을 이용해 작업(Job)을 처리하며 기본적으로는 FIFO 방식을 사용해서 작업을 수행한다.

그림2. Kafka Streams의 Processor Topology

그림2는 Kafka Streams의 스트림 처리 로직을 나타내고 있는데, 먼저 논리적인 DAG(Directed Acyclic Graph) 구조로 표현하는데, 세 가지 Processor(Source Processor, Stream Processor, Sink Processor)로 구성되어 있으며 Source Processor는 Topology의 시작 노드로 Kafka와 연결된 Processor이며 Kafka 토픽에서 데이터를 가져오는 역할을 한다. Stream Processor는 Source Processor, Stream Processor에서 반환하는 데이터를 처리하는 역할을 한다. Sink Processor는 Topology의 마지막 노드로 처리된 데이터를 Kafka의 토픽으로 전달하는 역할을 한다. 아래 표1은 Spark Streaming과 Kafka Streams의 특징들을 비교하고 있다.

표1. Spark Streaming과 Kafka Streams 비교

	Spark Streaming	Kafka Streams
Processing model	micro-batch	event-at-a-time
Latency	High	Low
Reliability	Exactly-once	At-least-once
Resource management	YARN, MESOS, etc.	
Windowing	tumbling, sliding window	tumbling, sliding, hopping, session window

2.2 지연 시간 및 처리율

애플리케이션에서 발생하는 이벤트를 실시간으로 처리하기 위해 Spark streaming, Kafka Streams 등의 스트림 처리 플랫폼이 사용되고 있다. 실시간 데이터 처리를 위한 방식 중 데이터 프로세싱 모델로는 데이터가 도착하는 대로 처리하는 이벤트 스트림 처리 방식(event-at-a-time)과 짧은

시간 단위 동안 데이터를 모아서 배치 프로세스를 처리하는 마이크로-배치 처리 방식이 있다. 이벤트 스트림 처리는 한 번에 하나의 데이터 요소를 처리하는 방식으로 동작하며 지속적인 데이터의 흐름을 처리한다. 마이크로-배치 처리 방식은 특정 주기를 가지고 데이터를 처리하기 위해 데이터를 작은 배치 단위로 묶어서 수집 후 처리하는 방식으로 사용되고 있다. Spark Streaming은 데이터를 메모리에 적재하여 사용하는 인-메모리 컴퓨팅 기반의 마이크로-배치 처리 방식을 사용하고 있다. 경우에 따라서 마이크로-배치는 충분히 실시간 처리로 사용되었다. 그러나 더 빠른 시간의 데이터 스트림 처리 요구에 대응하기 위해 Spark Streaming과는 달리 Kafka Streams는 데이터 이벤트가 생성되는 즉시 처리되는 이벤트 스트림 처리 방식을 사용하고 있다. 따라서 마이크로-배치 처리의 경우 상대적으로 긴 지연 시간을 소모하여 높은 처리율을 얻지만 Kafka Streams에서는 짧은 지연 시간만으로도 높은 처리율을 달성할 수 있는 처리 방식의 이점을 가진다.

2.3 전송 신뢰성

Spark Streaming은 메시지를 정확히 한 번(Exactly-once) 전송하는 방식을 사용하여 메시지 중복이나 누락이 없이 전송할 수 있다. Kafka Streams는 기본적으로 ACK와 오프셋 커밋을 이용해서 최소 한 번(At-least-once) 메시지 처리를 보장하는데 브로커가 메시지 수신 후 ACK 응답을 보내지 않으면 재전송하게 되고, 컨슈머에서는 메시지를 수신하고 오프셋을 확인해 메시지를 어디까지 수신했는지 확인할 수 있다.

2.4 자원 관리

Spark Streaming은 실시간 처리를 위해서 Spark 클러스터 내에서 애플리케이션이 계속 실행중이어야 하고 리소스를 관리, 할당하고 작업을 스케줄링 하는 등의 클러스터 관리자가 필요하다. 반면 Kafka Streams는 자바 라이브러리이기 때문에 의존성만 추가해주면 사용 가능하고 스트림 처리를 위한 별도의 클러스터 관리자도 필요하지 않다.

2.5 성능 비교를 위한 실험 설계

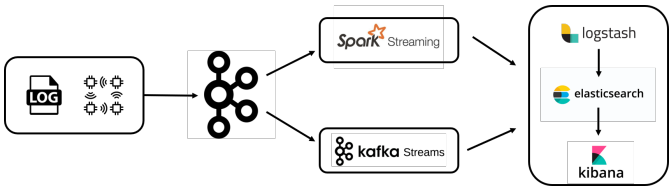

그림3. Spark Streaming과 Kafka Streams 실험 환경 설계

Spark streaming과 Kafka streams의 스트림 처리 성능을 비교하기 위하여 실험을 설계하였고 그림3은 이를 도식화하고 있다. 성능 측정을 위해 Docker 환경에서 Spark와 Kafka 클러스터를 각각 구축였으며 입력 데이터로 반정형 및 비정형 데이터를 사용하였다. 반정형 데이터로 IoT 센서 데이터를 사용하였는데, IoT 센서 데이터들의 상태 이벤트 등은 주기적이고 반복적으로 수집되는 경우가 많고, 비정형 데이터로 사용한 웹 로그와 같은 데이터들은 비주기적으로 수집된다.

다음 그림4와 5는 반정형 데이터의 예시인 IoT센서 데이터와 비정형 데이터의 예시인 웹 로그 데이터를 보여준다. 본 실험에서는 전송 속도, 지연 시간, 시간당 처리된 레코드 수 등을 비교분석 한다.

Kafka Streams는 IoT 센서 데이터와 같은 반정형 데이터를 키-값 형태로하여 센서ID를 키로, 읽은 데이터를 값으로 나타낼 수 있다. 그런 다음 Streams API를 사용하여 필터링, 변환 및 집계와 같은 작업을 수행할 수 있다. Spark streaming에서는 반정형의 IoT 센서 데이터를 처리하기 위해 RDD 혹은 Dataframes와 같이 추상화하여 데이터 변환 및 집계 등의 처리를 지원한다. Kafka Streams는 웹 로그와 같은 비정형 데이터를 Avro, JSON과 같은 데이터 형식으로 직렬화되어 Streams API에서 처리되고 추가 분석을 위해 원래 형식으로 역직렬화될 수 있다. Spark Streaming 또한 마찬가지로 비정형 데이터를 Avro 및 JSON과 같은 구조화된 형식으로 변환하고 필터링, 매핑 및 조인과 같은 연산으로 데이터를 처리할 수 있다.

```
1 "ts","device","co","humidity","light","lpg","motion","smoke","temp"
2 "1.5945120943859746E9","b8:27:eb:bf:9d:51","0.004955938648391245","51.0","false","0
```

그림4. IoT 센서 데이터의 예시 (반정형 데이터)

```
1 "ts","device","co","humidity","light","lpg","motion","smoke","temp"
2 "1.5945120943859746E9","b8:27:eb:bf:9d:51","0.004955938648391245","51.0","false","0

44.110.205.198 - - [2018-07-29T14:09:55.337Z] "GET /elasticsearch HTTP/1.1" 200 2162 "-" "Moz
illa/5.0 (X11; Linux i686) AppleWebKit/534.24 (KHTML, like Gecko) Chrome/11.0.696.50 Safari/5
34.24"
```

그림5. 웹 로그 데이터의 예시 (비정형 데이터)

Kafka Streams와 Spark Streaming 두 플랫폼 모두 정형, 반정형, 비정형 데이터를 처리하는데 사용될 수 있지만 Spark Streaming은 기계 학습과 그래프 처리 등의 알고리즘을 지원해 복잡한 데이터 처리에 사용하고, Kafka Streams는 인-메모리 처리 방식과 연속 스트림으로 데이터를 처리해 짧은 지연 시간을 갖는 특징 때문에 시간에 민감한 데이터를 처리하는데 사용한다.

Ⅲ. 결론

본 논문에서는 스트림 처리를 위한 플랫폼의 특징을 비교하였고 추후 성능 비교를 위한 실험을 설계하였다. 스트림 처리의 최적화에 관한 토폴로지 구성과 스케줄링 기법 등의 연구가 존재한다. 향후 연구에서 두 플랫폼의 성능 비교뿐만 아니라 토폴로지 및 스케줄러 최적화를 통한 성능 향상 연구를 진행하고자 한다.

ACKNOWLEDGMENT

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 한국연구재단 지원을 받아 수행된 연구임(NRF-2021R1A5A1021944).

참 고 문 헌

[1] N.Marz and J. Warren, Big Data: Principles and Best Practices of Scalable Realtime Data Systems (Part3 Speed Layer), 2015.

[2] Apache Kafka, https://kafka.apache.org/

[3] Apache Spark, https://spark.apache.org/

[4] Apache Flink, https://flink.apache.org/