Apache Kafka에서 효율적인 과부하 측정을 위한 모니터링 도구

방지원, 손시운, 문양세, 최미정
강원대학교 컴퓨터과학과
e-mail: {jiwonbang, ssw5176, ysmoon, mjchoi}@kangwon.ac.kr

Monitoring Tools for Efficient Overload Measurements in Apache Kafka

Jiwon Bang, Siwoon Son, Yang-Sae Moon, Mi-Jung Choi
Dept. of Computer Science, Kangwon National University

요 약

실시간으로 빠르게 발생하는 대용량 데이터를 다루기 위해 Apache Storm, Apache Spark 등 실시간 데이터 스트림 처리 기술에 대한 연구가 활발하다. 대부분의 실시간 처리 기술들은 단독으로 사용하기에 어려움이 있으며, 데이터 스트림의 입출력을 위해 메시징 시스템과 함께 사용하는 것이 일반적이다. Apache Kafka는 대표적인 분산 메시징 시스템으로써, 실시간으로 발생하는 대용량의 로그 데이터를 전달하는데 특화된 시스템이다. 현재 Kafka를 위한 다양한 성능 모니터링 도구들이 존재한다. 이러한 모니터링 도구들은 Kafka에서 처리되는 데이터의 양 이외에도 유입 데이터의 크기, 수집 속도, 처리 속도 등 다양한 데이터들을 관찰할 수 있다. 본 논문은 Kafka에서 제공하는 도구와 오픈 소스로 제공되는 여러 개의 도구들을 비교하여, 향후 Kafka의 로드 쉐딩에 대한 연구에 적용할 수 있는 최적의 모니터링 도구를 선별하고자 한다.

1. 서론[1]

최근 사물 인터넷(IoT)과 소셜 네트워크 서비스(SNS)의 실시간 데이터 발생량이 폭증함에 따라 빅데이터 처리에 대한 요구가 증가하였으며, 이에 발맞추어 빅데이터 기술들이 발전해왔다. 빅데이터 처리 기술들은 일반적으로 안정적인 데이터의 관리가 필요하여 메시징 시스템이 같이 사용되었다. 메시징 시스템은 큐(Queue)구조로, 발생하는 메시지를 임시로 저장해 두었다가 처리가 필요한 경우에만 큐에서 메시지를 꺼내어 작동 한다. 대표적인 메시징 시스템으로는 ActiveMQ[1], RabbitMQ[2], ZeroMQ[3], Apache Kafka[4] 등이 있다.

이 중, Apache Kafka의 경우 다른 메시징 시스템과 달리, 실시간 로그 처리에 특화되어 초당 처리량이 높고, 분산 환경을 사용하므로 확장이 쉽다는 장점이 있다. 그러나, 이렇게 특화된 Kafka도 장시간 동안 대용량의 데이터를 처리하게 되면 지연이 발생하게 된다. Kafka가 지연 없이 제대로 처리하고 있는지 확인하기 위한 다양한 모니터링 도구들이 존재한다.

본 논문에서는 Kafka에서 기본적으로 제공되는 JMX Interface를 통하여 MBeans(Managed Beans)[5]의 값들을 모니터링할 수 있는 여러 도구들을 소개하고, 향후 Kafka의 로드 쉐딩[6]에 적합한 모니터링 도구를 선별하고자 한다.

[1] 이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.R7117-17-0214, 데이터 스트림 정제를 위한 지능형 샘플링 및 필터링 기술 개발)

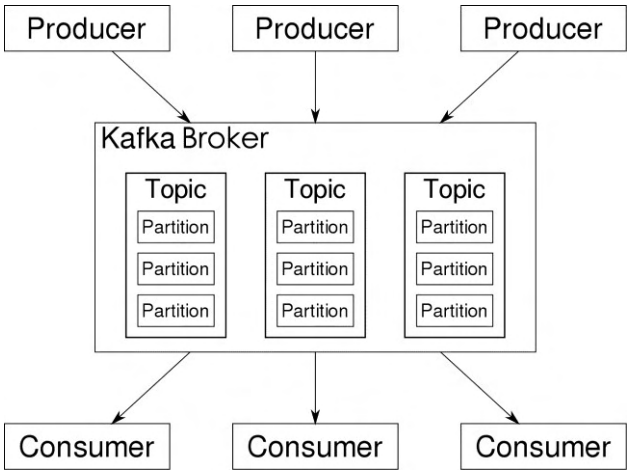
2. 관련 연구

Apache Kafka는 소셜 네트워크 서비스 회사인 Linkedin[7]에서 개발하였다. Linkedin은 비즈니스 중심의 소셜 네트워크 서비스 가입자가 급증하여 그에 따른 데이터 폭증으로 인한 자사의 데이터 처리 시스템의 한계를 겪고, 이를 보완하기 위해 Scala 기반의 대용량의 실시간 로그 처리 시스템을 개발하였다. Active-MQ, RabbitMQ 등의 기존 메시징 시스템은 단일 노드로 구성되어 동작하지만 Kafka는 분산 구조로 설계되어 확장(Scale Out)이 쉽다는 장점이 있다. 또한, 기존 시스템보다 높은 가용성(High Availability)을 제공하며, 메시지 처리 속도가 빠르다.

기존의 메시징 시스템은 메시지를 메모리에서 관리하지만, Kafka는 메시지를 파일 시스템에서 관리한다. 즉, 기존 메시지 시스템에서는 처리 되지 못하고 지연된 메시지의 수가 많을수록 메모리 자원을 많이 사용하므로 시스템의 성능을 크게 감소 시켰으나, Kafka는 디스크에 메시지를 저장하기 때문에 성능을 크게 향상시킬 수 있다. Kafka는 기존 메시징 시스템에 사용된 AMQP(Advanced Message Queueing Protocol)나

JMS API 기반의 메시지 헤더보다 단순하고 가벼운 헤더로 구성된 TCP 기반의 프로토콜을 사용하여 메시지의 크기를 확연히 줄였다[8].

그림 1은 Kafka의 구조[9]를 나타낸다. 그림 1에서 보듯이, Kafka는 발행-구독(Publish-Subscribe) 모델을 기반으로 동작하며 Producer, Broker, Consumer로 구성된다. Kafka를 이용하려면 먼저 Topic을 생성해야 하는데, Topic은 Kafka에서 각 메시지를 분류시켜주는 역할을 한다. Kafka의 동작 방식으로는 각각의 Producer가 Broker에게 메시지를 발행(Publish)하면, 각 Broker들은 큐에 Topic 별로 메시지를 구분해 저장한다. 해당 Topic을 구독(Subscribe)하는 Consumer가 필요한 만큼의 메시지를 가져와서 처리한다.



(그림 1) Apache Kafka의 구조.

3. 모니터링 도구

실시간 데이터 스트림 처리에 특화된 Kafka에서도 장시간 동안 대용량의 데이터를 처리함에 따라 지연이 발생하는 것은 필연적이다. 정상적인 성능을 발휘하기 위해서는 시스템을 모니터링할 수 있는 도구들이 필요하다. 본 장에서는 대표적인 Kafka 모니터링 도구들을 비교한다.

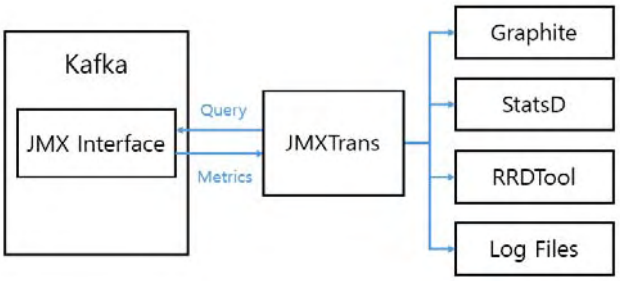
3.1 JConsole

JDK(Java Development Kit)에 포함 되어있는 JConsole은 간단하고 빠르게 정보를 제공해줄 수 있는 GUI 형식의 모니터링 도구이다. Kafka에서 JMX Interface를 통하여 제공 할 수 있는 모든 시스템 자원을 수집하여 보여준다. 사용 방법은 Kafka에서 기본적으로 지정되어있는 JMX의 포트번호와 IP 주소로 원격 접속하여 직접 모니터링하는 것이다.

3.2 JMX Reporter를 이용한 모니터링 도구

기본적으로 제공되는 JConsole은 빠르게 정보를 제공해 주는 도구지만, 장기적인 패턴의 변화나 패턴 규모가 커질 경우 모니터링하기 힘들고, 또한 여러 정보들을 한번에 보거나 원하는 정보만을 보기가 어렵다. 이를 해결하기 위하여 대부분의 모니터링 서비스와 도구들이 JMX Reporter를 이용하여 시스템 자원을 수집한다. 대표적으로 JMXTrans[10], Kafka-Ganglia[11] 등이 사용한다.

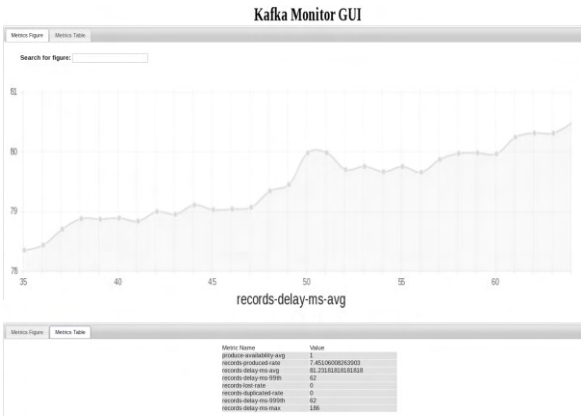
JMXTrans는 Kafka 외에도 다른 JVM에서 동작하는 프로세스를 기록하거나 모니터링하는 커넥터 역할을 하는 도구이다. JSON(또는 YAML) 형식의 파일을 이용하여 원하는 시스템 자원의 정보를 수집할 수 있다. JMXTrans은 웹으로 정보를 전송, 그래프형식으로 원하는 정보를 출력해주는 Graphite, StatsD 등의 도구들과 연동하는 기능을 제공한다. 또한, 실시간으로 시스템 자원을 수집할 수 있게 로그 파일로 저장할 수 있는 기능도 제공한다. Kafka-Ganglia는 Kafka와 고성능 컴퓨팅 시스템을 모니터링 하기 위한 분산 모니터링 시스템인 Ganglia를 연동 서비스 제공해주는 Plugin이다. 그림 2는 JMXTrans의 구성을 나타낸다.



(그림 2) JMXTrans의 구성.

3.3 Linkedin Kafka Monitor

Kafka를 개발한 Linkedin 회사에서 직접 제작한 모니터링 도구로써, Kafka가 적용된 시스템은 장시간 시스템이 실행되어있는 상태이기 때문에, 낮은 확률로 발생할 수 있는 버그나 오류를 포착하여 신속히 보완하기 위해 개발되었다. 또한 시스템에서 지속적인 메시지 송수신이 진행되면서 발생할 수 있는 성능 저하를 파악하여 할당량 분배, 권한 이전 등의 정상적인 작업이 진행되는지 테스트도 진행할 수 있다. 향후 디스크 오류, 데이터 손상과 같은 실패 시나리오 테스트와 Graphite 같은 유사한 모니터링 도구의 통합계획이 있다. 그림 3은 Web UI로 표시한 Kafka Monitor이다[12].



(그림 3) Linkedin Kafka Monitor GUI.

4. 결론 및 향후 연구

본 논문에서는 Apache Kafka의 다양한 모니터링 도구들 중에서 대표적인 도구들을 소개하고 비교하였다. JConsole은 원하는 정보만을 가져오기 힘들고, Kafka Monitor의 경우 아직 개발 된지 얼마 되지 않아서 정보를 추출하는 기능이 제공되지 않는다. 이 외에도 조사한 모니터링 도구 비교 결과, JMXTrans가 Kafka의 성능 모니터링에 적합하다고 판단하였다. JMXTrans를 이용하면 원하는 정보를 JSON 형식으로 간편하게 작성하여 지속적인 시스템 자원정보 수집이 용이하기 때문이다. 향후 연구로는 JMXTrans에서 수집한 모니터링 정보를 기반으로 Kafka의 로드 쉐딩을 구현할 예정이다.

참고문헌

[1] Apache ActiveMQ, http://activemq.apache.org/.
[2] Pivotal RabbitMQ, https://www.rabbitmq.com/.
[3] ZeroMQ, https://www.zeromq.org/.
[4] Apache Kafka, http://kafka.apache.org/.
[5] MBeans,https://docs.oracle.com/javase/tutorial/jmx/mbeans/index.html.
[6] Load Shedding, Tatbul, Nesime, et al. "Load Shedding in a Data Stream Manager." *VLDB*. 2003.
[7] Linkedin, https://www.linkedin.com/.
[8] J. Kreps, N. Narkhede, and R. Jun, “Kafka: a Distributed Messaging System for Log Processing,” In *Proc. of the NetDB*, Athens, Greece, pp. 1-7, June 2011.
[9] Structure of Kafka, https://en.wikipedia.org/wiki/Apache_Kafka.
[10] JMXTrans, http://www.jmxrans.org/.
[11] Kafka-Ganglia, http://crazyadmins.com/kafka-integration-with-ganglia/.
[12] Linkedin Kafka Offset Monitor, https://engineering.Linkedin.com/blog/2016/05/open-sourcing-kafka-monitor.