Apache Kafka

미들웨어운영팀 박재연 사원

목차

1. Apache Software Foundation
2. 개요
3. 아파치 라이선스
4. 아파치의 문화
5. 아파치 프로젝트
6. Apache Kafka
7. 유래 및 목표
8. 주요 인원 소개
9. 카프카 구성 요소 및 구조
10. 활용 방안
11. Apache Software Foundation ([www.apache.org](http://www.apache.org))
12. 개요

* 다양한 오픈소스 프로젝트를 지원하고 관리하는 비영리 재단.
* 1999년, 아파치 그룹과 델라웨어사의 합병으로 설립.
* 아파치 HTTPD 웹서버 개발 당시 기술 개선을 위해 메일링 리스트를 만들고, 많은 기여자와 협업한 것에서 아파치 소프트웨어 재단으로 발전.
* 프로젝트의 기술적 부분 외에도 법률적, 경제적 지원을 하고 있음.

1. 아파치 라이선스

* 아파치 재단이나 재단 내 프로젝트에 의해 제작된 소프트웨어는 아파치 라이선스 2.0에 의해 배포.
* 누구나 해당 소프트웨어에서 파생된 프로그램을 제작할 수 있으며, 저작권을 양도, 전송할 수 있는 라이선스.
* 아파치 소프트웨어를 다운 받아 부분/전체를 개인적/상업적 목적으로 이용할 수 있고, 재 배포시 아파치 라이선스를 포함시키고, 아파치 소프트웨어 재단에 의해 개발된 소프트웨어임을 밝혀야 함.

1. 아파치의 문화

* 아파치의 6가지 원칙
  + 소프트웨어 협업 개발
  + 상업적으로 활용할 수 있는 표준 라이선스
  + 고품질이 유지되는 소프트웨어
  + 서로 존경하며 정직하게 기술 교류를 할 것
  + 표준에 충실할 것
  + 보안 기능에 충실할 것
* 아파치 참여자
  + Apache Committer: 7032명 (6693명 활동 중)
  + ASF 멤버: 730명
  + 신임 멤버: 44명

|  |  |
| --- | --- |
| 분류 | 설명 |
| User | 아파치 소스를 사용하는 참여자 |
| Contributor | 자신의 아이디어나 소스를 아파치에 기여하는 참여자 |
| Committer | 소스의 반영 여부를 결정하는 참여자 |
| PMC | 프로젝트의 Leader |

1. 아파치 프로젝트

* 인큐베이터 프로젝트: 아파치 소프트웨어에 응모하여 일정 조건을 충족한 오픈 소스는 아파치의 명칭을 사용할 수 있으며, 실험 프로젝트로서 인큐베이터로 등록되고, 지원을 받을 수 있음.
* 탑 레벨 프로젝트: 전용 PMC 위원회가 조직되고, 아파치 정식 프로젝트로 운영됨.
* 숫자로 보는 2018 아파치에 따르면, 프로젝트와 하부 프로젝트는 총 328개.
* 그 중 198개가 탑 레벨 프로젝트로서 운영되고 있음.



그림 1. 인큐베이팅 프로젝트 발전

* 인큐베이터에서 Podling되고 있는 프로젝트는 2019년 4월 기준, 53개.
* 카테고리별로 보았을 때, 라이브러리, 빅데이터, 네트워크-서비스 순으로 많은 프로젝트가 관리되고 있음.
* 대표적인 아파치 소프트웨어 재단 프로젝트
  + Apache Lucene

자바 언어로 이루어진 정보 검색 라이브러리로, 추가적인 개발을 통해 자바 언어 이외에도 펄, C#, C++, 파이썬 등 다른 프로그래밍 언어를 사용할 수 있도록 변경. 이메일 검색, 온라인 문서 검색, 웹 페이지 검색, 버전 관리 및 콘텐츠 관리 등에 적용이 가능함.

* + Apache Hadoop

빅데이터를 저장, 처리, 분석할 수 있는 소프트웨어 프레임워크로 Apache Lucene의 하부 프로젝트. 분산처리 시스템인 구글 파일 시스템을 대체할 수 있는 하둡 분산 파일 시스템(HDFS)과 맵리듀스를 구현. 야후, 페이스북 등 대용량 데이터를 처리해야 하는 웹 사이트나, 데이터 분석 기업들이 하둡을 기반으로 플랫폼을 구축.

* + Apache Spark

분산된 메모리상의 데이터 처리 시스템으로, 자바, 스칼라, 파이썬에 대해 API 세트를 제공. 실시간 데이터 처리와 분석, 머신 러닝, 단일 클러스터 플랫폼 상의 거대한 양의 데이터 그래프 처리 등을 위한 어플리케이션 개발에 사용될 수 있음.

1. Apache Kafka (<https://kafka.apache.org/>)
2. 유래 및 목표

* 2011년, LinkedIn에서 개발된 **비동기 방식 분산 메시징 시스템**으로,   
  2014년에 아파치를 통해 오픈 소스화.
* LinkedIn 사내에서 포털의 사용자 활동 데이터를 수집하고, 사용자에게 적절한 정보를 제공.
* 기존 시스템의 경우, 통합된 전송 영역이 없기 때문에 복잡도가 증가하고,  
  파이프라인 관리가 어렵다는 단점이 있음.
* 카프카의 제작 목표
  + 메시지 프로듀서와 컨슈머 사이의 느슨한 연계
  + 다양한 형태의 데이터 사용 시나리오와 장애 처리 지원 위한 메시지 데이터 유지
  + 빠른 처리 시간을 지원하는 구성 요소로 시스템의 전반적 처리량 최대화
  + 이진 데이터 형식을 사용하여 다양한 데이터 형식과 유형 관리
  + 기존 클러스터 구성에 영향을 주지 않고 일정한 서버의 확장성 지원
* 기존 MOM(Message Oriented Middleware) 시스템인 ActiveMQ, RabbitMQ와 비교하였을 때, **대용량 실시간 로그 처리에 특화**되어 있는 시스템.

1. 주요 인원 소개(<https://kafka.apache.org/committers>)

Committer: 총 26명 (PMC 멤버: 14명)

주요 참여자

|  |  |  |
| --- | --- | --- |
| Confluent co-founders Jun Rao, Jay Kreps and Neha Narkhede  (Paul Sakuma Photography)  그림 2. Kafka 주요 개발진 | Jun Rao | LinkedIn Senior Staff SE  카산드라, 카프카 커미터  컨플루언트 공동 창업자 |
| Jay Kreps | NexTag Senior SE  LinkedIn Principal Staff Engineer  컨플루언트 CEO |
| Neha Narkhede | Oracle 기술 조직 회원  LinkedIn Principal SE  컨플루언트 공동 창업자, CTO |

1. 카프카 특징

* 고가용성 및 확장성

클러스터로 작동하기 때문에 장애 허용 시스템(시스템 구성의 일부에서 장애가 발생해도 정상적으로 기능하는 시스템)으로서 고가용성 서비스를 제공할 수 있으며, 서버를 수평적으로 늘려 안정성/성능을 향상하는 스케일 아웃이 가능.

* 디스크 순차 저장 및 처리

메시지를 메모리 큐에 저장하는 기존 메시지 시스템과 다르게 메시지를 디스크에 순차 저장. 이로서 서버에 장애가 나더라도 메시지가 디스크에 저장되어 있으므로 유실의 위험성이 감소. 디스크가 순차적으로 저장되어 있어 디스크 I/O가 감소하여 성능 향상.

* Pub/Sub 메시징 시스템

\* P2P 메시징 모델과 Pub/Sub 메시징 모델의 차이

|  |  |
| --- | --- |
| 그림 3. P2P 메시징 모델 | 그림 4. Pub/Sub 메시징 모델 |
| 한 지점에서 다른 지점으로 메시지 전송.  하나 이상의 리시버가 메시지를 읽을 수 있지만, 리시버가 읽은 메시지는 대기열에서 사라지므로, 특정 메시지는 오직 하나의 리시버만 읽을 수 있음.  대표적인 예) 주문 처리 시스템. | 하나의 메시지를 여러 컨슈머가 구독.  메시지를 읽으면서 큐에서 제거해 버리면 다른 컨슈머가 읽을 수 없으므로, 메시지는 큐에 잔존.  메시지가 늘어날 경우 저장공간이 부족하거나 데이터 처리의 효율이 떨어질 수 있으므로 일정 시간이 지난 메시지는 삭제.  대표적인 예) 뉴스 구독 시스템. |

1. 카프카 구성 요소 및 구조
   * Topic, Partition, Offset

카프카에 저장되는 메시지는 토픽으로 구분되며, 토픽은 여러개의 파티션으로 구분.

파티션 내 메시지 상대 위치를 나타내는 오프셋 정보를 통해 이전에 가져간 메시지 위치 정보를 알 수 있음.

동시에 들어오는 데이터를 여러개 파티션에 나누어 저장하므로 병렬로 처리 가능.

* + Producer, Consumer

프로듀서는 메시지를 생산, 작성하는 주체, 컨슈머는 메시지를 소비, 읽는 주체. 기존 메시징 시스템 (ActiveMQ, RabbitMQ)등에서 브로커가 컨슈머에게 메시지를 Push하는 방식이었으나, 카프카에서는 컨슈머가 브로커에서 메시지를 Pull하는 방식.

* + Broker, ZooKeeper

브로커는 카프카 서버를 칭하며, 동일 노드 내에 다수 브로커 서버를 올릴 수 있음.

주키퍼는 분산 메시지 큐 정보를 관리하는 역할로, 카프카 실행을 위해서는 주키퍼 실행이 선행되어야 함

* + Replication

카프카에서는 레플리케이션 수를 임의로 지정하여 토픽을 만들 수 있음.

브로커에 문제가 생겼을 경우 해당 브로커의 역할을 다른 브로커에서 즉각적으로 대신 수행할 수 있게 하기 위함.

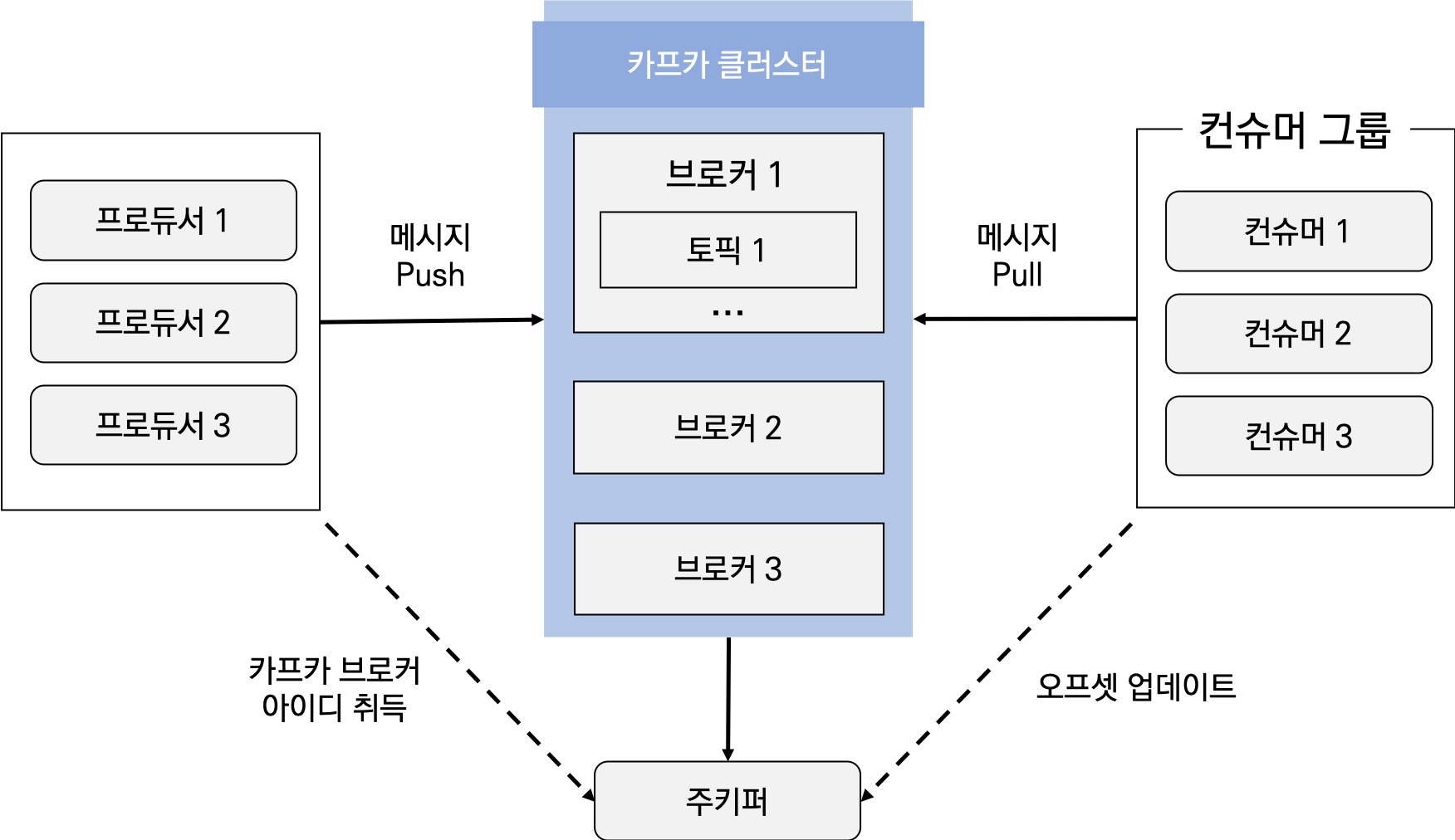


그림 5. Kafka 구조

1. 활용 방안

* 메시징 시스템

전통적인 메시지 브로커의 역할(MOM)을 수행할 수 있음.

기존 메시지 브로커보다 높은 처리량과 복제, 내결함성 등으로 대규모 메시지 처리 응용이 가능하다는 장점을 지님.

* 사용자의 웹사이트 활동 추적

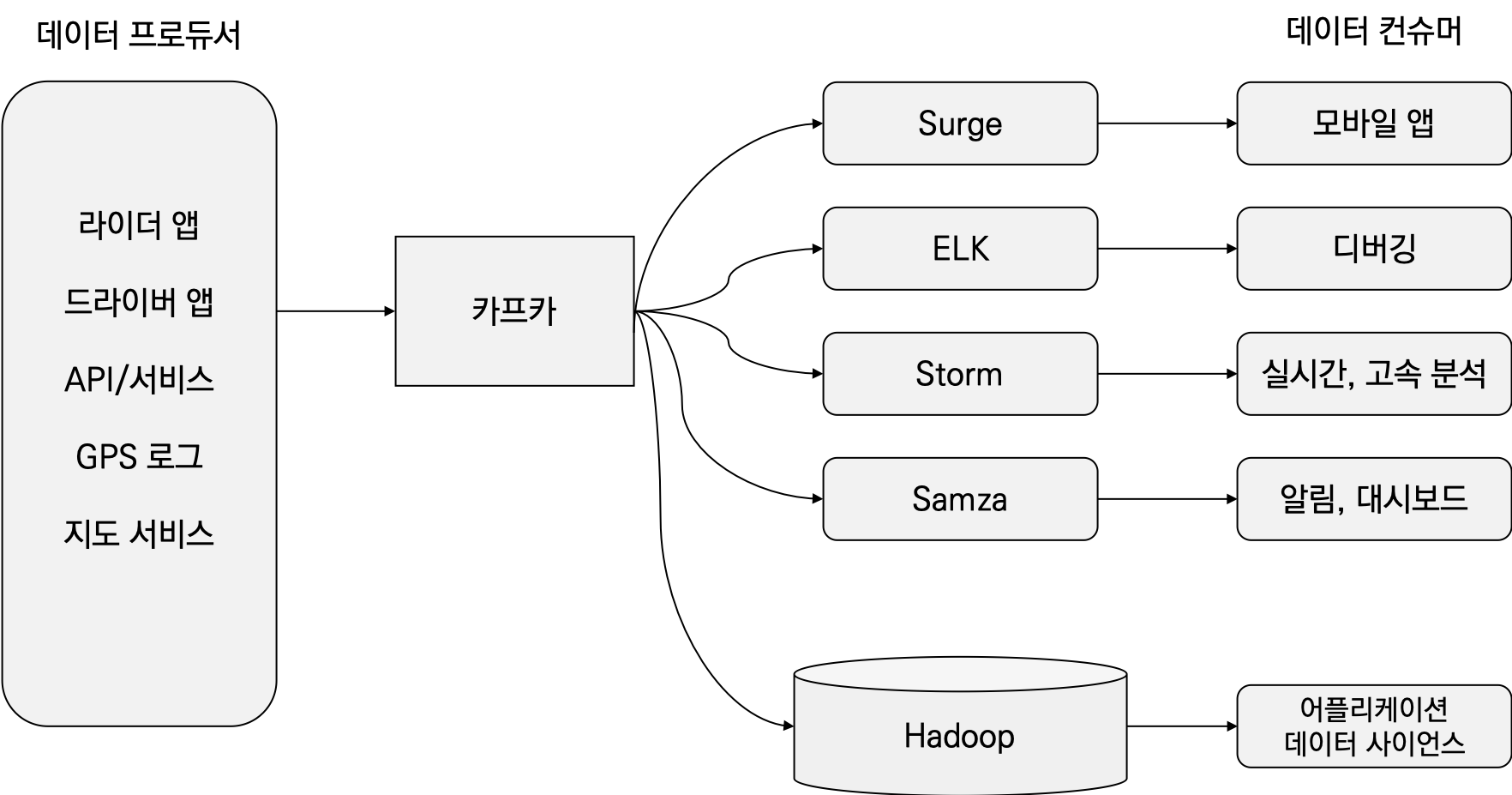
페이지 조회수, 검색, 사용자가 행할 수 있는 작업 등 웹사이트 활동을 추적.

실시간 모니터링, Hadoop을 통한 데이터 분석, 오프라인 데이터 웨어하우스 시스템으로의 로드 등 다양한 사례에 적용 가능.

* 로그 집계

일반적으로 물리적 로그 파일을 서버에서 수집하여 중앙 서버에 저장하여 처리하지만, 카프카는 로그 또는 이벤트 데이터를 메시지 스트림으로 추상화하여, 대기 시간이 낮아지고, 분산된 데이터의 소비를 쉽게 지원할 수 있음.

* 이 외에도 스트리밍 프로세싱, 운영 모니터링 등의 활용방안이 있음.
* 실제 활용 예시: 우버 (<https://eng.uber.com/ureplicator/>)



Uber에서는 Apache Kafka를 Ecosystem의 여러 부분을 연결하는 메시지 버스로 사용.

라이더 및 드라이버 앱에서 시스템 및 애플리케이션 로그, 이벤트 데이터 등을 수집.

Kafka를 통해 다양한 다운 스트림 컨슈머에게 이 데이터를 제공.

Kafka의 데이터는 실시간 파이프 라인과 배치 파이프 라인을 모두 지원하며, 실시간 파이프라인의 데이터는 컴퓨팅 비즈니스 측정, 디버깅, 알림 및 대시 보드와 같은 활동을 위한 것이고, 배치 파이프라인의 데이터는 하둡 등을 통한 데이터의 심층 분석을 위해 사용.