Apache Spark

개념

Apache Hadoop

- Apache Common
- Hadoop Distributed File System(HDFS)
  - 하둡 에코시스템에서 분산파일시스템을 지원하는 것을 목표로 한다.
  - 큰 파일들을 잘 저장하고 처리할 수 있도록 지원
  - Reliability를 위해서 HDFS는 copy본을 저장한다.

> HDFS
> - Good for : Large Files
> - Not Good for : Lots of small Files, Low latency access
> 디스크 I/O를 일으키는 작업이기 때문에 latency가 발생할 수 밖에 없음
> Spark는 이러한 부분을 메모리에 올려서 latency를 개선하였음

Master Node (Main Node) - 메타데이터를 다룬다.

Slave Node - 실제로 데이터를 저장하게 되는 데이터 노드들.

> 하둡 클러스터 데이터 노드의 수 - 스토리지 용량 및 컴퓨팅 파워를 의미

fault tolerance

- 데이터 노드들이 죽는 일을 방지하기 위해 안전하게 서비스 수행을 하기 위해 블럭을 카피.
- (특정 노드에 문제가 생겼을 때를 대비. 기본 설정은 3개의 파일을 가지도록 설정되어있음)
- (추가적으로, 블럭이 잘 분배되어 저장되어 있으면 병렬성이 높아진다)

HDFS는 블럭 단위로 나뉘게 된다.

크기가 어떠한 파일을 올리던 내부에서는 블럭 단위로 분산되어 저장됨

Apache Spark

목표 : 기존의 맵리듀스 프레임워크보다 클러스터 컴퓨팅 시스템에서 더욱 빠르게 컴퓨팅할 수 있다.

디스크 기준으로는 10배, In memory 방식으로는 100배의 성능차이를 낼 수 있다.

램이 싸지면서 메모리에서 처리할 수 있는 프레임워크가 많이 나오기 시작한 시기와 겹침

머신러닝 알고리즘에서는 반복적으로 데이터를 접근하여 연산을 해야하는데, 하둡 구조에서는 매번 요청이 있을때마다 디스크로 내려가야한다. 디스크 I/O를 모두 인메모리에서 처리하겠다는 것이 스파크의 컨셉이다.

빅데이터 프로세싱 프레임워크 중 가장 핫함.

스파크는 컴퓨터 클러스터에서 병렬 데이터 프로세싱을 하는 모든 라이브러리의 집합이며, 통합된 컴퓨팅 엔진이다.

Distributed Variables, RDDs는 Low-level API에 속한다. 다루는 데에 난이도가 있는 편.

스키마를 가지고있는 데이터(Structured Data)의 처리(Structured APIs) 에 있어서는 Datasets, DataFrame, SQL을 이용할 수 있다.

Spark에서 분산병렬 데이터 프로세싱에서 끝나는 것이 아니라 머신러닝 라이브러리도 제공을 하고 있다. 빅데이터 처리에서 머신러닝까지 이어질 수 있도록 지원하고 있음. 이러한 에코시스템이 다양하기 때문에 경쟁력이 있다.

Spark은 기본적으로 Scalar 언어로 개발이 되어있고 functional lang의 장점을 가진다.

Scalar, Java, python, R, sql 지원

코드량 자체도 상당히 간결화되어있고, 개발의 생산적 측면에서도 장점이 있다.

더 적은 노드를 가지고 sort를 더 빠르게 수행한다.

Big Data Processing

- Batch Processing

크고 복잡한 데이터의 summary table를 만드는 느낌

데이터는 크지만 latency를 가져도 되는 경우

(map-reduce가 활용될 수도 있고 spark batch processing이 활용될 수도 있다.)

- Stream Processing

데이터가 도착하자마자 처리 (간단하고 독립적으로 동작해야한다.)

실시간처리가 필요한경우 (sub-second latency)

Spark streaming은 in streaming processing이라기보다는 micro batch system과 비슷.

최종적으로 streaming processing에서는 sink를 시켜야한다.

in stream 방식에서는 source operator가 kafka라면 kafka에서 데이터를 가져온 다음 실시간으로 처리하고, 이것을 sink operator의 elastic search에 넣겠다.

micro batch 방식에서는 데이터가 들어오면, 각각의 인터페이스마다 receiver가 존재한다. (kafka라면 kafka receiver 존재) 그 안에서 kafka receiver가 존재, 그 안에서 streaming할 때 바로바로 데이터를 가져와서 처리하는 방식이 아니라, micro batching style로, 레코드가 작은 batch 스타일을 가지게 된다. (short batch) 그리고 그 작은 batch를 가져와서 sink operator에 넣는다.