Spark

1. Spark 설치 및 실행
    - Spark site download and install
    - databricks 사용
    - Docker 사용
    - code 예제
        - 원서 code 예제 : https://github.com/databricks/Spark-The-Definitive-Guide
        - 번역서 code 예제 : https://github.com/FVBros/Spark-The-Definitive-Guide
        - docker code 예제 : https://hub.docker.com/r/rheor108/spark_the_definitive_guide_practice

2. Apache Spark란
    - 아파치 스파크는 통합 컴퓨팅 엔진
    - 클러스터 환경에서 데이터를 병렬로 처리하는 라이브러리 집합
    - 빛처럼 빠른 통합 분석 엔진
    - Hadoop과 비교하면 100배 빠르다.
    - Hadoop은 disk에서의 분산 데이터 처리
    - Spark는 메모리에서의 분산 데이터 처리
    - Hadoop/Spark 둘다 병렬처리 구조

3. Apache Spark의 철학
    - 통합
        - 빅데이터 애플리케이션 개발에 필요한 통합 플랫폼을 제공하자.
        - 간단한 데이터 읽기, SQL, 머신러닝, 스트림 처리를 API로 처리
        - 필요할 경우 Spark기반의 라이브러리를 만들수 있음
        - Spark API의 경우에는 사용자 애플리케이션에서 다른 라이브러리의 기능을 조합해 더 나은 성능을 발휘할 수 있도록 설계되어 있다.
            1. 예) SQL 쿼리로 데이터를 읽고 ML 라이브러리로 머신러닝 모델을 평가해야 할 경우,
               Spark 엔진은 이 두 단계를 하나로 병합하고 데이터를 한번만 조회할 수 있게 해준다.
    - 컴퓨팅 엔진
        - 스파크는 저장소 시스템의 데이터를 연산하는 역할만 수행할 뿐 영구 저장소 역할은 수행하지 않는다.
        - 대신, 분산처리 파일 시스템인 아파치 하둡(Apache Hadoop), 클라우드 기반의 애저 스토리지(Azure Storage), 아마존 S3(Amazon S3), 키/값 저장소인 아파치 카산드라(Apache Cassandra), 메시지 서비스인 아파치 카프카(Apache Kafka)등의 저장소 지원
        - 스파크는 데이터 저장 위치에 상관없이 처리에 집중하도록 만들어짐.
        - 스파크는 하둡 저장소와 잘 호환된다.
        - 하둡 아키텍처를 사용할 수 없는 환경에서도 많이 사용된다.
    - 라이브러리
        - 스파크 컴포넌트는 데이터 분석 작업에 필요한 통합 API를 제공하는 통합 엔진 기반의 자체 라이브러리 이다.
        - 표준 라이브러리 + 외부 패키지

        1. 표준라이브러리 : 스파크 엔진에서 제공
        2. 외부 패키지 : 오픈소스 커뮤니티에서 서드파티 패키지 형태로 제공하는 라이브러리
    - 표준 라이브러리
        1. Spark SQL : SQL과 구조화된 데이터를 제공
        2. 머신러닝을 지원하는 MLlib
        3. 스트림 처리 기능을 제공하는 스파크 스트리밍 과 새롭게 선보인 구조적 스트리밍
        4. 그래프 분석엔진인 GraphX 라이브러리를 제공
    - 외부 패키지
        1. 다양한 저장소 시스템을 위한 커넥터(connector)
        2. 머신러닝을 위한 알고리즘까지 수백개의 외부 오픈소스 라이브러리도 존재한다.

4. 스파크의 등장 배경
    - 2005년경 하드웨어의 성능향상이 멈춤
    - 물리적인 방열 한계로 인해 하드웨어 개발자들은 단일프로세서의 성능을 향상시키는 방법 대신 모든 코어가 같은 속도로 동작하는 병렬 CPU 코어를 더 많이 추가하는 방향으로 선회
    - 데이터가 클러스터에서 처리해야 할 만큼 거대해짐.
    - 하여, 기존의 전통적인 프로그래밍 모델이 아니 새로운 프로그래밍 모델이 필요하게되어 아파치 스파크(Apache Spark)가 탄생하게 됨.

5. 스파크의 역사
    - UC 버클리 대학교에서 2009년 스파크 연구 프로젝트로 시작됨.
    - Spark: Cluster Computing with Working Sets 논문 발표로 알려짐
        - 마테이 자하리아(Matei Zaharia), 모샤라프 카우두리(Mosharaf Chowdhury), 마이클 프랭클린(Michael Franklin), 스콧 쉔커(Scott Shenker), 이온 스토이카(Ion Stoica)
        - 이상 AMPLab 소속
    - 문제
        - 첫번째 문제 : 클러스터 컴퓨팅이 엄청난 잠재력을 가지고 있다는 것, 하지만 경험이 없는 조직은 누군가가 성공하고 난 후에 사용할 수 있다는 것
        - 두번째 문제 : 맵리듀스 엔진을 사용하는 대규모 애플리케이션의 난이도와 효율성
    - 해결
        - 함수형 기반 API 설계
        - 연산 단계 사이에서 메모리에 저장된 데이터를 효율적으로 공유할 수 있는 새로운 엔진 기반의 API 구현
    - AMPLab팀은 스파크가 특정 업체에 종속되는 것을 막기위해 아파치 재단에 기부
    - AMPLab팀은 프로젝트를 성장시키기위해 데이터브릭스를 설립
    - 2014년 스파크 1.0 발표(by 아파치 스파크 커뮤니티)
    - 2016년 스파크 2.0 발표(by 아파치 스파크 커뮤니티)
    - Spark 초기버젼 1.0 이전 : 함수형 연산
    - Spark 1.0 부터 Spark SQL(구조화된 데이터 기반) 추가
    - DataFrame, 머신러닝 파이프라인 그리고 자동 최적화를 수행하는 구조적 스트리밍등 더 강력한 구조체 기반의 신규 API들이 추가됨.

6. 스파크의 현재와 미래
    - Spark의 고수준 스트리밍 엔진인 구조적 스트리밍(2016)을 기반으로 스파크의 영역을 꾸준히 넓혀가고 있다.

- 우버, 넷플릭스, NASA, CERN(유럽 입자 물리연구소) 같은 연구소에서 거대한 규모의 데이터 셋을 처리하기 위해 사용
- MIT, 하버드 같은 교육기관에서는 과학적 데이터 분석에 스파크를 사용

7. 스파크 실행하기
   - 파이썬, 자바, 스칼라, R, SQL 언어에서 스파크를 사용할 수 있다.
   - 스파크는 스칼라로 구현되어 자바 가상 머신 기반으로 동작한다.
   - 따라서 노트북(Notebook) or 클러스터 환경에서 스파크를 실행하려면 자바를 설치해야 한다.
   - 파이썬 API를 사용하려면, python 2.7버젼 이상을 설치해야 한다.
   - R을 사용하려면 R을 컴퓨터에 설치해야 한다.
   -
   - 스파크를 시작하는 두 가지 방법
     - Apache Spark를 노트북에 내려받아 설치
     - Spark 학습용 무료 클라우드 환경 웹 기반의 데이터 브릭스 커뮤니티

8. 스파크 실행 환경 구성하기
   - OS : 윈도우, 리눅스, Mac OS 환경을 지원
   - Java
     - 최신 버전의 스파크를 실행하려면 Java 8.0 이상이 설치되어 있어야 한다. Java를 설치한 다음 시스템 PATH나 JAVA_HOME 환경변수에 자바가 설치된 경로가 설정되어 있는지 확인한다.
   - Scala(스칼라)
     - 스파크 2.3 버전부터는 스칼라 2.11 버전 이상은 지원한다. 스칼라 설치 방법은 스칼라 홈페이지(https://www.scala-lang.org/download/)를 참고 하면된다.
     - 현재 사용하는 책은 스파크 2.2 버전을 기준으로 작성했기 때문에 스칼라 2.10 버전도 사용 가능하다.
     - 하지만 문제없이 사용하려면 스칼라 2.11 버전을 설치하는 것이 좋다.
   - python
     - PySpark를 실행하려면 파이썬이 설치되어 있어야 한다. 파이썬은 2.7 이상의 버전이나 3.4이상의 버전을 사용해야 한다.
   - R
     - SparkR 또는 sparklyr를 실행하기 위해서는 R 언어가 설치되어 있어야 한다. 스파크는 R 3.1 이상을 지원한다. R 설치 방법은 R-project(https://www.r-project.org/) 공식 웹사이트를 참고.

9. Spark 설치 - local
   - Spark 공식 홈페이지 접속
   - Spark 공식 홈페이지 Download 링크 클릭
   - Spark Download 링크의 절차에 따라 실행환경을 설정
     - 스파크 버전 선택, 스파크 2.2버전 선택
     - spark-2-2-2-bin-hadoop2.7.tgz
   - Download Spark을 클릭해서 스파크를 내려받는다.
   - 내려받은 스파크 바이너리 파일을 원하는 경로에 압축을 푼다.
   - 내려받은 스파크 디렉토리 구조
   - 스파크 셸을 실행하기 위해 bin 디렉토리로 이동한다. 다양한 언어와 방식으로 스파크를 실행하는데 필요한 각종 셸 파일이 존재한다.

- cmd 파일은 윈도우용이며, 확장자가 없는 파일은 리눅스나 Mac OS를 위한 스파크 실행파일이다.
-
- 스파크 기본 실행 방식인 spark-shell 명령을 실행하면 다음과 같이 출력된다.

10. Spark 설치 - databricks
    - 회사이름이자 스파르 런타임 솔루션
    - 클라우드 환경을 기반으로 스파크 실행 환경을 제공
    - 관리형 스파크 클러스터 환경
    - 대화형 데이터 탐색 및 시각화 기능
    - 운영용 파이프라인 스케줄러
    - 선호하는 스파크 기반 애플리케이션을 위한 플랫폼

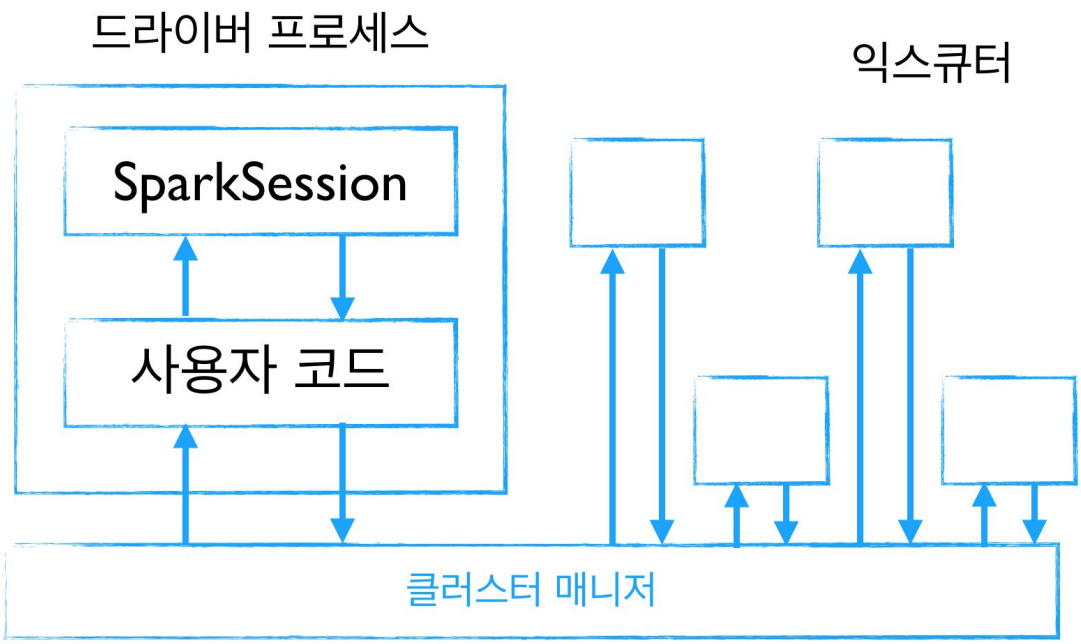
    - 데이터브릭스 공식홈페이지 접속
    - 메인 화면에서 TRY DATABRICKS FREE 링크를 클릭
    - databricks free trial
    - databricks community edition

11. Spark 설치 - docker
    - Docker
        - 컨테이너 기반의 오픈소스 가상화 플랫폼
        - 컨테이너는 HostOS 위에 구성된 고립된 환경을 의미한다.
        - Host OS 환경과 다른 컨테이너의 환경에 영향을 받지 않는 OS 공간이다.
        -
    - Docker 설치하기
    - Docker 이미지 실행하기
        - Docker 이미지 내려 받기
            1. docker pull docker.io/rheor108/spark_the_definitive_guide_practice
        - Docker 이미지 실행 하기
            1. docker run -p 8080:8080 -p 4040:4040 rheor108/spark_the_definitive_guide_practice
        - 웹 브라우저를 통해 제플린 UI에 접속하기
            1. http://localhost:8080
    -
    -
    -

12. 스파크의 기본 아키텍쳐(DataFrame, SQL, Spark Application, 구조적 API)
    - 스파크는 데이터의 클러스터의 데이터 처리 작업을 관리하고 조율하는 프레임워크
    - 클러스터 : 여러 컴퓨터의 자원을 모아 하나의 컴퓨터처럼 사용할 수 있게 만들어 놓은것
    - 스파크 스탠드얼론(standalone) 클러스터 매니저, 하둡 YARN, 메소스(Mesos)같은 클러스터 매니저에서 스파크 클러스터를 관리한다.
    - 사용자-> 스파크 애플리케이션 제출 -> 클러스터 매니저 -> 자원 할당 -> 작업처리

13. 스파크 애플리케이션(Spark Application)
    - 구성 : 드라이버 프로세스(driver process) 하나, 다수의 익스큐터(executor) 프로세스
    - 드라이버 프로세스(driver process)는 클러스터 노드 중 하나에서 실행되며, main()함수 실행
    - 드라이버 프로세스 : 스파크 애플리케이션의 심장
    - 익스큐터(executor)
        - 드라이버 프로세스가 할당한 작업을 수행한다.
        - 드라이버가 할당한 코드를 실행하고 진행 상황을 다시 드라이버 노드에 보고 하는 역할

    - 스파크 애플리케이션의 아키텍쳐

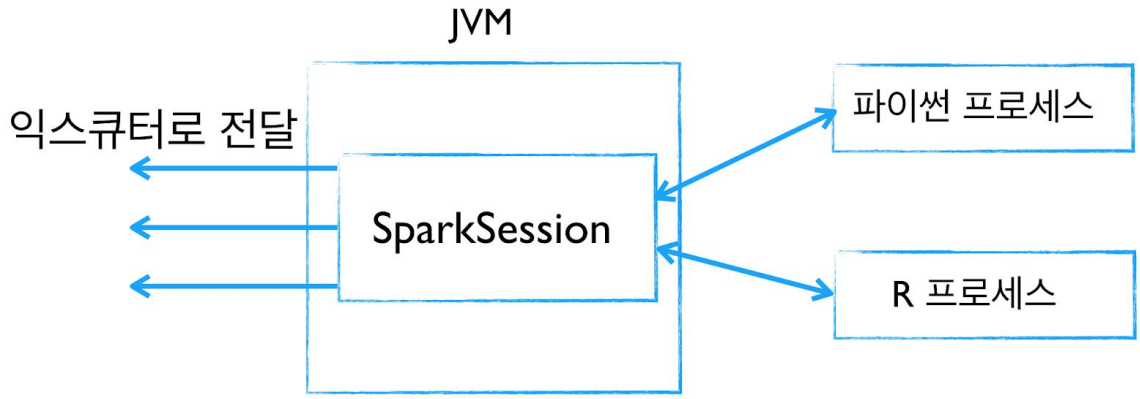


        - 클러스터 매니저가 물리적 머신을 관리하고, 스파크 애플리케이션에 자원을 할당
        - 클러스터 매니저는 스파크 스탠드얼론 클러스터 매니저, 하둡 YARN, 메소스 중 하나 선택가능
        - 하나의 클러스터에서 여러 개의 스파크 애플리케이션을 실행가능
        - 위 그림에서는 클러스터 노드의 개념은 나타내지 않음
        - 사용자는 각 노드에 할당할 익스큐터 수를 지정할 수 있다.

- 스파크는 클러스터 모드와 로컬모드를 지원한다.
- 드라이버와 익스큐터는 단순 프로세스이므로 같은 머신이나 서로 다른 머신에서 실행가능함.
- 단, 로컬모드에서 실행을 하면 단일머신에서 스레드 형태로 실행
- 드라이버는 스파크의 언어 API를 통해 다양한 언어로 실행가능

- **스파크 애플리케이션의 핵심**
  - 사용 가능한 자원을 파악하기 위해 클러스터 매니저를 사용
  - 드라이버 프로세스는 주어진 작업을 완료하기 위해 드라이버 프로그램의 명령을 익스큐터에서 실행할 책임이 있다.

14. 스파크의 다양한 언어 API



-
- 사용자는 스파크 코드를 실행하기위해 SparkSession 객체를 진입점으로 사용할 수 있다.
- 파이썬이나 R로 스파크를 사용할 때는 JVM 코드를 명시적으로 작성하지 않는다.
- 스파크는 사용자를 대신해 파이썬이나 R로 작성한 코드를 익스큐터의 JVM에서 실행할 수 있는 코드로 변환한다.
-
- **스파크에서 사용 가능한 언어**
  - 스칼라 : 스파크는 스칼라로 개발되어 있으므로 스칼라가 스파크의 '기본' 언어이다.
  - 자바 : 스파크가 스칼라로 되어 있지만, 스파크 창시자들은 자바를 이용해 스파크 코드를 작성할 수 있도록 심혈을 기울였다.
  - 파이썬 : 파이썬은 스칼라가 지원하는 거의 모든 구조를 지원한다.

        - SQL : ANSI SQL 2003 표준 중 일부를 지원한다. 분석가나 비프로그래머도 SQL을 이용해 스파크의 강력한 빅데이터 처리 능력을 쉽게 활용할 수 있다.
        - R : 스파크에는 일반적으로 사용하는 두 개의 R 라이브러리가 있다. 하나는 스파크 코어에 포함된 SparkR이고, 다른 하나는 R 커뮤니티 기반 패키지인 sparklyr이다.

15. Spark API
    - 다양한 언어로 스파크를 사용할 수 있는 이유는 스파크가 크게 2가지 API를 제공하기 때문
        - 저수준의 비구조적(unstructured) API
        - 고수준의 구조적(structured) API
    - 여기서는 고수준의 구조적 API를 중심으로 진행할 예정임

16. 스파크 시작하기
    - ./bin/spark-shell 명령을 사용해 스칼라 콘솔에 접속할 수 있다.
    - 대화형 모드로 스파크를 시작하면 스파크 애플리케이션을 관리하는 SparkSession이 자동으로 생성된다.
    - 스탠드얼론 애플리케이션으로 스파크를 시작하면 사용자 애플리케이션 코드에서 SparkSession 객체를 직접 생성해야 한다.

17. SparkSession
    - 스파크 애플리케이션은 SparkSession이라 불리는 드라이버 프로세스로 제어한다.
    - SparkSession 인스턴스는 사용자가 정의한 처리 명령을 클러스터에서 실행한다.
    - 하나의 SparkSession은 하나의 스파크 애플리케이션에 대응한다.
    - 스칼라나 파이썬에서 SparkSession을 확인하기 위해 다음 코드를 실행해보자.
    - spark
        - // 스칼라에서 실행
        - res0: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@4751cd3
        - // 파이썬에서 실행
        - <pyspark.sql.session.SparkSession object at 0x103767bd0>

    - (code)
        - // 스칼라 코드
        - val myRange = spark.range(1000).toDF("number")
        - // 파이썬 코드
        - myRange = spark.range(1000).toDF("number")
    - 이 코드는 한 개의 컬럼과 1000개의 로우로 구성되며 각 로우에는 0부터 999까지의 값이 할당되어 있다.
    - 이 숫자들은 분산 컬렉션을 나타낸다.
    - 클러스터 모드에서 실행하면 숫자 범위의 각 부분이 서로 다른 익스큐터에 할당된다.
    - 이것이 스파크의 DataFrame이다.

18. DataFrame
    - DataFrame은 가장 대표적인 구조적 API이다.
    - DataFrame은 테이블의 데이터를 로우와 컬럼으로 단순하게 표현한다.
    - 컬럼과 컬럼의 타입을 정의한 목록을 스키마(Schema)라고 부른다.
    - 스프레드 시트는 한 대의 컴퓨터에 있지만, 스파크의 DataFrame은 수천 대의 컴퓨터에 분산되어 있다.

- 여러 컴퓨터에 데이터를 분산하는 이유는? 단일 컴퓨터에 저장하기에는 데이터가 너무 크거나 계산에 너무 오랜 시간이 걸릴 수 있기 때문이다.
- 스파크는 Dataset, DataFrame, SQL 테이블 그리고 RDD라는 몇 가지 핵심 추상화 개념을 가지고 있다.(모두 분산 데이터 모음을 표현)
- 이 중 가장 쉽고 효율적인 DataFrame은 모든 프로그래밍 언어에서 사용할 수 있다.

19. 파티션
    - 스파크는 모든 익스큐터가 병렬로 작업을 수행할 수 있도록 파티션이라 불리는 청크 단위로 데이터를 분할한다.
    - 파티션은 클러스터의 물리적 머신에 존재하는 로우의 집합을 의미한다.
    - DataFrame의 파티션은 실행중에 데이터가 컴퓨터 클러스터에서 물리적으로 분산되는 방식을 나타낸다.
    - 파티션이 하나라면, 스파크에 수천 개의 익스큐터가 있더라도 병렬성은 1이 된다.
    - 또한 수백개의 파티션이 있더라도 익스큐터가 하나밖에 없다면 병렬성은 1이 된다.
    - DataFrame을 사용하면 파티션을 수동 혹은 개별적으로 처리할 필요가 없다.
    - 물리적 파티션에 데이터 변환용 함수를 지정하면 스파크가 실제 처리 방법을 결정한다.
    - RDD 인터페이스를 이용하는 저수준 API역시 제공된다.