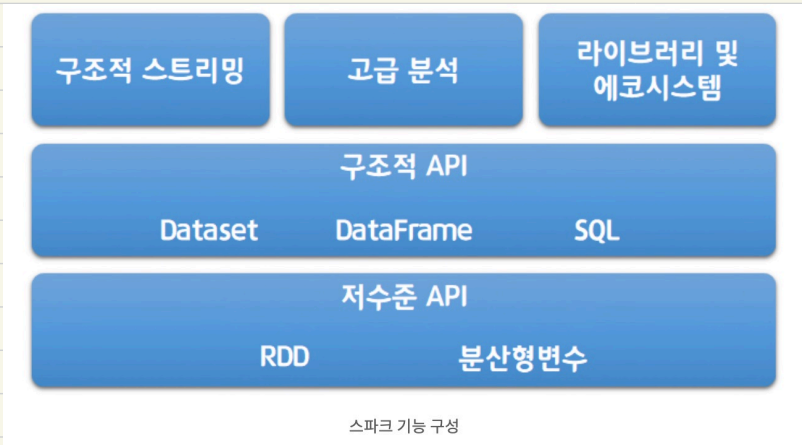
스파크 완벽 가이드 정리해보세

Part 1 - 빅데이터와 스파크 간단히 살펴보기

Chapter 1 - 아파치 스파크란

1. 통합 컴퓨팅 엔진, 클러스터 환경에서 데이터를 병렬로 처리하는 라이브러리 집합
   - 파이썬, 자바, 스칼라, R의 4가지 언어 지원
   - SQL, 머신러닝, 스트리밍 등의 라이브러리 지원



스파크 기능 구성

1.1 아파치 스파크의 철학

- 통합

  Biodata application에 필요한 통합(SQL, 머신러닝, 스트림 등) 플랫폼 제공 목표

- 컴퓨팅 엔진

  데이터 연산 역할 O, 영구 저장소 역할X But 여러 저장소 연동 지원

  사용자 API는 여러 저장소 시스템을 유사하게 볼 수 있도록 개발됨

- 라이브러리(* - 뉘앙스가 중요한가봐..?)

  표준 라이브러리(스파크 엔진 제공), 외부 라이브러리(서드파티 패키지 형태) 제공

  라이브러리 : SPARK SQL, MLlib, 구조적 스트리밍, Graphx, 커넥터, 머신러닝 알고리즘

  외부 라이브러리 목록 : spark-package.org 에서 확인 가능

궁극의 스파크 컴포넌트는 데이터 분석 작업에 필요한 통합 API를 제공하는 통합 엔진 기반의 자체 라이브러리입니다. — 무슨말이지..

Part 1 - 빅데이터와 스파크 간단히 살펴보기

Chapter 1 - 아파치 스파크란

1. 2 스파크의 등장 배경
- 데이터 분석 처리 엔진과 프로그래밍 모델이 필요한 근본적 이유는?
  - 컴퓨터 애플리케이션과 하드웨어의 바탕을 이루는 경제적 요인의 변화 때문
  - 컴퓨터 속도 -> 프로세서 성능 향상으로 빨라짐 -> 하드웨어 성능 향상 저하 (2005년 이후) -> 병렬처리 (CPU 코어 추가) -> 스파크 등장 배경
  - 데이터 수집에 필요한 기술 비용 갈수록 저렴 But 데이터는 급격히 증가
  - 과거의 하드웨어 성능 및 프로그래밍 모델 적용 어려움 -> 새로운 모델 필요

1. 3 스파크의 역사
- UC버클리 2009년 스파크 연구 프로젝트로 시작
- Spark: cluster computing with working sets 논문으로 처음 알려짐(2010년)
- 전통적인 맵리듀스 맵리듀스 엔진은 디스크 I/O를 반복 처리 -> 스파크는 연산 단계 사이에서 메모리에 저장된 데이터를 효율적으로 공유할 수 있는 새로운 엔진 기반 API 개발
- 이후 ad-hoc query 기능 제공(이 이아디어로 샤크가 개발된다능..)
- SPARK SQL 성공 후 MLlib, spark streamming, Graphx 등 개발
- AMPLap 아파치 재단 기부 및 1.0 버전 공개 (2014) -> 2.0 공개 (2016)
- 함수형 연산 (1.0 이전) -> 구조화된 데이터 (1.0 이후) 기반 SparkSQL 추가 -> 신규 API 추가(DataFrame, 머신러닝 파이프라인 등)

1. 4 스파크의 현재와 미래 - 대충 스파크 쓰라는 내용

1. 5 스파크 실행하기
- 실제로 설치하고 실행해보자는 내용과 데이터는 저자 깃허브를 참고하라는 내용

Part 1 - 빅데이터와 스파크 간단히 살펴보기

Chapter 2 - 스파크 간단히 살펴보기 : 스파크 핵심 용어와 개념, 사용법 탐구

2.1 스파크의 기본 아키텍처

- 클러스터 : 여러 컴퓨터의 자원을 모아 하나의 컴퓨터처럼 사용
- 스파크 : 클러스터의 데이터 처리 작업을 관리하고 조율하는 프레임워크

2.2 스파크 애플리케이션

- 구성 : 드라이버 프로세스와 다수의 익스큐터 프로세스
- 드라이버 프로세스(*) : main() 함수 실행

  -> 1 스파크 애플리케이션 정보 유지 관리

  2 사용자 프로그램이나 입력에 대한 응답

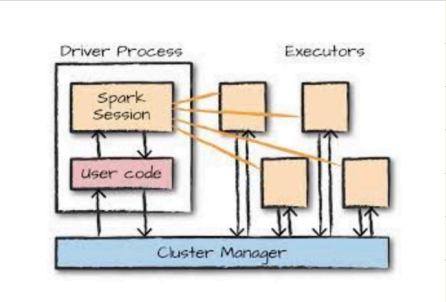
  3 익스큐터 프로세스의 작업 관련 분석

  4 배포 스케줄링 역할 수행 등등

- 익스큐터 프로세스 : 드라이버 프로세스가 할당한 작업을 수행

  -> 1 드라이버가 할당한 코드 실행

  2 진행상황 드라이버 노드에 보고



클러스터 매니저

- 물리적 머신 관리 스파크 어플리케이션 자원 할당
- 하둡, yarn, 메소스, 스파크 스탠드얼론 클러스터 매니저 선택 가능
- 하나의 클러스터에서 여러개 스파크 애플리케이션 실행

* 스파크는 사용 가능한 자원을 파악하기 위해 클러스터 매니저를 사용
* 드라이버 프로세스는 드라이버 프로그램의 명령을 익스큐터에서 실행

Part 1 - 빅데이터와 스파크 간단히 살펴보기

Chapter 2 - 스파크 간단히 살펴보기 : 스파크 핵심 용어와 개념, 사용법 탐구

2. 2 스파크의 다양한 언어 API

- 스파크는 모든 언어에 핵심개념(클러스터 머신에서 실행되는 스파크 코드 변환)을 제공
- 스칼라: 스파크의 기본 언어(스칼라로 개발)
- 자바: 창시자들은 자바를 이용해 스칼라 코드를 작성할 수 있도록 개발
- 파이썬: 스칼라가 지원하는 거의 모든 구조 지원
- SQL: ANSI SQL:2003 표준 일부 지원
- R: SparkR(스파크 코어), sparklyr(R커뮤니티 기반) 지원(라이브러리 통합 가능)

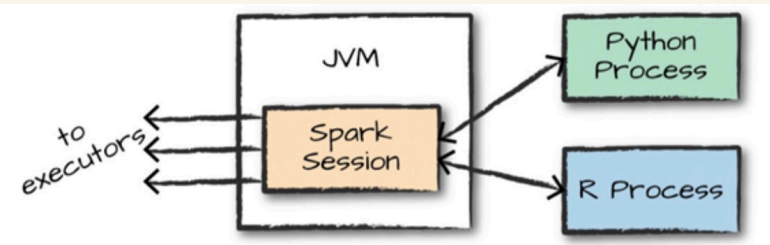


Figure 2-2. The relationship between the SparkSession and Spark's Language API

2. 3 스파크 API

- 스파크는 비구조적 API(저수준), 구조적 API(고수준) 두 가지를 사용

2. 4 스파크 시작하기

- 스파크 개발을 위해서는 사용자 명령과 데이터를 스파크 애플리케이션에 전송하는 방법을 알아야 함

-stand alone: SparkSession 직접 생성(대화형 모드는 SparkSession 자동 생성)

2. 5 SparkSession

- Spark application은 SparkSession이라 불리는 드라이버 프로세스로 제어(1 : 1 대응)

Part 1 - 빅데이터와 스파크 간단히 살펴보기

Chapter 2 - 스파크 간단히 살펴보기 : 스파크 핵심 용어와 개념, 사용법 탐구

2.6 DataFrame

- 가장 대표적인 구조적 API, 테이블의 데이터를 로우와 컬럼으로 단순 표기
- 스키마: 컬럼과 컬럼 타입을 정의한 목록
- 스파크의 DataFrame은 수천대의 컴퓨터에 분산되어 저장됨
- Pandas, R의 데이터 프레임을 스파크의 데이터 프레임으로 쉽게 변환 가능

2.6.1 파티션

- 파티션: 청크 단위로 데이터 분할한 것(익스큐터가 병렬로 작업 수행 가능하도록)
- 클러스터의 물리적 머신에 존재하는 로우의 집합
- DataFrame의 파티션: 실행중에 데이터가 컴퓨터 클러스터에서 물리적으로 분산되는 방식(파티션 한개라면 스파크에 익스큐터가 수천대라도 병렬성은 1)
- DataFrame 사용하면 파티션을 수동/개별적으로 처리할 필요 없음

2.7 트랜스포메이션* (스파크 비즈니스 로직 표현하는 핵심 개념)

- immutable(불변성): 스파크 핵심 데이터 구조, 한번 생성하면 변경할 수 없음
- Transformation: DataFrame을 변경할때 사용하는 명령
- 액션(Action)을 호출하기 전까지 트랜스포메이션을 수행하지 않음
- 좁은 의존성(narrow dependency)
  각 입력 파티션이 하나의 출력 파티션에만 영향을 미치는 것, 파이프라이닝 자동 수행
- 넓은 의존성(wide dependency)
  하나의 입력 파티션이 여러 출력 파치션에 영향을 미치는 것
  셔플(shuffle)이 발생함(스파크는 셔플 결과를 디스크에 저장)
  셔플 최적화**는 중요한 이슈

Part 1 - 빅데이터와 스파크 간단히 살펴보기

Chapter 2 - 스파크 간단히 살펴보기 : 스파크 핵심 용어와 개념, 사용법 탐구

2.7.1 지연 연산(lazy evaluation)

- 스파크가 연산 그래프를 처리하기 직전까지 기다리는 동작방식
- 스파크는 특정 연산 명령이 내려진 즉시 데이터를 수정하지 않고 원시 데이터에 적용할 트랜스포메이션의 실행계획을 생성. 실행의 마지막 순간까지 대기하다가 원형의 데이터 프레임 트랜스포메이션을 물리적 실행계획으로 컴파일
- DataFrame의 Predicate pushdown(조건절 푸시다운)
- 복잡한 스파크 잡이 원시 데이터에서 하나의 로우만 가져오는 필터가 있다면 레코드 하나만 읽는 것이 효율적

2.8 액션

- 트랜스포메이션을 사용해 논리적 실행계획을 세웠다면 실제 연산은 액션 명령을 사용
- 트랜스포메이션으로부터 결과를 계산하도록 지시하는 명령
- 액션 명령의 예: df.count()
- 액션의 유형:
  1. 콘솔에서 데이터를 확인
  2. 각 언어로 된 네이티브 객체
  3. 출력 데이터소스에 저장하는 액션
- 액션을 지정하면 스파크 잡(Job) 실행. 필터(좁은 트랜스포메이션) 수행 후 파티션별로 레코드 수를 카운트(넓은 트랜스포메이션)

2.9 스파크 UI

- 스파크 잡 진행상황 모니터링 할 때 사용(4040포트)
- 튜닝 및 디버깅시 유용***

Part 1 - 빅데이터와 스파크 간단히 살펴보기

chapter 2 - 스파크 간단히 살펴보기 : 스파크 핵심 용어와 개념, 사용법 탐구

2.10 종합 예제

- 스파크 내부에서 일어나는 일을 살펴보자
- 처음 데이터를 읽어올땐 스키마 추론(schema inference) 기능을 사용함
- sort 메서드는 트랜스포메이션이기 때문에 호출시 데이터 변화는 없음
- 실행계획 읽는 법: 위에서 아래 방향으로 읽고 최종 결과는 가장 위에 있음
- 스파크는 셔플 수행시 200개 파티션 생성(기본)
- spark.conf.set("spark.sql.shuffle.partitions", "5") 명령어로 설정 변경 가능

2.11 DataFrame과 SQL

- 스파크는 언어에 상관없이 같은 방식으로 트랜스포메이션을 실행 가능
- 실행계획은 트랜스포메이션의 지향성 비순환 그래프
  (directed acyclic graph, DAG)

Part 1 - 빅데이터와 스파크 간단히 살펴보기

Chapter 3 - 스파크 기능 둘러보기

- 다음과 같은 내용을 설명함
- spark-submit 명령으로 운영용 애플리케이션 실행
- Dataset: 타입 안정성(type-safe)을 제공하는 구조적 API
- 구조적 스트리밍
- 머신러닝과 고급분석
- RDD: 스파크의 저수준 API
- SparkR
- 서드파티 에코시스템

3.1 운영용 애플리케이션 실행하기

Spark-submit 명령: 애플리케이션 코드를 클러스터에 전송해 실행시키는 역할

- 스파크 애플리케이션은 stand-alone, YARN, mess 클러스터 매니저를 이용해 실행

3.2 Dataset

- 정적 타입 코드(statically typed code)를 지원하기 위해 고안된 스파크 구조적 API, 파이썬과 R에서는 사용 불가
- 데이터프레임의 레코드를 사용자가 자바나 스칼라로 정의한 클래스에 할당하고 자바의 ArrayList 또는 스칼라의 Seq 객체 등의 고정 타입형 컬렉션으로 다룰 수 있는 기능 제공
- 다수의 소프트웨어 엔지니어가 잘 정의된 인터페이스로 상호작용하는 대규모 애플리케이션 개발하는데 특히 유용
- 타입 안정성을 지원하므로 초기화에 사용한 클래스 대신 다른 클래스 사용해 접근할 수 없음
- Dataset은 필요한 경우에 선택적으로 사용 가능. 스파크 처리를 마치고 결과를 DataFrame으로 자동 변환해 반환 가능

Part 1 - 빅데이터와 스파크 간단히 살펴보기

Chapter 3 - 스파크 기능 둘러보기

spark-packages.org 2.2 버전에서 안정화(production-ready)된 스트림 처리용 고수준 API

- 로컬 모드에서 코드를 실행할경우 셔플 파티션 수를 5개 정도로 줄여 사용
  spark.conf.set("spark.sql.shuffle.partitions", "5")
- 코드는 거의 바뀌지 않음. read 대신 readStream 메서드 사용
- maxFilesPerTrigger 옵션 추가 지정 가능

3.4 머신러닝과 고급분석

- MLlib 활용하여 전처리, 멍잉, 모델학습 및 예측 가능
- 분류, 회귀, 군집화, 딥러닝 등 정교한 API 제공
- 스파크에서 머신러닝 모델을 학습시키는 과정은 크게 두 단계로 진행
  1. 학습되지 않은 모델 초기화
  2. 모델 학습

3.5 저수준 API

- 스파크의 거의 모든 기능은 RDD 기반으로 개발됨
- RDD를 이용해 파티션과 같은 물리적 실행 특성을 결정 가능
  (dataframe보다 세밀하게 제어 가능)
- 원시 데이터를 읽거나 다루는 용도로 사용할 수 있지만 구조적 API 사용 추천

3.6 SparkR

- 스파크를 R언어로 사용하기 위한 기능
- SparkR은 파이썬 API와 매우 유사함

3.7 스파크의 에코시스템과 패키지

- 스파크 패키지 목록은 spark-packages.org 에서 확인 가능