스파크

맵 리듀스를 부분적으로 대체한다.

스파크는 실행 엔진으로 맵 리듀스를 사용하지 않는다 .

대신 스파크는 클러스트 기반으로 작업을 실행하는 자체 분산 런타임 엔진

잡 사이의 대용량 작업 데이터셋을 메모리상에 유지

인메모리이기에 맵 리듀스 워크플로에 비해 10배 이상 빠르다.

이득을 얻는 에플리케이션 상황

- 반복적 알고리즘 (종료 조건을 만족할 떄까지 데이터셋에 함수를 반복 적용)
- 대화형 분석 (사용자가 데이터셋에 일련의 대화식 커리 생성)

인메모리 캐싱을 사용하지 않더라도 , DAG 엔진 , 사용자 경험 제공
DAG 엔진 : 연산자 중심의 파이프라인을 처리 → 사용자를 위한 단일잡으로 변환

조인과 같은 다양한 일반적인 데이터 처리 작업 수행 , 풍부한 API

지원 언어 : 스칼라 ,JAVA ,파이썬 등등

read-eval-print loop(REPL) 커맨드 창 이용해서 코드 작성하는 거 말하는 듯

스파크 include (MLlib,GraphX,Spart Streaming, Spark Sql)

spark-shell (REPL 실행 명령어)

val lines = sc.textFile(”input/ncdc/micro-tab/sample.txt)

↑ 예시에 불과

lines 변수는 탄력적인 분산 테이터셋(RDD)를 참조

다수의 머신에 분할되어 저장된 읽기 전용 컬렉션
RDD(읽기전용)→ETL → 목표 RDD 집합으로 변형 , 영구정 저장소에 저장

val records = lines.map(_.split(”\t”))

위 lines에서 읽어온 데이터셋을 한줄 씩 변환, 한 줄은 또 \t로 분할되어짐

val filtered = records.filter(rec ⇒ (rec(1) = “9999”) && rec(2).mathes(”[01459]”)))

9999로 표기된 기록 제거,

코드 보니간 파이썬으로 하는게 더 나을듯?

<생략>

maxTemps.saveAsTextFile(”output”)으로 저장

ex) output/part-* 양식으로 지정한 디렉터리 밑에 결과 파일들이 생성됨

스파크는 맵 리듀스와 유사하게 잡이라는 개념이 있다.

but, 하나의 맵과 하나의 리듀스로 구성된 단일 맵리듀스의 잡과 달리 스파크의 잡은 임의의 방향성 비순환 그래프(DAG)인 stage로 구성된다.

스테이지는 실행될때 다수의 TASK로 분할 → 각 태스크는 mapredue의 태스크와 같이 클러스터에 분산된 RDD 파티션에서 병렬로 실행된다.

잡은 항상 RDD 및 공유변수를 제공하는 SparkContext내에서 실행된다.

하나의 에플리케이션은 여러개의 잡으로 구성될 수 있고 , 동일 에플리케이션에서 수행된 이전잡에서 캐싱된 RDD에 접근할 수 있는 메커니즘을 제공한다 . ?

그래서 RDD는 뭐지?

RDD를 만드는 방법은 세가지

- 객체의 인메모리 컬렉션으로 생성
- HDFS와 같은 기존 외부 저장소의 데이터셋을 사용
- 기존의 RDD를 변환

우리는 2번을 쓰던지 , 외부 데이터를 받아와 인모메리 컬렉션으로 변환해 쓰던지

스파크는 내부적으로 파일을 읽을 떄 , 구버전 맵리듀스 API의 TextInputFormat을 이용, 파일을 스플릿하는 방식은 하둡과 동일

RDD에서 트랜스포메이션과 액션이라는 두종류를 통해 새로운 RDD 생성

val text = sc.textFile(inputPath)

val lower : RDD[String] = text.map(_.toLowerCase())

lower.foreach(println(_))

map 은 트렌스포메이션

map에 정의된 함수는 액션이 foreach를 만나기 전까지 실제로 호출되지 않는다.

스파크에서는 풍부한 연산자 집합을 제공한다.

근데 mapreduce의 map,reduce 를 구현할려하면 메소드에 주의하라고 한다.

map

flatMap

집계 트랜스포메이션

reduceByKey

foldByKey

aggregateByKey

1 13

1 14

1 15

2 16

2 20 ….

tuples.reduceByKey((a,b) ⇒ Math.min(a,b)).foreach(println(_))

을 하면 1,2 키 중 가장 작은 값 출력

맵 리듀스는 다른 계산을 수행하기 위해서는 입력 데이터셋을 디스크에서 다시 불러와야한다.

따라서 I/O 발생 , 스파크는 여러 머신에 있는 메모리에 데이터셋을 분산하여 캐싱할 수 있어 빠르게 처리가능 ,cache()를 호출해야함

메모리 작다고 RDD 파티션 저장 수행을 실패하지는 않음 , 하지만 연산 과정중 실패하면 다시 계산을 수행해야할 경우 휘발되니깐 다시 해야함, 중간 중간 저장하는 persist메서드를 사용가능

직렬화

공유변수

브로드 캐스트 변수

어큐뮬레이터

YARN 에서 스파크 실행

YARN 클러스터 모드 : 운녕에 유리 ,

YARN 클라이언트 모드 : 대화형 컴포넌트 필요, 디버깅 용이

요약 : map reduce 비슷하게 사용가능, 문법이다름 , 더 빠름(특정상황 ex 작을 때) , 연속적 처리가 필요할 떄(map reduce는 한 시퀀스로 끝나니깐)