Chapter8. Tuning and Debugging Spark

아꿈사

overview

- SparkConf로 스파크 설정하기
- 실행을 구성하는 것 : Jobs, Tasks and Stages
- 정보 찾기
- 성능에 관한 핵심 고려 사항

SparkConf

- Spark application의 설정을 변경할 수 있음
- key, value값으로 구성
- 아래의 방법으로 설정 가능
  - SparkConf 오브젝트로 구성
  - spark-submit설정

SparkConf오브젝트로 설정

```
val conf = new SparkConf()
conf.set("spark.app.name", "My Spark App")
conf.set("spark.master", "local[4]")
conf.set("spark.ui.port", "36000")
val sc = new SparkContext(conf)
```

spark-submit설정

```
$ bin/spark-submit \
 --class com.example.MyApp \
 --master local[4] \
 --name "My Spark App" \
 --conf spark.ui.port=36000 \
 myApp.jar
```

spark-submit설정

```
$ bin/spark-submit \

 --class com.example.MyApp \
 --properties-file my-config.conf \
 myApp.jar
```

실행을 구성하는 것 : Jobs, Tasks, Stages

- RDD시각화

RDD가 자신이 어떤 타입의 관계를 갖고 있는지 보여준다.

각 함수들이 내부적으로 어떤 일을 하는지 알 수 있다.

RDD 시각화

```
scala> input.toDebugString
res0: String =
(2) MapPartitionsRDD[1] at textFile at <console>:21 []
 |  input.txt HadoopRDD[0] at textFile at <console>:21 []

scala> counts.toDebugString
res1: String =
(2) ShuffledRDD[5] at reduceByKey at <console>:25 []
 +-(2) MapPartitionsRDD[4] at map at <console>:25 []
    |  MapPartitionsRDD[3] at map at <console>:23 []
    |  MapPartitionsRDD[2] at filter at <console>:23 []
    |  MapPartitionsRDD[1] at textFile at <console>:21 []
    |  input.txt HadoopRDD[0] at textFile at <console>:21 []
```

RDD 모으기

- collect()함수를 사용해 명시적으로 액션을 실행
- RDD의 모든 파티션이 실체화, 드라이버 프로그램으로 전송
- 파이프라이닝 : 데이터 이동 없이, 하나의 부모로부터 연산가능 할 때,
  물리단계에서 합쳐질 수 있음
- 건너뛰기(short-circuit) : 캐싱 작업 건너 뛰기

scala> counts.collect()
res0: Array[(String, Int)] = Array((She,1), (�Perhaps,1), (There,1), (�Papa?,1), (Elsewhere,,1), (As,1), (Fascinus,3), (�Do,1), (�Good,1), (With,2), (His,2) …..

파이프라이닝

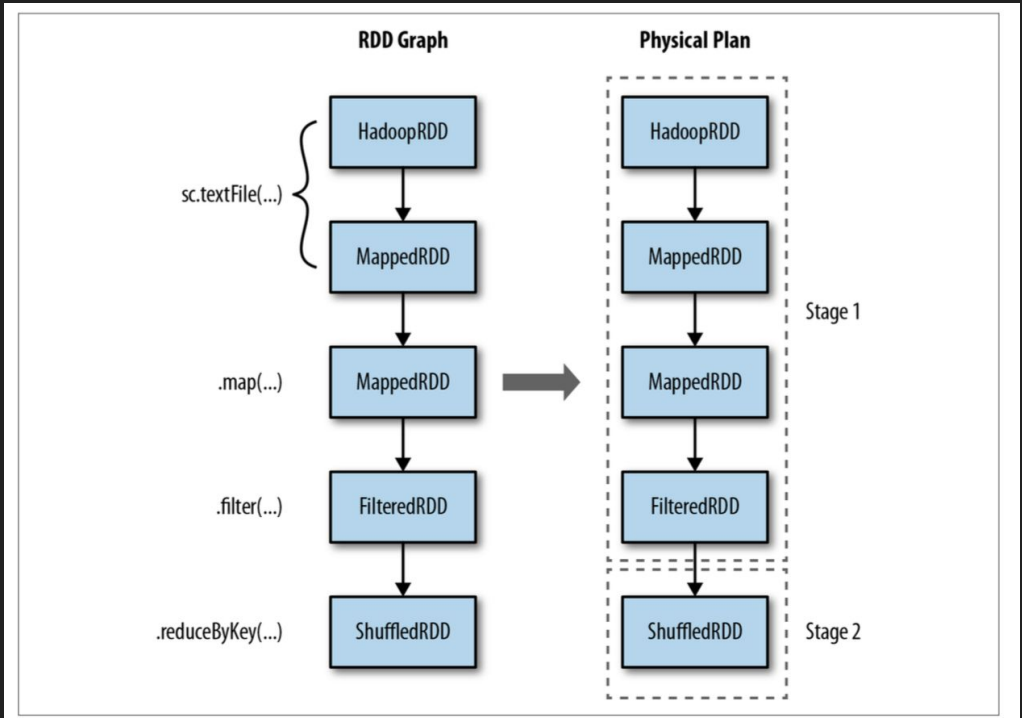

Figure 8-1. RDD transformations pipelined into physical stages

정보 찾기

- 스파크 웹 UI

웹을 통해서 스파크 어플리케이션 성능을 모니터링 지원

Jobs : 진행 상황과 작업 단계, 테스크 등에 대한 수치

Storage : 영속화된 RDD의 정보

Executors : 어플리케이션에 존재하는 익스큐터 목록
익스큐터가 스고있는 저장 장치의 용량, 스레드덤프 기능 제공

성능에 관한 핵심 고려 사항

- 병렬화 수준

목적 : 스파크의 RDD의 병렬화 수준을 조절하여

너무 작으면 리소스가 놀게 되고, 너무 많으면 오버헤드 발생

- coalesc() : 병렬화 정도를 인자로 줄 수 있다.
- repartition() : 무작위로 섞어 다시 나눠준다

성능에 관한 핵심 고려 사항

- 직렬화 포맷

스파크는 네트워크로 데이터를 전송하거나 디스크에 쓸 때, 객체들을 직렬화 시킴

자바의 내장된 직렬화를 사용

서드파티 Kyro가 더 나은 성능을 제공

메모리 관리

- 메모리를 여러 가지 목적으로 사용된다. 목적에 따라 비율 조절 필요
  RDD저장용(60%)
  셔플 및 집합 연산 버퍼(20%)
  사용자 코드(20%)

- 캐싱 동작에 따른 옵션 지정 가능
  - MEMORY_AND_DISK : 메모리, 디스크 저장
  - MEMORY_AND_DISK_SER : 모메뢰, 디스크 저장(직렬화된 상태)