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

- 물리작업을 실행 하기 위해서, 어떻게 실행 계획을 세우는지
- RDD시각화
- 예제

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.

RDD저장용(60%)

셔플 및 집합 연산 버퍼(20%)

사용자 코드(20%)

하드웨어 프로비저닝

스파크에 부여하는 자원은 어플리케이션 완료 시간에 큰 영향

클러스터 사이징에 영향을 미치는 인자

- 각 익스큐터에 지정되는 메모리양
- 각 익스큐터에 쓰는 코어 개수
- 익스큐터의 총 개수
- 데이터에 사용되는 로컬 디스크 개수