March 14, 2019

1 Abstract

RDD 큰 클러스터 안에서 in-memory 연산을 하도록 해준다. 클러스터의 일부분에서 장애가 발생하더라도 정상적 혹은 부분적으로 기능을 수행할 수 있도록 한다(fault–tolerance) 현재의 computing framework가 반복되는 알고리즘이나 상호작용하는 데이터마이닝 툴에서 비효율적으로 연산하는 부분에서 시작되었다. Fault-tolerance를 달성하기위해서 공유 메모리의 제약된 형식을 제공한다.

2 Introduction

Clustering computing frameworks 로 Mapreduce와 Dryad가 널리 사용되고 있다. 이러한 프레임워크들은 사용자가 작업 할당이나 fault-tolerance에 대한 걱정없이 병렬컴퓨팅을 사용가능하게 해준다. 그러나 메모리할당과 관련된 부분이 빠져있다. 이러한 부분은 비효율성을 야기한다. Data reuse는 많은 iterative한 머신러닝 알고리즘들에서 발생한다. 또한 interactive한 데이터 마이닝에서 문제가 발생한다. 이러한 알고리즘들을 처리하는 방법은 데이터를 외부의 정적인 저장소에 저장하는 방법뿐이다. 이것은 데이터 반복으로 인한 잠재적인 문제를 야기한다. 결과적으로 연산시간이 증가하게 된다. 이러한 문제들을 발견하고 데이터 재사용을 위한 프레임워크를 만들었다. 그러나 이러한 프레임워크들은 특정한 컴퓨팅 패턴에 대해서만 지원되는 문제가 발생한다. 따라서 RDDs를 소개한다

3 RDDs

RDDs는 read-only, 분할된 기록들의 집합이다. RDDs는 정적인 저장공간 혹은 다른 RDDs에 있는 data를 사용한 deterministic한 operation에 의해서만 생성된다. RDDs는 항상 연산되어질 필요가 없다. 일련의 프로세스에 대한 정보(lineage)를 바탕으로 한번에 연산된다. Lineage는 DAG 방향성이 있고 순환되지 않는 그래피컬 모델로 디자인이 되어있다. 따라서 RDDs는 데이터를 저장하는 방식이 아닌 DAG를 저장하기 때문에 cost가 크지 않다. 일련의 과정들을 기록하고 최종단계에서 연산이 한번만 발생하기 때문이다. RDD의 Operation은 transforamtion과 action 두가지가 있다. 데이터의 값을 변경시키는 작업을 transformation이라고 하고 결과를 출력하는 과정을 action이라고 한다. Action이 발생할 때에만 연산을 수행하는 것을 lazy-execution이라고한다 (pyhon의 map 함수나 range함수를 생각하면 이해하기 쉽다)

4 Pros

장점을 분석하기 위해서 DSM(distributed memory abstraction)과의 비교를 한다. DSM은 전체의 메모리 공간에서 임의 공간을 할당하는 작업을 수행한다. 이는 일반적으로 효율적인 연산을 어렵게 하고 fault-tolerance또한 어렵게 한다. 두 방법간의 주요한 차이점은 RDDs는 coarse-grained transformation을 통해서만 생성된다는 점이다. 따라서 RDDs는 check point에 대한 간접비용이 발생하지 않는다. 또한 문제가 발생 할 경우 전체 프로그램을 다시 롤백할 필요없이 문제가 발생한 부분부터 병렬컴퓨팅으로 처리한다. 두번째 장점은 변환불가한 RDD의 특성이 느린 노드의 특성을 back up tasks를 활용해서 완화시킨다

5 RDD Abstraction

5.1 restriction

공유 메모리 분배에 대한 제약된 형식으로 변형이 불가능하고, record들의 분할된 집합이다. coarse-grained deterministic transformations를 통해서만 생성된다

5.2 Efficient fault-tolerance using lineage

fine-grained data를 업데이트하는 것이 아닌 coarse-grained operations을 기록한다 RDD는 어떻게 다른 데이터셋으로부터 도출되었는지 충분한 정보를 가지고 있다. 또한 문제가 발생할 시 발생한 파티션에 대한 부분만 재연산한다.