Spark 简介以及与 Hadoop 的对比

1 Spark 简介

1.1 Spark 概述

Spark 是 UC Berkeley AMP lab 所开源的类 Hadoop MapReduce 的通用的并行计算框架，Spark 基于 map reduce 算法实现的分布式计算，拥有 Hadoop MapReduce 所具有的优点；但不同于 MapReduce 的是 Job 中间输出和结果可以保存在内存中，从而不再需要读写 HDFS，因此 Spark 能更好地适用于数据挖掘与机器学习等需要迭代的 map reduce 的算法。

1.2 Spark 核心概念

1.2.1 弹性分布数据集（RDD）

RDD 是 Spark 的最基本抽象,是对分布式内存的抽象使用 ,实现了以操作本地集合的方式来操作分布式数据集的抽象实现。RDD 是 Spark 最核心的东西，它表示已被分区，不可变的并能够被并行操作的数据集合，不同的数据集格式对应不同的 RDD 实现。RDD 必须是可序列化的。RDD 可以 cache 到内存中，每次对 RDD 数据集的操作之后的结果，都可以存放到内存中，下一个操作可以直接从内存中输入，省去了 MapReduce 大量的磁盘 IO 操作。这对于迭代运算比较常见的机器学习算法, 交互式数据挖掘来说，效率提升比较大。

1.2.2 RDD 的转换与操作

对于 RDD 可以有两种计算方式 :转换(返回值还是一个 RDD)与操作(返回值不是一个 RDD)

1. 转换(Transformations) (如：map, filter, groupBy, join 等)，Transformations 操作是 Lazy 的，也就是说从一个 RDD 转换生成另一个 RDD 的操作不是马上执行，Spark 在遇到 Transformations 操作时只会记录需要这样的操作，并不会去执行，需要等到有 Actions 操作的时候才会真正启动计算过程进行计算。
2. 操作(Actions) (如：count, collect, save 等)，Actions 操作会返回结果或把 RDD 数据写到存储系统中。Actions 是触发 Spark 启动计算的动因。

1.2.3 血统（Lineage）

利用内存加快数据加载,在众多的其它的 In-Memory 类数据库或 Cache 类系统中也有实现，Spark 的主要区别在于它处理分布式运算环境下的数据容错性（节点实效/数据丢失）问题时采用的方案。为了保证 RDD 中数据的鲁棒性 ,RDD 数据集通过所谓的血统关系(Lineage)记住了它是如何从其它 RDD 中演变过来的。相比其它系统的细颗粒度的内存数据更新级别的备份或者 LOG 机制，RDD 的 Lineage 记录的是粗颗粒度的特定数据转换（Transformation）操作（filter, map, join etc.)行为。当这个 RDD 的部分分区数据丢失时，它可以通过 Lineage 获取足够的信息来重新运算和恢复丢失的数据分区。这种粗颗粒的数据模型，限制了 Spark 的运用场合，但同时相比细颗粒度的数据模型，也带来了性能的提升。

RDD 在 Lineage 依赖方面分为两种 Narrow Dependencies 与 Wide Dependencies 用来解决数据容错的高效性。Narrow Dependencies 是指父 RDD 的每一个分区最多被一个子 RDD 的分区所用，表现为一个父 RDD 的分区对应于一个子 RDD 的分区或多个父 RDD 的分区对应于一个子 RDD 的分区，也就是说一个父 RDD 的一个分区不可能对应一个子 RDD 的多个分区。Wide Dependencies 是指子 RDD 的分区依赖于父 RDD 的多个分区或所有分区，也就是说存在一个父 RDD 的一个分区对应一个子 RDD 的多个分区。对与 Wide Dependencies，这种计算的输入和输出在不同的节点上，lineage 方法对与输入节点完好，而输出节点宕机时，通过重新计算，这种情况下，这种方法容错是有效的，否则无效，因为无法重试，需要向上其祖先追溯看是否可以重试（这就是 lineage，血统的意思），Narrow Dependencies 对于数据的重算开销要远小于 Wide Dependencies 的数据重算开销。

1.2.4 容错

在 RDD 计算 ,通过 checkpint 进行容错 ,做 checkpoint 有两种方式 ,一个是 checkpoint data ,一个是 logging the updates。用户可以控制采用哪种方式来实现容错 默认是 logging the updates 方式，通过记录跟踪所有生成 RDD 的转换（transformations）也就是记录每个 RDD 的 lineage（血统）来重新计算生成丢失的分区数据。

2 Spark 与 Hadoop 对比

2.1 快速

Spark 的中间数据放到内存中，对于迭代运算效率更高。Spark 更适合于迭代运算比较多

的 ML 和 DM 运算。因为在 Spark 里面，有 RDD 的抽象概念。

2.2 灵活

1. Spark 提供的数据集操作类型有很多种，不像 Hadoop 只提供了 Map 和 Reduce 两种操作。比如 map, filter, flatMap, sample, groupByKey, reduceByKey, union, join, cogroup, mapValues, sort,partionBy 等多种操作类型，Spark 把这些操作称为 Transformations。同时还提供 Count, collect, reduce, lookup, save 等多种 actions 操作。
2. 这些多种多样的数据集操作类型，给给开发上层应用的用户提供了方便。各个处理节点之间的通信模型不再像 Hadoop 那样就是唯一的 Data Shuffle 一种模式。用户可以命名，物化，控制中间结果的存储、分区等。可以说编程模型比 Hadoop 更灵活。
3. 由于 RDD 的特性，Spark 不适用那种异步细粒度更新状态的应用，例如 web 服务的存储或者是增量的 web 爬虫和索引。就是对于那种增量修改的应用模型不适合。

2.3 容错性

在 RDD 计算，通过 checkpoint 进行容错，做 checkpoint 有两种方式，一个是 checkpoint data，一个是 logging the updates。用户可以控制采用哪种方式来实现容错，默认是 logging the updates 方式，通过记录跟踪所有生成 RDD 的转换（transformations）也就是记录每个 RDD 的 lineage（血统）来重新计算生成丢失的分区数据。