12 Spark 分布式计算

- 问题
  - 数据生成速度远快于处理速度
  - 只能在大规模集群上并行处理
- 大纲
  - 数据流 vs 传统网络编程
  - MapReduce的局限性
  - Spark计算
  - 机器学习的例子
  - Spark生态系统的现状
  - 内置库
- **Spark 与 Hadoop 的区别和联系**
  - 它扩展了广泛使用的 MapReduce 计算模型。高效的支撑更多计算模式，包括交互式查询和流处理。spark 的一个主要特点是能够在内存中进行计算，及时依赖磁盘进行复杂的运算，Spark 依然比 MapReduce 更加高效。
- 数据流与传统网络编程
  - 传统网络编程——节点间消息传递MPI，不咋用啦
  - 数据流模型——MapReduce
    - 限制编程接口，使系统可以做更多的自动化。
    - 将作业表示为高级运算符的图
  - 为什么使用数据流引擎?
    - 编程简单
    - 广泛部署
    - 可扩展到非常大的集群
- **MapReduce的局限性**
  - MapReduce 擅长one-pass计算，但对于多程算法效率低下（只扫描一遍）
  - 没有有效的数据共享原语
    - 步骤之间的状态进入分布式文件系统
    - 由于复制和磁盘存储而变慢
  - 虽然 MapReduce 很简单，但它可能需要渐近更多的通信或 I/O
- Spark计算引擎
  - 使用分布式集合数据结构扩展编程语言
    - 弹性分布式数据集（RDD）
  - 在 Apache 开源

- 有清晰的API
- **Spark 中的 RDD 如何理解**
  - RDD 是 Spark 的核心数据模型，但是个抽象类，全称为 Resillient Distributed Dataset，即弹性分布式数据集。
  - RDD 在抽象上来说是一种元素集合，包含了数据。它是被分区的，分为多个分区，每个分区分布在集群中的不同节点上，从而让 RDD 中的数据可以被并行操作。（分布式数据集）
  - RDD 通常通过 Hadoop 上的文件，即 HDFS 文件或者 Hive 表，来进行创建；有时也可以通过应用程序中的集合来创建。
  - RDD 最重要的特性就是，提供了容错性，可以自动从节点失败中恢复过来。即如果某个节点上的 RDDpartition，因为节点故障，导致数据丢了，那么 RDD 会自动通过自己的数据来源重新计算该 partition。这一切对使用者是透明的。
  - RDD 的数据默认情况下存放在内存中的，但是在内存资源不足时，Spark 会自动将 RDD 数据写入磁盘。（弹性）
  - 不可变的对象集合，分布在集群中
  - 静态类型：RDD [T], 具有 T 类型的对象
  - 跨集群的对象集合，具有用户控制的分区和存储
  - 通过并行转换 构建
  - 只允许您制作 RDD，以便它们可以在失败时自动重建
- 容错
  - 跟踪lineage信息以重建丢失的数据
- 划分——函数

- 机器学习的例子
  - spark比hadoop块

以上内容整理于 幕布文档