# Spark

Spark是为大规模分布式数据处理而设计的一站式计算引擎。

其设计哲学主要为四点：

1. Simple 易用：

提供支持Java、Scala、Python、R的高级API。内部各种数据抽象都基于RDD，Spark为其提供两种算子：Transformation（转换）、Action（行动），相对MR可以更灵活轻松地满足需求。

2. Fast 快速

所有中间结果都可保存在内存中，相对MR减少了大量的磁盘IO，数据复用效率更高，且支持将查询计算构建为DAG，并进行查询优化。

3. Scalable 可扩展

Spark解耦了存储和计算，因此便于扩展。存储方面可以灵活读取在各种数据源中的数据，计算方面可以将任务分发到集群中各个节点进行并行计算。

4. Unified 通用

提供了各种模块，支持多种场景需求，可用于结构式查询（SparkSQL）、流处理（Spark Streaming、Structured Streaming）、机器学习（Spark MLlib）、图计算（GraphX）。

## 应用模块

1. Spark Core

Spark生态系统中的核心数据处理框架，拥有Java、Scala、Python、R四种语言的API，实现了Spark中最基本的功能。其核心数据抽象为RDD，代表了分布在集群中的数据集合。Spark Core通过提供RDD API来对RDD进行定义和操作。此外，任务调度、内存管理、错误恢复、存储系统交互等功能也由Spark Core提供。

2. Spark SQL

Spark处理结构化数据的模块

3. Spark Streaming

4. Structured Streaming

5. MLlib

6. GraphX

## 核心概念

RDD（Resilient Distributed Dataset 弹性分布式数据集）

Spark中的基本数据抽象，是分布式计算的基础。RDD是不可变的、分布式的数据集合，Spark利用它在集群中高效地分布处理数据。

特性

1. 不可变

RDD是不可变的，一旦创建就不能被修改。这意味着无法对RDD的数据进行原地修改，而是通过转换操作创建新的RDD来表示修改后的数据。不可变性有助于确保数据的一致性，同时也能帮助Spark实现容错性。

2. 分区

RDD将数据划分成多个分区，每个分区可以在不同的计算节点上处理，分区是Spark并行计算的基本单元。

3. 惰性求值

RDD是惰性求值的，也就是转换算子不会立即执行，而只有在行动算子触发计算时才会真正执行。这个特点有利于Spark优化计算计划，避免不必要的数据处理开销，提高性能，同时也有利于容错。

4. 容错

RDD通过维护一个表示数据转换历史的依赖关系图来实现容错性。如果某个分区的数据丢失或计算失败，Spark可以根据依赖关系重新计算该分区，确保计算的正确性。

5. 缓存

RDD支持将数据缓存到内存中，以便在后续操作中重复使用。这对于频繁访问相同数据的计算非常有用，可以避免重复计算，也是Spark快速的重要支撑。

属性

1. partitions()

一个分区(Partition)列表，即数据集的基本组成单位。对于 RDD 来说，每个分区都会被一个计算任务处理，分区数决定并行度。用户可以在创建 RDD 时指定 RDD 的分区个数，如果没有指定，那么就会采用默认值。

2. iterator(p, parentIters)

通过输入参数某个分区的标识就可以获得这个分区的数据集合的迭代器。当执行行动算子时，就会

3. dependencies()

一个 RDD 会依赖于其他多个 RDD。RDD 的每次转换都会生成一个新的 RDD，所以 RDD 之间就会形成类似于流水线一样的前后依赖关系。在部分分区数据丢失时，Spark 可以通过这个依赖关系重新计算丢失的分区数据，而不是对 RDD 的所有分区进行重新计算。

4. partitioner()

可选项，对于 KV 类型的 RDD 会有一个 Partitioner，即 RDD 的分区函数，默认为 HashPartitioner。

5. preferredLocations(p)

可选项,一个列表，存储存取每个 Partition 的优先位置(preferred location)。对于一个 HDFS 文件来说，这个列表保存的就是每个 Partition 所在的块的位置。按照"移动数据不如移动计算"的理念，Spark 在进行任务调度的时候，会尽可能选择那些存有数据的 worker 节点来进行任务计算。

创建方式

1. 通过对现有的RDD或DataFrame、Dataset执行转换操作；

2. 从SparkContext创建。

操作类型

1. Transformations（转化算子）

转换算子是对RDD进行变换的操作，生成一个新的RDD作为结果。转换算子是惰性的，不会立即执行，只有在行动算子触发计算时才会执行。

2. Action（行动算子）

行动算子是触发计算的操作，每个Spark程序都必须包含至少一个行动算子。

依赖类型

1. 宽依赖

2. 窄依赖

## 运行机制

1. 部署模式

单机模式：

Local模式

集群模式：

Standalone模式：

Yarn-cluster模式：

Mesos模式（已于3.2.0版本弃用）

客户端模式：

YARN-client模式：

部署模式一览表（摘自《Spark快速大数据分析（Learning Spark, 2nd Edition）》）

|  |  |  |  |
| --- | --- | --- | --- |
| 模式 | Driver | Executor | Cluster Manager |
| Local | 在一个JVM中运行，比如在笔记本计算机或者单节点上运行 | 与Spark驱动器在同一个JVM中运行 | 在同一台主机上运行 |
| Standalone | 可以在集群内的任意节点上运行 | 集群中的每个节点都会启动自己的执行器JVM | 可能随机分配到集群内的任意主机上 |
| YARN-client | 在client内运行，不属于集群的一部分 | YARN的NodeManager所启动的容器 | YARN的ResourceManager与YARN的ApplicationMaster协作，从NodeManager上为Executor分配容器 |
| YARN-cluster | 在YARN的ApplicationMaster内运行 | 与YARN-client模式相同 | 与YARN-client模式相同 |
| Kubernetes（k8s） | 在一个Kubernetes pod内运行 | 各个Executor在各自的pod中运行 | Kubernetes Master |

2. 集群模式下应用运行原理



Spark SQL

PySpark

字典

|  |  |
| --- | --- |
| 术语 | 含义 |
| Application |  |
| Driver |  |
| Executor |  |
|  |  |
|  |  |
|  |  |
|  |  |