Spark RDD

点击一键运行代码→ 获取更多速查表→ 使用说明书 [视频 & 图文]→ bilibili Show Me AI

1. 初始化 Spark

SparkContext

```
> from pyspark import SparkContext
  sc = SparkContext(master = 'local[2]')
```

SparkContext 信息获取

```
> sc.version # 获取 SparkContext 版本
> sc.pythonVer # 获取 Python 版本
> sc.master # 要连接的 MasterURL
> str(sc.sparkHome) #Spark 在工作节点的安装路径
> str(sc.sparkUser()) # 获取 SparkContext 的 Spark 用户名

> sc.appName # 返回应用名称
> sc.applicationId # 获取应用程序 ID
> sc.defaultParallelism # 返回默认并行级别
> sc.defaultMinPartitions #RDD 默认最小分区数
```

配置

```
> from pyspark import SparkConf, SparkContext
  conf = (SparkConf()
          .setMaster("local")
          .setAppName("My app")
          .set("spark.executor.memory", "1g"))
  sc = SparkContext(conf = conf)
```

使用 Shell

PySpark Shell 已经为 SparkContext 创建了名为 sc 的变量。

```
# 命令行启动 spark
$ ./bin/spark-shell --master local[2]

# 命令行提交 spark 脚本任务
$ ./bin/pyspark --master local[4] --py-files code.py
```

用 --master 参数设定 Context 连接到哪个 Master 务器，通过传递逗号分隔列表至 --py-files 添加 Python.zip、.egg 或 .py 文件到 Runtime 路径。

2. 加载数据

并行集合

```
> rdd = sc.parallelize([('a',7),('a',2),('b',2)])
> rdd2 = sc.parallelize([('a',2),('d',1),('b',1)])
> rdd3 = sc.parallelize(range(100))
> rdd4 = sc.parallelize([("a",["x","y","z"]), ("b",["p", "r"])])
```

外部数据

使用 textFile() 函数从 HDFS、本地文件或其它支持 Hadoop 的文件系统里读取文本文件，或使用 wholeTextFiles() 函数读取目录里文本文件。

```
> textFile = sc.textFile("/my/directory/*.txt")
> textFile2 = sc.wholeTextFiles("/my/directory/")
```

3. 提取 RDD 信息

基础信息

```
> rdd.getNumPartitions() # 列出分区数

> rdd.count() # 计算 RDD 实例数量
3

> rdd.countByKey() # 按键计算 RDD 实例数量
defaultdict(<type 'int'>,{'a':2,'b':1})

> rdd.countByValue() # 按值计算 RDD 实例数量
defaultdict(<type 'int'>,
            {('b',2):1,('a',2):1,('a',7):1})

> rdd.collectAsMap() # 以字典形式返回键值
{'a': 2,'b': 2}

> rdd3.sum() #RDD 元素求和
4950

# 检查 RDD 是否为空
> sc.parallelize([]).isEmpty()
True
```

汇总

```
> rdd3.max() #RDD 元素的最大值
99

> rdd3.min() #RDD 元素的最小值
0

> rdd3.mean() #RDD 元素的平均值
49.5

> rdd3.stdev() #RDD 元素的标准差
28.866070047722118

> rdd3.variance() #RDD 元素的方差
833.25

> rdd3.histogram(3) # 分箱 (Bin) 生成直方图
([0,33,66,99],[33,33,34])

# 综合统计包括：计数、平均值、标准差、最大值和最小值
> rdd3.stats()
```

ShowMeAI

4. 应用函数

map 与 flatmap 函数

```
> rdd.map(lambda x: x+(x[1],x[0])).collect() # 对每个 RDD 元素执行函数
[('a',7,7,'a'),('a',2,2,'a'),('b',2,2,'b')]

> rdd5=rdd.flatMap(lambda x: x+(x[1],x[0])) # 对每个 RDD 元素执行函数，并拉平结果
  rdd5.collect()
['a',7,7,'a','a',2,2,'a','b',2,2,'b']

> rdd4.flatMapValues(lambda x: x).collect() # 不改变键，对 rdd4 每个键值对执行 flatMap 函数
[('a','x'),('a','y'),('a','z'),('b','p'),('b','r')]
```

5. 选择数据

获取

```
> rdd.collect() # 返回包含所有 RDD 元素的列表
[('a', 7), ('a', 2), ('b', 2)]

> rdd.filter(lambda x: "a" in x) .collect() # 提取前两个 RDD 元素
[('a',7),('a',2)]

> rdd.first() # 提取第一个 RDD 元素
('a', 7)

> rdd5.distinct().collect() # 提取前两个 RDD 元素
['a',2,'b',7]
```

抽样

```
> rdd3.sample(False, 0.15, 81).collect() # 返回 rdd3 的采样子集
[3,4,27,31,40,41,42,43,60,76,79,80,86,97]
```

筛选

```
> rdd.filter(lambda x: "a" in x) .collect() # 筛选 RDD
[('a',7),('a',2)]

> rdd5.distinct().collect() # 返回 RDD 里的唯一值
['a',2,'b',7]

> rdd.keys().collect() # 返回 RDD 键值对里的键
['a', 'a', 'b']
```

6. 迭代

foreach 函数迭代

```
> def g(x):
      print(x)
  rdd.foreach(g) # 为所有 RDD 应用函数
('a', 7)
('b', 2)
('a', 2)
```

7. 改变数据形状

Reduce 操作

```
> rdd.reduceByKey(lambda x,y : x+y).collect() # 合并每个键的 RDD 值
[('a',9),('b',2)]

> rdd.reduce(lambda a, b: a + b) # 合并 RDD 的值
('a',7,'a',2,'b',2)
```

分组

```
> rdd3.groupBy(lambda x: x % 2).mapValues(list).collect() # 返回 RDD 的分组值
> rdd.groupByKey().mapValues(list).collect() # 按键分组 RDD
[('a',[7,2]),('b',[2])]
```

聚合

```
> seqOp = (lambda x,y: (x[0]+y,x[1]+1))
> combOp = (lambda x,y:(x[0]+y[0],x[1]+y[1]))
> add = (lambda x,y:x+y)

> rdd3.aggregate((0,0), seqOp, combOp) # 汇总每个分区里的 RDD 元素，并输出结果
(4950,100)

> rdd.aggregateByKey((0,0), seqOp, combOp).collect() # 汇总每个 RDD 的键的值
[('a',(9,2)), ('b',(2,1))]

> rdd3.fold(0,add) # 汇总每个分区里的 RDD 元素，并输出结果
4950

> rdd.foldByKey(0, add).collect() # 合并每个键的值
[('a',9),('b',2)]

> rdd3.keyBy(lambda x: x+x).collect() # 通过执行函数，创建 RDD 元素的元组
```

8. 数学运算

RDD 运算

```
> rdd.subtract(rdd2).collect() # 返回在 rdd2 里没有匹配键的 rdd 键值对
[('b',2),('a',7)]
> rdd2.subtractByKey(rdd).collect() # 返回 rdd2 里的每个 ( 键，值 ) 对，rdd 中没有匹配的键
[('d', 1)]
> rdd.cartesian(rdd2).collect() # 返回 rdd 和 rdd2 的笛卡尔积
```

9. 排序

RDD 排序

```
> rdd2.sortBy(lambda x: x[1]).collect() # 按给定函数排序
[('d',1),('b',1),('a',2)]
> rdd2.sortByKey().collect() #RDD 按键排序 RDD 的键值对
[('a',2),('b',1),('d',1)]
```

10. 重分区

repartition 函数

```
> rdd.repartition(4) # 新建一个含 4 个分区的 RDD
> rdd.coalesce(1) # 将 RDD 中的分区数缩减为 1 个
```

12. 终止 SparkContext

停止 SparkContext

```
> sc.stop()
```

13. 执行脚本程序

提交脚本执行

```
$ ./bin/spark-submit examples/src/main/python/pi.py
```

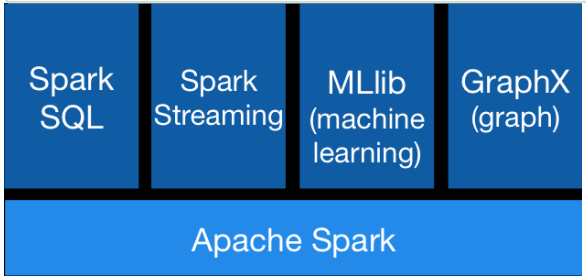
Spark 是基于内存计算的大数据并行计算框架。它包含 MapReduce 计算模型，而且高效地支持更多计算模式，包括交互式查询和流处理。

Spark 适用于各种各样原先需要多种不同的分布式平台的场景，包括批处理、迭代算法、交互式查询、流处理。

Spark 生态系统已经发展成为一个包含多个子项目的集合，其中包含 SparkSQL、Spark Streaming、GraphX/ GraphFrame、MLlib、SparkR 等子项目。

PySpark 是 Spark 的 PythonAPI，允许 Python 调用 Spark 编程模型。

作者 | 韩信子 @ShowMeAI
设计 | 南 乔 @ShowMeAI
参考 | datacamp cheatsheet



11. 保存

存储 RDD 到本地或 HDFS

```
> rdd.saveAsTextFile("rdd.txt")
> rdd.saveAsHadoopFile("hdfs://namenodehost/parent/child", 'org.apache.hadoop.mapred.TextOutputFormat')
```