呆鸟译

Python 数据科学 速查表

PySpark - RDD 基础

天善智能 商业智能与大数据社区 www.hellobi.com

Spark

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

初始化 Spark

SparkContext

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

核查 SparkContext

```
>>> sc.version                    获取 SparkContext 版本
>>> sc.pythonVer                  获取 Python 版本
>>> sc.master                     要连接的 Master URL
>>> 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。

```
$ ./bin/spark-shell --master local[2]
$ ./bin/pyspark --master local[4] --py-files code.py
```

用 --master 参数设定 Context 连接到哪个 Master 服务器，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。

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/")
```

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()                      汇总 RDD 元素
 4950
>>> sc.parallelize([]).isEmpty()    检查 RDD 是否为空
  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()                 计算 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])
>>> rdd3.stats()                    综合统计
                                    包括：计数、平均值、标准差、最大值和最小值
```

应用函数

```
>>> rdd.map(lambda x: x+(x[1],x[0]))      对每个 RDD 元素执行函数
       .collect()
  [('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)       不改变键，对 rdd4 的每个键值对执行 flatMap 函数
       .collect()
  [('a','x'),('a','y'),('a','z'),('b','p'),('b','r')]
```

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.take(2)                     提取前两个 RDD 元素
  [('a', 7), ('a', 2)]
>>> rdd.first()                     提取第一个 RDD 元素
  ('a', 7)
>>> rdd.top(2)                      提取前两个 RDD 元素
  [('b', 2), ('a', 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)     筛选 RDD
       .collect()
  [('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']
```

迭代

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

改变数据形状

规约

```
>>> rdd.reduceByKey(lambda x,y : x+y)     合并每个键的 RDD 值
       .collect()
  [('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)         返回 RDD 的分组值
        .mapValues(list)
        .collect()
>>> rdd.groupByKey()                      按键分组 RDD
       .mapValues(list)
       .collect()
  [('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]))
>>> 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)   汇总每个 RDD 的键的值
       .collect()
  [('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)                 合并每个键的值
       .collect()
  [('a',9),('b',2)]
>>> rdd3.keyBy(lambda x: x+x)             通过执行函数，创建 RDD 元素的元组
       .collect()
```

数学运算

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

排序

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

重分区

```
>>> 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
```

保存

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

终止 SparkContext

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

执行程序

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