DIST

Spark RDD详解

颜 晖
2017.4.10

目录 CONTENTS

1 RDD概念及理解

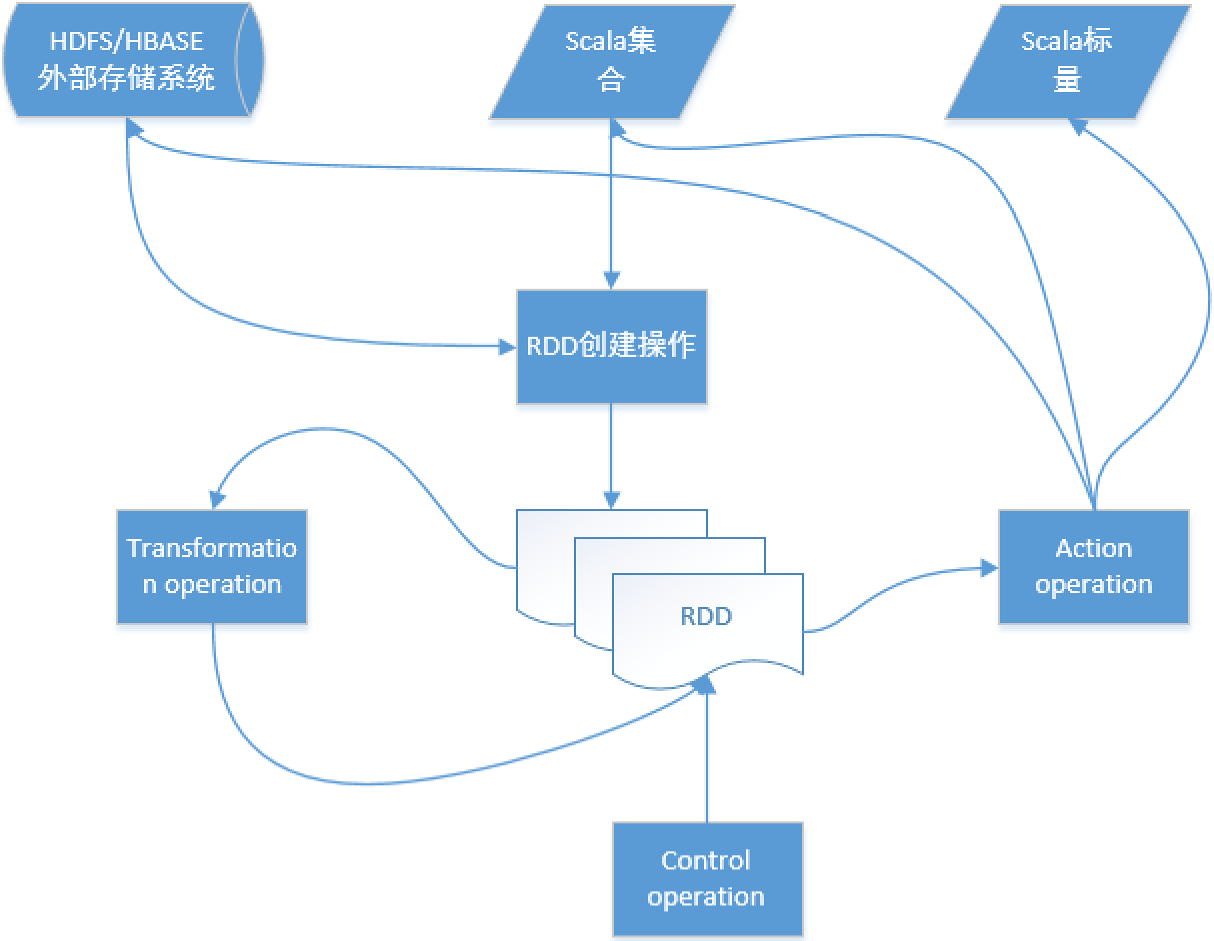
2 RDD特性解读

3 RDD重要接口

4 RDD的基本使用

1、RDD概念及理解

- Spark程序流程



1、RDD概念及理解

- RDD （Resilient Distributed Dataset）弹性分布式数据集
  - □ 元素的（只读）集合；
  - □ 集合是基于分区（partition）的；
  - □ 支持并行化操作；
  - □ 看作是驻于内存的Spark对象；

2、RDD特性解读

- 只读、抽象的数据集
  - □ 只支持粗粒度的操作，应用在RDD的所有数据上；
- 高容错性
  - □ 通过构建RDD的继承关系（lineage），丢失或操作失败后可以重建；

2、RDD特性解读

- 基于分区的
  - □ 一个RDD会有一个或多个分区（partition）；
- 实现自定义的RDD

3、RDD重要接口

- partttion
  - □ 分区的多少决定这并行计算的粒度；
  - □ 对每一个RDD分区的操作都在一个单独的任务中执行；

```
scala> val rdd = sc.parallelize(1 to 100,2)
rdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[5] at parallelize at <console>:21


scala> rdd.partitions.size
res7: Int = 2
```

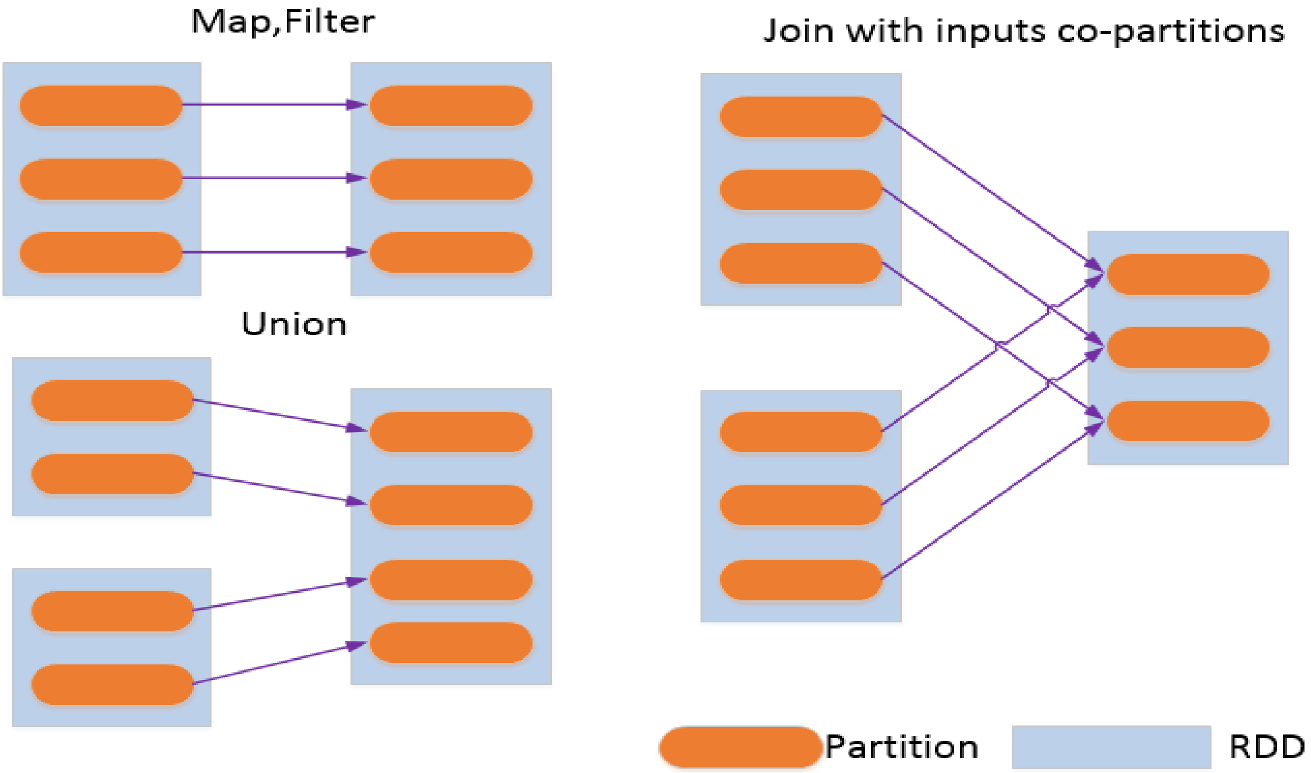
3、RDD重要接口

- preferredLocations
  - 每个partiton所存储的位置，与spark中的调度有关;

```
scala> val rdd=sc.textFile("hdfs://23.36.126.11/user/cqgps/data/gps/merge/alg
orithm/gpsspeed/index/2017/04/08/20170408135000.txt/p*")

scala> val hadoopRDD = rdd.dependencies(0).rdd
hadoopRDD: org.apache.spark.rdd.RDD[_] = hdfs://23.36.126.11/user/cqgps/
data/gps/merge/algorithm/gpsspeed/index/2017/04/08/20170408135000.txt/p*
HadoopRDD[0] at textFile at <console>:21

scala> hadoopRDD.partitions.size
17/04/08 14:49:02 INFO mapred.FileInputFormat: Total input paths to process
: 1
res3: Int = 2

scala> hadoopRDD.preferredLocations(hadoopRDD.partitions(0))
res4: Seq[String] = ListBuffer(datasvr2.bigdata.cqtpi.org, datasvr5.bigdata.cqt
pi.org)
```

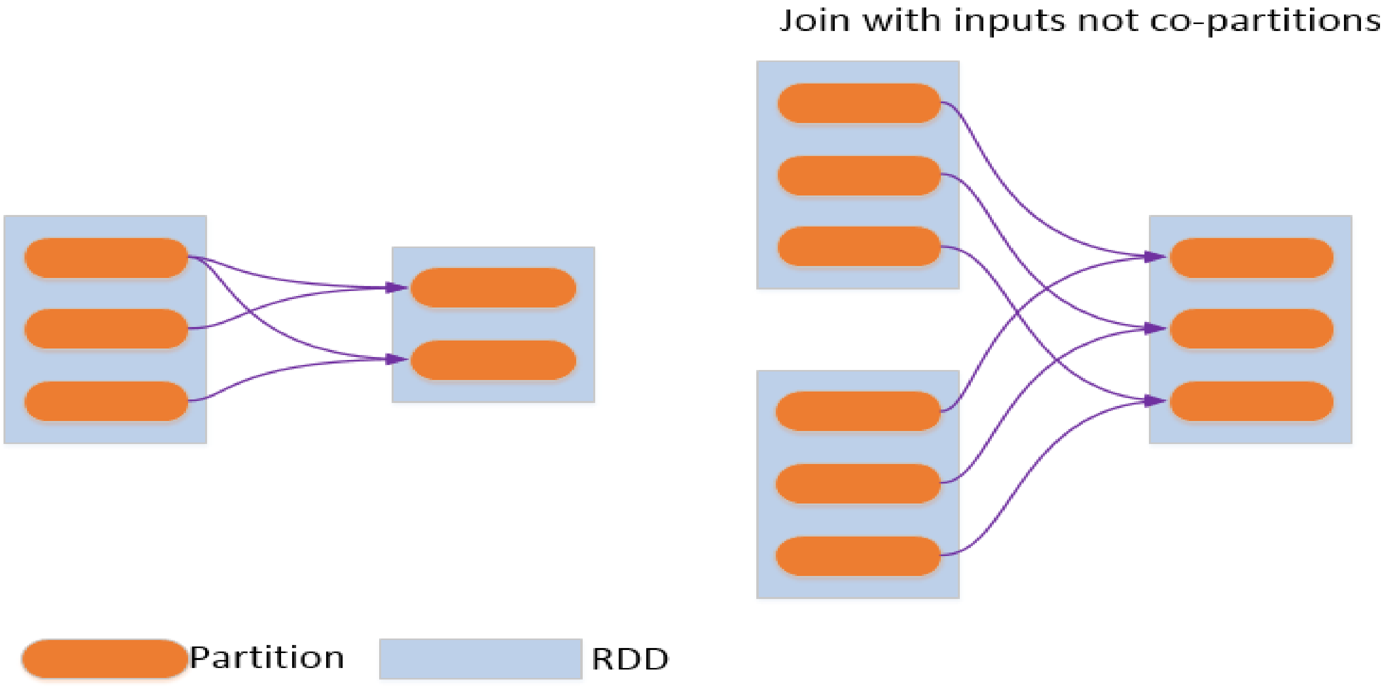
3、RDD重要接口

- dependencies
  - 窄依赖（Narrow Dependencies）：每一个父RDD的分区最多只被子RDD的一个分区所使用。



3、RDD重要接口

- dependencies
  - 宽依赖（Wide Dependencies）：多个子RDD的分区会依赖于同一个父RDD的分区。



3、RDD重要接口

- pageRank算法示例

```
val sparkConf = new SparkConf().setMaster("local[*]").setAppName(“pageRa
nk")
val sparkContext  = new SparkContext(sparkConf)

val links  = sparkContext.parallelize(Array(("A",Array("D")),("B",Array("A")),("C"
,Array("A","B")),("D",Array("A","C"))),2)
  .map(x => (x._1,x._2)).cache
var ranks  = sparkContext.parallelize(Array(("A",1.0),("B",1.0),("C",1.0),("D",1.0
)),2)

for( i <- 1 to 3){
  val contribs  = links.join(ranks,2).flatMap{
    case (url,(links,rank))  => links.map(dest  => (dest,rank/links.size))
  }
  ranks  = contribs.reduceByKey(_  + _,2).mapValues(0.15  + 0.85*_)
}

ranks.saveAsTextFile("file:///I:/data/ranks")
```

3、RDD重要接口

- 分区函数partitioner
  - □ 只存在于（K,V）类型的RDD中，对于非（K,V）类型的RDD来说该属性为None；

```
scala> val rdd = sc.makeRDD(1 to 10,2).map( x=> (x,x))
rdd: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[13] at map at <
console>:21

scala> rdd.partitioner
res10: Option[org.apache.spark.Partitioner] = None

scala> val group_rdd = rdd.groupByKey(new org.apache.spark.HashPartition
er(3))
group_rdd: org.apache.spark.rdd.RDD[(Int, Iterable[Int])] = ShuffledRDD[14] a
t groupByKey at <console>:23

scala> group_rdd.partitioner
res11: Option[org.apache.spark.Partitioner] = Some(org.apache.spark.HashP
artitioner@3)
```

4、RDD基本使用

- RDD创建

  - 由scala集合创建，例如parallelize、makeRDD 方法

```
scala> val collect = Seq((1 to 10,Seq("host1","host3")),(11  to 20,Seq("hos
t2")))

collect: Seq[(scala.collection.immutable.Range.Inclusive,  Seq[String])]  =
List((Range(1,  2, 3, 4, 5, 6, 7, 8, 9, 10),List(host1,  host3)),  (Range(11,  12
, 13, 14, 15, 16, 17, 18, 19, 20),List(host2)))

scala> val rdd = sc.makeRDD(collect)
rdd: org.apache.spark.rdd.RDD[scala.collection.immutable.Range.Inclusi
ve] = ParallelCollectionRDD[15]  at makeRDD  at <console>:23

scala> rdd.preferredLocations(rdd.partitions(0))
res15: Seq[String] = List(host1,  host3)

scala> rdd.preferredLocations(rdd.partitions(1))
res14: Seq[String] = List(host2)
```

4、RDD基本使用

■ RDD创建

□ 从HDFS、HBASE中读取文件创建RDD，例如hadoopRDD、newwHadoopRDD

◆ 主要参数:

✓ InputFormat

✓ 键类型

✓ 值类型

✓ 分区值

4、RDD基本使用

- transformation operation

□ map

defmap[U](f: (T) ⇒ U)(*implicit* arg0: ClassTag[U]): RDD [U]

□ distinct

def distinct(): RDD[T]

□ flatMap

def flatMap[U](f: (T) ⇒ TraversableOnce[U]): RDD[U]

□ coalesce

def coalesce(numPartitions: Int, shuffle: Boolean = false): RDD[T]

□ repartition

def repartition(numPartitions: Int): RDD[T]

4、RDD基本使用

- transformation operation

□ randomSplit

def randomSplit(weights: Array[Double], seed: Long = Utils.random.nextLong): Array[RDD[T]]

□ glom

def glom(): RDD[Array[T]]

□ union

def union(other: RDD[T]): RDD[T]

□ intersection

def intersection(other: RDD[T]): RDD[T]

□ subtract

def subtract(other: RDD[T]): RDD[T]

4、RDD基本使用

- transformation operation

□ mapPartitions

def mapPartitions[U](f: (Iterator[T]) ⇒ Iterator[U], preservesPartitioning: Boolean = false)

□ mapPartitionsWithIndex

def mapPartitionsWithIndex[U](f: (Int, Iterator[T]) ⇒ Iterator[U], preservesPartitioning: Boolean = false)(implicit arg0: ClassTag[U]): RDD[U]

□ zip

def zip[U](other: RDD[U]): RDD[(T, U)]

□ zipPartitions

def zipPartitions[B, C, D, V](rdd2: RDD[B], rdd3: RDD[C], rdd4: RDD[D])(f: (Iterator[T], Iterator[B], Iterator[C], Iterator[D]) ⇒ Iterator[V]): RDD[V]

□ zipWithIndex

zipWithIndex(): RDD[(T, Long)]

4、RDD基本使用

- transformation operation
  - □ zipWithUniqueId

    zipWithUniqueId(): RDD[(T, Long)]

  - □ partitionBy

    def partitionBy(partitioner: Partitioner): RDD[(K, V)]

  - □ mapValues

    def mapValues[U](f: (V) ⇒ U): RDD[(K, U)]

  - □ flatMapValues

    def flatMapValues[U](f: (V) ⇒ TraversableOnce[U]): RDD[(K, U)]

  - □ combineByKey

    def combineByKey[C](createCombiner: (V) ⇒ C, mergeValue: (C, V) ⇒ C, mergeCombiners: (C, C) ⇒ C): RDD[(K, C)]

4、RDD基本使用

- transformation operation
  - □ foldByKey

    def foldByKey(zeroValue: V)(func: (V, V) ⇒ V): RDD[(K, V)]

  - □ reduceByKey

    def reduceByKey(func: (V, V) ⇒ V): RDD[(K, V)]

  - □ groupByKey

    def groupByKey(): RDD[(K, Iterable[V])]

  - □ cogroup
  - □ join
  - □ leftOuterJoin
  - □ rightOuterJoin
  - □ subtractByKey

4、RDD基本使用

- control operation

  - □ persist

    def reduceByKey(func: (V, V) ⇒ V): RDD[(K, V)]

    有不同的storage level

  - □ cache

    将RDD持久化，storage level为内存

  - □ checkpoint

    将RDD持久化在HDFS中，会切断RDD之前的依赖关系；

4、RDD基本使用

- action operation

✓ 将标量或者集合返回给Spark的客户端程序

✓ 将RDD直接保存到外部文件系统或者数据库中

□ first

□ count

□ reduce

□ collect

□ take

□ top

□ takeOrdered

4、RDD基本使用

- action operation
  - □ aggregate
  - □ fold
  - □ lookup
  - □ saveAsTextFile
  - □ saveAsObjectFile
  - □ saveAsHadoopFile

- 问答&交流

http://spark.apache.org/docs/1.4.1/api/scala/index.html#package

Thanks