Spark总结&分享

邓志华

目录

1. Spark RDD
2. Spark Stage && Task
3. Spark如何执行用户定义的函数
4. Spark Tune

1. Spark RDD

RDD(Resilient Distributed Datasets)

A read-only, partitioned collection of records.

(1) Fault Tolerance

RDD has enough information about how it was derived from other RDDs.

(2) Coarse-grained Transformations

A good fit for many parallel applications.

(3) Data Reuse

Refer to: http://www-bcf.usc.edu/~minlanyu/teach/csci599-fall12/papers/nsdi_spark.pdf

RDD is an Interface

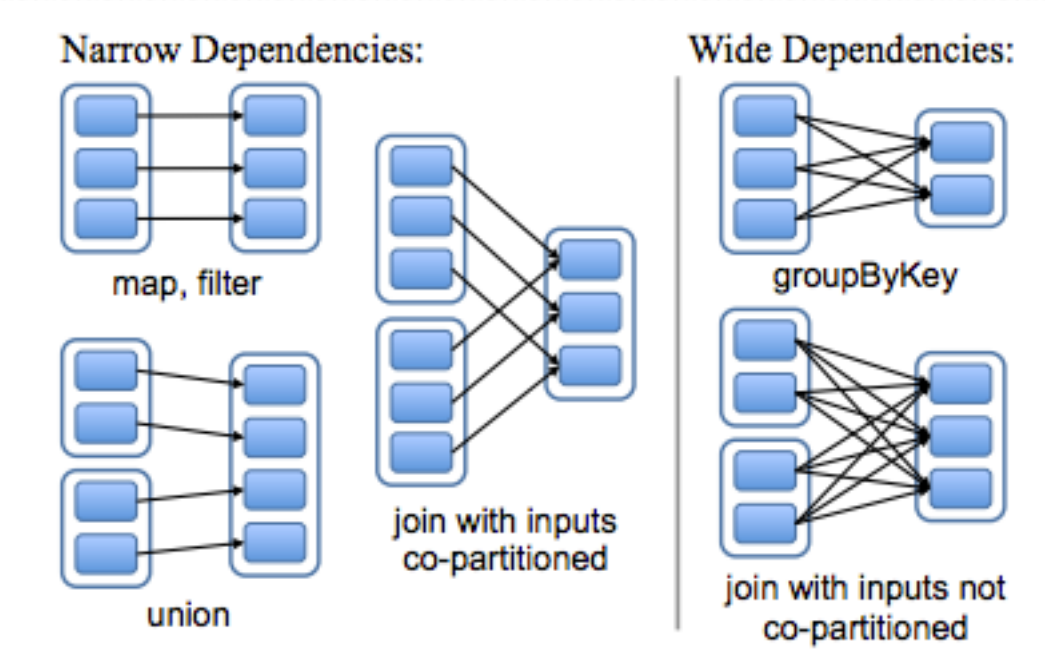
基本要素：

1. Set of partitions (“splits” in Hadoop)
2. List of dependencies on parent RDDs
3. Function to compute a partition (as an Iterator) given its parent(s)

优化要素

4. (Optional) partitioner (hash, range)
5. (Optional) preferred location(s) for each partition

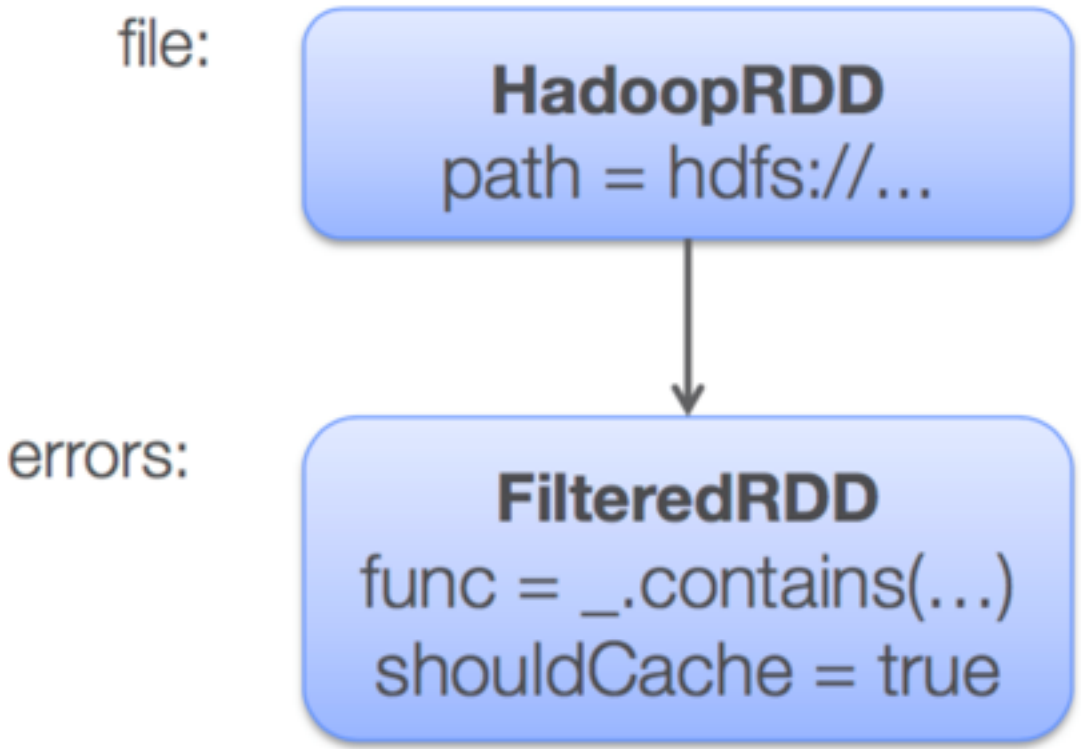
RDD&RDD



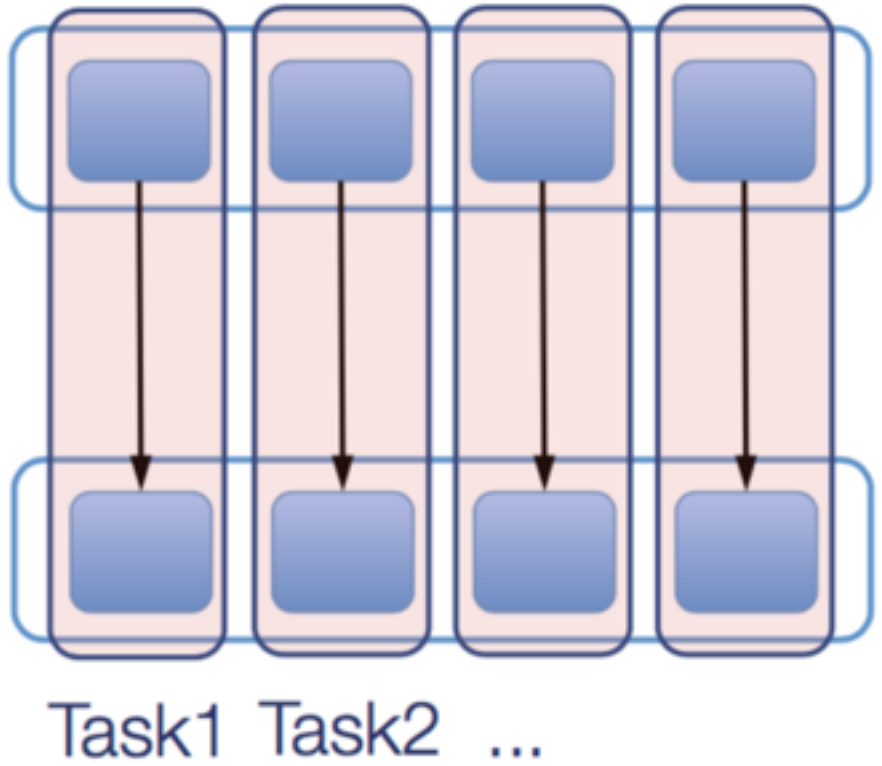
Each box is an RDD, with partitions shown as shaded rectangles.

RDD DAG

Dataset-level view:



Partition-level view:



RDD Op

```
val conf = new SparkConf().setAppName("Spark Pi").setMaster("local")
val spark = new SparkContext(conf)
val wcount = spark.textFile("file:///Users/yixin/test").flatMap(line => line.split("\\s+"))
```

```
def flatMap[U: ClassTag](f: T => TraversableOnce[U]): RDD[U] = withScope {
 val cleanF = sc.clean(f)
 new MapPartitionsRDD[U, T](this, (context, pid, iter) => iter.flatMap(cleanF))
}
```

```
private[spark] class MapPartitionsRDD[U: ClassTag, T: ClassTag](
  extends RDD[U](prev) {          定义了其依赖的RDD
  override def compute(split: Partition, context: TaskContext): Iterator[U] =
    f(context, split.index, firstParent[T].iterator(split, context))
=>RDD( ...).flatMap(line=>line.split("\\s+")
}
```

RDD

1， 每个RDD使用迭代器模式使外部用户访问其内部的数据。

2， RDD实例保存了用户定义的具体逻辑。

3， 调用每个RDD的compute方法并不会生成属于该RDD实例上的数据。

4， RDD上保存了其依赖的关系.(ShuffleDependency, OneToOneDependency...)。

5，一个作业的提交 <=> rdd.action(驱动Iterator)

Stage&Task

查找每个Stage的parent stages

将每个Stage中最尾的rdd放在栈中

while(栈不为空) {

1，弹出栈顶rdd

2，该rdd是否被访问过，如果没有，则依次进行第3,4,5步骤

3，遍历rdd的dependency，如果该dependency为ShuffleDependency，否则，跳到第5步

3,0 如果该dependency已被解析，则返回解析好的ShuffleMapStage

3,1 以ShuffleDependency中rdd0为跟节点，查找可达的ShuffleDependency，如果一个dependency还未被解析，则将其加入到一个栈中。

3,2 遍历上一步得到的栈，以每个ShuffleDependency中rdd为基础，查找其依赖的parent stages，创建一个ShuffleMapStage，加入到已经解析的Map中。

3,3 查找得到rdd0的parent stages，创建一个ShuffleMapStage，并返回。

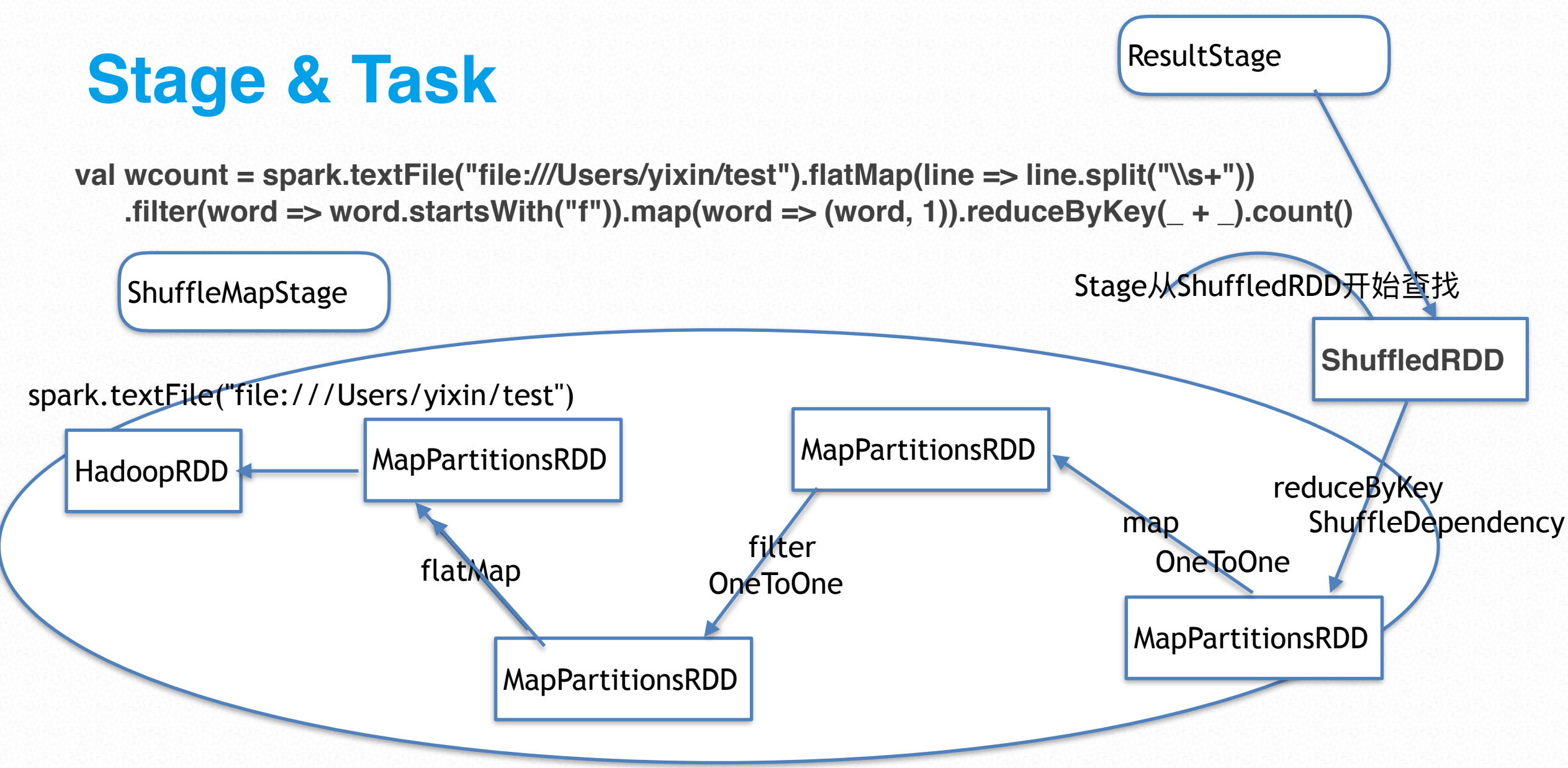
4，将第3步返回的stage加入的parent stages列表中

5，将该rdd压入栈中

}

Stage & Task

```
val wcount = spark.textFile("file:///Users/yixin/test").flatMap(line => line.split("\\s+"))
    .filter(word => word.startsWith("f")).map(word => (word, 1)).reduceByKey(_ + _).count()
```



Stage&Task

Stage提交：

1，从ResultStage开始，递归调用，当一个Stage所有的parent stages都完成时，这个stage才有机会被提交。

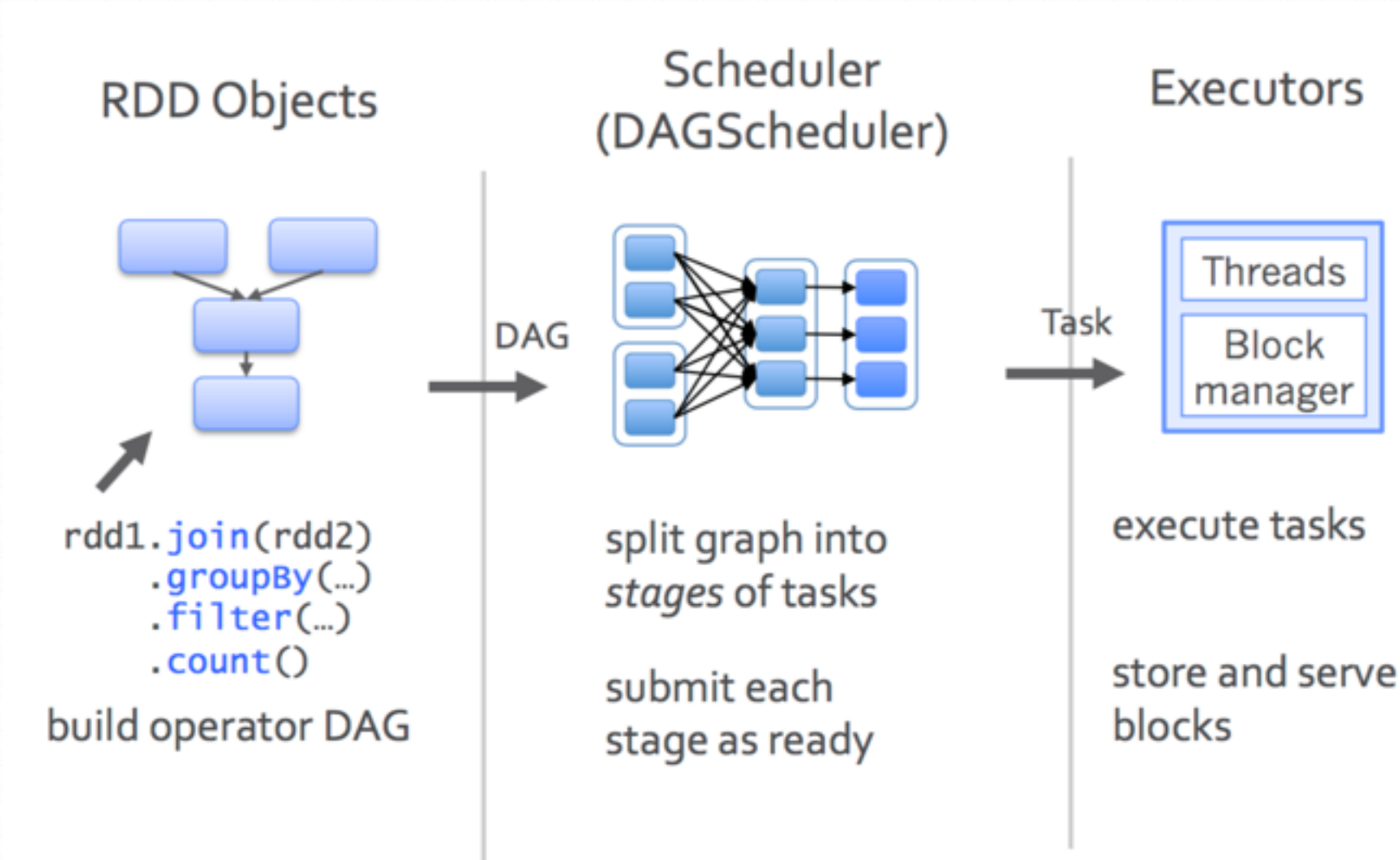
2，将stage最尾部的RDD序列化，并将序列化得到的子节数组初始化相应的一组Task。

```
case stage: ShuffleMapStage =>
      closureSerializer.serialize((stage.rdd, stage.shuffleDep): AnyRef).array()
case stage: ResultStage =>
      closureSerializer.serialize((stage.rdd, stage.func): AnyRef).array()
```

3，将这样的一组Task提交到Task任务管理器中，根据不同的部署模式，序列化Task 后进行分发。

4，Executor发序列化得到Task之后，在通过反序列化，得到RDD以及相应的信息。

总结

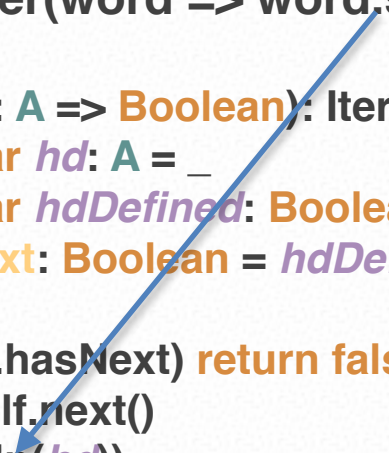

RDD Objects
Scheduler
(DAGScheduler)
Executors
DAG
Task
Threads
Block
manager
rdd1.join(rdd2)
.groupBy(…)
.filter(…)
.count()
build operator DAG
split graph into
stages of tasks
submit each
stage as ready
execute tasks
store and serve
blocks

User Defined Logic

```
spark.textFile(“file:///…”).filter(word => word.startsWith("f"))
```

```
MapPartitionsRDD:
 override def compute(split: Partition, context: TaskContext): Iterator[U] =
     var iter = RDD(spark.textFile(“file:///…”)).iterator(split,context)
     iter.filter(word => word.startsWith(“f"))

def filter(p: A => Boolean): Iterator[A] = new AbstractIterator[A] {
  private var hd: A = _
  private var hdDefined: Boolean = false
 def hasNext: Boolean = hdDefined || {
    do {
      if (!self.hasNext) return false
      hd = self.next()
    } while (!p(hd))
    hdDefined = true
    true
  }
  def next() = if (hasNext) { hdDefined = false; hd } else empty.next()
}
```



User Defined Logic

```
val conf = new SparkConf().setAppName("Spark Pi").setMaster("local")
val spark = new SparkContext(conf)
val wcount = spark.textFile("file:///Users/yixin/test").flatMap(line =>
line.split("\\s+"))
     .filter(word => word.startsWith("f")).map(word => (word, 1))
 wcount.reduceByKey(_ + _).count()

val patitions = wcount.mapPartitions(iter => {
  var buf = new ListBuffer[String]();
  while (iter.hasNext) {
    var (word, count) = iter.next()
     buf += (word + "_" + count)
  }
  buf.toList.toIterator
})

val iter: Iterator[String] = patitions.compute(new Partition {
  override def index: Int = wcount.partitions.apply(0).index
},
 new TaskContextImpl(0, 0, 0, 0,
    new TaskMemoryManager(SparkEnv.get.memoryManager, 0),
        SparkEnv.get.metricsSystem,
        internalAccumulators = Seq.empty))

while (iter.hasNext) {
   println("--------->" + iter.next())
}

patitions.saveAsTextFile("file///user/yixin/res")
```

RDD的定义

得到某个RDD实例上的迭代器

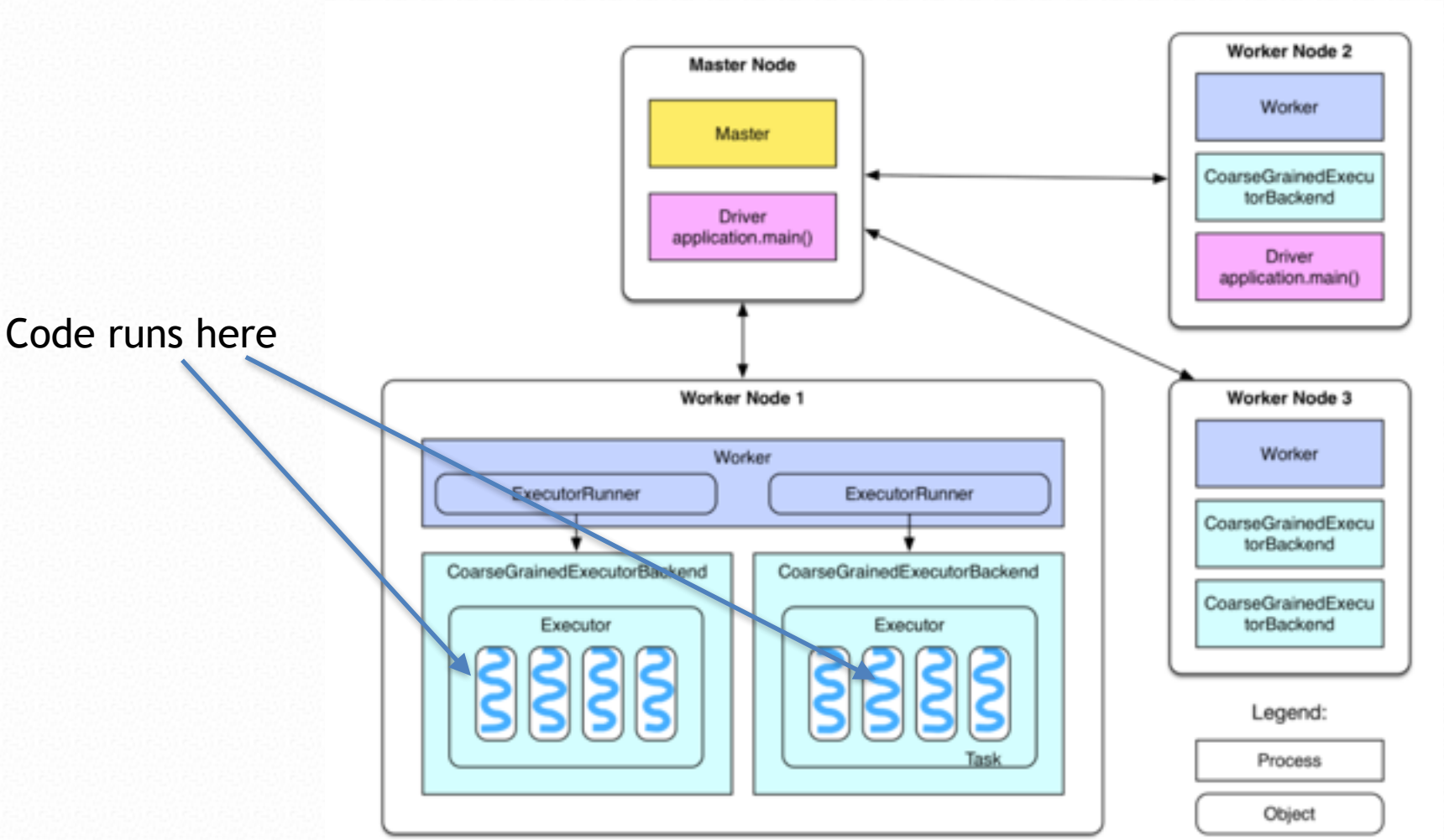
遍历RDD的纪录

User Defined Logic

```
var counter = 0
var rdd = sc.parallelize(data)

// Wrong: Don't do this!!
rdd.foreach(x => counter += x)
println("Counter value: " + counter)
```

User Defined Logic



Spark&MapReduce

1，相同处

Distributed，Batch…

2，不同处

计算函数，Shuffle，Data...

Spark Tune

1，GC

2，Serialization

3，RDD Reuse

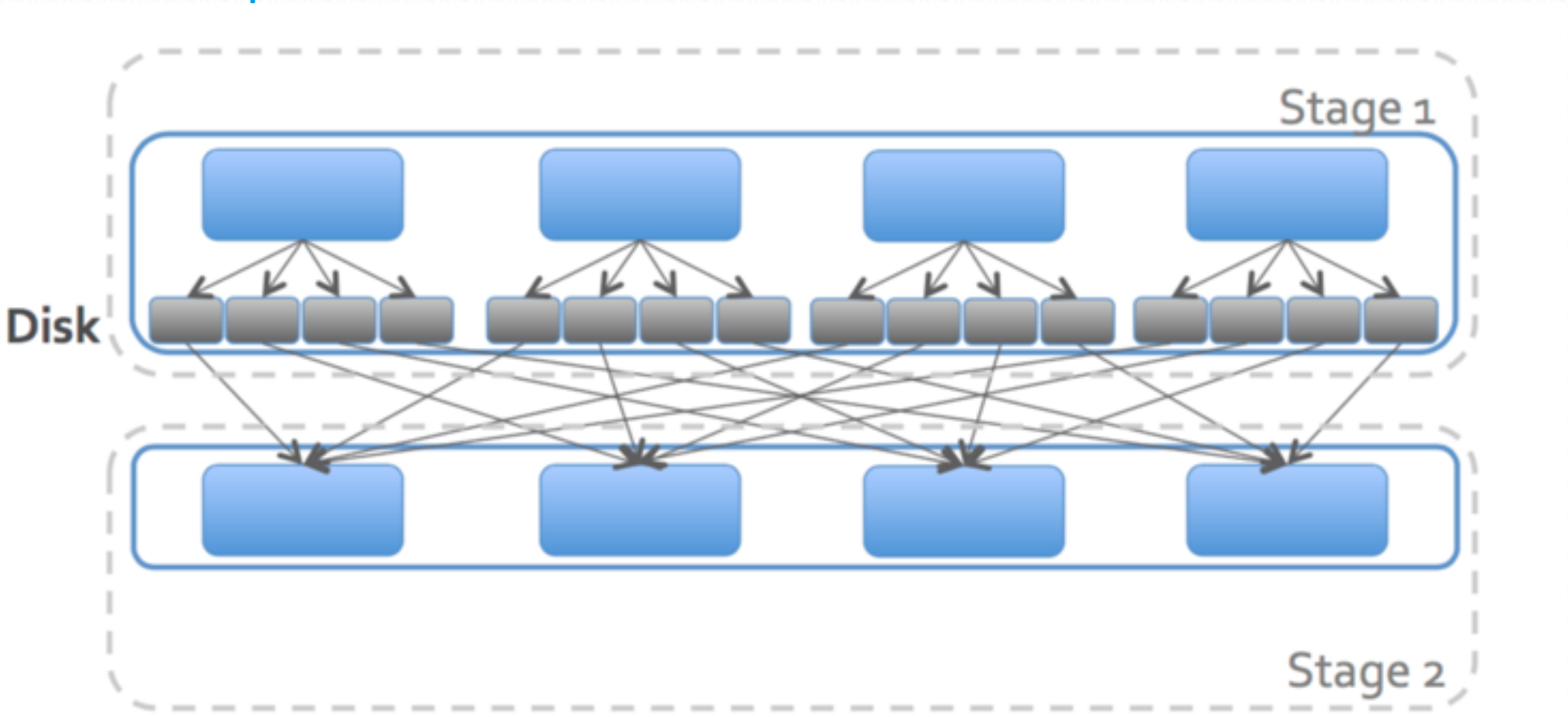
4，广播大变量 (>20K)

5，计算资源分配

6，Turn speculation on to mitigates Stragglers

Spark Tune

Avoid Shuffle if possible



Q&A

宜信 CreditEase 10周年 2006-2016

Q: ?

A: ……

谢谢！