1. spark shuffle过程

spark中管理shuffle的过程有一个shuffleManage负责管理，在spark 2.X 之后,主要负责的是sortshufflemanager，其中主要的是sortshuffle,shuffle分为两个阶段,shufflewriter和shuffleread

a) map task 的计算结果会写入到一个内存数据结构里面，内存数据结构默认是5M

b) 在shuffle的时候会有一个定时器，不定期的去估算这个内存结构的大小，当内存结构中的数据超过5M时，比如现在内存结构中的数据为5.01M，那么他会申请5.01*2-5=5.02M内存给内存数据结构。

c) 如果申请成功不会进行溢写，如果申请不成功，这时候会发生溢写磁盘。

d) 在溢写之前内存结构中的数据会进行排序分区

e) 然后开始溢写磁盘，写磁盘是以batch的形式去写，一个batch是1万条数据，

f) map task执行完成后，会将这些磁盘小文件合并成一个大的磁盘文件，同时生成一个索引文件。

g) reduce task去map端拉取数据的时候，首先解析索引文件，根据索引文件再去拉取对应的数据。

> https://blog.csdn.net/qq_21835703/article/details/79104733

2. spark streaming 读取kafka数据的两种方式

这两种方式分别是：

1. Receiver-base

使用Kafka的高层次Consumer API来实现。receiver从Kafka中获取的数据都存储在Spark Executor的内存中，然后Spark Streaming启动的job会去处理那些数据。然而，在默认的配置下，这种方式可能会因为底层的失败而丢失数据。如果要启用高可靠机制，让数据零丢失，就必须启用Spark Streaming的预写日志机制（Write Ahead Log，WAL）。该机制会同步地将接收到的Kafka数据写入分布式文件系统（比如HDFS）上的预写日志中。所以，即使底层节点出现了失败，也可以使用预写日志中的数据进行恢复。

2. Direct

Spark1.3中引入Direct方式，用来替代掉使用Receiver接收数据，这种方式会周期性地查询Kafka，获得每个topic+partition的最新的offset，从而定义每个batch的offset的范围。当处理数据的job启动时，就会使用Kafka的简单consumer api来获取Kafka指定offset范围的数据。(使用比received更底层的api)

3.Spark提供的两种共享变量

Spark 程序的大部分操作都是 RDD 操作，通过传入函数给 RDD 操作函数来计算，这些函数在不同的节点上并发执行，内部的变量有不同的作用域，不能相互访问，有些情况下不太方便。

1. 广播变量，**是一个只读对象**，在所有节点上都有一份缓存，创建方法是 SparkContext.broadcast()。创建之后再更新它的值是没有意义的，一般用 val 来修改定义。

2. 计数器，**只能增加，可以用计数或求和，支持自定义类型。**创建方法是 SparkContext.accumulator(V, name)。只有 Driver 程序可以读这个计算器的变量，RDD 操作中读取计数器变量是无意义的。但节点可以对该计算器进行增加（？？？）

以上两种类型都是 Spark 的共享变量。

> https://zhuanlan.zhihu.com/p/49169166

4. 解释一下Spark Master的选举过程

与hadoop一样，spark也存在单点故障问题，为此，spark的standalone模式提供了master的HA，与hadoop一样，一个是active，一个是standby状态，当active挂了，standby会顶上。

spark HA的主备切换主要基于两种机制：

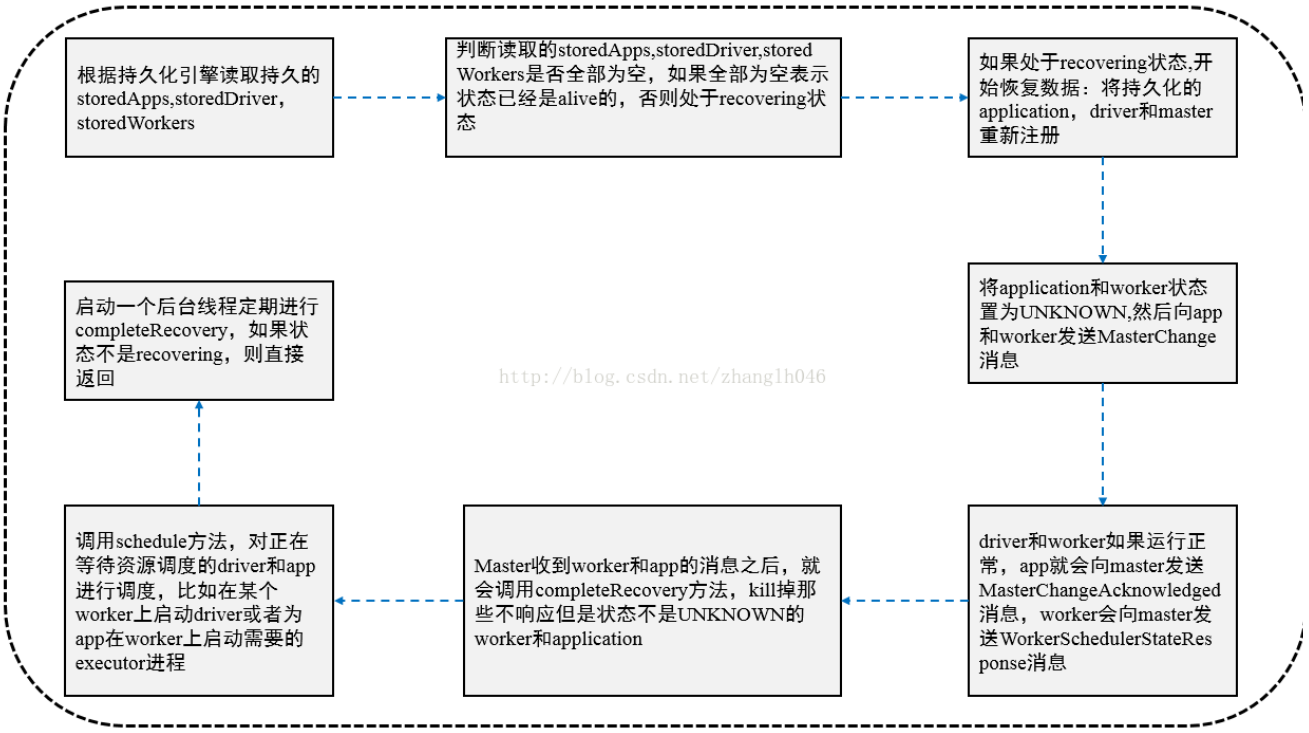
1. 基于文件系统

2. 基于zk集群。

前者在挂了后需要手动切换，而基于zk的HA可以自动实现切换。策略可以通过配置文件spark-env.sh配置spark.deploy.recoveryMode

切换的过程如下：

首先，standby的master去读取持久化（可能是磁盘或者zk）的storedAPP，storeddriver，storedworker的相关信息。读取后，如果storedAPP，storeddriver，storedworker有任何一个非空，那么就会启动master恢复机制，将持久化的APP，driver，worker信息重新注册到内存中，将application和worker的状态修改为UNknown，然后向该APP对应的driver，worer发送自己的地址信息，driver，worer如果没有挂，那么在接收到master发送的地址后，就会返回响应消息给新的master。此时，master在接收到driver，worker的消息后，会使用completeRecory对没有响应的组件进行清理，最后调用master的schedul方法，对正在等待的driver和app调度，如启动executor。



> 原文链接：https://blog.csdn.net/englishsname/article/details/50631372
> https://blog.csdn.net/zhanglh046/article/details/78485745

5. Spark Streaming小文件问题

使用 Spark Streaming 时，如果实时计算结果要写入到 HDFS，那么不可避免的会遇到一个问题，那就是在默认情况下会产生非常多的小文件，这是由 Spark Streaming 的微批处理模式和 DStream(RDD) 的分布式(partition)特性导致的，Spark Streaming 为每个 Partition 启动一个独立的线程（一个 task/partition 一个线程）来处理数据，一旦文件输出到 HDFS，那么这个文件流就关闭了.

处理 Spark Streaming 小文件的典型方法(大致如下):

1. 增加 batch 大小
2. Coalesce大法好 ， 小文件的基数是 batch_number * partition_number, 所以就是合并分区数
3. Spark Streaming 外部来处理: 用 Hive 或者 Spark Sql 这样的“sql on hadoop”系统类进一步进行数据分析，而这些表一般都是按照半小时或者一小时、一天
4. 自己调用 foreach 去 append, 写入文件时,不要新建,而是追加已有的

> https://zhuanlan.zhihu.com/p/49169166