一 要点

1 mapper() 分片中的每一个kv都要调用一次，多个map的数据都是内部有序外部无序

2 reduce() 相同的key 放到一起(s,(1,1,1,1,))只调用一次，且reduce拉取数据时也需要进行排序

3 如果mapper的输入输出和reduce一致，mapper可以不用

4 **combine**:

1. 在从环形缓冲区（100M 达到80时进行溢写）溢写（分区、排序）到本地时就会进行combine。
2. 若一个map产生多个溢写文件时，在多个小文件合并成一个大文件时也会进行溢写。

5 自定义分区（partition）

在程序main方法中job.setPartitionerClass(xxx.class ) ，且需要设定reduce的个数， hash分区容易造成数据倾斜，且一个reduce对应一个输出文件，默认按照字典序排序。

6 二次排序针对key 从缓冲区写入文件需要使用该排序

1. 首先自定义排序的实体类需要实现writeableComparable 接口（该接口实现了writeable接口和comparable接口） 传入对应泛型
2. 重写compareTo方法，实现自定义排序规则。
3. 重写write(DataOutput dataOutput)方法（序列化）和readFields(DataInput dataInput)（反序列化），注意在两个方法中字段的顺序需要一致，避免序列化出现问题。
4. 在map中将数据与实体类中的字段进行对应，并创建实体类对象，在其内部实现排序对比，最终context.write(person, NullWritable.get());

7 数据倾斜（分区倾斜）？

1. **数据倾斜的诊断** 在Reduce类中的setup方法中设置阈值 ，将大于此阈值的key进行打印。

7.2 数据倾斜解决方案

（1）根据数据key的分布实现自定义分区。

（2）使用combine操作，来减少数据输入量达到合并精简数据的目的。

（3）抽样和范围分区

Hadoop默认的分区器是HashPartitioner，基于map输出键的哈希值分区。这仅在数据分布比较均匀时比较好。在有数据倾斜时就很有问题。

使用分区器需要首先了解数据的特性。TotalOrderPartitioner中，可以通过对原始数据进行抽样得到的结果集来预设分区边界值。TotalOrderPartitioner中的范围分区器可以通过预设的分区边界值进行分区。因此它也可以很好地用在矫正数据中的部分键的数据倾斜问题。

分布式缓存：将数据缓存到计算节点数据量不能大，类似广播变量，一个executor一个文件

