面试回来之后把其中比较重要的问题记了下来写了个总结：   
（答案在后面）

1、简答说一下hadoop的map-reduce编程模型

2、hadoop的TextInputFormat作用是什么，如何自定义实现

3、hadoop和spark的都是并行计算，那么他们有什么相同和区别

4、为什么要用flume导入hdfs，hdfs的构架是怎样的

5、map-reduce程序运行的时候会有什么比较常见的问题

6、简单说一下hadoop和spark的shuffle过程

以下是自己的理解，如果有不对的地方希望各位大侠可以帮我指出来~：

**1、简答说一下hadoop的map-reduce编程模型**

首先map task会从本地文件系统读取数据，转换成key-value形式的键值对集合

使用的是hadoop内置的数据类型，比如longwritable、text等

将键值对集合输入mapper进行业务处理过程，将其转换成需要的key-value在输出

之后会进行一个partition分区操作，默认使用的是hashpartitioner，可以通过重写hashpartitioner的getpartition方法来自定义分区规则

之后会对key进行进行sort排序，grouping分组操作将相同key的value合并分组输出，在这里可以使用自定义的数据类型，重写WritableComparator的Comparator方法来自定义排序规则，重写RawComparator的compara方法来自定义分组规则

之后进行一个combiner归约操作，其实就是一个本地段的reduce预处理，以减小后面shufle和reducer的工作量

reduce task会通过网络将各个数据收集进行reduce处理，最后将数据保存或者显示，结束整个job

**2、hadoop的TextInputFormat作用是什么，如何自定义实现**

InputFormat会在map操作之前对数据进行两方面的预处理   
1是getSplits，返回的是InputSplit数组，对数据进行split分片，每片交给map操作一次   
2是getRecordReader，返回的是RecordReader对象，对每个split分片进行转换为key-value键值对格式传递给map

常用的InputFormat是TextInputFormat，使用的是LineRecordReader对每个分片进行键值对的转换，以行偏移量作为键，行内容作为值

自定义类继承InputFormat接口，重写createRecordReader和isSplitable方法   
在createRecordReader中可以自定义分隔符

**3、hadoop和spark的都是并行计算，那么他们有什么相同和区别**

两者都是用mr模型来进行并行计算，hadoop的一个作业称为job，job里面分为map task和reduce task，每个task都是在自己的进程中运行的，当task结束时，进程也会结束

spark用户提交的任务成为application，一个application对应一个sparkcontext，app中存在多个job，每触发一次action操作就会产生一个job

这些job可以并行或串行执行，每个job中有多个stage，stage是shuffle过程中DAGSchaduler通过RDD之间的依赖关系划分job而来的，每个stage里面有多个task，组成taskset有TaskSchaduler分发到各个executor中执行，executor的生命周期是和app一样的，即使没有job运行也是存在的，所以task可以快速启动读取内存进行计算

hadoop的job只有map和reduce操作，表达能力比较欠缺而且在mr过程中会重复的读写hdfs，造成大量的io操作，多个job需要自己管理关系

spark的迭代计算都是在内存中进行的，API中提供了大量的RDD操作如join，groupby等，而且通过DAG图可以实现良好的容错

**4、为什么要用flume导入hdfs，hdfs的构架是怎样的**

flume可以实时的导入数据到hdfs中，当hdfs上的文件达到一个指定大小的时候会形成一个文件，或者超过指定时间的话也形成一个文件

文件都是存储在datanode上面的，namenode记录着datanode的元数据信息，而namenode的元数据信息是存在内存中的，所以当文件切片很小或者很多的时候会卡死

**5、map-reduce程序运行的时候会有什么比较常见的问题**

比如说作业中大部分都完成了，但是总有几个reduce一直在运行

这是因为这几个reduce中的处理的数据要远远大于其他的reduce，可能是因为对键值对任务划分的不均匀造成的数据倾斜

解决的方法可以在分区的时候重新定义分区规则对于value数据很多的key可以进行拆分、均匀打散等处理，或者是在map端的combiner中进行数据预处理的操作

**6、简单说一下hadoop和spark的shuffle过程**

hadoop：map端保存分片数据，通过网络收集到reduce端   
spark：spark的shuffle是在DAGSchedular划分Stage的时候产生的，TaskSchedule要分发Stage到各个worker的executor

减少shuffle可以提高性能

部分答案不是十分准确欢迎补充:-)

——-补充更新———

**1、Hive中存放是什么？**   
表。   
存的是和hdfs的映射关系，hive是逻辑上的数据仓库，实际操作的都是hdfs上的文件，HQL就是用sql语法来写的mr程序。

**2、Hive与关系型数据库的关系？**   
没有关系，hive是数据仓库，不能和数据库一样进行实时的CURD操作。   
是一次写入多次读取的操作，可以看成是ETL工具。

**3、Flume工作机制是什么？**   
核心概念是agent，里面包括source、chanel和sink三个组件。   
source运行在日志收集节点进行日志采集，之后临时存储在chanel中，sink负责将chanel中的数据发送到目的地。   
只有成功发送之后chanel中的数据才会被删除。   
首先书写flume配置文件，定义agent、source、chanel和sink然后将其组装，执行flume-ng命令。

**4、Sqoop工作原理是什么？**   
hadoop生态圈上的数据传输工具。   
可以将关系型数据库的数据导入非结构化的hdfs、hive或者bbase中，也可以将hdfs中的数据导出到关系型数据库或者文本文件中。   
使用的是mr程序来执行任务，使用jdbc和关系型数据库进行交互。   
import原理：通过指定的分隔符进行数据切分，将分片传入各个map中，在map任务中在每行数据进行写入处理没有reduce。   
export原理：根据要操作的表名生成一个java类，并读取其元数据信息和分隔符对非结构化的数据进行匹配，多个map作业同时执行写入关系型数据库

**5、Hbase行健列族的概念，物理模型，表的设计原则？**   
行健：是hbase表自带的，每个行健对应一条数据。   
列族：是创建表时指定的，为列的集合，每个列族作为一个文件单独存储，存储的数据都是字节数组，其中的数据可以有很多，通过时间戳来区分。   
物理模型：整个hbase表会拆分为多个region，每个region记录着行健的起始点保存在不同的节点上，查询时就是对各个节点的并行查询，当region很大时使用.META表存储各个region的起始点，-ROOT又可以存储.META的起始点。   
rowkey的设计原则：各个列簇数据平衡，长度原则、相邻原则，创建表的时候设置表放入regionserver缓存中，避免自动增长和时间，使用字节数组代替string，最大长度64kb，最好16字节以内，按天分表，两个字节散列，四个字节存储时分毫秒。   
列族的设计原则：尽可能少（按照列族进行存储，按照region进行读取，不必要的io操作），经常和不经常使用的两类数据放入不同列族中，列族名字尽可能短。

**6、Spark Streaming和Storm有何区别？**   
一个实时毫秒一个准实时亚秒，不过storm的吞吐率比较低。

**7、mllib支持的算法？**   
大体分为四大类，分类、聚类、回归、协同过滤。

**8、简答说一下hadoop的map-reduce编程模型？**   
首先map task会从本地文件系统读取数据，转换成key-value形式的键值对集合。   
将键值对集合输入mapper进行业务处理过程，将其转换成需要的key-value在输出。   
之后会进行一个partition分区操作，默认使用的是hashpartitioner，可以通过重写hashpartitioner的getpartition方法来自定义分区规则。   
之后会对key进行进行sort排序，grouping分组操作将相同key的value合并分组输出。   
在这里可以使用自定义的数据类型，重写WritableComparator的Comparator方法来自定义排序规则，重写RawComparator的compara方法来自定义分组规则。   
之后进行一个combiner归约操作，其实就是一个本地段的reduce预处理，以减小后面shufle和reducer的工作量。   
reduce task会通过网络将各个数据收集进行reduce处理，最后将数据保存或者显示，结束整个job。

**9、Hadoop平台集群配置、环境变量设置？**   
zookeeper：修改zoo.cfg文件，配置dataDir，和各个zk节点的server地址端口，tickTime心跳时间默认是2000ms，其他超时的时间都是以这个为基础的整数倍，之后再dataDir对应目录下写入myid文件和zoo.cfg中的server相对应。

hadoop：修改   
hadoop-env.sh配置java环境变量   
core-site.xml配置zk地址，临时目录等   
hdfs-site.xml配置nn信息，rpc和http通信地址，nn自动切换、zk连接超时时间等   
yarn-site.xml配置resourcemanager地址   
mapred-site.xml配置使用yarn   
slaves配置节点信息   
格式化nn和zk。

hbase：修改   
hbase-env.sh配置java环境变量和是否使用自带的zk   
hbase-site.xml配置hdfs上数据存放路径，zk地址和通讯超时时间、master节点   
regionservers配置各个region节点   
zoo.cfg拷贝到conf目录下

spark：   
安装Scala   
修改spark-env.sh配置环境变量和master和worker节点配置信息

环境变量的设置：直接在/etc/profile中配置安装的路径即可，或者在当前用户的宿主目录下，配置在.bashrc文件中，该文件不用source重新打开shell窗口即可，配置在.bash\_profile的话只对当前用户有效。

**10、Hadoop性能调优？**

调优可以通过系统配置、程序编写和作业调度算法来进行。   
hdfs的block.size可以调到128/256（网络很好的情况下，默认为64）   
调优的大头：mapred.map.tasks、mapred.reduce.tasks设置mr任务数（默认都是1）   
mapred.tasktracker.map.tasks.maximum每台机器上的最大map任务数   
mapred.tasktracker.reduce.tasks.maximum每台机器上的最大reduce任务数   
mapred.reduce.slowstart.completed.maps配置reduce任务在map任务完成到百分之几的时候开始进入   
这个几个参数要看实际节点的情况进行配置，reduce任务是在33%的时候完成copy，要在这之前完成map任务，（map可以提前完成）   
mapred.compress.map.output,mapred.output.compress配置压缩项，消耗cpu提升网络和磁盘io   
合理利用combiner   
注意重用writable对象

**11、Hadoop高并发？**   
首先肯定要保证集群的高可靠性，在高并发的情况下不会挂掉，支撑不住可以通过横向扩展。   
datanode挂掉了使用hadoop脚本重新启动。

**12、hadoop的TextInputFormat作用是什么，如何自定义实现？**   
InputFormat会在map操作之前对数据进行两方面的预处理。   
1是getSplits，返回的是InputSplit数组，对数据进行split分片，每片交给map操作一次 。   
2是getRecordReader，返回的是RecordReader对象，对每个split分片进行转换为key-value键值对格式传递给map。   
常用的InputFormat是TextInputFormat，使用的是LineRecordReader对每个分片进行键值对的转换，以行偏移量作为键，行内容作为值。   
自定义类继承InputFormat接口，重写createRecordReader和isSplitable方法 。   
在createRecordReader中可以自定义分隔符。

**13、hadoop和spark的都是并行计算，那么他们有什么相同和区别？**   
两者都是用mr模型来进行并行计算，hadoop的一个作业称为job，job里面分为map task和reduce task，每个task都是在自己的进程中运行的，当task结束时，进程也会结束。   
spark用户提交的任务成为application，一个application对应一个sparkcontext，app中存在多个job，每触发一次action操作就会产生一个job。   
这些job可以并行或串行执行，每个job中有多个stage，stage是shuffle过程中DAGSchaduler通过RDD之间的依赖关系划分job而来的，每个stage里面有多个task，组成taskset有TaskSchaduler分发到各个executor中执行，executor的生命周期是和app一样的，即使没有job运行也是存在的，所以task可以快速启动读取内存进行计算。   
hadoop的job只有map和reduce操作，表达能力比较欠缺而且在mr过程中会重复的读写hdfs，造成大量的io操作，多个job需要自己管理关系。   
spark的迭代计算都是在内存中进行的，API中提供了大量的RDD操作如join，groupby等，而且通过DAG图可以实现良好的容错。

**14、为什么要用flume导入hdfs，hdfs的构架是怎样的？**   
flume可以实时的导入数据到hdfs中，当hdfs上的文件达到一个指定大小的时候会形成一个文件，或者超过指定时间的话也形成一个文件。   
文件都是存储在datanode上面的，namenode记录着datanode的元数据信息，而namenode的元数据信息是存在内存中的，所以当文件切片很小或者很多的时候会卡死。

**15、map-reduce程序运行的时候会有什么比较常见的问题？**   
比如说作业中大部分都完成了，但是总有几个reduce一直在运行。   
这是因为这几个reduce中的处理的数据要远远大于其他的reduce，可能是因为对键值对任务划分的不均匀造成的数据倾斜。   
解决的方法可以在分区的时候重新定义分区规则对于value数据很多的key可以进行拆分、均匀打散等处理，或者是在map端的combiner中进行数据预处理的操作。

**16、简单说一下hadoop和spark的shuffle过程？**   
hadoop：map端保存分片数据，通过网络收集到reduce端。   
spark：spark的shuffle是在DAGSchedular划分Stage的时候产生的，TaskSchedule要分发Stage到各个worker的executor。   
减少shuffle可以提高性能。

**17、RDD机制？**   
rdd分布式弹性数据集，简单的理解成一种数据结构，是spark框架上的通用货币。   
所有算子都是基于rdd来执行的，不同的场景会有不同的rdd实现类，但是都可以进行互相转换。   
rdd执行过程中会形成dag图，然后形成lineage保证容错性等。   
从物理的角度来看rdd存储的是block和node之间的映射。

**18、spark有哪些组件？**   
（1）master：管理集群和节点，不参与计算。   
（2）worker：计算节点，进程本身不参与计算，和master汇报。   
（3）Driver：运行程序的main方法，创建spark context对象。   
（4）spark context：控制整个application的生命周期，包括dagsheduler和task scheduler等组件。   
（5）client：用户提交程序的入口。

**19、spark工作机制？**   
用户在client端提交作业后，会由Driver运行main方法并创建spark context上下文。   
执行add算子，形成dag图输入dagscheduler，按照add之间的依赖关系划分stage输入task scheduler。   
task scheduler会将stage划分为task set分发到各个节点的executor中执行。

**20、spark的优化怎么做？**   
通过spark-env文件、程序中sparkconf和set property设置。   
（1）计算量大，形成的lineage过大应该给已经缓存了的rdd添加checkpoint，以减少容错带来的开销。   
（2）小分区合并，过小的分区造成过多的切换任务开销，使用repartition。

**21、kafka工作原理？**   
producer向broker发送事件，consumer从broker消费事件。   
事件由topic区分开，每个consumer都会属于一个group。   
相同group中的consumer不能重复消费事件，而同一事件将会发送给每个不同group的consumer。

**22、ALS算法原理？**   
答：对于user-product-rating数据，als会建立一个稀疏的评分矩阵，其目的就是通过一定的规则填满这个稀疏矩阵。   
als会对稀疏矩阵进行分解，分为用户-特征值，产品-特征值，一个用户对一个产品的评分可以由这两个矩阵相乘得到。   
通过固定一个未知的特征值，计算另外一个特征值，然后交替反复进行最小二乘法，直至差平方和最小，即可得想要的矩阵。

**23、kmeans算法原理？**   
随机初始化中心点范围，计算各个类别的平均值得到新的中心点。   
重新计算各个点到中心值的距离划分，再次计算平均值得到新的中心点，直至各个类别数据平均值无变化。

**24、canopy算法原理？**   
根据两个阈值来划分数据，以随机的一个数据点作为canopy中心。   
计算其他数据点到其的距离，划入t1、t2中，划入t2的从数据集中删除，划入t1的其他数据点继续计算，直至数据集中无数据。

**25、朴素贝叶斯分类算法原理？**   
对于待分类的数据和分类项，根据待分类数据的各个特征属性，出现在各个分类项中的概率判断该数据是属于哪个类别的。

**26、关联规则挖掘算法apriori原理？**   
一个频繁项集的子集也是频繁项集，针对数据得出每个产品的支持数列表，过滤支持数小于预设值的项，对剩下的项进行全排列，重新计算支持数，再次过滤，重复至全排列结束，可得到频繁项和对应的支持数。

作者：[@小黑](http://www.xiaohei.info/)

以下是自己的理解，如果有不对的地方希望各位大侠可以帮我指出来~：

**1、简答说一下hadoop的map-reduce编程模型**

首先map task会从本地文件系统读取数据，转换成key-value形式的键值对集合

使用的是hadoop内置的数据类型，比如longwritable、text等

将键值对集合输入mapper进行业务处理过程，将其转换成需要的key-value在输出

之后会进行一个partition分区操作，默认使用的是hashpartitioner，可以通过重写hashpartitioner的getpartition方法来自定义分区规则

之后会对key进行进行sort排序，grouping分组操作将相同key的value合并分组输出，在这里可以使用自定义的数据类型，重写WritableComparator的Comparator方法来自定义排序规则，重写RawComparator的compara方法来自定义分组规则

之后进行一个combiner归约操作，其实就是一个本地段的reduce预处理，以减小后面shufle和reducer的工作量

reduce task会通过网络将各个数据收集进行reduce处理，最后将数据保存或者显示，结束整个job

**2、hadoop的TextInputFormat作用是什么，如何自定义实现**

InputFormat会在map操作之前对数据进行两方面的预处理   
1是getSplits，返回的是InputSplit数组，对数据进行split分片，每片交给map操作一次   
2是getRecordReader，返回的是RecordReader对象，对每个split分片进行转换为key-value键值对格式传递给map

常用的InputFormat是TextInputFormat，使用的是LineRecordReader对每个分片进行键值对的转换，以行偏移量作为键，行内容作为值

自定义类继承InputFormat接口，重写createRecordReader和isSplitable方法   
在createRecordReader中可以自定义分隔符

**3、hadoop和spark的都是并行计算，那么他们有什么相同和区别**

两者都是用mr模型来进行并行计算，hadoop的一个作业称为job，job里面分为map task和reduce task，每个task都是在自己的进程中运行的，当task结束时，进程也会结束

spark用户提交的任务成为application，一个application对应一个sparkcontext，app中存在多个job，每触发一次action操作就会产生一个job

这些job可以并行或串行执行，每个job中有多个stage，stage是shuffle过程中DAGSchaduler通过RDD之间的依赖关系划分job而来的，每个stage里面有多个task，组成taskset有TaskSchaduler分发到各个executor中执行，executor的生命周期是和app一样的，即使没有job运行也是存在的，所以task可以快速启动读取内存进行计算

hadoop的job只有map和reduce操作，表达能力比较欠缺而且在mr过程中会重复的读写hdfs，造成大量的io操作，多个job需要自己管理关系

spark的迭代计算都是在内存中进行的，API中提供了大量的RDD操作如join，groupby等，而且通过DAG图可以实现良好的容错

**4、为什么要用flume导入hdfs，hdfs的构架是怎样的**

flume可以实时的导入数据到hdfs中，当hdfs上的文件达到一个指定大小的时候会形成一个文件，或者超过指定时间的话也形成一个文件

文件都是存储在datanode上面的，namenode记录着datanode的元数据信息，而namenode的元数据信息是存在内存中的，所以当文件切片很小或者很多的时候会卡死

**5、map-reduce程序运行的时候会有什么比较常见的问题**

比如说作业中大部分都完成了，但是总有几个reduce一直在运行

这是因为这几个reduce中的处理的数据要远远大于其他的reduce，可能是因为对键值对任务划分的不均匀造成的数据倾斜

解决的方法可以在分区的时候重新定义分区规则对于value数据很多的key可以进行拆分、均匀打散等处理，或者是在map端的combiner中进行数据预处理的操作

**6、简单说一下hadoop和spark的shuffle过程**

hadoop：map端保存分片数据，通过网络收集到reduce端   
spark：spark的shuffle是在DAGSchedular划分Stage的时候产生的，TaskSchedule要分发Stage到各个worker的executor

减少shuffle可以提高性能