1. 提空题
2. df -h
3. Uptime
4. HDFS
5. FIFO 、Capacity Scheduler（容量调度器）和Fair Sceduler（公平调度器）
6. 半数机制
7. Kafka机器数量=2\*（峰值生产速度\*副本数/100）+1
8. 长度原则，散列原则，唯一原则
9. Local，Standalone，Yarn，Mesos
10. 128M
11. \_version的乐观锁机制
12. 简答题
13. RDD在Lineage依赖方面分为两种Narrow Dependencies与Wide Dependencies用来解决数据容错时的高效性以及划分任务时候起到重要作用。
14. LZO不是Hadoop自带的压缩格式，LZO支持对文件块的split，GIP不支持，LZO的压缩率比GZIP低。LZO常用在Mapper端输出时压缩，GZIP常用在reduce端输出时压缩
15. 发现拖后腿的任务，比如某个任务运行速度远慢于任务平均速度。为拖后腿任务启动一个备份任务，同时运行。谁先运行完，则采用谁的结果。
16. 删除一个管理表时，Hive会删除该表在Mysql中的元数据和hdfs上的数据。删除外部表只会删除掉元数据，不过删除hdfs上的数据。
17. (1)首先要有一个main()线程

(2)在main线程中创建Zookeeper客户端，这时就会创建两个线程，一个负责网络连接通信（connet），一个负责监听（listener）。

(3)通过connect线程将注册的监听事件发送给Zookeeper。

(4)在Zookeeper的注册监听器列表中将注册的监听事件添加到列表中。

(5)Zookeeper监听到有数据或路径变化，就会将这个消息发送给listener线程。

(6)listener线程内部调用了process（）方法。

6、第一次执行的时候被持久化到内存中,后面的业务代码就可以复用,能有效提高效率,cache内部调用了persist方法,persist方法又调用了存储级别(MEMORY\_ONLY,MEMORY\_AND\_DISK,MEMORY\_ONLY\_SER,MEMORY\_AND\_DISK\_SER,DISK\_ONLY,MEMORY\_ONLY\_2,MEMORY\_AND\_DISK\_2)

7、Stage：根据RDD之间的依赖关系的不同将Job划分成不同的Stage，遇到一个宽依赖则划分一个Stage。

Task：Stage是一个TaskSet，将Stage根据分区数划分成一个个的Task。

1. Elasticsearch使用的是 term frequency/inverse document frequency算法，简称为TF/IDF算法。
2. Term frequency：搜索文本中的各个词条在field文本中出现了多少次，出现次数越多，就越相关。
3. Inverse document frequency：搜索文本中的各个词条在整个索引的所有文档中出现了多少次，出现的次数越多，就越不相关。
4. (1)数据写入buffer

(2)commit point

(3)buffer中的数据写入新的index segment

(4)等待在os cache中的index segment被fsync强制刷到磁盘上

(5)新的index sgement被打开，供search使用

(6)buffer被清空

10、Flink实现Exactly once的策略: Flink会持续地对整个系统做snapshot,然后把global state(根据config文件设定)储存到master node或HDFS.当系统出现failure,Flink会停止数据处理,然后把系统恢复到最近的一次checkpoint.

三、优化题

1、在Reduce阶段完成join。容易发生数据倾斜，可以用MapJoin把小表全部加载到内存在map端进行join，避免reducer处理。

2、首先出现写入文件过量导致宕机，那么可能有以下几个原因

(1)写入小文件过多，导致NameNode内存不够。

处理方式：i)增加NameNode的数量,ii)增加缓存层将小文件聚合成一个大文件再存入HDFS

(2)写入的不是小文件

处理方式:扩大NameNode容量

3、(1)缓存层缓存空值。给个空值过期时间,后台设置时主动删除空值，并缓存把值进去

(2)将数据库中所有的查询条件，放布隆过滤器中。当一个查询请求来临的时候，先经过布隆过滤器进行查，如果请求存在这个条件中，那么继续执行，如果不在，直接丢弃。

4、先用top命令找出CPU占比最高的，使用linux管道找出是具体的进程ID。使用jstack找出程序具体位置。

5、（1）如果是Kafka消费能力不足，则可以考虑增加Topic的分区数，并且同时提升消费组的消费者数量，消费者数=分区数。（两者缺一不可）

（2）如果是下游的数据处理不及时：提高每批次拉取的数量。批次拉取数据过少（拉取数据/处理时间<生产速度），使处理的数据小于生产的数据，也会造成数据积压。