1.redis分布式实现原理?集群模式存在的问题?

1、redis分布式实现原理:

memcache只能说是简单的kv内存数据结构,而redis支持的数据类型比较丰富。Redis在3.0以后实现集群机制。目前Redis实现集群的方法主要是采用一致性哈稀分片（Shard），将不同的key分配到不同的redis server上，达到横向扩展的目的。

使用了一致性哈稀进行分片，那么不同的key分布到不同的Redis-Server上，当我们需要扩容时，需要增加机器到分片列表中，这时候会使得同样的key算出来落到跟原来不同的机器上，这样如果要取某一个值，会出现取不到的情况，对于这种情况，Redis的提出了一种名为Pre-Sharding的方式:

2、redis集群模式存在的问题:

A：扩容问题:

Pre-Sharding方法是将每一个台物理机上，运行多个不同断口的Redis实例，假如有三个物理机，每个物理机运行三个Redis实际，那么我们的分片列表中实际有9个Redis实例，当我们需要扩容时，增加一台物理机，步骤如下:

(1)在新的物理机上运行Redis-Server;

(2)该Redis-Server从属于(slaveof)分片列表中的某一Redis-Server（假设叫RedisA）;

(3)等主从复制(Replication)完成后，将客户端分片列表中RedisA的IP和端口改为新物理机上Redis-Server的IP和端口;

(4)停止RedisA.

B: 单点故障问题

将一个Redis-Server转移到了另外一台上。Prd-Sharding实际上是一种在线扩容的办法，但还是很依赖Redis本身的复制功能的，如果主库快照数据文件过大，这个复制的过程也会很久，同时会给主库带来压力。

2.各完成一个awk和sed的例子，最简单的应用即可，并说明?

3.Flume工作机制是什么?

1. Flume用于从大量不同的源有效的收集、聚合、移动大量日志数据进行集中式存储。

2. Flume的核心是Agent，Agent中包括Source、Channel和Sink。Agent是最小的独立运行单位。在Agent中数据流向为Source>Channel>Sink。

a. Source负责收集数据，传递给Channel。支持多种收集方式，比如Avro、Thrift、Spooling Directory、Taildir、Kafka、HTTP等。其中Taildir Source：观察指定的文件，并在监测到添加的每个文件的新行后几乎实时的尾随它们；Spooling Directory Source：监测配置的目录下新增的文件，并将文件中的数据读取出来。需要注意两点，拷贝到spool目录下的文件不可以再打开编辑，并且spool目录下不可包含相应的子目录。

b. Channel作为数据通道，接受Source的数据并储存，传递给Sink。Channel中的数据会在被Sink消费前一直保存，等Sink成功把数据发送到下一跳Channel或最终目的地后才会删除缓存的数据。支持多种类型的Channel，包括Memory、JDBC、Kafka、File等。其中Memory Channel可以实现高速的吞吐，但是无法保证数据的完整性；File Channel是一个持久化的隧道，它持久化所有的事件，并将其存储到磁盘中。

c. Sink消费Channel中的数据，传递到下一跳Channel或最终目的地，完成后将数据从Channel中移除。支持多种类型的Sink，包括HDFS、Hive、Hbase、Kafka等。Sink在设置存储数据的时候，可以向文件系统、数据库、hadoop存数据；在日志数据比较少时，可以将数据存储在文件系统中，并且设定一定的时间间隔保存数据。在日志数据较多时，可以将相应的日志数据存储到Hadoop中，便于日后进行相应的数据分析。

3. Flume支持多个Agent相连，形成多极Agent。此时上一级Sink和下一集Source都必须使用Avro协议。使用多级Flume可以实现日志的聚合，第一层Agent接收日志，第二层Agent统一处理。

4. Flume还支持将一个流从一个Source扇出到多个Channel。有两种模式的扇出，复制和复用。在复制流程中，事件被发送到所有配置的通道。在复用的情况下，事件仅发送到合格信道的子集。

4.kafka收集数据的原理?

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。

5.Flume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。

6.Sqoop工作原理是什么?

1. 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。

    a. 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。

    b. 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

7.kafka接收spark streaming数据的两种方式?

1. 第一种是基于Receiver的直接方式。这种方式使用Receiver来获取数据，receiver从Kafka中获取的数据是存储在Spark Executor的内存中，然后Spark Streaming启动job去处理那些数据。需要开启WAL预写日志机制，该机制会同步将接收到的Kafka数据写入分布式文件系统上的预写日志中。
2. 第二种是基于Direct的方式。这种方式会周期性的查询Kafka，来获得每个topic+partition的最新的offset，从而定义每个batch的offset的范围，来获取Kafka指定offset范围的数据。

    ɑ. 优势在于:

        i. 首先简化并行读取，如果要读取多个partition，不需要创建多个输入DStream然后对其进行union操作。Spark会创建跟Kafka partition一样多的RDD partition，并且会并行从Kafka中读取数据。所以在Kafka partition和RDD partition之间，有一对一的映射关系;

3. 然后是高性能，如果要保证零数据丢失，在基于receiver的方式中，需要开启WAL机制。这种方式效率低下，因为数据被复制了两份。Kafka本身就有高可靠机制，会对数据复制一份，而这里又会复制一份到WAL中。而基于direct的方式，不依赖Receiver，不需要开启WAL机制，只要Kafka中做了数据的复制，那么就可以通过Kafka的副本进行恢复;

i. 最后是一次仅且一次的事务机制。基于receiver的方式，是使用Kafka的高阶API在Zookeeper中保存消费过的offset。这是消费Kafka数据的传统方式，这种方式配合WAL机制可以保证数据零丢失的高可靠性，但是却无法保证数据被处理一次且仅一次，可能会处理两次。因为Spark和Zookeeper之间可能不是同步的。基于direct的方式，spark streaming自己负责追踪消费的offset，并保存在checkpoint中。Spark自己一定是同步的，因此可以保证数据是消费一次且仅一次的。

8.Kafka机制?

1. Kafka是一个生产-消费模型。
2. Producer：生产者，负责数据生产，生产者的代码可以集成到任务系统中。数据的分发策略由producer决定；
3. Broker：当前服务器上的Kafka进程，只管数据
4. Topic：

9.对于kafka来讲，groupid的作用是什么?

多个作业同时消费同一个topic时，每个作业拿到完整数据，计算互不干扰；每个作业拿到一部分数据，相当于进行了负载均衡。