尚硅谷大数据技术之大数据基础阶段考试题（一）

(作者：大海哥)

官网：[www.atguigu.com](http://www.atguigu.com)

版本：V1.0

## 一 Linux

### Linux常用命令

find、df、tar、ps、top、netstat

### CentOS查看版本的命令

cat /etc/issue

### linux 查看端口调用

netstat –anp | grep :端口号

### linux命令 查看内存 磁盘 io 端口 进程

查看内存：top

查看磁盘存储情况：df -h

查看磁盘IO读写情况：iotop -o（查看正在进行磁盘读写程序）

查看端口占用情况：netstat -anp | grep 端口号

查看进程：ps aux | ps -ef

### 使用Linux命令查询file1里面空行的所在行号

awk '/^$/{print NR}' file1

### 有文件chengji.txt内容如下:

张三 40

李四 50

王五 60

请使用Linux命令计算第二列的和并输出

awk -v sum=0 -F " " '{sum+=$2} END{print sum}' chengji.txt

### 在Linux环境下有文件/home/dim\_city.txt如何加载dim\_city外部表中，HDFS路径/user/dim/dim\_city

①建表时指定表的类型为外部表，指定表的location属性:

create external table dim\_city (…) location ‘/user/dim/dim\_city’;

②加载数据：

load data local inpath ‘/home/dim\_city.txt’ into table dim\_city

### Shell脚本里如何检查文件是否存在，如果不存在该如何处理？Shell里如何检查一个变量是否是空？

if [ -f file.txt ]; then

echo "文件存在!"

else

echo "文件不存在!"

fi

检查变量是否为null: [ ! $变量 ] && echo "变量 is null"

### Shell脚本里如何统计一个目录下（包含子目录）有多少个java文件？如何取得每一个文件的名称（不包含路径）

ls -lR 目录名 | grep ".java$" | wc –l

获取文件名： basename path

## 二 Hadoop入门

### 简要描述如何安装配置apache的一个开源[Hadoop](http://lib.csdn.net/base/hadoop)，只描述即可，无需列出具体步骤，列出具体步骤更好。

①根据Hadoop版本安装匹配的JDK，配置JAVA\_HOME

②解压安装Hadoop，配置HADOOP\_HOME，将bin/sbin加入到path，方便以后使用

③配置hadoop的配置文件，core-site.xml,hdfs-site.xml,mapred-site.xml,yarn-site.xml

④如果为完全分布式的集群，则需要配置所有机器的hosts映射信息，配置rm所在机器到其他机器的ssh免密登录

⑤在rm所在主机编辑$HADOOP\_HOME/etc/hadoop/slave文件，配置集群中的所有的主机名

⑥分发安装的hadoop到其他节点

### Hadoop中需要哪些配置文件，其作用是什么？

xxx-env.sh文件为配置hadoop中各个组件运行的环境信息！

core-site.xml： 用户自定义核心组件，例如namenode的rpc地址

hdfs-site.xml： 用户自定义和hdfs相关的参数

mapred-site.xml： 用户自定义和mapreduce相关的参数

yarn-site.xml： 用户自定义和yarn相关的参数

### 请列出正常工作的Hadoop集群中Hadoop都分别需要启动哪些进程，它们的作用分别是什么?

ResourceManager： 负责整个集群中所有计算资源（cpu,内存，IO，硬盘）的管理!

NodeManager： 负责单个节点中所有计算资源（cpu,内存，IO，硬盘）的管理!领取RM中的Task任务，分配container运行Task

Namenode: 负责HDFS中元数据的管理和处理客户端的请求

Datanode:以块为单位存储HDFS中文件。

SecondaryNamenode: 帮助namenode定期合并fsimage文件和edits文件，HA中可以省略此进程！

### 简述Hadoop的几个默认端口及其含义。

|  |  |
| --- | --- |
| 50070 | namenode的http服务端口 |
| 9000 | namenode的接受客户端rpc调用端口 |
| 8088 | yarn的http服务端口 |
| 19888 | MR运行历史收集服务的http服务端口 |
| 8042 | nodemanager的http服务端口 |

## 三 Hadoop的HDFS

### HDFS的存储机制（读写流程）。

写流程：

①客户端创建一个分布式文件系统客户端对象，向nn发送请求，请求上传文件

②NN处理请求，对请求进行合法性检查（权限，文件路径是否存在...）,请求合法后，响应客户端，通知进行写操作

③客户端创建一个输出流，输出流在写文件时，以块(128M)为单位，块又由packet(64k)作为基本单位,packet由多个chunk(512B+4B校验位)组成！

④开始第一块的上传，在上传时，会请求NN，根据网络拓扑距离，和上传的副本数，分配指定数量的距离客户端最近的Datanode节点列表

⑤客户端请求距离最近的一个DN建立通道，DN列表中的DN依次请求建立通道！全部通道建立完成，开始传输！客户端将第一个块的0-128M信息，以packet形式进行封装，将封装好的packet放入 data\_queue的队列中！输出流在传输时，会新建一个ack\_queue，将data\_queue要传输的packet依次放入ack\_queue中！

⑥客户端只负责将当前的packet发送给距离最近的DN，DN会在收到packet后进行存盘，并将packet依次发往下一个DN！当所有的DN在收到当前的packet后，向客户端的流对象发送ack指令，当ack\_queue中的packet已经被所有的DN收到，那么在当前队列中就会删除此packet！

⑦第一块上传完毕后，会上报NN，当前块已经成功发送到了哪些DN上！开始传输第二块（128M-...），和第一块的流程一样....

⑧当所有的数据都上传完毕，关闭流，等待NN的一个响应

读流程：

①客户端向NN发送请求，请求读取指定路径的文件

②NN处理请求，返回当前文件所有的块列表信息

③客户端创建一个输入流，根据块的信息，从第一块开始读取，根据拓扑距离选择距离最近的一个节点进行读取，剩余块依次读取

④所有块信息读取完毕，关闭流

### SecondaryNameNode 工作机制。

2NN和NN不是主从关系，是两个不同的进程！2NN负责辅助NN工作，定期合并NN中产生的edits文件和fsimage文件！

2NN基于两个触发条件，执行CheckPoint(合并)，每间隔dfs.namenode.checkpoint.period秒合并一次，默认为1小时，每间隔dfs.namenode.checkpoint.txns次合并一次，默认为100w

2nn默认每间隔60秒向NN发送请求，判断CheckPoint的条件是否满足，如果满足，向NN发送请求立刻滚动日志（产生一个新的日志，之后的操作都向新日志写入）！将历史日志和fsiamge文件拷贝到2NN工作目录中，加载到内存进行合并！合并后，将新的fsiamge文件，传输给NN，覆盖老的fsiamge文件！

### NameNode与SecondaryNameNode 的区别与联系？

联系：

①2NN需要配合NN工作，NN启动，2NN工作才有意义！

②2NN可能会保存部分和NN一致的元数据，可以用来NN的容灾恢复！

区别：

这是两个功能不同的进程，不是主备关系！

### 服役新数据节点和退役旧节点步骤

服役新节点： 加机器，安装软件，配置环境，启动进程！

退役旧节点： 使用黑白名单！

### NameNode元数据损坏怎么办？

①如果配置了NN的多目录配置，还是可以照常启动

②如果多个目录的元数据都损坏，可以查看是否启用了HA，或者查看是否启用2NN

③可以通过另外一个NN或者2NN中的元数据进行恢复

## 四 Hadoop的MapReduce

### 谈谈Hadoop序列化和反序列化及自定义bean对象实现序列化?

①在Hadoop中如果有reduce阶段，那么需要Mapper和Reducer中的key-value实现序列化

②Hadoop采用的是自己的序列化机制，即Writeable机制，是一种轻量级的序列化机制，保存的信息少，适合大数据量的网络传输

③无论是Map还是Reduce阶段，key-value需要实现Writeable接口即可，重写readFileds()和writeFields()方法即可

### FileInputFormat切片机制

①将输入目录中的所有文件，以文件为单位进行切片

②根据isSplitable()方法，以文件的后缀名为依据，判断文件是否使用了压缩格式！

如果是普通文件，可切。否则继续判断是否是一个可切的压缩格式！

③如果文件不可切，整个作为一片

④可切，确定每片的大小（默认是块大小），之后以此大小为根据，循环进行切片

⑤除了最后一片有可能为切片大小的1.1倍，其余每片切片大小为片大小。

### 自定义InputFormat流程

需要继承InputFormat类！通常可以继承FileInputFormat以节省方法的实现!

①如果需要实现自定义的切片逻辑，实现或重写createSplits()方法

②实现createRecordReader()方法，返回一个RecordReader对象

③RecordReader负责将切片中的数据以key-value形式读入到Mapper中。

④其核心方法是nextKeyValue()，这个方法负责读取一对key-value，读到则返回true，否则返回false！

⑤可以根据需要实现isSplitable()方法！

### 如何决定一个job的map和reduce的数量?

maptask数量取决于切片数，可以通过调整切片的大小，来控制map的数量！

reducetask数量取决于Job.setNumReduceTasks()的值！

### MapTask工作机制

①Map阶段： 使用InputFormat中的RecordReader读取切片中的每一对key-value

每一对key-value都会调用mapper的map()处理

②Sort阶段： 在Mapper的map()方法处理后，输出的key-value会先进行分区，之后被收集到缓冲区，当缓冲区达到一定的溢写阀值时，每个区的key-value会进行排序，之后溢写到磁盘。多次溢写的文件，最后会进行合并为一个总的文件，这个文件包含若干区，而且每个区内都是有序的！

### ReduceTask工作机制

①copy阶段： ReduceTask启动shuffle进程，到指定的maptask拷贝指定区的数据。

拷贝之后，会进行合并，合并为一个总的文件

②sort阶段：在合并时，保证所有的数据都是合并后有序的，所以会进行排序

③reduce阶段： 合并后的数据，会进行分组，每一组数据，调用reducer的reduce()，之后reduce通过OutputFormat的RecordWriter将数据输出

### 请描述mapReduce有几种排序及排序发生的阶段

两种排序： 快速排序，归并排序

MapTask处理的阶段：每次溢写前，进行快速排序！

最后合并时，使用归并排序

ReduceTask处理的阶段：在sort和merge时，使用归并排序！

### 请描述mapReduce中shuffle阶段的工作流程，如何优化shuffle阶段

shuffle阶段从mapper的map()处理完，到reducer的reduce()方法前属于shuffle!

工作流程： sort()—copy()—sort()

优化：

本质就是减少磁盘IO（减少溢写次数和减少每次溢写的数据量）和网络IO（减少网络数据传输量）

MapTask端优化：

①map端减少溢写次数：调大mapreduce.task.io.sort.mb和mapreduce.map.sort.spill.percent

②map端减少合并的次数：调大io.sort.factor

③在合适的情况下使用Combiner对数据在map端进行局部合并！

④使用压缩，减少数据传输量！

ReduceTask端优化：

⑤reduce端减少溢写次数：调大mapred.job.reduce.input.buffer.percent

### 请描述mapReduce中combiner的作用是什么，一般使用情景，哪些情况不需要，及和reduce的区别？

①作用就是在每次溢写数据到磁盘时，对数据进行局部合并，减少溢写数据量！

②求和，汇总等场景适用！不适合使用的场景例如求平均数之类！

③和reducer的唯一区别，就是Combiner运行在shuffle阶段，且主要是MapTask端的shuffle阶段，而reducer运行在reduce阶段！

### 如果没有定义partitioner，那数据在被送达reducer前是如何被分区的？

①如果reduceTask个数为1，那么所有的key-value都是0号区！

②如果reduceTask个数大于1，默认使用HashPartitioner，根据key的hashcode()方法 和Interger最大值做&运算，之后模除reduceTask的个数！

③所有数据的区号介于0和reduceTask个数-1的范围内！

### MapReduce 怎么实现 TopN？

在Map端对数据，根据排名的字段，进行排序：

①合理设置Map的key，key中需要包含排序的字段

②通过使key实现WritableComparable接口或者自定义key的RawComparator类型比较器！归根到底，在排序时都是使用用户实现的compareTo()方法进行比较！

在reduce端，输出数据：

③reduce端处理的数据已经自动排序完成，只需要控制输出N个key-value即可！

### 有可能使 Hadoop 任务输出到多个目录中么？如果可以，怎么做？

可以，通过自定义OutPutFormat进行实现，核心是实现OutPutFormat中的相关的RecordWriter!

通过实现其write()方法就需要的数据输出到指定的目录！

### 简述hadoop实现join的几种方法及每种方法的实现方法

ReduceJoin: 在Map阶段，对所有的输入文件进行组装，打标记输出。到reduce阶段

只处理需要join的字段，进行合并即可！

MapJoin： 在Map阶段，将小文件以分布式缓存的形式进行存储，在Mapper的map()方法处理前，读取小文件的内容，和大文件进行合并即可！不需要有reduce阶段！

### 请简述hadoop怎样实现二级排序

key实现WritableComparable()接口，实现CompareTo()方法，先根据一个字段比较，如果当前字段相等继续按照另一个字段进行比较。

### 参考下面的MR系统的场景:

--hdfs块的大小为64MB

--输入类型为FileInputFormat

--有三个文件的大小分别是:64KB 65MB 127MB

Hadoop框架会把这些文件切多少片？

4片！

### Hadoop中RecordReader的作用是什么？

读取每一片中的记录为key-value，传给Mapper！

### 给你一个1G的数据文件。分别有id,name,mark,source四个字段，按照mark分组，id排序，简述排序的核心逻辑思路?其中启动几个MapTask？

①Map阶段key的比较器，使用根据mark和id进行二次排序！

②Reduce阶段分组比较器，根据mark进行比较，mark相同视为key相同！

默认启动8个MapTask!

## 五 Hadoop的Yarn

### 简述Hadoop1与Hadoop2 的架构异同。

①Hadoop1使用JobTracker调度MR的运行！

②Hadoop2提供YARN框架进行资源的调度！

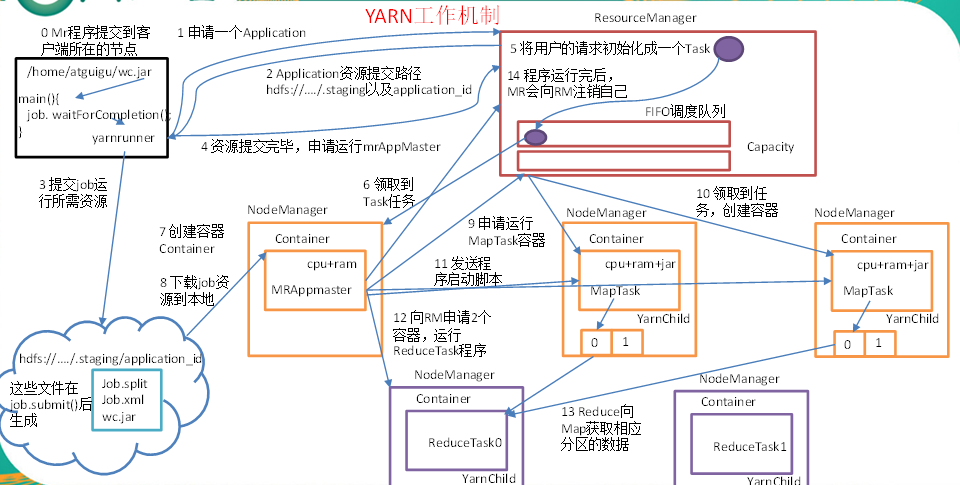
③Hadoop2支持HA集群搭建！

### 为什么会产生yarn,它解决了什么问题，有什么优势？

①YARN为了将MR编程模型和资源的调度分层解耦！

②使用YARN后软件维护方便，YARN还可以为其他的计算框架例如spark等提供资源的调度！

### MR作业提交全过程



### HDFS的数据压缩算法？

系统自带： deflate和gzip和bzip2

额外安装： lzo,snappy

压缩率高： bzip2

速度快： snappy，lzo

可切片的： lzo,bzip2

使用麻烦的： lzo

### Hadoop的调度器总结。

FIFO调度器：

单队列，按照Job提交的顺序先进先出！容易出现单个用户的大Job独占资源，而其他用户的小Job无法被及时处理！

容量调度器：

①多个队列，队列内部FIFO，每个队列可以指定容量

②资源利用率高，处理灵活，空闲队列的资源可以补充到繁忙队列

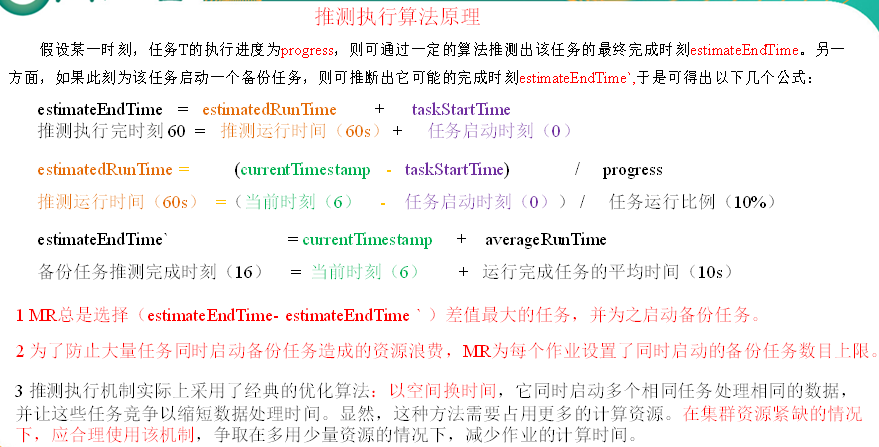
③可以设置单个用户的资源限制，防止单个用户独占资源

④动态调整，维护方便

公平调度器：

在容量调度器的基础上，改变了FIFO的调度策略，默认参考集群中内存资源使用最大最小公平算法，保证小Job可以及时处理，大Job不至于饿死。小Job有优势！

### mapreduce推测执行算法及原理。



## 六 Hadoop优化

### mapreduce 跑的慢的原因？

①Task运行申请的资源少，可以通过调节相关参数解决

②程序逻辑复杂，可以将一个复杂逻辑拆分为多个Job，串行执行

③产生了数据倾斜，可以通过合理设置切片策略和设置分区及调节reducetask数量解决

④shuffle过程漫长，可以通过合理使用combiner，使用压缩，调大Map端缓冲区大小等解决

### mapreduce 优化方法。

见HadoopMapReduce资料的第6章6.2

### HDFS小文件优化方法。

①在源头处理，就小文件压缩或打包

②使用Har进行归档，Har归档后的文件只能节省Namenode的内存空间，在进行MR计算时，依然以小文件的形式存在

③使用CombineTextInputFormat

④使用紧凑的文件格式例如SequeceFile

### MapReduce怎么解决数据均衡问题，如何确定分区号？

①Map端避免数据倾斜：抽样数据，避免有不可切分的数据！

小文件过多，使用CombineTextInputFormat!

②Reduce端避免数据倾斜： 抽样数据，合理设置数据的分区！合理设置reduceTask的个数！

③使用Partitioner的getPartition()确定分区号！

### Hadoop中job和Tasks之间的区别是什么？

一个Job在运行期间，会启动多个task来完成每个阶段的具体任务！

## Zookeeper

### 请简述ZooKeeper的选举机制

①只有zookeeper以集群模式启动时，需要选举Leader

②集群中leader在集群启动时，选举产生或者是leader挂掉，集群中的其他机器重新选举产生

③满足半数以上机制，按照启动顺序，id大的有优势

### ZooKeeper的监听原理是什么？

①首先要有一个main()线程，在main线程中创建Zookeeper客户端，这时就会创建两个线程，一个负责和zookeeper服务端通信（connet），一个负责监听zookeeper服务端的通知（listener）。

②通过connect线程将注册的监听事件发送给Zookeeper。

③在Zookeeper的注册监听器列表中将该客户端注册的监听事件添加到列表中。

④Zookeeper监听到有数据或路径变化，通知listener线程。

⑤listener线程内部调用了Watcher的process()方法。

### ZooKeeper的部署方式有哪几种？集群中的角色有哪些？集群最少需要几台机器？

standalone模式和集群模式！

Leader和Follower!

2台！

### ZooKeeper的常用命令

ls,ls2,stat,get,create,set

## Hive

### Hive表关联查询，如何解决数据倾斜的问题？

①提前清洗数据，将不合法数据例如key为null的数据进行过滤

②如果没用清洗则在查询时先过滤再关联表

③如果一个数据中有很多数量级多的key,可以增加reducetask个数，避免多个大key集中到一个reducetask

④且将无意义的nullkey进行随机替换处理，同时增加reducetask的个数，以分散数据

⑤转换为mapjoin

⑥大key和其他数据分开处理

⑦hive.optimize.skewjoin：会将一个join sql 分为两个job。另外可以同时设置下hive.skewjoin.key，默认为10000。参考：

https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties

参数对full outer join无效。

⑧调整内存设置：适用于那些由于内存超限内务被kill掉的场景。通过加大内存起码能让任务跑起来，不至于被杀掉。该参数不一定会明显降低任务执行时间。如：

setmapreduce.reduce.memory.mb=5120 ;

setmapreduce.reduce.java.opts=-Xmx5000M -XX:MaxPermSize=128m ;

### 请谈一下Hive的特点，Hive和RDBMS有什么异同？

Hive的特点：基于OLAP设计的数据仓库软件，不适合实时计算，适合大数据量的计算！底层本质是MR！

和RDBMS的异同：

核心异同为RDBMS为OLTP（事务，实时），Hive为OLAP（分析，延迟）！

具体可参考Hive文档第1章1.4。

### 请说明Hive中 Sort By，Order By，Cluster By，Distrbute By各代表什么意思？

①Sort By：部分排序，一个Job有多个Reducer，每个Reducer处理的数据内部有序；

②Order By：全局排序，一个Job只有一个Reducer；

③Distrbute By：类似MR中partition，进行分区，结合sort by使用，Hive要求DISTRIBUTE BY语句要写在SORT BY语句之前，即先分区再排序。

④Cluster By：当distribute by和sorts by字段相同时，可以使用cluster by方式。但是排序只能是升序排序，不能指定排序规则为ASC或者DESC。

### Hive有哪些方式保存元数据，各有哪些特点？

①默认存放在derby

②修改存储在其他的关系型数据库例如mysql,oracle,mss,progresql

③通过使用thrift调用metaServer服务进行元数据的读写

### Hive内部表外部表的区别？

内部表也成为管理表，可以管理数据的生命周期，删除管理表时，数据也会随之删除！

外部表，只删除表结构（元数据）！

### 写出将 text.txt 文件放入 Hive中 test 表‘2016-10-10’ 分区的语句，test 的分区字段是 l\_date。

load data local inpath ‘/text.txt’ into table test partition(l\_date=’ 2016-10-10’);

### Hive自定义UDF函数的流程?

①自定义一个Java类,继承UDF类,重写一个或多个evaluate方法

②evaluate()的返回值不能为void，可以是null

③打成jar包, 在hive中使用add jar pathofjar 或者直接将jar包放入到hive的lib目录中

④在hive中使用create function 函数名 as ‘主类名’声明函数

### 对于Hive，你写过哪些udf函数，作用是什么？

dayofyear，作用是返回当前日期是一年中的第几天！

### Hive 中的压缩格式TextFile、SequenceFile、RCfile 、ORCfile各有什么区别？

TextFile: 默认格式，数据不做压缩，磁盘开销大，数据解析开销大。可结合压缩格式进行压缩。

SequenceFile: SequenceFile是Hadoop API 提供的一种二进制文件，它将数据以<key,value>的形式序列化到文件中。这种二进制文件内部使用Hadoop 的标准的Writable 接口实现序列化和反序列化。Hive 中的SequenceFile 继承自Hadoop API 的SequenceFile，不过它的key为空，使用value 存放实际的值， 这样是为了避免MR 在运行map 阶段的排序过程。

RCFile: RCFile是Hive推出的一种专门面向列的数据格式。 它遵循“先按列划分，再垂直划分”的设计理念。当查询过程中，针对它并不关心的列时，它会在IO上跳过这些列。需要说明的是，RCFile在map阶段从 远端拷贝仍然是拷贝整个数据块，并且拷贝到本地目录后RCFile并不是真正直接跳过不需要的列，并跳到需要读取的列， 而是通过扫描每一个row group的头部定义来实现的，但是在整个HDFS Block 级别的头部并没有定义每个列从哪个row group起始到哪个row group结束。所以在读取所有列的情况下，RCFile的性能反而没有SequenceFile高。

ORCfile: ORC是列式存储，有多种文件压缩方式，并且有着很高的压缩比。文件是可切分（Split）的。因此，在Hive中使用ORC作为表的文件存储格式，不仅节省HDFS存储资源,查询任务的输入数据量减少，使用的MapTask也就减少了。提供了多种索引，row group index、bloom filter index。ORC可以支持复杂的数据结构（比如Map等）

### Hive join过程中大表小表的放置顺序？

小表join大表，当设置了set hive.auto.convert.join=true后，hive会自动调整顺序！

### Hive的两张表关联，使用MapReduce怎么实现？

ReduceJoin:

①在Map端使用一个通用的bean来封装数据，这个bean中包含了两个表的所有字段

②在Map端处理时，为每个bean打上标记，标记当前数据的来源

③在Reduce端，根据数据的来源将数据分类

④在Reduce端进行字段的关联，且只处理需要处理的数据

MapJoin:

①根据两张表的数据的数据量，将两张表划分为大表和小表！

②小表使用分布式缓存提前缓存，大表作为MR的输入，进行切片后读入到MapTask!

③在Mapper的map()方法处理之前，提前从分布式缓存中读取小表中的数据！

④在Mapper的map()方法中，对大表的数据进行关联操作！

### 所有的Hive任务都会有MapReduce的执行吗？

取决于hive.fetch.task.conversion的配置，默认配置为more!

即当一个查询中只有select ,where ,包括带分区字段的过滤查询和limit走fetchtask，不走MR!

### Hive的函数：UDF、UDAF、UDTF的区别？

UDF: 用户定义函数，一进一出。

UDAF: 用户定义的聚集函数，多进一出。

UDTF: 用户定义的表生成函数，一进多出。

### 说说对Hive桶表的理解？

分桶也是为了分散数据！分桶的作用，可以按照字段将数据分散到多个文件！

可以使用抽样查询结合分桶操作，只选择指定的桶进行查询！

### Hive可以像关系型数据库那样建立多个库吗？

可以!

### Hive实现统计的查询语句是什么？

count(),max(),avg(),sum(),min()

rank(),row\_number(),ntile(),dense\_ran()

### Hive优化措施

①在简单查询时使用fetchtask

②在测试和小文件的试验中，本地模式速度快！

③能够使用Mapjoin尽量使用mapjoin

④聚合操作如果数据量过大！开启map端聚合，将原先的一个MR通过两个MR实现！

hive.map.aggr = true

hive.groupby.mapaggr.checkinterval = 100000

hive.groupby.skewindata = true

⑤count(distinct)，在数据量过大时，可以先group by 再执行count操作！

⑥开启严格模式，避免无效hql的执行

⑦行列过滤

列过滤： 按需查询！避免select \*

行过滤： 在查询时尽量先通过where将数据集的范围缩小，再进行关联等计算！

⑧在执行hive时，提前对数据进行抽样和调查，合理设置map和reduce个数避免数据倾斜

⑨在机器性能好的前提下，可以设置MR的并行执行！

set hive.exec.parallel=true;

set hive.exec.parallel.thread.number=16;

⑩针对小文件过多造成的小任务过多，开启jvm重用！

mapreduce.job.jvm.numtasks=10-20

⑩①在必要是更换hive的执行引擎，在tez或者spark上执行hql语句！

### Hive中，建的表为压缩表，但是输入文件为非压缩格式，会产生怎样的现象或者结果？

如果使用load加载数据，那么文件在上传到表中时依然为非压缩格式！

如果使用insert into 的方式插入，那么会以压缩格式存在！

### 已知表a是一张内部表，如何将它转换成外部表？请写出相应的hive语句。

alter table a set tblproperties(‘EXTERNAL’=’TRUE’);

### Hive中mapjoin的原理和实际应用？

原理同11题的MapJoin原理！

作用是为了避免出现在Reducer端的数据倾斜。

### 订单详情表ord\_det(order\_id订单号，sku\_id商品编号，sale\_qtty销售数量，dt日期分区)任务计算2016年1月1日商品销量的Top100，并按销量降级排序

select sku\_id,sum(sale\_qtty) sum\_sale

from ord\_det

where dt=20160101

group by sku\_id

order by sum\_sale desc

limit 100

### 一个表STG.t\_ORDER，有如下字段:OrderDate，Order\_id，User\_id，amount。请给出sql进行统计:数据样例:2017-01-01,10029028,1000003251,33.57。

1. 给出2017年每个月的订单数、用户数、总成交金额。

select month(orderdate),count(order\_id)ordercount,count(distinct user\_id) userCount,sum(amount) sum

from t\_order

where year(orderdate)=2017

group by month(orderdate)

1. 给出2017年11月的新客数(指在11月才有第一笔订单)。

select count(\*)

from

(select user\_id,orderdate,order\_id, lag(orderdate,1,'无') over(partition by user\_id order by orderdate) last\_orderdate

from t\_order)tmp

where

last\_orderdate='无'

and month(orderdate)=11

and year(orderdate)=2017

## Flume

### flume有哪些组件，flume的source、channel、sink具体是做什么的

flume的组件有source,channel,sink,interceptor,channel selector,sink processor

source :对接不同的数据源，将数据封装为event，传输给channel!

sink: 负责从channel中获取event，将event写入到指定的目标！

channel:介于source和sink中的一个缓冲！负责临时存储event!

### 你是如何实现flume数据传输的监控的

使用ganglia来监控；

或者使用Json Reporting将Flume的相关数据以json格式生成，交给前端进行可视化！

自定义监控类，继承MonitoredCounterGroup，来自定义监控的逻辑！

### flume的source,sink,channel的作用？你们source是什么类型？

使用tairdirsource监控一个实时写入的文件！

使用netcat source监控一个网络端口！

### Flume的Channel Selectors

默认为Replicating Channel Selector，它将event复制发送到所有的channel!

Multiplexing Channel Selector则根据event中指定的header信息，将event发送到指定的channel!

### Flume参数调优

①Source

增加Source个数（使用Tair Dir Source时可增加FileGroups个数）可以增大Source的读取数据的能力。例如：当某一个目录产生的文件过多时需要将这个文件目录拆分成多个文件目录，同时配置好多个Source 以保证Source有足够的能力获取到新产生的数据。

batchSize参数决定Source一次批量运输到Channel的event条数，适当调大这个参数可以提高Source搬运Event到Channel时的性能。

②Channel

type 选择memory时Channel的性能最好，但是如果Flume进程意外挂掉可能会丢失数据。type选择file时Channel的容错性更好，但是性能上会比memory channel差。

使用Kafka Channel兼顾性能和安全！

使用file Channel时dataDirs配置多个不同盘下的目录可以提高性能。

Capacity 参数决定Channel可容纳最大的event条数。transactionCapacity 参数决定每次Source往channel里面写的最大event条数和每次Sink从channel里面读的最大event条数。transactionCapacity需要大于Source和Sink的batchSize参数。

③Sink

增加Sink的个数可以增加Sink消费event的能力。Sink也不是越多越好够用就行，过多的Sink会占用系统资源，造成系统资源不必要的浪费。

batchSize参数决定Sink一次批量从Channel读取的event条数，适当调大这个参数可以提高Sink从Channel搬出event的性能。

使用Kafka Sink构建合理的数据流拓扑结构，提高数据并发消费能力！

### Flume的事务机制

event从source到channel为put事务！put事务在一批event被拦截器处理后，准备存储到channel时开启事务，全部存储完毕后提交事务，如果失败则回滚事务！

event从channel到sink为take事务！同理，sink开始写入数据时开启事务，一批event被sink从channel中全部写入到指定目标后提交事务，然后清除在channel中存储的event!发生异常时则回滚事务，保证event的安全！

### Flume采集数据会丢失吗?

会！

使用exec source有丢失数据的风险！

使用memory channel会在agent故障时丢失阶段性数据！