1. 在非局域网中配置Hadoop集群ip，外节点使用公网ip，本节点使用私有ip。

2、在配置hadoop2.3.0集群时，在yarn-site.xml配置文件中需要显示指定resourcemanager的相关信息，否则即使各个节点上启动了nodemanager，也会因为找不到resourcemanager而去连接0.0.0.0的ip的相关yarn的80xx端口而报错，在8088的web页面上看到的集群内存分配也是0（即使在配置文件中已经配置），需要配置的信息如下：

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>your\_rm'host:8031</value>

</property>

<property>

<name>yarn.resourcemanager.address</name>

<value>your\_rm'host:8032</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>your\_rm'host:8030</value>

</property>

<property>

<name>yarn.resourcemanager.admin.address</name>

<value>your\_rm'host:8033</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address</name>

<value>your\_rm'host:8088</value>

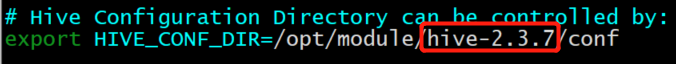
</property>

3、配置hive的mysql matestores时，需要先初始化数据库

schematool -dbType mysql -initSchema

4、hive配置mysql的metastore时失败

Hive-env.sh的不要直接复制笔记



4、在hive中使用beeline，需要

1）在core-site.xml配置：

<property>

<name>hadoop.proxyuser.zrf.hosts</name>

<value>\*</value>

</property>

<property>

<name>hadoop.proxyuser.zrf.groups</name>

<value>\*</value>

</property>

2）在hdfs-site.xml中配置：

<property>

<name>dfs.webhdfs.enabled</name>

<value>true</value>

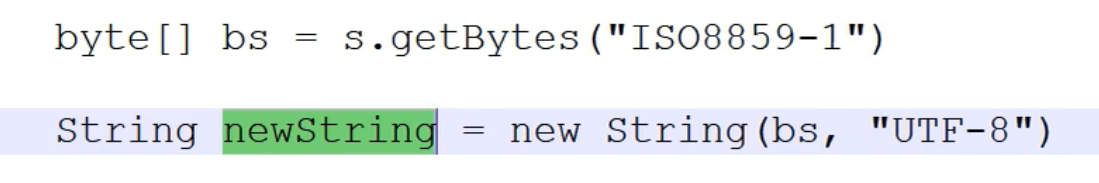
</property>

6、kafka API连不上？

答：更改advertised.listeners=PLAINTEXT://192.168.113.128:9092

7、双亲委派机制

8、乱码问题



9、什么时候应该启动hiveserver2？

答：当通过jdbc连接hive时需要，比如beeline、一些可视化客户端。

什么时候启动metestore？

答：获得元数据两种方式：客户端支链、启动metestore

10、如果从MySQL向hdfs导入数据失败，可能原因是安全模式正在开启

11、使用脚本启动hadoop104的消费flume，进程启动成功但不起作用，必须在控制台直接输入命令？

答：创建/etc/profile.d/my\_env.sh 文件，添加环境变量

login shell启动时会加载/etc/profile。

non-login shell启动时会加载~/.bashrc。

但是无论在加载~/.bashrc（实际上是加载了~/.bashrc中的/etc/bashrc）或/etc/profile时，都会执行/etc/profile.d/\*.sh，所以，无论在login shell或non-login shell环境中，都会加载/etc/profile.d/\*.sh文件，这样我们可以自定义一个my\_env.sh文件用来存放java或者其他的环境变量，一劳永逸！

12、hive on spark中spark会话不能创建？查看hive的两个配置文件的路径、端口号是否错误

**离线数仓项目知识汇总**

1. hive的分组查询语句中，select字段只能有三种：用来分组的字段、聚合函数（如count）、常数、一进一出的函数但函数参数必须是分组字段。
2. hive常用函数
3. with as
4. sum(if(,))
5. group by可用来去重，尽量少用destinct
6. 求分组后的topN

开窗->rank->where rank<10

1. partition by 和 group by的区别

group by是计算每步的累计值，而不是只有一个最终值，而group by 只获得该组最后的最终计算值。

1. 遇到偶尔ssh连接不到linux的情况，重启一下linux的network服务
2. Azkaban启动后还需要激活才能使用，每次重启都需要重新激活
3. Azkaban 使用流程

创建两个文件：azkaban.project、gmall.flow，打包zip，上传至azkaban-web

1. superset 轻量级BI分析工具

**即席查询**

Kylin：

Cube构筑优化

1. 使用衍生维度

其实并没有减少数据量，只是计算的时间改变

1. 聚合组（强大的剪枝工具）
2. 强制维度：指定某个维度为强制维度，只计算包含强制维度的cuboid
3. 层级维度：低级维度必须依赖高级维度才能存在
4. 联合维度：联合维度必须全部存在或都不存在
5. Row Key优化
6. 查询时被用作过滤的维度放在前面（优化查询）
7. 基数大的维度放在基数小的维度前面（因为hbase是优先选择roekeyid小的cuboid进行降维的）
8. 并发粒度优化

Cuboid分片到多个分区

Presto：

1. 需要启动hive的metestore服务
2. 优点：基于内存运算，减少了硬盘IO，计算更快

能连接多个数据源，可以跨数据源传输

1. 注意事项：
   1. 原生不支持lzo，需要添加lzo依赖，但即使如此也不支持单独的lzo，需要lzo和列式存储同时存在，需要重新编译依赖包
   2. 时间字符串不能直接比较
   3. 不支持insert overwrite
   4. 支持Parquet，但只支持查询，不支持insert
2. 优化存储：
3. 合理设置分区
4. 使用列式存储：建议hive使用ORC格式存储（presto对ORC进行了特定优化）
5. 使用压缩：建议snappy

优化查询：

1. 只选择使用的字段
2. 过滤条件尽量加上分区字段
3. Group By 语句优化：字段按照基数降序
4. OrderBy时使用Limit：不会使任一一个worker压力过大
5. 使用join时大表放在小表左边

**集群监测**

Zabbix 和（Ganglia）功能类似

一般使用模板

单例模式

三大范式

Sql优化

二级索引

隔离级别

Hdfs写流程

读流程

hive调优的思路

1 好的模型设计

2 解决数据倾斜

3 减少job数

4 设置合理的map reduce的task数

5 对小文件进行合并

6 单个作业最优不如整体最优

写Sql

手写wordcount

手写 spark 求 π、wordcount、二次排序

简单总结一下：

消费端重复消费：很容易解决，建立去重表。

消费端丢失数据：也容易解决，关闭自动提交offset，处理完之后受到移位。

生产端重复发送：这个不重要，消费端消费之前从去重表中判重就可以。

生产端丢失数据：这个是最麻烦的情况。

Kafka ISR机制

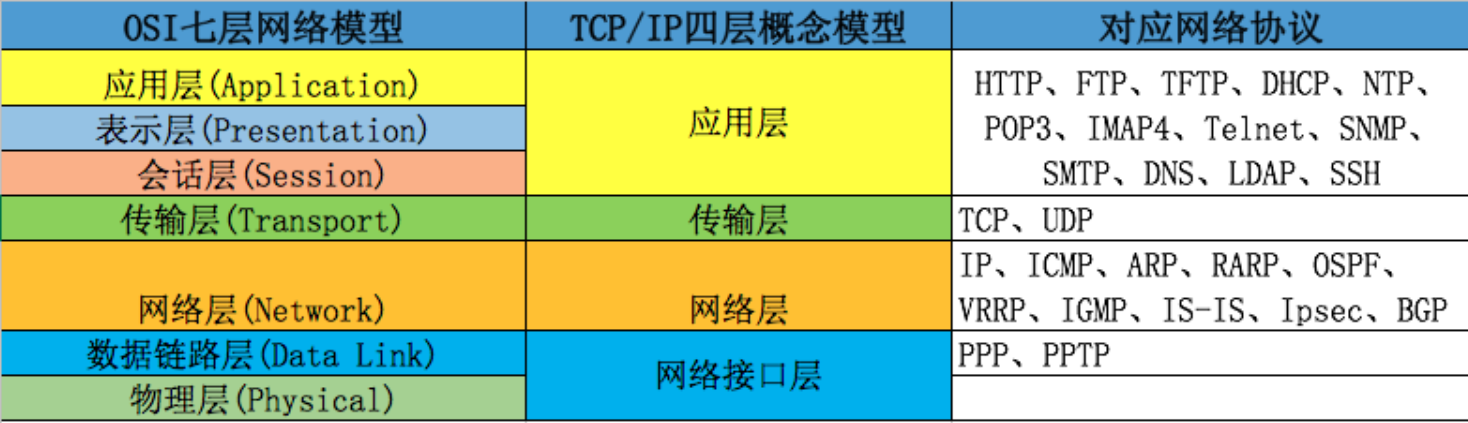
解决spark数据倾斜方法：

1. hive ETL预处理：将数据倾斜提前到ETL阶段，治标不治本
2. 过滤掉少数导致倾斜的key：主要用在产生大量异常数据的情况下
3. 提高shuffle的 并行度：防止多个key在同一task中计算
4. 两阶段聚合（局部聚合+全局聚合）：在key前加上前缀，使key平均，但只适用于reducebykey和groupby这类shuffle中
5. 使用map join：适合小表join大表
6. 采样倾斜key并分拆join操作：将导致倾斜的key单独拆成RDD算子，并加前缀拆分成n个任务，适合只有少数key倾斜的情况

Js跨域

1. jsonp
2. document.domain
3. windows.name
4. 等

TCP/IP



Spark效率比mapreduce高的原因：

1、Spark是基于内存的，而MapReduce是基于磁盘的迭代

2、有向无环图是指：一个图从顶点出发，无法再回到原点，那么这种图叫做有向无环图。

3、DAG计算模型在spark任务调度

4、Spark是粗粒度资源调度，MapReduce是细粒度资源调度

**1、Spark vs MapReduce ≠ 内存 vs 磁盘**

其实Spark和MapReduce的计算都发生在内存中，区别在于：

* MapReduce通常需要将计算的中间结果写入磁盘，然后还要读取磁盘，从而导致了频繁的磁盘IO。
* Spark则不需要将计算的中间结果写入磁盘，这得益于Spark的RDD（弹性分布式数据集，很强大）和DAG（有向无环图），其中DAG记录了job的stage以及在job执行过程中父RDD和子RDD之间的依赖关系。中间结果能够以RDD的形式存放在内存中，且能够从DAG中恢复，大大减少了磁盘IO。

**2、Spark vs MapReduce Shuffle的不同**

Spark和MapReduce在计算过程中通常都不可避免的会进行Shuffle，但Spark因为DAG，Shuffle尽可能减少；

**3、多进程模型 vs 多线程模型的区别**

* MapReduce采用了多进程模型，而Spark采用了多线程模型。多进程模型的好处是便于细粒度控制每个任务占用的资源，但每次任务的启动都会消耗一定的启动时间。就是说MapReduce的Map Task和Reduce Task是进程级别的，而Spark Task则是基于线程模型的，就是说mapreduce 中的 map 和 reduce 都是 jvm 进程，每次启动都需要重新申请资源，消耗了不必要的时间（假设容器启动时间大概1s，如果有1200个block，那么单独启动map进程事件就需要20分钟）
* Spark则是通过复用线程池中的线程来减少启动、关闭task所需要的开销。（多线程模型也有缺点，由于同节点上所有任务运行在一个进程中，因此，会出现严重的资源争用，难以细粒度控制每个任务占用资源）

总结：关于Spark为什么比MapReduce快，或者Spark速度快于MapReduce的原因，总结至少有这几点不同之处吧。

**1、消除了冗余的HDFS读写**  
Hadoop每次shuffle操作后，必须写到磁盘，而Spark在shuffle后不一定落盘，可以cache到内存中，以便迭代时使用，而且针对于spark-shuffle spark使用的是bypass的sortshuffle机制，进一步提高效率。如果操作复杂，很多的shufle操作，那么Hadoop的读写IO时间会大大增加，相对的spark就很快。  
**2、消除了冗余的MapReduce阶段**  
Hadoop的shuffle操作一定连着完整的MapReduce操作，冗余繁琐。而Spark基于RDD提供了丰富的算子操作，且reduce操作产生shuffle数据，可以缓存在内存中。  
**3、JVM的优化**  
Hadoop每次MapReduce操作，启动一个Task便会启动一次JVM，基于进程的操作。而Spark每次MapReduce操作是基于线程的，只在启动Executor是启动一次JVM，内存的Task操作是在线程复用的。每次启动JVM的时间可能就需要几秒甚至十几秒，那么当Task多了，这个时间Hadoop不知道比Spark慢了多少。  
**4.序列化方式**  
Spark序列化方式提供了kryo序列化–是一种轻量级的序列化，MR只有java的序列化–重量级。  
**5. DAG Scheduler**  
Spark 计算比 MapReduce 快的根本原因在于 DAG 计算模型。一般而言，DAG 相比MapReduce 在大多数情况下可以减少 shuffle 次数。Spark 的 DAGScheduler 相当于一个改进版的 MapReduce，如果计算不涉及与其他节点进行数据交换，Spark 可以在内存中一次性完成这些操作，也就是中间结果无须落盘，减少了磁盘 IO 的操作。