新版 HUE 使用须知

一 spark 和 hive 的语法区别

二 目前 spark 使用限制

三 spark 使用常见问题集

四 常见返回错误参考

spark 和 hive 语法区别

1.某些 hive 可使用函数 spark 不支持或者只能部分支持
1)join...on...的条件中无法使用随机数获取函数 rand()，但在 select 中是支持的
2.由于会有数据倾斜的问题，在 hive 中应尽量避免使用 count(distinct xxx)语法，采用类似 select count(1) from(select id from spark_sam group by id) t
的语句进行优化，在 spark 中无需这样做，系统会自动对 count(distinct xxx)进行优化
3.from
(select * from default.RPS__H_INDEX_USER_LIST_P_VCDZ where p_event_date='2015-12-07' and app_key regexp '^[0-9A-Z]{12}$'
) tmp
insert ...
insert ...
…
此类型语句会导致多次文件扫描源表，效率较低，当 insert 数量》=2 时请改为
Create table tmp as select * from default.RPS__H_INDEX_USER_LIST_P_VCDZ where p_event_date='2015-12-07' and app_key regexp '^[0-9A-Z]{12}$';
Cache table tmp;
Insert… from tmp;
Insert… from tmp;
…
Drop table tmp;
即创建临时表 并将其 cache 到内存，极大的提升执行效率
4.计算输入数据量(日志条数)过大时，应使用 hive 进行计算(参考值 50 亿)
5.新建表推荐使用 PARQUET 文件格式
建表语句为：

create table spark_xxx(
……
)STORED AS PARQUET
目前集群的 hive 版本对 PARQUET 支持不好，应当避免交叉使用
6.spark 不支持子查询，即不能使用
select * from spark_xxx where a1 in (select b1 from spark_yyy)
这样的语法，可用 join 实现上述功能：
select * from
spark_xxx a join
spark_yyy b
on a.a1=b.b1
如果条件是 not in,则应当用 left outer join：
select * from
spark_xxx a left outer join
spark_yyy b
on a.a1=b.b1
where b.b1 is null
7.使用 select xxxx 语句进行临时查询时，sql 末尾应加入 limit 语句对输出规模进行限制，可提高结果返回速度
8.当前集群 spark 版本在读取 orc 格式表的时候，会有概率触发没有表权限的 BUG，在实际使用中应尽量避免用 spark 读取 orc 表

目前 spark 使用限制

为保证运行 spark-sql 得到最佳体验，请遵守以下约束：
1.参与计算数据量小于 1T、50 亿条、数据文件数小于 1 万，三者需同时满足
2.查询分区表要加分区条件
3.Hue 执行 select 语句时需加 limit 限制返回结果数，目前最多返回 1000 条
4.确保执行 join 或 group by 语句时无严重的数据倾斜问题
5.不要使用全外链接，即笛卡尔积
6.若某个分区的数据量均小于 100M，请修改业务逻辑将该字段作为普通字段，不作为分区字段

spark 使用常见问题

Q：我要运行一个 sql，应该用 spark 还是用 hive
A：若要查询表加分区条件后数据量小于 1T 并且小于 50 亿条、数据文件数小于 1 万，那么使用 spark 是最佳选择，若超过两个条件任意一个请使用 hive，另外 spark 不支持 hive 的某些语法，目前已知不支持 where 子句中的 if 条件，除此之外 spark 完全兼容 hive-sql 语法

Q：为什么 hue 上看不到 spark 执行的实时 log

A：spark 服务目前不向客户端返回 log，我们正在开发这方面功能

Q：一次修改分区或分区文件过多对 spark 有什么影响，为什么同样的逻辑对 hive 没有影响
A：spark 默认是不支持动态分区的，认为分区作为动态是不合理的业务逻辑，所以没有在这方面进行优化，所以使用 spark 的时候尽量不要使用动态分区，产生过多文件。

Q：如无法避免，必须使用动态分区，产生过多文件，应如何优化
A：动态分区在业务逻辑上都可以避免，若产生文件数量过多且文件数量较小请使用 set spark.sql.shuffle.partitions=n，n=输出数据量/128M 以控制文件数量

Q：spark 是否支持 hive 的 udf\udaf\udtf 函数
A：支持，这些函数重新编译后可以直接使用

Q：spark-sql 在使用的时候与 hive 有哪些重要的注意事项
A：spark 与 hive 不同，没有根据数据量预估 shuffle 的并行度（reduce 的数量），目前设置的默认值为 600，可以满足绝大多数业务，当任务量过大 shuffle 并行度不足或任务很小，产生过多结果小文件是可以通过配置 set spark.sql.shuffle.partitions=n，n=输出数据量/128M 来调节并行度

Q：创建表时有哪些不同，或优化的策略
A：推荐新创建的表使用 PARQUET 文件格式，语句为
create table spark_(
……
)STORED AS PARQUET
该格式基于列存储，并且会压缩数据，经测试在扫描个别字段（非全表扫描）时可以带来 30%左右的性能提升

Q：为何 spark 与 hive 相比限制这么多，为什么 spark 跑大任务不稳定并且与 hive 相比优势不明显
A：shuffle 时 spark 发送端（maper）需要写与接收端一一对应的文件，当任务很大时造成很大的内存压力与磁盘 io 压力，可能导致节点停止响应，从而任务失败甚至节点丢失
同时 dag 与 mr 模型的多次迭代相比，迭代次数明显减少，导致计算更集中，更容易产生 gc oom 等问题

常见返回错误日志参考

※用户执行 hue 时返回结果数过多，异常信息如下：
Total size of serialized results of 1271 tasks (1028.8 MB) is bigger than spark.driver.maxResultSize (1024.0 MB)

※大任务 shuffle 过程中文件丢失（一般都是节点失败造成的），异常信息如下：
org.apache.spark.shuffle.FetchFailedException: java.io.FileNotFoundException: /data3/hadoop/yarn/local/usercache/spark/appcache/application_1459747388810_21728/block

mgr-4532f97c-2fe7-4780-89bc-8998b5e32ce1/00/shuffle_3_2193_0.index (No such file or directory)
该问题比较严重，会导致未完成的 shuffle 重新计算重新 shuffle，严重影响效率，一般出现这种情况的任务不建议使用 spark 计算。

※节点异常退出，异常信息如下：
Container exited with a non-zero exit code 143
具体原因需要到 nodemanager 日志去看，可能是 oom、gclimit 等一系列原因，目前发现这种错误多了会导致 spark-server 假死，其根源是跑了不适合 spark 平台的任务。

※后台报分区不存在错误
该问题是由元数据库内容与实际 hdfs 文件目录不一致造成的，目前大集群使用的 hive 版本较低，不会报这类错误，所以造成一些 hive 可以正常运行的任务在 spark 平台报错，如遇到这类问题，但这类问题可能会影响到统计数据的准确性，如遇到请尝试修复分区文件，弄清到底是该分区真的不存在还是导入时产生了错误。

※元数据获取失败错误，异常信息如下：
org.apache.spark.shuffle.MetadataFetchFailedException: Missing an output location for shuffle 6
shuffle 发生了数据倾斜，或笛卡尔积，导致某 reduce 端因数据处理量过大而失败，遇到这种问题时请修改业务逻辑，优化 sql。

※树节点异常，异常信息如下：
org.apache.spark.sql.catalyst.errors.package$TreeNodeException: execute, tree: TungstenAggregate(key=[], functions=[(first(if ((gid#22146 = 0)) count(1)#22149L else null) ignore nulls,mode=Final,isDistinct=false),(count(if ((gid#22146 = 1)) package_name#22147 else null),mode=Final,isDistinct=false)], output=[appNum#22139L,packaNum#22140L]) +- TungstenExchange SinglePartition, None +- TungstenAggregate(key=[], functions=[(first(if ((gid#22146 = 0)) count(1)#22149L else null) ignore nulls,mode=Partial,isDistinct=false),(count(if ((gid#22146 = 1)) package_name#22147 else null),mode=Partial,isDistinct=false)], output=[first#22158L,valueSet#22159,count#22160L]) +- TungstenAggregate(key=[package_name#22147,gid#22146], functions=[(count(1#22148),mode=Final,isDistinct=false)], output=[package_name#22147,gid#22146,count(1)#22149L]) +- TungstenExchange(coordinator id: 710510090) hashpartitioning(package_name#22147,gid#22146,600), Some(coordinator[target post-shuffle partition size: 500000000]) +- TungstenAggregate(key=[package_name#22147,gid#22146], functions=[(count(1#22148),mode=Partial,isDistinct=false)], output=[package_name#22147,gid#22146,count#22154L]) +- Expand [List(null, 0, 1),List(package_name#22141, 1, null)], [package_name#22147,gid#22146,1#22148] +- HiveTableScan [package_name#22141], MetastoreRelation d_dmp, base_app_type_data, None
该问题是有查询的表文件格式为 orcfile 格式引起的，出现这类异常可以使用 hive 执行即可。若想用 spark 执行，请将源表改为 parquet 或 textfile 格式。

如果下载 Excel 结果文件为乱码，按以下步骤进行转换：
打开 Excel，执行“数据”->“自文本”，选择 CSV 文件，出现文本导入向导，选择“分隔符号”，下一步，勾选“逗号”，去掉“ Tab 键”，下一步，完成，在“导入数据”对话框里，直接点确定