Shark的实践探索

亚信科技（中国）有限公司

2014年8月

AsiaInfo 亚信

关于我

小名：锎（Cf）

大名：刘昊

From:亚信大数据部门的化学家

喜欢钻研，SPARK的钟爱者

Spark

目录

Hadoop平台目前在电信经营分析系统中的应用

ETL	• 数据清洗，数据转换 • 开发自定义的MR完成
ODS→DW的数据汇总计算	• 编写HQL实现数据模型的计算,汇总 • 通过编写tcl或python脚本连接hive server
专题应用数据的计算	• 通过Sqoop或者DB自带的工具导入传统数据库 • 通过SQL在传统数据库中计算,展现

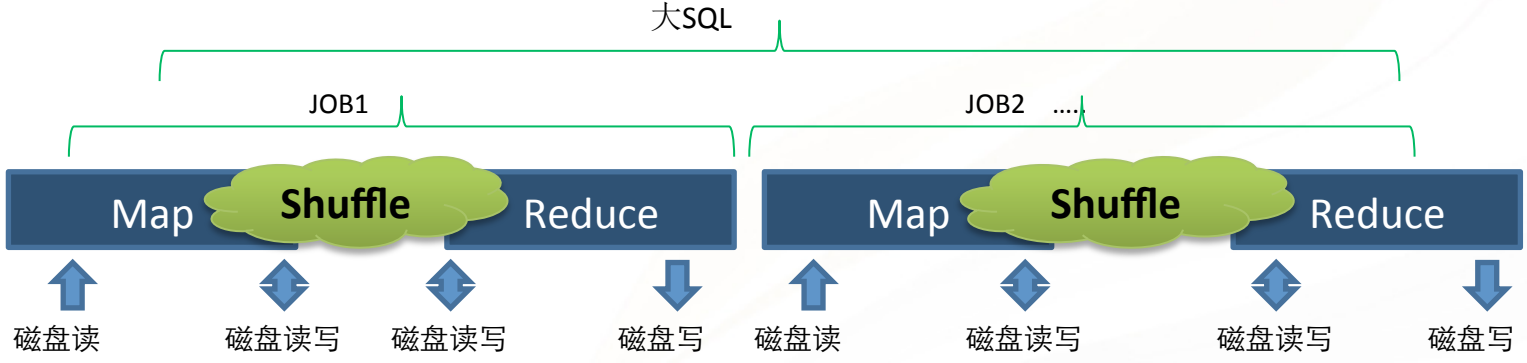
一个难以逾越的问题：慢 ！

- 一些小数据量的SQL在hive上执行时，与DB2相比效率差了很多
  -- 分析原因主要是由于hadoop基于心跳的任务调度和基于jvm进程的任务启停消耗性能

- 一些相对复杂的SQL，一个SQL会分解为多个JOB，虽然每个JOB我们已经做了充分的优化，job的执行时间并不长，但要等待所有这些JOB都执行完却要很长的时间

现状与问题

- Hive HQL的执行过程



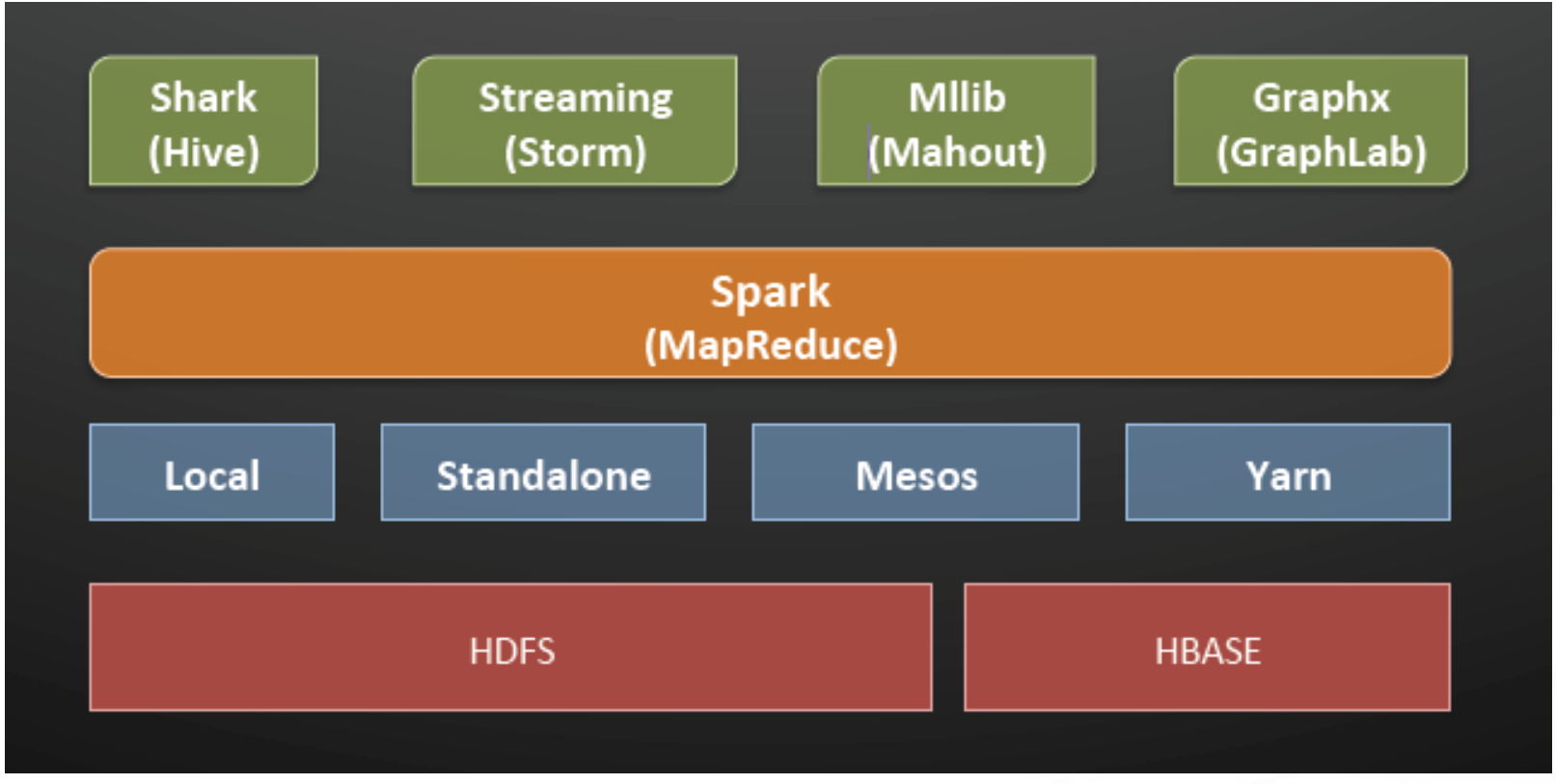
大量的磁盘读写及序列化、反序列化操作，使得执行效率非常低，若出现反复的迭代运算，现象更加明显

- Hadoop生态系统中有没有更适合处理这种场景的架构？

目录

- Hadoop + Hive 在生产应用中的问题
- 新一代大数据处理引擎 Spark && Shark
- Spark + Shark 在生产环境中的实践
- 总结与思索

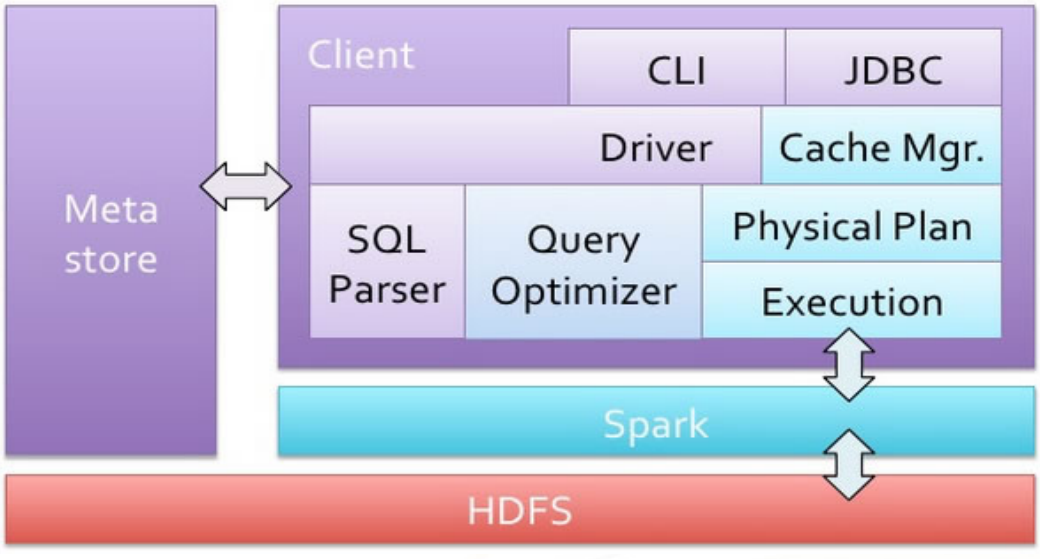
Spark生态系统


Shark
(Hive)
Streaming
(Storm)
Mllib
(Mahout)
Graphx
(GraphLab)
Spark
(MapReduce)
Local
Standalone
Mesos
Yarn
HDFS
HBASE

Spark对比MapReduce的优势

MapReduce	Spark
数据存储结构：磁盘hdfs文件系统的split	使用内存构建弹性分布式数据集RDD，对数据进行运算和cache
编程范式：Map + Reduce	DAG(有向无环图)：Transformation + action
计算中间数据落磁盘，io及序列化、反序列化代价大	计算中间数据在内存中维护，存取速度是磁盘的多个数量级
Task以进程的方式维护，任务启动就有数秒	Task以线程的方式维护，对小数据集的读取能达到亚秒级的延迟

准实时SQL查询引擎Shark



- Shark是运行在Spark上的Hive
- 将sql解析为在Spark上运行的task

可以无缝对接HIVE Queries, 重用HIVE的SQL Parser & Metastore & Query Optimizer，并支持CACHE Table

重写sql解析执行的operator ， 底层应用Spark引擎来加速计算

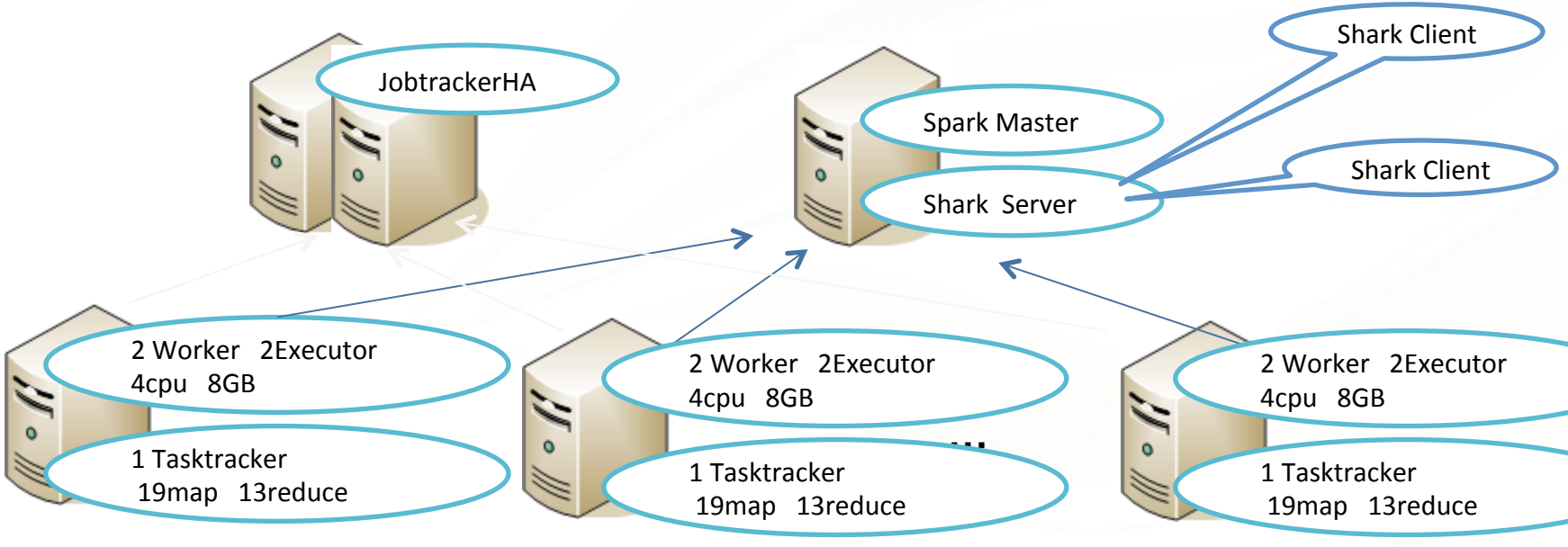
目录

- Hadoop + Hive 在生产应用中的困境
- 新一代大数据处理引擎 Spark && Shark
- Spark + Shark 在生产环境中的实践
- 总结与思索

XX现场Spark集群情况

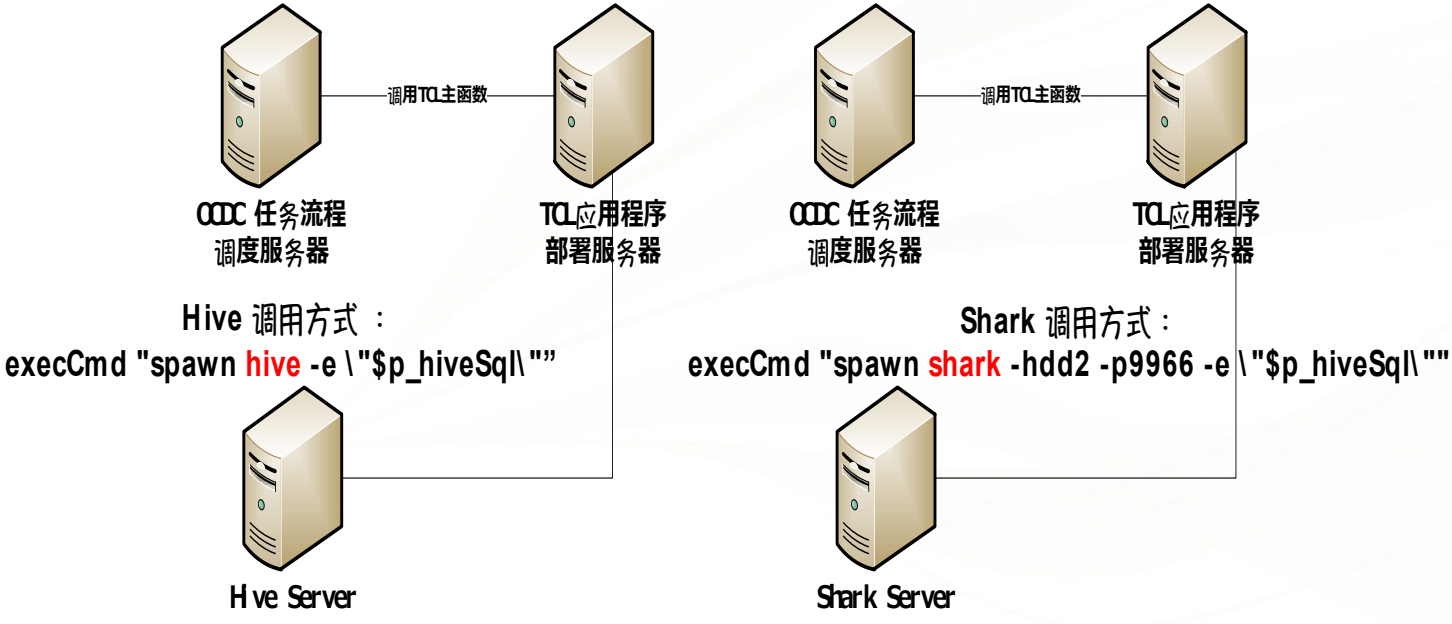
与MapReduce集群共享系统资源:

- 主节点(1台): spark master + shark server
- 子节点(15台): standalone模式，2 spark worker + 2 StandaloneExecutor (2cpu + 4GB)
- spark、shark共占用系统10%的cpu和15%的内存资源



应用程序改造内容

- 由于shark和hive的兼容性，只需改造tcl主函数调用脚本，所有用hive运行的流程就可以切换成shark上运行了



集群性能调优

1. GC

   -- Minor GC

   新生代过大：一次minor GC 时间过长

   新生代过小：老年代负担大，fullgc

   E = block * compRatio * parallel

   -Xmn = *4/3*E

   -- Full GC（数据量不是特别巨大）

   spark.storage.memoryFraction=0.2

   spark.default.parallelism=core*(2~3)

   mapred.reduce.tasks=200

2. Not Serialized（Agg）

   spark.shuffle.spill=true

   spark.shuffle.memoryFraction=0.3

集群性能调优

3. Hql改造，去除冗余的中间临时表

4. akka 网络延时，Worker丢失

5. 数据文件Block大小，造成单任务执行时间较长或过短
   --过大（256）： 单任务执行时间过长，集群资源利用不完全；
   --过小和大量小文件： 任务数过多，任务调度切换耗资源。

6. Shuffle reduce 数只有两个任务
   --过大：map 端拆分bucket过多，中间对象和缓冲区等耗资源，碎文件多
   --过小：shuffle 数据集过大，写磁盘或溢出
   mapred.reduce.tasks=200

运行情况对比

12个tcl由hive运行时长与shark运行时长的对比

TCL名		Hive	Shark
dw_new_comp_town_ds.tcl	生成dw层竞争对手新增用户归属营销区域表	77m	21m
co_dtdsM_gd.tcl	竞争对手区县变更	35m	9m
co_dtdyM_new.tcl	竞争对手当天话单表	26m	18m
nb_atvdt_td.tcl	生成每天所有使用TD业务的用户活动表	32m	12m
pr_stcdr.tcl	统计话单的延时情况	31m	8m
ac_dwPrtAcctdM.tcl	集团客户日帐单——集团非标准化产品帐单	92m	20m
dw_new_chuanka_user_town_ds.tcl	生成dw层移动新增用户营销区域归属表	95m	22m
ac_dwGrpPtdMnew.tcl	集团客户日帐单——集团统一付费部分	67m	26m
dw_new_chuanka_lt_town_ds.tcl	生成dw层竞争对手(联通)新增用户归属营销区域表	55m	18m
nb_dwmmsdM.tcl	生成短信日DW表中的数据	34m	13m
nb_wland.tcl	处理wlan 业务的日呼叫数据	33m	13m
cr_24houroutdM.tcl	省外语音详单日st聚合表	24m	16m

结论：SQL

越复杂，提升的效果越明显

目前存在的问题

改造后结果不尽如人意的tcl

dw_new_user_town_ds.tcl	nb_atvdt.tcl	fbcard_imei_black.tcl
co_dtdsM.tcl	nb_dwgdM.tcl	cr_dwcdM.tcl
co_dodsM.tcl	..	

现象：tcl中某些SQL的运行长时间不结束，并且不断的有FullGC出现，最终出现OutOfMemory的现象任务失败。

问题分析

原因：

1. shark 在做 join 操作；
2. join 相关数据表数据量过大；
3. reduce 数量设置为200。

分析：shark 在做join 操作时缺少spill 机制

1. spark 在 agg 和 cogroup 时，shuffle 数据可溢出磁盘；
2. shark 重写了CoGroupRDD类，通过维护一个hashMap进行数据存储，但不具备spill功能；
3. shark 还没提供自动计算reduce任务数的方法；

当shuffle数据量巨大且同时多任务运行，很容易出现OOM。

解决： 我们已经提出相应patch

目录

- Hadoop + Hive 在生产应用中的困境
- 新一代大数据处理引擎 Spark && Shark
- Spark + Shark 在生产环境中的实践
- 总结与思索

我们在社区上的贡献

发现shark的两个bug，并成功将patch提交社区；逐步实现扩展功能，功能性能测试及提交社区

现象	原因
执行数据量较大的复杂SQL时，有时会出现NullPointerException的异常	当按PartitionKey值做Combine后的结果集还需在做Combine时缺少方法定义
执行row_number()函数有时会丢失一些记录	PartitionTableFunctionOperator 中的LazyPTFIterator的逻辑有问题，当最后一行记录的PartitionKey与上一行的PartitionKey不同时，这行记录就丢掉了
实现Shark的虚拟列功能	由于Shark与Hive的执行机制相差较大，Shark未实现对虚拟列的查询功能。但在我们的生产场景中有对该功能的需求，尝试实现该功能。

关于Shark总结

- Spark + Shark 最适合的场景是什么？
  - 相对于Hive而言，小数据处理、迭代计算
  - 需要对静态表进行频繁访问，使用cache表
  - 包含大量临时表的处理流程

- Spark + Shark 是否已经能够替换 MR + Hive ?
- 优点：
  - 性能高，效率快
  - 完全兼容Hive,迁移平滑

  缺点：
  - 内存溢出问题
  - shark后续停止演进

我们认为很难完全替换MR+Hive，在SparkSQL等完善前，依然是个不错的选择

关于Spark思考

- Spark在YARN上的粗粒度资源使用
  - 只要shark client或shark server运行着就会占用系统的内存资源，不会随任务的多少而变化
  - 社区应该会在3-6个月内能够解决这个问题
- 在YARN的资源管控框架下与其他计算框架的协同竞争资源
- Shark的继任者
  - SparkSQL (git branch spark-1.0-jdbc)
  - Hive on Spark (JIRA HIVE-7292)

Thanks

AsiaInfo 亚信