

QCon 全球软件开发大会 【北京站】2016

Spark在360的大规模实践与经验分享

李远策

QCon

2016.10.20~22
上海·宝华万豪酒店

全球软件开发大会 2016

[上海站]



购票热线: 010-64738142
会务咨询: qcon@cn.infoq.com
赞助咨询: sponsor@cn.infoq.com
议题提交: speakers@cn.infoq.com
在线咨询 (QQ): 1173834688

团·购·享·受·更·多·优·惠

7折 优惠 (截至06月21日)
现在报名, 立省2040元/张

题纲

360Spark 平台介绍

业务及应用案例

经验 & 改进分享

360-Spark集群概况

集群类型	节点数	内存	部署方式	业务	作业数
通用计算	3000+	64G	与MR混部	SparkSQL/ 通用计算	1.5W+/day
机器学习	500+	288G	独占	机器学习/ 图计算	5000+/day

Spark在360的演进

2015.05

2015.08

2015.11

2016.02



题纲

360Spark平台介绍

业务及应用案例

经验 & 改进分享

360-Spark应用

➤ *MLlib*

- 算法: LDA、LR、FP-Growth、ALS、KMeans、随机深林等。
- 业务: 新闻主题分类、新闻推荐、APP推荐、恶意代码识别、恶意域名检测等。

➤ *GraphX*

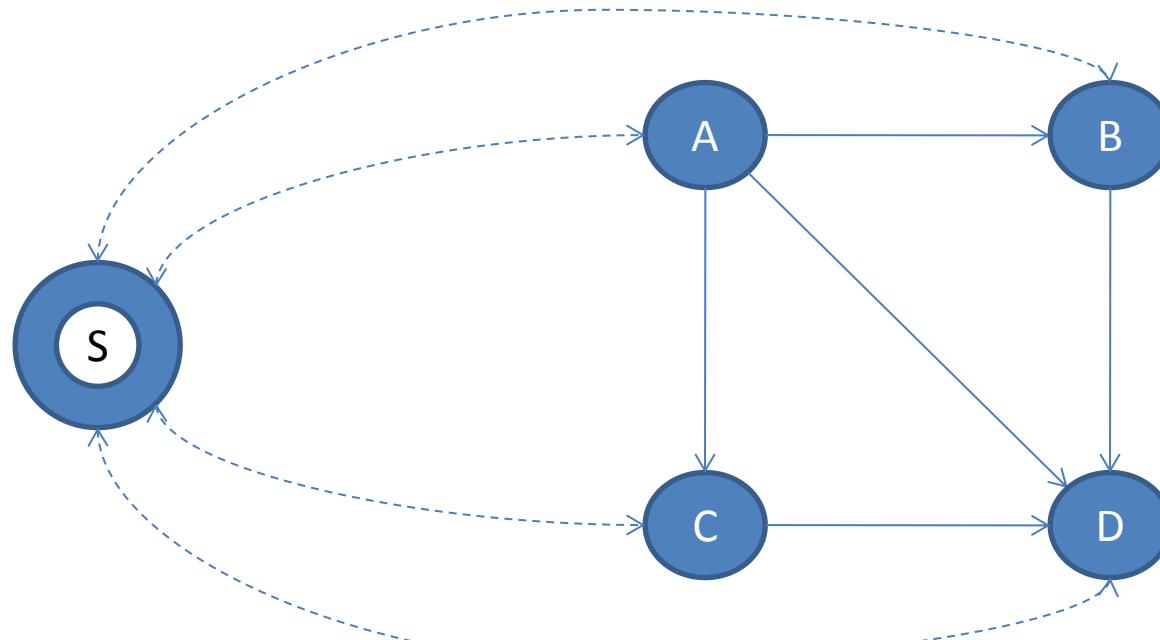
- 算法: PageRank、Louvain、LPA、连通子图等。
- 业务: 搜索PageValue、网站安全监测等。

➤ *SparkSQL*

- 采用HiveContext替换公司90%以上的Hive作业，每天例行1.5W+作业。
- 每个Hive SQL平均3轮MR作业，平均性能提升2~5倍。

PageRank on Spark

千亿量级



计算公式: $W = \alpha * W_p + (1 - \alpha) * \text{Random}$

PageRank on Spark

内置org.apache.spark.graphx.lib.Pagerank算法不适合 

GraphX在千亿边规模上难以胜任 

自己动手

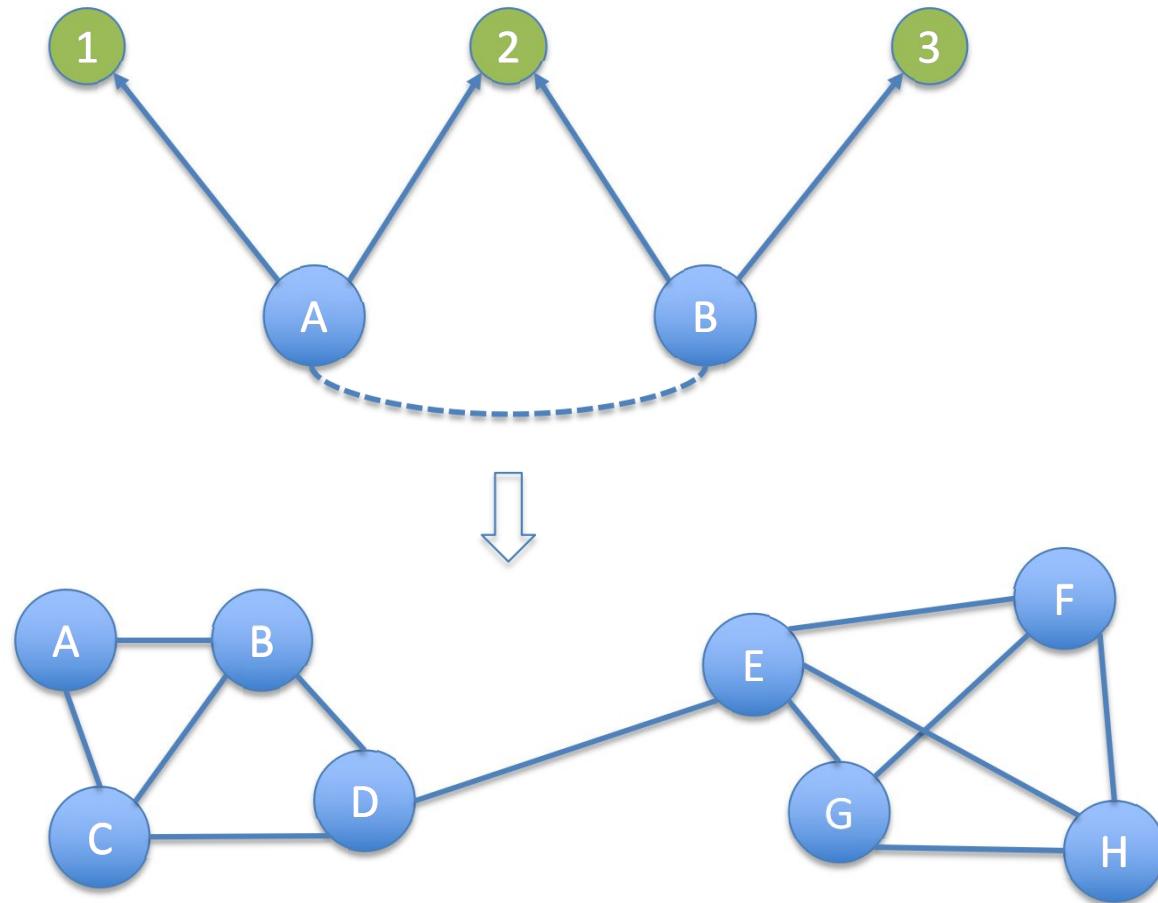


PageRank on Spark (续)

核心代码:

```
val links = ... //网页传递概率RDD [(pageId, [(pageId, ratio)])]
val ranks = ... //网页value RDD [(pageId, rank)]
val ratios = ... //网页赋值权重RDD [(pageId, ratio)]
val totalZ = ... //网页权重总和 Long
val alpha = ... //网页权重转义比例 float
循环 {
    val contribs = links.join(ranks).values.flatMap{ case (urls, rank) =>
    urls.map(url => (url._1, url._2*rank))}
    val ranksTmp = contribs.reduceByKey(_ + _).mapValues(alpha * _)
    val contribsZ = ranksTmp.map(x => x._2).sum()
    val superZ = totalZ - contribsZ
    val ranksZ = ratios.map{x => (x._1, x._2*superZ)}
    ranks = ranksTmp.rightOuterJoin(ranksZ).map{case(pageId, (rankTmp,
    rankZ)) => (pageId, rankTmp.getOrElse(0.0) + rankZ) }
}
```

社群发现在网络安全的应用-恶意域名发现



社群发现在网络安全的应用-恶意域名发现

社群发现结果：

社群1：A、B、C、D

社群2：E、F、G、H

K-means聚类结果：

簇1：A、B

簇2：C、D

簇3：E、F、G、H

(E、F、G、H) \longrightarrow (E、F、G、H)

Spark做社群发现的历程（1）

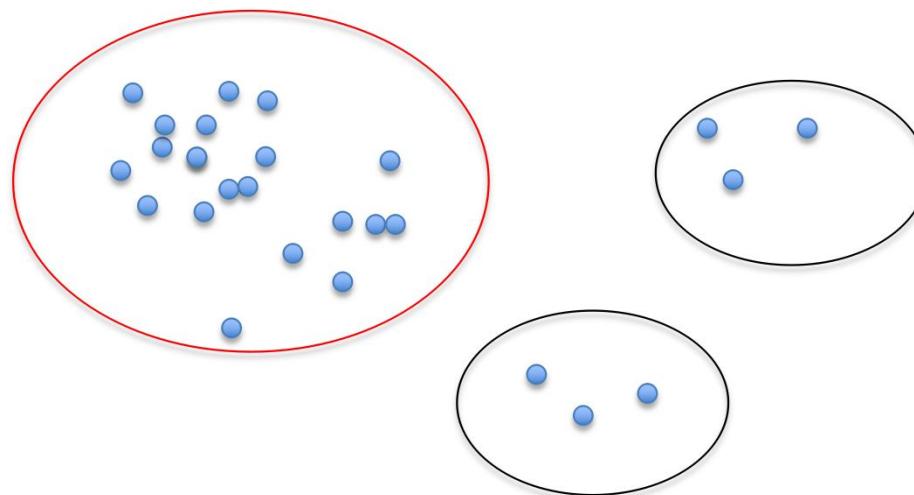
内置org.apache.spark.graphx.lib.LabelPropagation在大规模场景下性能太差，原因：

```
def mergeMessage(count1: Map[VertexId, Long], count2: Map[VertexId, Long])  
  : Map[VertexId, Long] = {  
  (count1.keySet ++ count2.keySet).map { i =>  
    val count1Val = count1.getOrElse(i, 0L)  
    val count2Val = count2.getOrElse(i, 0L)  
    i => (count1Val + count2Val)  
  }.toMap  
}
```

存在大量的内存拷贝开销，使用scala.collection.mutable.Map
做内存优化性能提升数倍

Spark做社群发现的历程（2）

内置org.apache.spark.graphx.lib.LabelPropagation经常出现超大社区的问题，效果不太理想。



Spark做社群发现的历程（3）

自研Louvain算法，支持：

- 1、边权重
- 2、有向图
- 3、分层结果保存
- 4、增量计算

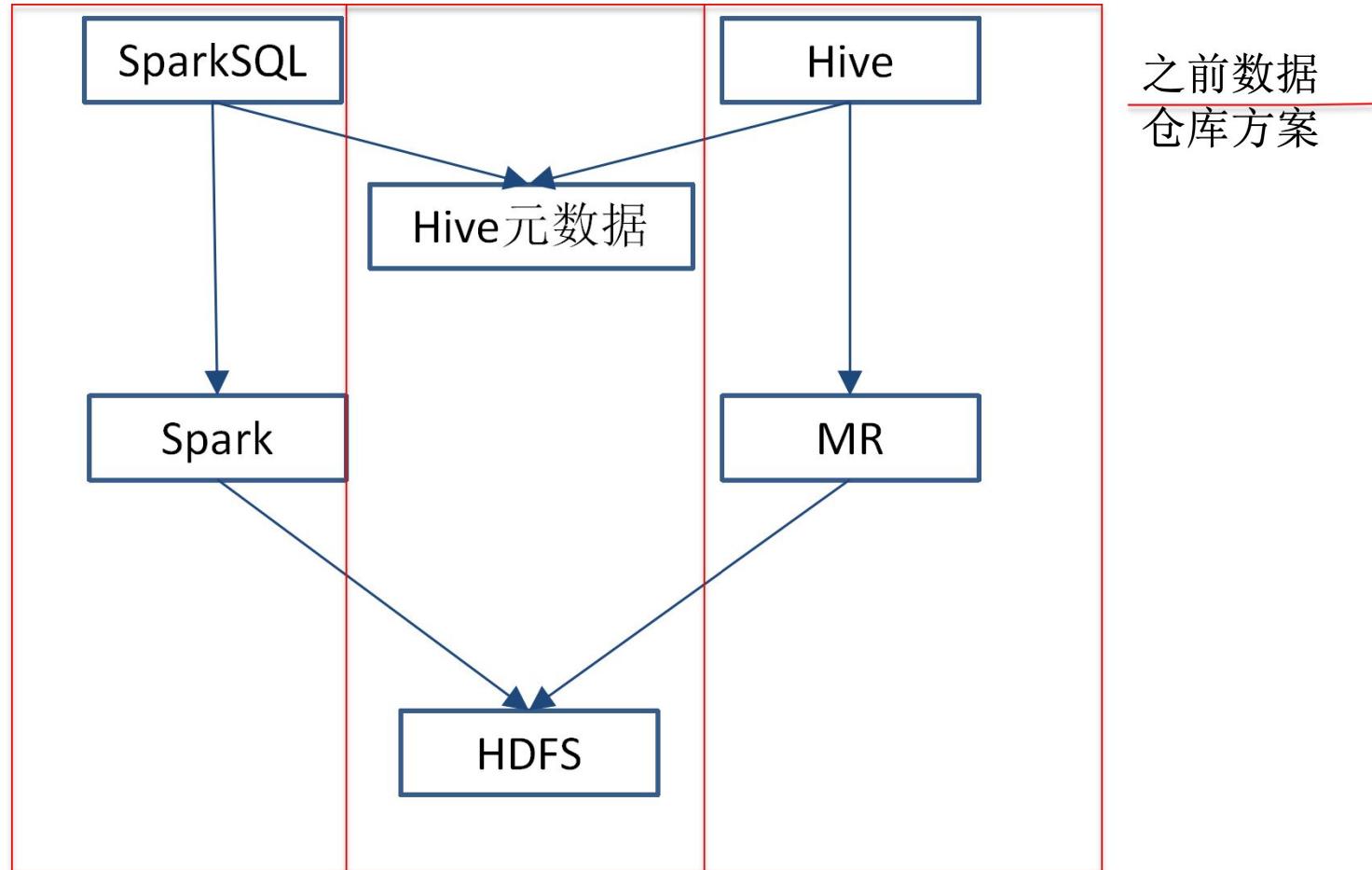
SparkSQL替换Hive

Hive迁移到SparkSQL的“正确打开方式”：

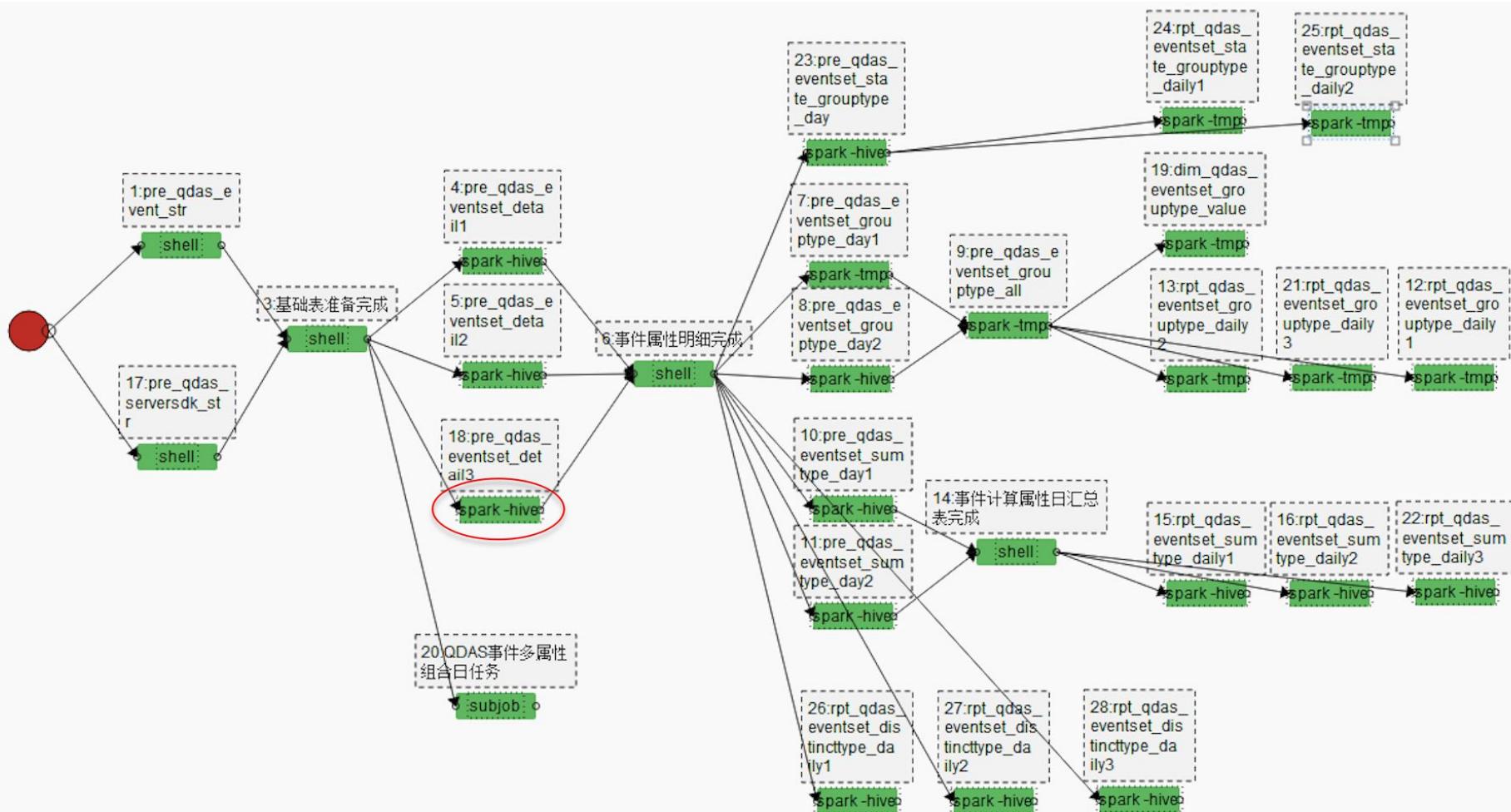
- 1、编译Spark加上-Phive -Phive-thriftserver参数
- 2、部署Spark（Yarn）集群
- 3、配置SparkSQL共用Hive的元数据库
- 4、用spark-hive（spark-sql）工具替换原有的hive命令
- 5、-e/-f 或者thriftserver提交作业。

SparkSQL部署方案

新版数据仓库方案



Hive迁移SparkSQL – 迁移方法



SparkSQL 版本选择

SpakSQL-1.4 VS SparkSQL-1.6

	1.4.1	1.6.0
Join	默认HashJoin， 支持BroadcastHashJoin和SortMergeJoin	默认SortMergeJoin， 支持BroadcastHashJoin
outer join	仅支持HashOuterJoin	支持SortMergeOuterJoin，BroadcastHashOuterJoin
Aggregation	HashAggregation	SortBasedAggregation
count(distinct)	单节点	分布式

Hive版本选择

Spark版本	支持的Hive <i>metastore</i> 版本	hive execution版本
1.4.1	0.12/0.13	0.13.1
1.6.0	0.12/0.13/1.0/1.1/1.2	1.2.1

`spark.sql.hive.metastore.version`

`x.y.z`

`spark.sql.hive.metastore.jars`

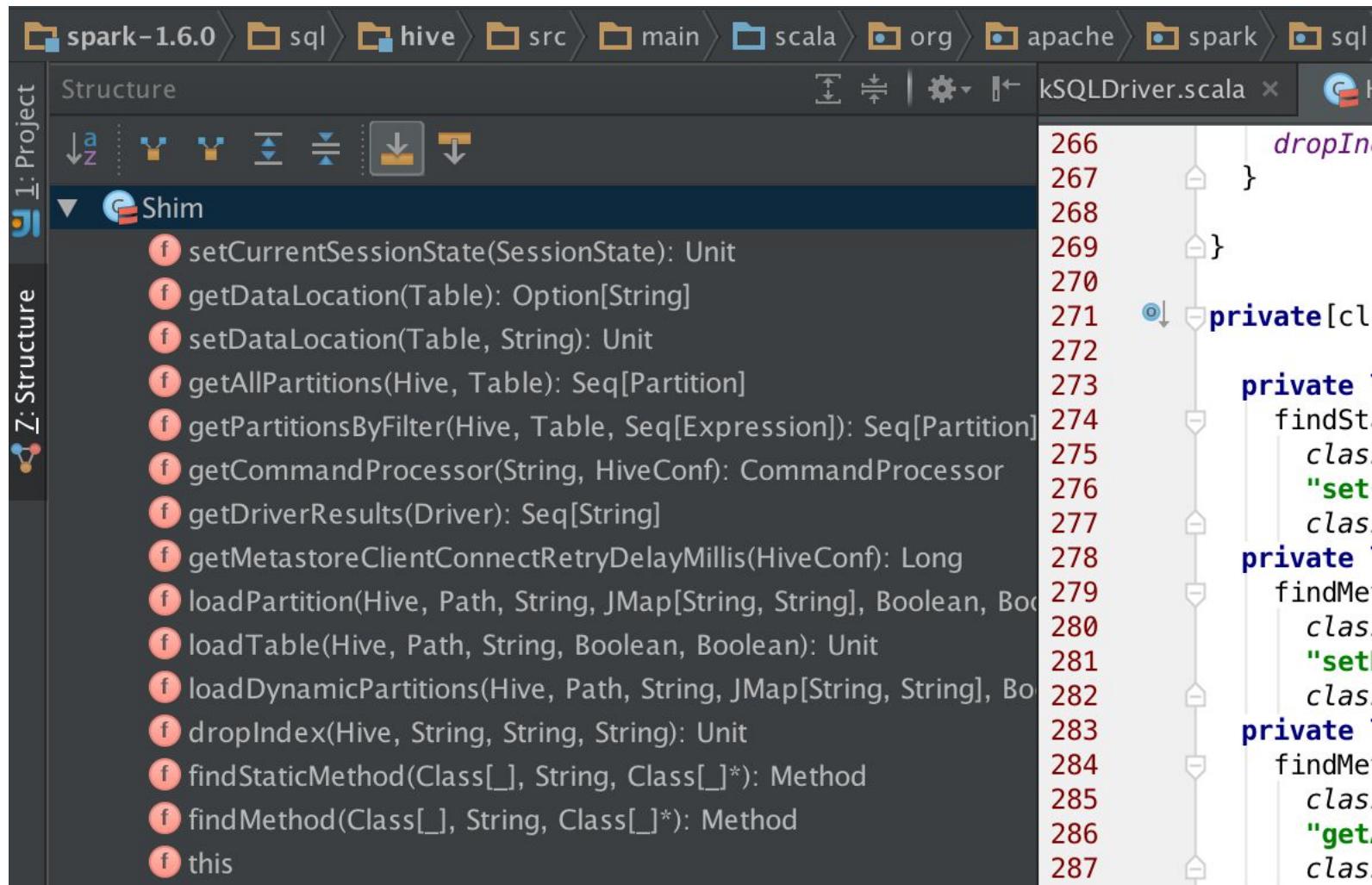
`/path/your/hiveJars`

支持多Hive MetaStore版本

实现原理

- 1、HiveContext中创建一个URLClassLoader作为clientLoader加载不同版本的Hive metastore classes。
- 2、在ClientWrapper中调用Hive的接口时会切换到clientLoader中，然后借助HiveShim去反射Hive中对应函数。

自定义Hive MetaStore版本



The screenshot shows an IDE interface with the following details:

- Project Path:** spark-1.6.0 > sql > hive > src > main > scala > org > apache > spark > sql
- Structure View:** Shows the `Shim` class with its methods:

- `f setCurrentSessionState(SessionState): Unit`
- `f getDataLocation(Table): Option[String]`
- `f setDataLocation(Table, String): Unit`
- `f getAllPartitions(Hive, Table): Seq[Partition]`
- `f getPartitionsByFilter(Hive, Table, Seq[Expression]): Seq[Partition]`
- `f getCommandProcessor(String, HiveConf): CommandProcessor`
- `f getDriverResults(Driver): Seq[String]`
- `f getMetastoreClientConnectRetryDelayMillis(HiveConf): Long`
- `f loadPartition(Hive, Path, String, JMap[String, String], Boolean, Boolean): Unit`
- `f loadTable(Hive, Path, String, Boolean, Boolean): Unit`
- `f loadDynamicPartitions(Hive, Path, String, JMap[String, String], Boolean, Boolean): Unit`
- `f dropIndex(Hive, String, String, String): Unit`
- `f findStaticMethod(Class[_], String, Class[_]*): Method`
- `f findMethod(Class[_], String, Class[_]*): Method`
- `f this`

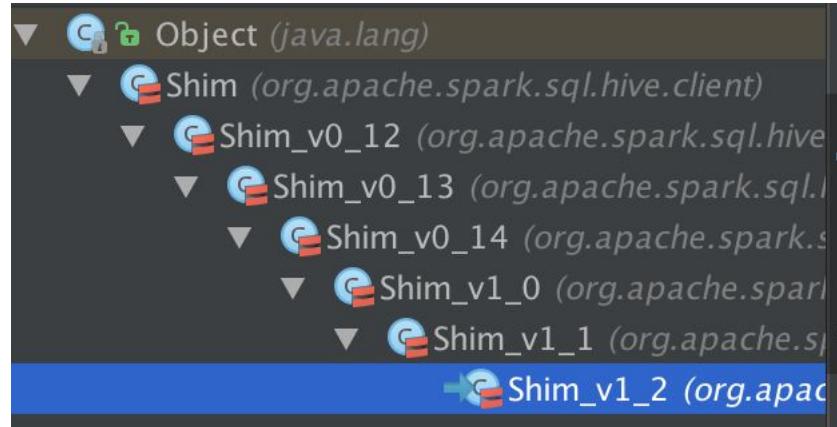
The code editor on the right shows the `kSQLDriver.scala` file with the following content:

```
266 } dropIn
267 }
268 }
269 }
270 }
271 private[cl
272 }
273 private[cl
274 findSt
275 clas
276 "set
277 clas
278 private[cl
279 findMe
280 clas
281 "set
282 clas
283 private[cl
284 findMe
285 clas
286 "get
287 clas
```

自定义Hive MetaStore版本

例如

```
private[client] class Shim_v1_1 extends Shim_v1_0 {  
  
  private lazy val dropIndexMethod =  
    findMethod(  
      classOf[Hive],  
      "dropIndex",  
      classOf[String],  
      classOf[String],  
      classOf[String],  
      JBoolean.TYPE,  
      JBoolean.TYPE)  
  
  override def dropIndex(hive: Hive, dbName: String, tableName: String, indexName: String): Unit = {  
    dropIndexMethod.invoke(hive, dbName, tableName, indexName, true: JBoolean, true: JBoolean)  
  }  
}
```



题纲

360Spark平台介绍

业务及应用案例

经验 & 改进分享

Hive迁移SparkSQL – 坑 & 改进

- SQL兼容（`Insert overwrite [local] directory`的支持）

例如：`insert overwrite directory '/tmp/testdir' select * from T1;` Hive中支持，SparkSQL暂时不支持。

因为SparkSQL-HiveContext的SQL解析调用了Hive的
`ParseDriver. parse`完成，所以语法解析上不存在问题。

Hive迁移SparkSQL – 坑 & 改进

- SQL兼容（Insert overwrite [local] directory的支持）

解决方案：

- 1、解析AST中的TOK_DIR和TOK_LOCAL_DIR将其转化成新定义的逻辑计划WriteToDirectory
- 2、将逻辑计划WriteToDirectory转换成新定义的物理计划WriteToDirectory。
- 3、在物理计划WriteToDirectory执行方法中复用InsertIntoHiveTable中的saveAsHiveFile逻辑将结果写到HDFS中。
- 4、如果是local directory则将结果再拉回到本地

Hive迁移SparkSQL – 坑 & 改进

- SQL兼容 (SQL二义性问题)

例如：

```
select C.id from (
    select A.id from testb as A
    join
    (select id from testb ) B
    on A.id=B.id) C;
```

C.id is A.id or B.id ?

Hive迁移SparkSQL – 坑 & 改进(续)

- transformation bugs (行尾部空列导致的数组越界)

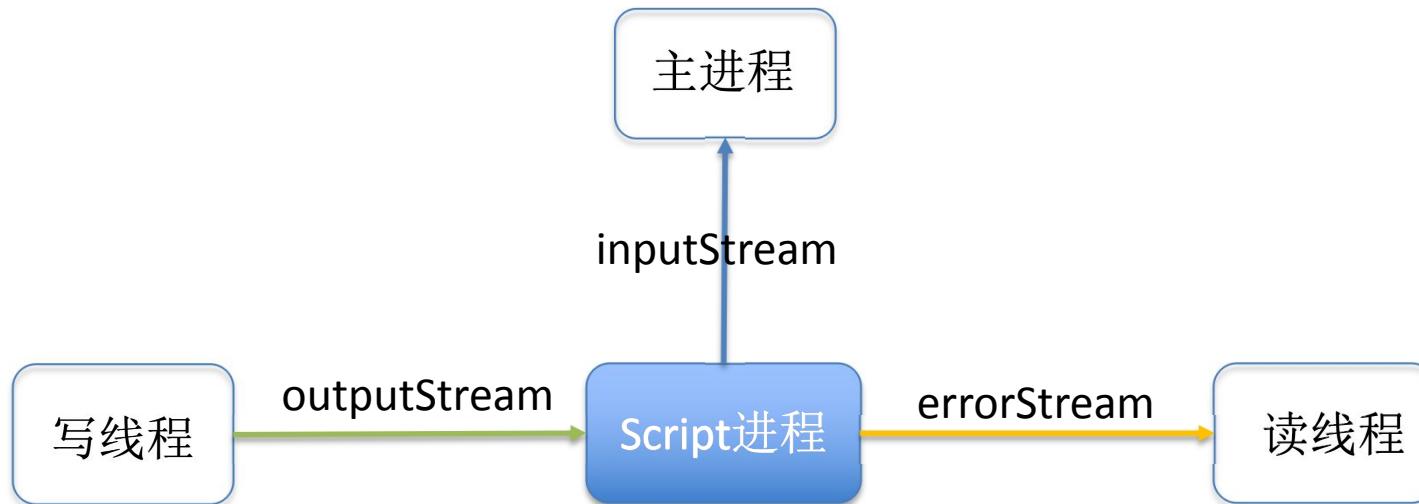
例如：

001\tABC\t002\t	→	[001, ABC, 002]
003\tEFG\t\t	→	[003, EFG]

```
new GenericInternalRow(  
    prevLine.split(ioschema.outputRowFormatMap("TOK_TABLEROWFORMATFIELD"))  
    .map(CatalystTypeConverters.convertToCatalyst))
```

Hive迁移SparkSQL – 坑 & 改进(续)

- transformation bugs (Script的标准错误缓冲区打满导致transform流程卡住)



Hive迁移SparkSQL – 坑 & 改进(续)

- 输入小文件合并的改进 (增加支持自定义inputFormat类)
默认采用建表时指定的InputFormat, 如果是默认的TextInputFormat, 当小文件比较多是可能会导致RDD的partition数太多, 导致性能下降。

解决办法: 通过参数允许用户指定InputFormat, 在TableReader中反射生成对应的InputFormat对象并传入到HadoopRDD的构造函数中。

使用方法: `set spark.sql.hive.inputformat=org.apache.hadoop.mapred.lib.CombineTextInputFormat;`

Hive迁移SparkSQL – 坑 & 改进(续)

- 输出小文件合并的改进（增加自动合并结果文件）
当spark.sql.shuffle.partitions设置的比较大且结果数据集比较小时，会产生大量的小文件（文件数等同spark.sql.shuffle.partitions）。

解决办法：在最后的执行计划中加入一个repartition transformation。通过参数控制最终的partitions数且不影响shuffle partition的数量。

使用方法： set spark.sql.result.partitions=10;

Hive迁移SparkSQL – 坑 & 改进(续)

- 支持yarn-cluster模式，减小client的负载
默认的yarn-client模式下Scheduler会运行在client上，加重client机器的负载。
解决办法：让sparkSQL工具支持yarn-cluster模式。
 - 在Yarn集群上部署SparkSQL依赖的hive metastore jar包。
 - 开通Yarn nodemanager节点访问Hive metastore数据库的权限。
 - 解决“\”转义问题。如 spark-hive -e “select * from user where name = \"张三\"”;在yarn-cluster模式中会触发两次command执行从而导致“\”被转义两次。

Hive迁移SparkSQL – 坑 & 改进(续)

Join数据倾斜

RDD1 Partition_1

id=1	value=A
id=2	value=B
id=3	value=C
id=1	value=D
id=2	value=E
id=3	value=F

Iterator

RDD2 Partition_1

id=1	value=A
id=2	value=B
id=1	value=C
id=2	value=D
id=1	value=E
id=1	value=F

HashMap

Spark-1.4.1 HashJoin

Hive迁移SparkSQL – 坑 & 改进(续)

Join数据倾斜

RDD1 Partition_1

id=1	value=A
id=2	value=B
id=3	value=C
id=1	value=D
id=2	value=E
id=3	value=F

Iterator

RDD2 Partition_1

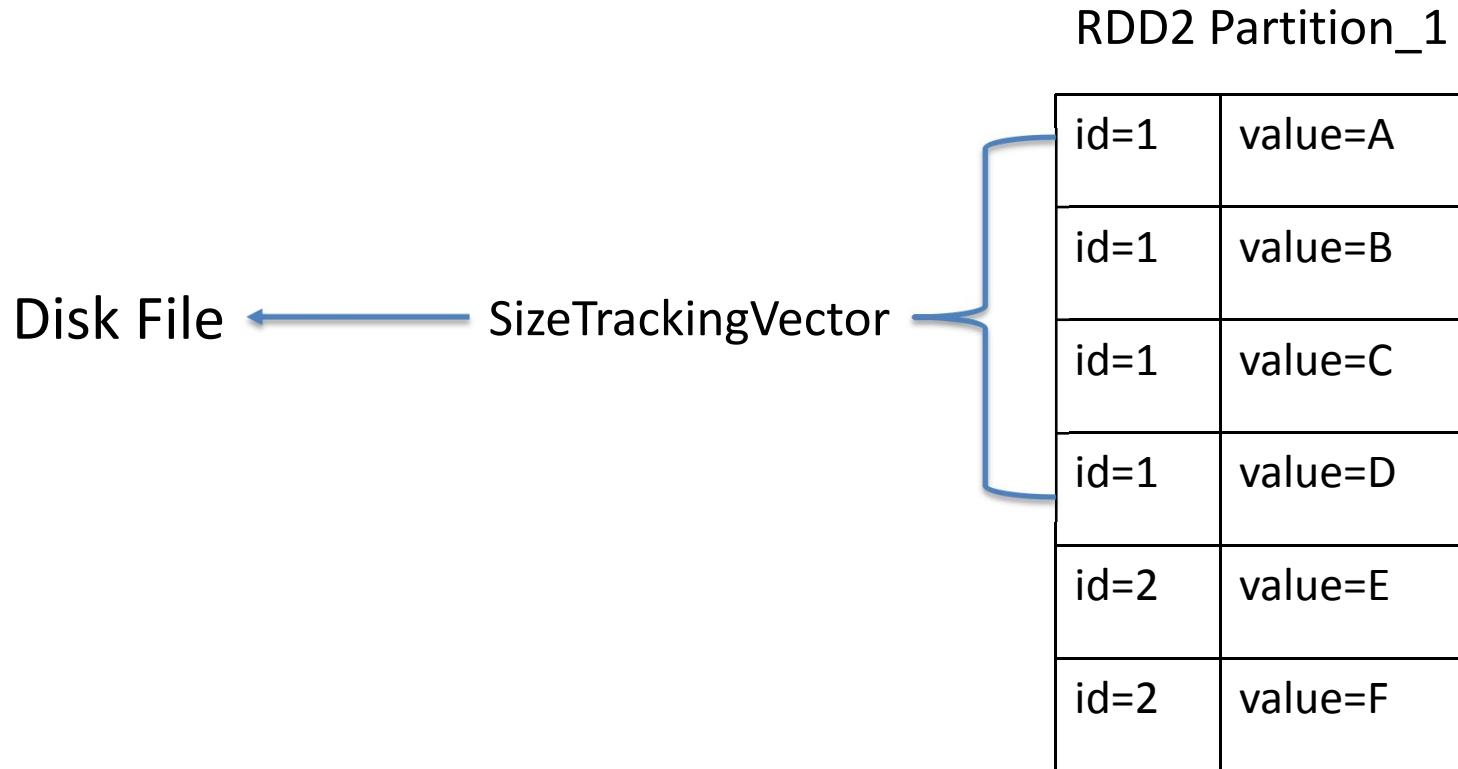
id=1	value=A
id=1	value=B
id=1	value=C
id=1	value=D
id=2	value=E
id=2	value=F

ArrayBuffer

Spark-1.6.0 SortMergeJoin

Hive迁移SparkSQL – 坑 & 改进(续)

Join数据倾斜



解决办法

Spark平台推广的一点感悟

- 平台推广要找到用户的痛点（性能、便捷、0/1问题）
- 新平台的推广要小处入手，单点突破
- 及时响应用户的问题，保证平台口碑
- deadline是第一生产力
- 对用户问题的总结和梳理

联系信息

姓名：李远策

电话：15201453364

邮箱：liyuance@gmail.com

liyuance@360.cn



急招大数据运维和运维开发人员，谢谢



THANKS!

for your listening and sleeping